

**Total Annihilation and
Total Annihilation: Kingdoms**
Reference Document

Technical Reference Document

By Arthur Keller
This version amended by Andrew L. Crystall
Portions Created by other individuals and used with permission

Version 2.5

Questions regarding this information can be directed to dawnfalcon@zoom.co.uk.

All work included herein is retained as the copyright of the original authors. Certain portions and technical data are Copyright © Cavedog Entertainment, GT Interactive, Humongous Entertainment and Infogrames, Inc., all rights reserved by copyright holders.

REVISION HISTORY	7
Project Status Report	8
Team Members	9
TOTAL ANNIHILATION	10
Directory Structure and File Formats	10
Directory Structure.....	10
Totala1.hpi Contents.....	11
Totala2.hpi Contents.....	39
HPI File Format Documentation	41
FBI Commands	51
Editable Categories.....	52
ARM Abbreviations.....	54
CORE Abbreviations.....	56
BOS Functions	57
BOS Script Tutorial – TA	59
static-var.....	60
#define.....	62
#include.....	63
Create().....	63
StartMoving().....	64
StopMoving().....	65
AimPrimary(heading_pitch).....	66
AimSecondary(heading_pitch).....	66
AimTetriary(heading_pitch).....	66
AimFromPrimary(piecenum).....	68
AimFromSecondary(piecenum).....	68
AimFromTetriary(piecenum).....	68
QueryPrimary(piecenum).....	68
QuerySecondary(piecenum).....	68
QueryTetriary(piecenum).....	68
FirePrimary(piecenum).....	69
FireSecondary(piecenum).....	69
FireTetriary(piecenum).....	69
Activate().....	70
Deactivate().....	70
StartBuilding().....	75
StopBuilding().....	75
TargetHeading(heading).....	76
QueryNanoPiece(piecenum).....	76
QueryBuildInfo(piecenum).....	77
QueryTransport(piecenum).....	77
BeginTransport(height).....	78
EndTransport().....	78
SweetSpot(piecenum).....	78
Demo().....	79

Killed(severity, corpsetype)	79
GUI File Format	80
Generic Infos about gui files :	80
COMMON Tag descriptions :	83
id :	83
Name :	83
width/height	85
xpos/ypos : Read, not so obvious detail inside	85
active :	85
fontnumber :	85
attribs :	85
assoc :	85
UNCOMMON Tag descriptions :	86
Headers (ID 0) :	86
Buttons (ID 1) :	88
Listbox (ID 2) :	89
Textbox (ID 3) :	90
Scrollbar (ID 4) :	91
Labels (ID 5) :	92
Blank Surfaces (ID 6) :	93
Fonts (ID 7) :	94
Picture Box (ID 12) :	94
GAF Format	95
SCT Format	101
OTA Format	102
BugFix Information	104
Unit ID Number Changes:	104
3rd party units don't work with Bugfix?	107
Speed Benifit:	107
AI fixes:	108
Ai profile changes:	110
Single-player Mission changes:	110
New control options!	111
Category changes in unit FBI files:	112
COB script file changes:	113
Core Necro changes:	115
Antinuke silo changes:	116
Arm Stunner EMP silo changes:	117
Krogoth changes:	117
Landmines Changes:	118
LRPC Changes:	119
Bugfix for the Arm Pelican:	119
Bugfix for ALL Hovercraft:	120
Bugfix for Naval Defensive Structures:	120
Bugfix for the Naval Dragons Teeth:	120
Bugfix for the Arm Penetrator:	121
EMG weapon changes:	121
Core Pyro changes:	121
Crawling Bomb changes:	122
Arm Fibber changes:	122

Core Leviathan Super-Sub changes:	122
Cruiser and Destroyer Changes:	122
Missile Frigate Changes:	123
Advanced bomber changes:	123
Anti-Aircraft missiles changed:	123
Flakker changes:	124
Zero-tolerance bug:	124
Weapons Changes:	124
Accumulating scars bug:	126
Corpse Changes:	126
Build Menu Changes:	127
Unit Name Changes:	127
Unit Changes:	128
Mobility changes on units:	129
GAMEDATA dir (in REV31.GP3) file changes:	129
Unfinished work:	130
Map Tutorials:	132
Terragen Map Tutorial	132
WHAT YOU WILL NEED	132
PART ONE - SETTING UP TERRAGEN	132
PART TWO - DESIGNING YOUR MAP	133
PART THREE - MAKING IT A MAP	139
Creating Custom Tilesets	143
Create a concept	143
Design Your Texture and create a template	144
Create the Terrain Features	144
Create A Pool of water	146
Create the acid	148
Create a River of Acid	149
Converting the sections to the TA color palette	151
Importing the bitmaps	151
Editing Height	153
Finishing Touches	155
Now its time to create a Terrain Archive and make your map!	156
Make your Map	157
AI TWEAKING GUIDE:	160
What is the ai?	160
Cavedog's ai, and its problems:	160
General problems with all the ai's made by Cavedog:	161
But that can be changed!	163
What are ai's and Where do I put them?	163
How does the computer know which ai profile to use?	164
How do I change the ai?	164
Then what does make a good ai?	166
So, a good ai should build nothing but resources?	167
Obviously wasted resources are bad?	167
It's not supposed to do too much at one time, but is ALSO supposed to do everything?	167
So optimally, the ai should have 0 metal and 0 energy in storage?	167
How much resources should the ai be using to stay balanced?	168
It starts building lots of solars even though it has max energy!	169
Why is the ai slow building a fusion reactor?	169
How come the ai built 5 Advanced Aircraft Plants - I LIMITED it to just 3!	169

The difficult MATH side of ai profiles:.....	170
The ai's clockwork build patterns:.....	171
How come my ARM ai *SMOKES* my CORE ai?.....	172
Ok, I caught all of that. But my ai still seems slow to buildup!.....	172
testing. Testing. TESTING!.....	173
I'm doing a lot of testing. but what am I looking for to know what to change?.....	174
What kind of ai profile *DOES* a map need?.....	175
About MY ai.....	177
TA Weapons Creation.....	178
Types of weapons.....	178
Important values and behaviour of the weapon.....	178
Weapon Characteristics.....	179
Special weapon stuff.....	179
Looks of a weapon.....	180
The sounds it makes.....	180
Weapon Controls.....	180
Needed resources.....	180
TOTAL ANNIHILATION: KINGDOMS.....	181
Directory Structure and File Formats.....	181
Directory Structure.....	181
.....	181
.....	181
V2 Rocket.hpi Contents.....	186
V3Rocket.hpi Contents.....	187
FBI Functions.....	204
FBI TABLE.....	205
HPI Format Documentation– TA:K.....	214
CHECKSUM CALCULATION.....	217
DECOMPRESSION OF BLOCKS AND FILES.....	217
TNT Format and Conversion.....	219
.....	219
.....	219
Using the Exporter:.....	219
Heightmap:.....	220
The Minimap:.....	220
The Voidmap:.....	220
The Roadmap:.....	220
The Jpeg Key.....	221
The Auto Functions:.....	221
The Terrain Reader.....	221
Hex Keys and Hex Values.....	222
Package and Deployment!.....	222
Conversion Tutorial.....	223
Units Editor Tutorial – TDF Edit.....	223

INTRODUCTION.....	223
HOW TO USE THIS TUTORIAL ?.....	225
HOW TO USE TDF EDIT.....	226
ANIMS.....	227
CANBUILD.....	228
FEATURES.....	231
Info.....	232
GAMEDATA.....	234
OBJECTS3D.....	237
SCRIPT.....	250
SOUNDS.....	251
TRANSLATE.....	252
UNITS.....	255
MISC.....	264
COMPILATION OF THE UNIT.....	264
INSTALLATION AND CONFIGURATION OF THE PROGRAMS.....	265
3DO BUILDER +.....	272
SHORT NAME LIST.....	280
Creating Build Pictures.....	281
Customizing TA:K to Allow Third Party Races without Iron Plague.....	283

Revision History

Version	Date	Changes
2.2	9/10/01	Major Document revision, first public release
2.3	9/28/01	Added map tutorial sections, tileset creation tutorial, corrected some FBI entries, added AI tweaking section, corrected some formatting.
2.4	9/30/01	Added Weapons Creation, massive layout modifications, fixed BOS tutorial, added TA:K third party race installation instructions, reformatted TA FBI List and deleted the weapons section (redundant with newer weapons section),

Conversion Project Information

Project Status Report

Report Date: 10 Sep 2001

Task Description	Date Added	Start Date	Estimated Completion Date	Percent Complete
Recruitment of Staff		10/1/00	Ongoing	
Research Activites				
Investigation of 2nd Resource		12/1/00	5/15/01	100%
Compilation of TA FBI Commands		12/1/00	12/15/00	100%
Compilation of TA:K FBI Commands		12/1/00		60%
Compilation of TA BOS Commands		12/1/00	12/15/00	100%
Compilation of TA:K BOS Commands		12/1/00		35%
Decoding/Implementation of .gui files		3/1/01		55%
File/system layouts - TA		5/1/01	5/2/01	100%
File/system layouts - TA:K		5/1/01	5/2/01	100%
Compilation of Documents		1/1/01	Ongoing	
Research of usability of demo		6/15/01		0%
Application Development				
GUI converter		1/1/01		65%
TA - TA:K hpi converter		1/1/01		10%
Map converter		1/1/01		0%
Conversion				
Tile Conversion		6/1/01		20%
Unit Conversion		9/15/01		0%
Map Conversion		7/15/01		0%
GUI conversion		6/15/01		40%
Creation				
Create TA:K AI for TA Units		7/15/01		0%
Creation of other packages		8/1/01		0%
Create Install Package		8/7/01		0%
Testing				
Alpha testing		9/30/01		0%
Beta Testing (closed)		11/1/01		0%
Beta Testing (open)		12/1/01		0%
Final Release		1/1/02		0%

Team Members

(This list is incomplete... If your name is missing, let me know)

Name	Responsibilities	Email
Arthur "Aslan" Keller	Project Lead, Research	aslan@tauniverse.com
Andrew "Dawn Falcon" Cyrstall	Map Conversion Lead	dawnfalcon@annihilated.com
Jerry20000	Map Conversion, Tilesets	Jerry60000@hotmail.com
HANSOLO	Unit Conversion Lead	
GenghisKhanX	Unit Conversion	
Dark Rain	GUI, Units, just about everything else	rochdenis@hotmail.com

Total Annihilation

Directory Structure and File Formats

Directory Structure

```
\CAVEDOG
├── TOTALA
│   ├── BACKUP
│   └── CC
```

Directory PATH listing

Volume serial number is 0012FC94 00A3:DCB5

D:.

```
├── TOTALA
│   ├── ccdata.ccx
│   ├── ccmaps.ccx
│   ├── mptaext.dll
│   ├── online.dll
│   ├── readme.doc
│   ├── setup.exe
│   ├── setup.ini
│   ├── smackw32.dll
│   ├── Taccread.doc
│   ├── Taccread.txt
│   ├── tadwngox.dll
│   ├── taheatx.dll
│   ├── takalix.dll
│   ├── tamplayx.dll
│   ├── tatenx.dll
│   ├── tawirepx.dll
│   ├── TotalA.exe
│   ├── totala1.hpi
│   └── totala2.hpi
│
│   └── BACKUP
│       ├── online.dll
│       ├── tadwngox.dll
│       ├── taheatx.dll
│       ├── takalix.dll
│       ├── tamplayx.dll
│       ├── tatenx.dll
│       ├── tawirepx.dll
│       └── TotalA.exe
│
│   └── CC
│       ├── Ccquery.exe
│       ├── INSTALL.LOG
│       └── UNCC.EXE
```

Totala1.hpi Contents

TOTALA1.HPI

-ai

AirBattle.txt
DEFAULT.TXT
MISSIONS.TXT
SeaBattle.TXT

-anim

ALLIES.GAF
ANYMSN.GAF
Archibrief.GAF
ArchiFoli.GAF
ArchiFronds.GAF
ArchiMetal.GAF
ArchiPalms.GAF
Archipelago.GAF
ArchiTrees.GAF
Archivents.GAF
ARMAAC1.GAF
ARMAAC2.GAF
ARMAAP1.GAF
ARMAAP2.GAF
ARMACA1.GAF
ARMACA2.GAF
ARMACK1.GAF
ARMACK2.GAF
ARMACS1.GAF
ARMACS2.GAF
ARMACV1.GAF
ARMACV2.GAF
ARMALAB.GAF
ARMALAB1.GAF
ARMALAB2.GAF
ARMAMD1.GAF
ARMAP1.GAF
ARMASY1.GAF
ARMASY2.GAF
ARMAVP1.GAF
ARMAVP2.GAF
ARMAVP3.GAF
ARMBLDG.GAF
ARMCA1.GAF
ARMCA2.GAF
ARMCA3.GAF
ARMCAMO.GAF
ARMCK1.GAF
ARMCK2.GAF
ARMCK3.GAF
ARMCK4.GAF
ARMCOM1.GAF
ARMCOM2.GAF
ARMCS1.GAF
ARMCV1.GAF
ARMCV2.GAF
ARMCV3.GAF
ARMCV4.GAF
ARMINT.GAF
ARMLAB1.GAF
ARMOPT.GAF
ARMSHARE.GAF
ARMSHIPS.GAF
ARMSILO1.GAF
ARMSY1.GAF
ARMVEHIC.GAF
ARMVP1.GAF
BRIEF.GAF
COMMBOOM.GAF
commongui.GAF
commongui_french.GAF

commongui_german.GAF
CORAAC1.GAF
CORAAC2.GAF
CORAAP1.GAF
CORAAP2.GAF
CORACA1.GAF
CORACA2.GAF
CORACK1.GAF
CORACK2.GAF
CORACV1.GAF
CORACV2.GAF
CORALAB1.GAF
CORALAB2.GAF
CORAP1.GAF
CORASY1.GAF
CORASY2.GAF
CORAVP1.GAF
CORAVP2.GAF
CORAVP3.GAF
CORBLDG.GAF
CORCA1.GAF
CORCA2.GAF
CORCA3.GAF
CORCAMO.GAF
CORCK1.GAF
CORCK2.GAF
CORCK3.GAF
CORCK4.GAF
CORCOM1.GAF
CORCOM2.GAF
CORCS1.GAF
CORCV1.GAF
CORCV2.GAF
CORCV3.GAF
CORCV4.GAF
COREBLDG.GAF
CORFMD1.GAF
CORINT.GAF
CORLAB1.GAF
CORMAIN.GAF
CORSHIPS.GAF
CORSILO1.GAF
CORSY1.GAF
CORVEHIC.GAF
CORVP1.GAF
CURSORS.GAF
Desertbrief.GAF
DRYCRUSH.GAF
DRYMETAL.GAF
DRYROCKS.GAF
DRYSCARS.GAF
DRYVENTS.GAF
ENDMSN.GAF
FOG.GAF
FOGTILES.GAF
FRONTEND.GAF
FX.GAF
GEOTHERM.GAF
Greenbrief.GAF
greenvents.GAF
grnventest.GAF
hattfont11.GAF
hattfont12.GAF
HOLES.GAF
HUMANS.GAF
IceChunks.GAF
ICEMETAL.GAF
ICESCARS.GAF
ICEVENTS.GAF
IGTITLES.GAF
Lavabrief.GAF
LavaRockA.GAF
LavaRockB.GAF

LavaRockC.GAF
LavaScars.GAF
LavaSpires.GAF
LavaStuff.GAF
LavaStuff2.GAF
LOADGAME.GAF
LOGOS.GAF
LOUNGE.GAF
LOUNGE2.GAF
Lunar2Brief.GAF
Lunarbrief.GAF
MAINMENU.GAF
Marsbrief.GAF
MarsGlyphs.GAF
MarsPlants.GAF
MarsPlants2.GAF
MarsPlants3.GAF
MarsRockGone.GAF
MarsRockGoneShad.GAF
MarsRockHit.GAF
MarsRockHitShad.GAF
MarsRocks.GAF
MarsRocks2.GAF
MarsRocks3.GAF
MarsVents.GAF
Metalbrief.GAF
MISCART.GAF
mooncraters.GAF
mooncrush.GAF
moonmetal.GAF
moonrocks.GAF
moonscars.GAF
MUSIC.GAF
MUSICRT.GAF
NEWGAME.GAF
OLDMAIN.GAF
PIPES.GAF
PREFS.GAF
ROCKREC.GAF
rockrecscars.GAF
ROCKS.GAF
rocksdead.GAF
rockshurt.GAF
rockshurt2.GAF
RUINS1.GAF
SCARS.GAF
SELGAME.GAF
SELPROV.GAF
SELSIDE.GAF
SHARE.GAF
SINGLE.GAF
SKIRMISH.GAF
TALK.GAF
TALK2.GAF
TERRAIN.GAF
TITLES.GAF
TOWERS.GAF
towercars.GAF
TREES.GAF
VENT.GAF
VENTS.GAF
VISMASK.GAF
VISMASKS.GAF
WDesertbrief.GAF
WETCRUSH.GAF
WETMETAL.GAF
WETROCKS.GAF
WETSCARS.GAF
WETVENTS.GAF
WRECKAGE.GAF

—bitmaps

ARMBKG.PCX

armguibottile.pcx
armguisidetile.pcx
armguitoptile.pcx
battleroom2.pcx
battlestart.pcx
BUTTONS2.PCX
CDLOG256.PCX
CONSOLE.PCX
COREBKG.PCX
corecamp0.PCX
corecamp1.pcx
corguibottile.pcx
corguisidetile.pcx
corguitoptile.pcx
createnew.pcx
DHELP.PCX
DLoadgame2.pcx
DLoadList.pcx
DRESTART.PCX
DSavegame2.pcx
DSaveList.pcx
DSelectmap2.pcx
DVIEWMAP.PCX
ENDCAMP.PCX
FAILURE.PCX
FRONTBG.PCX
Frontbgold.pcx
Frontend1F.PCX
FrontendX.pcx
GROMMETS.PCX
Hattfont10.PCX
Hattfont11.PCX
IGButtons.pcx
IGMBRIEF.PCX
IGOPT0X.PCX
IGOPT1X.PCX
Igoptintx.pcx
igoptionsTEMP.PCX
Igoptmusx.pcx
Igoptsoux.pcx
IgoptTEMP.pcx
Igoptvisx.pcx
IGPATCH.PCX
Installgame.pcx
InstallgameJ.pcx
Installglam.pcx
LOADBAR.PCX
Loadgame2bg.pcx
LOGOTEST.BMP
mbriefarm.pcx
mbriefcor.pcx
Mission02Win.PCX
Mission02Win2.PCX
Mission02WinBW.pcx
newcampaign4.pcx
newcampaign4x.pcx
newcamplogos.pcx
OptInterface4x.pcx
Options4x.pcx
Optmusic4x.pcx
OptSound4x.pcx
OptVisual4x.pcx
OUTCOME0.PCX
OUTCOME1.PCX
playanygame4.pcx
Playgame2.pcx
Playgame2J.pcx
pressedbone.PCX
PREVIEW.PIX
SAVEGAME.PCX
selconnect2.pcx
selectgame2x.pcx
SINGLEBG.PCX

Skirmsetup4x.pcx
SMALLDog.BMP
small_cavedog_logo.pcx
stagebuttons.pcx
STAR.PCX
TEMP.PCX
temptrans.pcx
TITLSCRN.PCX
UnitRestrict.pcx
UnitRestrict4x.pcx
UnitRestrict5x.pcx

features

all worlds
DragonsTeeth.tdf
scars.tdf

archi

FOLIAGE.TDF
METAL.TDF
TREES.TDF
TREES2.TDF
VENTS.TDF

corpses

arm_corpses.tdf
arm_heaps.tdf
core_corpses.tdf
core_heaps.tdf

desert

METAL.TDF
ROCKS.TDF
RUIN.TDF
VENTS.TDF

green

METAL.TDF
ROCKS.TDF
SHRUBS.TDF
SMUDGES.TDF
steamvents.tdf
TREES.TDF

ice

CHUNKS.TDF
METAL.TDF
SCARS.TDF
VENTS.TDF

lava

METAL.TDF
NODES.TDF
ROCKS.TDF
SPIRES.TDF
VENTS.TDF

mars

GLYPHS.TDF
METAL.TDF
PLANTS.TDF
ROCKS.TDF
VENTS.TDF

metal

buildings.tdf
PIPES.TDF
steamvents.tdf

moon

CRATERS.TDF
METAL.TDF
ROCKS.TDF

wetdesert

metal.tdf
rocks.tdf
vents.tdf

fonts

ARMBUTT.FNT
ARMCONTR.FNT
ARMFONT.FNT
armfont827.FNT
BUTTONS.FNT
COMIX.FNT
CONSOLE.FNT
CORBUTT.FNT
CORBUTTX.FNT
CORCONTR.FNT
COREFONT.FNT
Corefont827.FNT
COURIER.FNT
HATT12.FNT
HATT14.FNT
MSCRIPT.FNT
ROMAN10.FNT
ROMAN12.FNT
SMLFONT.FNT

gamedata

ALLSOUND.TDF
BuildInfo.tdf
CATEGORY.TDF
HELP.TDF
LOS.TDF
METEOR.TDF
MOVEINFO.TDF
SIDEDATA.TDF
SOUND.TDF
Translate.tdf
UNITVIEW.TDF

guis

ALLIES.GUI
ANYMSN.GUI
ARMAAP1.GUI
ARMACA1.GUI
ARMACA2.GUI
ARMACK1.GUI
ARMACK2.GUI
ARMACV1.GUI
ARMACV2.GUI
ARMALAB1.GUI
ARMAMD1.GUI
ARMAP1.GUI
ARMASY1.GUI
ARMAVP1.GUI
ARMAVP2.GUI
ARMBUILD.GUI
ARMBUTT.FNT
ARMCA1.GUI
ARMCA2.GUI
ARMCA3.GUI
ARMCK1.GUI
ARMCK2.GUI
ARMCK3.GUI
ARMCOM1.GUI
ARMCOM2.GUI
ARMCS1.GUI
ARMCV1.GUI
ARMCV2.GUI
ARMCV3.GUI
ARMGEN.GUI
ARMLAB1.GUI
ARMMAIN.GUI

ARMMAIN2.GUI
ARMOPT.GUI
ARMSILO1.GUI
ARMSY1.GUI
ARMVP1.GUI
BLANK.GUI
BRIEF.GUI
BRIEFING.GUI
BRIEFX.GUI
BUILDING.GUI
BUTTONS.FNT
CDCHECK.GUI
CHATSEL.GUI
CMENU.GUI
CONTROL.GUI
CORAAP1.GUI
CORACA1.GUI
CORACA2.GUI
CORACK1.GUI
CORACK2.GUI
CORACV1.GUI
CORACV2.GUI
CORALAB1.GUI
CORAP1.GUI
CORASY1.GUI
CORAVP1.GUI
CORAVP2.GUI
CORBUILD.GUI
CORBUTT.FNT
CORCA1.GUI
CORCA2.GUI
CORCA3.GUI
CORCK1.GUI
CORCK2.GUI
CORCK3.GUI
CORCOM1.GUI
CORCOM2.GUI
CORCS1.GUI
CORCV1.GUI
CORCV2.GUI
CORCV3.GUI
CORFMD1.GUI
CORGEN.GUI
CORLAB1.GUI
CORMAIN.GUI
CORMAIN2.GUI
CORSILO1.GUI
CORSY1.GUI
CORVP1.GUI
CREDITS.GUI
ENDGAME.GUI
ENDMSN.GUI
ENDMULTI.GUI
ENERGY.GUI
EXITMENU.GUI
GAMMA.GUI
HELP.GUI
LOADGAME.GUI
LOADLIST.GUI
LOGOSEL.GUI
LOUNGE.GUI
LOUNGE2.GUI
MAINMENU.GUI
METAL.GUI
MISSION.GUI
MISSIONX.GUI
MODEM.GUI
MOREINFO.GUI
MSGBOX.GUI
MSNBRIEF.GUI
MUSIC.GUI
MUSICRT.GUI
NEWCAMP.GUI

NEWGAME.GUI
NEWMULTI.GUI
OPTION.GUI
PREFS.GUI
RESTART.GUI
restrict2.GUI
ROMAN10.FNT
ROMAN12.FNT
SAVEGAME.GUI
SAVELIST.GUI
SCORE.GUI
SELCAMP.GUI
SELCAMPX.GUI
SELGAME.GUI
SELMAP.GUI
SELPROV.GUI
SELPROVX.GUI
SELSIDE.GUI
SELVMODE.GUI
SERIAL.GUI
SHARE.GUI
SIDESEL.GUI
SINGLE.GUI
SKIRMISH.GUI
SOUND.GUI
SOUNDS.GUI
SOUNDSRT.GUI
SPEEDS.GUI
SPEEDSRT.GUI
SSIDESEL.GUI
ssidesell.gui
STARTOPT.GUI
TABMENU.GUI
TALK.GUI
TALK2.GUI
TCP.GUI
TIMEOUT.GUI
UNITINFO.GUI
Unitinfox.GUI
VIEWMAP.GUI
VISUALRT.GUI
VISUALS.GUI
WARP.GUI
YESORNO.GUI

objects3d

1x1A.3do
1x1B.3do
1x1C.3do
1x1D.3do
1x1E.3do
1x1F.3do
2x2A.3do
2x2B.3do
2x2C.3do
2x2D.3do
2x2E.3do
2x2F.3do
3x3A.3do
3x3B.3do
3x3C.3do
3x3D.3do
3x3E.3do
3x3F.3do
4x4A.3do
4x4B.3do
4x4C.3do
4x4D.3do
4x4E.3do
4x4F.3do
5x5A.3do
5x5B.3do
5x5C.3do

5x5D.3do
6x6A.3do
6x6B.3do
6x6C.3do
6x6D.3do
7x7A.3do
7x7B.3do
7x7C.3do
7x7D.3do
amdrocket.3do
ARMaap.3do
armaap_dead.3do
ARMaca.3do
armaca_dead.3do
armack.3do
armack_dead.3do
armacv.3do
armacv_dead.3do
armalab.3do
armalab_dead.3do
ARMamd.3do
armamd_dead.3do
armanni.3do
armanni_dead.3do
armap.3do
armap_dead.3do
armarad.3do
armarad_dead.3do
armaser.3do
armaser_dead.3do
armasp.3do
armasp_dead.3do
armasy.3do
armasy_dead.3do
armatlas.3do
armatlas_dead.3do
armavp.3do
armavp_dead.3do
armbats.3do
armbats_dead.3do
armbrawl.3do
armbrawl_dead.3do
armbrtha.3do
armbrtha_dead.3do
armbull.3do
armbull_dead.3do
armca.3do
armcarry.3do
armcarry_dead.3do
armca_dead.3do
armck.3do
armck_dead.3do
armcom.3do
armcroc.3do
armcroc_dead.3do
armcrus.3do
armcrus_dead.3do
armcs.3do
armcs_dead.3do
armcv.3do
armcv_dead.3do
armdrag.3do
armestor.3do
armestor_dead.3do
armfast.3do
armfast_dead.3do
armfav.3do
armfav_dead.3do
armfido.3do
armfido_dead.3do
armfig.3do
armfig_dead.3do
armflash.3do

armflash_dead.3do
armfus.3do
armfus_dead.3do
armgate.3do
armgate_dead.3do
ARMgeo.3do
armgeo_dead.3do
armguard.3do
armguard_dead.3do
armham.3do
armham_dead.3do
armhawk.3do
armhawk_dead.3do
armhlt.3do
armhlt_dead.3do
armjam.3do
armjam_dead.3do
armjeth.3do
armjeth_dead.3do
armlab.3do
armlab_dead.3do
armlance.3do
armlance_dead.3do
armlt.3do
armlt_dead.3do
armmakr.3do
armmakr_dead.3do
armmart.3do
armmart_dead.3do
armmerl.3do
armmerl_dead.3do
armmex.3do
armmex_dead.3do
armmoho.3do
armmoho_dead.3do
ARMMship.3do
armmship_dead.3do
armmstor.3do
armmstor_dead.3do
armpeep.3do
armpeep_dead.3do
armpnix.3do
armpnix_dead.3do
armpt.3do
armpt_dead.3do
armpw.3do
armpw_dead.3do
armrad.3do
armrad_dead.3do
armrl.3do
armrl_dead.3do
armrock.3do
armrock_dead.3do
ARMroy.3do
armroy_dead.3do
armsam.3do
armsam_dead.3do
armseer.3do
armseer_dead.3do
armsilo.3do
armsilo_dead.3do
armsolar.3do
armsolar_dead.3do
armsonar.3do
armsonar_dead.3do
armspid.3do
armspid_dead.3do
armstump.3do
armstump_dead.3do
armsub.3do
armsubk.3do
armsubk_dead.3do
armsub_dead.3do

armsy.3do
armsy_dead.3do
armthund.3do
armthund_dead.3do
armtree.3do
armtree_dead.3do
armt1.3do
armt1_dead.3do
armtship.3do
armtship_dead.3do
armvader.3do
armvader_dead.3do
armvp.3do
armvp_dead.3do
armwin.3do
armwin_dead.3do
armzeus.3do
armzeus_dead.3do
Asteroid1.3do
Asteroid2.3do
Asteroid3.3do
Asteroid4.3do
Asteroid5.3do
Asteroid6.3do
ballmiss.3do
bomb.3do
bomb1.3do
bomb2.3do
bomb3.3do
coraap.3do
coraap_dead.3do
coraca.3do
coraca_dead.3do
corack.3do
corack_dead.3do
coracv.3do
coracv_dead.3do
corak.3do
corak_dead.3do
coralab.3do
coralab_dead.3do
corap.3do
corape.3do
corape_dead.3do
corap_dead.3do
corarad.3do
corarad_dead.3do
corasp.3do
corasp_dead.3do
corasy.3do
corasy_dead.3do
coravp.3do
coravp_dead.3do
corbats.3do
corbats_dead.3do
corbuild.3do
corbuild_dead.3do
corca.3do
corcan.3do
corcan_dead.3do
corcarry.3do
corcarry_dead.3do
corca_dead.3do
corck.3do
corck_dead.3do
corcom.3do
corcrash.3do
corcrash_dead.3do
corcрус.3do
corcрус_dead.3do
corcs.3do
corcs_dead.3do
corcv.3do

corcv_dead.3do
cordoom.3do
cordoom_dead.3do
cordrag.3do
cordrag_dead.3do
corestor.3do
corestor_dead.3do
coreter.3do
coreter_dead.3do
corfav.3do
corfav_dead.3do
corfink.3do
corfink_dead.3do
corfmd.3do
corfmd_dead.3do
corfus.3do
corfus_dead.3do
corgate.3do
corgate_dead.3do
corgator.3do
corgator_dead.3do
corgeo.3do
corgeo_dead.3do
corgol.3do
corgol_dead.3do
corhlt.3do
corhlt_dead.3do
corhurc.3do
corhurc_dead.3do
corint.3do
corint_dead.3do
corlab.3do
corlab_dead.3do
corllt.3do
corllt_dead.3do
cormakr.3do
cormakr_dead.3do
cormart.3do
cormart_dead.3do
cormex.3do
cormex_dead.3do
cormist.3do
cormist_dead.3do
cormoho.3do
cormoho_dead.3do
cormship.3do
cormship_dead.3do
cormstor.3do
cormstor_dead.3do
corpt.3do
corpt_dead.3do
corpun.3do
corpun_dead.3do
copyro.3do
copyro_dead.3do
corrad.3do
corrad_dead.3do
corraid.3do
corraid_dead.3do
correap.3do
correap_dead.3do
corrl.3do
corrl_dead.3do
corroach.3do
corroach_dead.3do
corroy.3do
corroy_dead.3do
corseal.3do
corseal_dead.3do
corshad.3do
corshad_dead.3do
corshark.3do
corshark_dead.3do

corshiprckt1.3do
corsilo.3do
corsilo_dead.3do
corsolar.3do
corsolar_dead.3do
corsonar.3do
corsonar_dead.3do
corspec.3do
corspec_dead.3do
corstorm.3do
corstorm_dead.3do
corsub.3do
corsub_dead.3do
corsy.3do
corsy_dead.3do
corthud.3do
corthud_dead.3do
cortide.3do
cortide_dead.3do
cortitan.3do
cortitan_dead.3do
cortl.3do
cortl_dead.3do
cortruck.3do
cortruck_dead.3do
cortship.3do
cortship_dead.3do
corvalk.3do
corvalk_dead.3do
corvamp.3do
corvamp_dead.3do
corveng.3do
corveng_dead.3do
corvp.3do
corvp_dead.3do
corvrad.3do
corvrad_dead.3do
corvroc.3do
corvrocket.3do
corvroc_dead.3do
corwin.3do
corwin_dead.3do
crblmssl.3do
deadtank.3do
DepthCharge.3do
dgun.3do
fmdmisl.3do
jeffttest.3do
missile.3do
missile1.3do
missile2.3do
missile3.3do
nrgyshla.3do
nrgyshlb.3do
plasma.3do
rocket.3do
torpedo.3do

—palettes

GUIPAL.PAL
GUIPAL.PCX
PALETTE.ALP
PALETTE.LHT
PALETTE.PAL
PALETTE.SHD

—scripts

ARMAAP.BOS
ARMAAP.COB
ARMACA.BOS
ARMACA.COB
ARMACK.BOS
ARMACK.COB

ARMACV . BOS
ARMACV . COB
ARMALAB . BOS
ARMALAB . COB
ARMAMD . BOS
ARMAMD . COB
ARMANNI . BOS
ARMANNI . COB
ARMAP . BOS
ARMAP . COB
ARMARAD . BOS
ARMARAD . COB
ARMASER . BOS
ARMASER . COB
ARMASP . BOS
ARMASP . COB
ARMASY . BOS
ARMASY . COB
ARMATLAS . BOS
ARMATLAS . COB
ARMAVP . BOS
ARMAVP . COB
ARMBATS . BOS
ARMBATS . COB
ARMBRAWL . BOS
ARMBRAWL . COB
ARMBRTHA . BOS
ARMBRTHA . COB
ARMBULL . BOS
ARMBULL . COB
ARMCA . BOS
ARMCA . COB
ARMCARRY . BOS
ARMCARRY . COB
ARMCK . BOS
ARMCK . COB
ARMCOM . BOS
ARMCOM . COB
ARMCROC . BOS
ARMCROC . COB
ARMCRUS . BOS
ARMCRUS . COB
ARMCS . BOS
ARMCS . COB
ARMCV . BOS
ARMCV . COB
ARMDRAG . BOS
ARMDRAG . COB
ARMESTOR . BOS
ARMESTOR . COB
ARMFAS . BOS
ARMFAS . COB
ARMFV . BOS
ARMFV . COB
ARMFIDO . BOS
ARMFIDO . COB
ARMFID . BOS
ARMFID . COB
ARMFUS . BOS
ARMFUS . COB
ARMFUS . BOS
ARMFUS . COB
ARMGATE . BOS
ARMGATE . COB
ARMGEO . BOS
ARMGEO . COB
ARMGUARD . BOS
ARMGUARD . COB
ARMHAM . BOS
ARMHAM . COB
ARMHAWK . BOS
ARMHAWK . COB
ARMHLT . BOS

ARMHLT . COB
ARMJAM . BOS
ARMJAM . COB
ARMJETH . BOS
ARMJETH . COB
ARMLAB . BOS
ARMLAB . COB
ARMLANCE . BOS
ARMLANCE . COB
ARMLLT . BOS
ARMLLT . COB
ARMMAKR . BOS
ARMMAKR . COB
ARMMART . BOS
ARMMART . COB
ARMMERL . BOS
ARMMERL . COB
ARMDEX . BOS
ARMDEX . COB
ARMMOHO . BOS
ARMMOHO . COB
ARMMSHIP . BOS
ARMMSHIP . COB
ARMSTOR . BOS
ARMSTOR . COB
ARMPEEP . BOS
ARMPEEP . COB
ARMPNIX . BOS
ARMPNIX . COB
ARMPT . BOS
ARMPT . COB
ARMPW . BOS
ARMPW . COB
ARMRAD . BOS
ARMRAD . COB
ARMRL . BOS
ARMRL . COB
ARMROCK . BOS
ARMROCK . COB
ARMROY . BOS
ARMROY . COB
ARMSAM . BOS
ARMSAM . COB
ARMSEER . BOS
ARMSEER . COB
ARMSILO . BOS
ARMSILO . COB
ARMSOLAR . BOS
ARMSOLAR . COB
ARMSONAR . BOS
ARMSONAR . COB
ARMSPID . BOS
ARMSPID . COB
ARMSTUMP . BOS
ARMSTUMP . COB
ARMSUB . BOS
ARMSUB . COB
ARMSUBK . BOS
ARMSUBK . COB
ARMSY . BOS
ARMSY . COB
ARMTHUND . BOS
ARMTHUND . COB
ARMTIDE . BOS
ARMTIDE . COB
ARMTL . BOS
ARMTL . COB
ARMTSHIP . BOS
ARMTSHIP . COB
ARMVADER . BOS
ARMVADER . COB
ARMVP . BOS
ARMVP . COB

ARMWIN.BOS
ARMWIN.COB
ARMZEUS.BOS
ARMZEUS.COB
COMPILE.BAT
CORAAP.BOS
CORAAP.COB
CORACA.BOS
CORACA.COB
CORACK.BOS
CORACK.COB
CORACV.BOS
CORACV.COB
CORAK.BOS
CORAK.COB
CORALAB.BOS
CORALAB.COB
CORAP.BOS
CORAP.COB
CORAPE.BOS
CORAPE.COB
CORARAD.BOS
CORARAD.COB
CORASP.BOS
CORASP.COB
CORASY.BOS
CORASY.COB
CORAVP.BOS
CORAVP.COB
CORBATS.BOS
CORBATS.COB
CORBUILD.BOS
CORBUILD.COB
CORCA.BOS
CORCA.COB
CORCAN.BOS
CORCAN.COB
CORCARRY.BOS
CORCARRY.COB
CORCK.BOS
CORCK.COB
CORCOM.BOS
CORCOM.COB
CORCRASH.BOS
CORCRASH.COB
CORCRUS.BOS
CORCRUS.COB
CORCS.BOS
CORCS.COB
CORCV.BOS
CORCV.COB
CORDOOM.BOS
CORDOOM.COB
CORDRAG.BOS
CORDRAG.COB
CORESTOR.BOS
CORESTOR.COB
CORETER.BOS
CORETER.COB
CORFAV.BOS
CORFAV.COB
CORFINK.BOS
CORFINK.COB
CORFMD.BOS
CORFMD.COB
CORFUS.BOS
CORFUS.COB
CORGATE.BOS
CORGATE.COB
CORGATOR.BOS
CORGATOR.COB
CORGEO.BOS
CORGEO.COB

CORGOL.BOS
CORGOL.COB
CORHLT.BOS
CORHLT.COB
CORHURC.BOS
CORHURC.COB
CORINT.BOS
CORINT.COB
CORLAB.BOS
CORLAB.COB
CORLLT.BOS
CORLLT.COB
CORMAKR.BOS
CORMAKR.COB
CORMART.BOS
CORMART.COB
CORMEX.BOS
CORMEX.COB
CORMIST.BOS
CORMIST.COB
CORMOHO.BOS
CORMOHO.COB
CORMSHIP.BOS
CORMSHIP.COB
CORMSTOR.BOS
CORMSTOR.COB
CORPT.BOS
CORPT.COB
CORPUN.BOS
CORPUN.COB
CORPYRO.BOS
CORPYRO.COB
CORRAD.BOS
CORRAD.COB
CORRAID.BOS
CORRAID.COB
CORREAP.BOS
CORREAP.COB
CORRL.BOS
CORRL.COB
CORROACH.BOS
CORROACH.COB
CORROY.BOS
CORROY.COB
CORSEAL.BOS
CORSEAL.COB
CORSHAD.BOS
CORSHAD.COB
CORSHARK.BOS
CORSHARK.COB
CORSILO.BOS
CORSILO.COB
CORSOLAR.BOS
CORSOLAR.COB
CORSONAR.BOS
CORSONAR.COB
CORSPEC.BOS
CORSPEC.COB
CORSTORM.BOS
CORSTORM.COB
CORSUB.BOS
CORSUB.COB
CORSY.BOS
CORSY.COB
CORTHUD.BOS
CORTHUD.COB
CORTIDE.BOS
CORTIDE.COB
CORTITAN.BOS
CORTITAN.COB
CORTL.BOS
CORTL.COB
CORTRUCK.BOS

CORTRUCK.COB
CORTSHIP.BOS
CORTSHIP.COB
CORVALK.BOS
CORVALK.COB
CORVAMP.BOS
CORVAMP.COB
CORVENG.BOS
CORVENG.COB
CORVP.BOS
CORVP.COB
CORVRAD.BOS
CORVRAD.COB
CORVROC.BOS
CORVROC.COB
CORWIN.BOS
CORWIN.COB
EXPTYPE.H
HELP.H
HITWEAP.H
ROCKUNIT.H
SFXTYPE.H
SMOKEUNIT.H
STATECHG.H
STDSCRPT.H
STDANK.H
YARD.H

—sounds

ANNI.WAV
ANNIGUN1.WAV
ARMSML1.WAV
ARMSML2.WAV
ARMSML3.WAV
ARMSML4.WAV
BEEP1.WAV
BEEP2.WAV
BEEP3.WAV
BEEP4.WAV
BEEP5.WAV
BEEP6.WAV
BERTHA1.WAV
BERTHA2.WAV
BERTHA3.WAV
BERTHA4.WAV
BERTHA5.WAV
BERTHA6.WAV
BOMBREL.WAV
BUILD1.WAV
BUILD2.WAV
BURN01.WAV
BURN02.WAV
BURN03.WAV
BURN1.WAV
BURNLRG.WAV
BURNMED.WAV
BURNSML.WAV
BUTMAIN.WAV
BUTMAIN1.WAV
BUTMAIN2.WAV
BUTMAIN3.WAV
BUTMAIN4.WAV
BUTNAGR1.WAV
BUTNAGR2.WAV
BUTNMBL1.WAV
BUTNMBL2.WAV
BUTNOPTN.WAV
BUTNSID1.WAV
BUTNSID2.WAV
BUTNSID3.WAV
BUTNSKIR.WAV
BUTOPTN.WAV
BUTSCROL.WAV

BUTSCRO2.WAV
BUTTN01.WAV
BUTTN02.WAV
BUTTN06.WAV
BUTTON1.WAV
BUTTON10.WAV
BUTTON11.WAV
BUTTON12.WAV
BUTTON13.WAV
BUTTON2.WAV
BUTTON3.WAV
BUTTON4.WAV
BUTTON5.WAV
BUTTON6.WAV
BUTTON7.WAV
BUTTON8.WAV
BUTTON9.WAV
CANCEL1.WAV
CANCEL2.WAV
CANLITE1.WAV
CANLITE2.WAV
CANLITE3.WAV
CANNHVY1.WAV
CANNHVY2.WAV
CANNHVY3.WAV
CANNHVY4.WAV
CANNHVY5.WAV
CANNON1.WAV
CANNON2.WAV
CANNON3.WAV
CANTDO.WAV
CANTDO1.WAV
CANTDO2.WAV
CANTDO3.WAV
CANTDO4.WAV
CANZIP1.WAV
CANZIP2.WAV
CAPTURE1.WAV
CAPTURE2.WAV
CDOGGY.WAV
COMFIRE1.WAV
COMFIRE2.WAV
COUNT1.WAV
COUNT2.WAV
COUNT3.WAV
COUNT4.WAV
COUNT5.WAV
COUNT6.WAV
DEBRIS1.WAV
DEBRIS2.WAV
DEBRIS3.WAV
DEBRIS4.WAV
DEBRIS5.WAV
DEBRIS6.WAV
DEPTH1.WAV
DEPTH2.WAV
DEPTH3.WAV
DISGUN1.WAV
DOOM.WAV
DRONE1.WAV
DRONE2.WAV
DRONE3.WAV
EMGPULS1.WAV
EMGPULSE.WAV
ENERGYN1.WAV
ENERGYN2.WAV
EXPLODE.WAV
EXPLODE2.WAV
FLAMHVY1.WAV
FLAMHVY2.WAV
FLAMHVY3.WAV
FLAMLIT1.WAV
FLAMLIT2.WAV

FLAMLIT3.WAV
FUSION1.WAV
FUSION2.WAV
GEOTHRM1.WAV
GEOTHRM2.WAV
GUN110.WAV
GUNHUGE.WAV
GUNRAPID.WAV
HONK.WAV
KBARMMOV.WAV
KBARMSE1.WAV
KBARMSEL.WAV
KBARMSTP.WAV
KBCORMOV.WAV
KBCORSEL.WAV
KBCORSTP.WAV
KCARMACT.WAV
KCARMMOV.WAV
KCARMSEL.WAV
KCARMSTP.WAV
KCCORSEL.WAV
KCCORSTP.WAV
KCORMOV.WAV
KLOAK1.WAV
KLOAK1UN.WAV
KLOAK2.WAV
KLOAK2UN.WAV
LAMEBURN.WAV
LARGEGUN.WAV
LASER.WAV
LASHIT.WAV
LASRCAN1.WAV
LASRCAN2.WAV
LASRFAST.WAV
LASRFIR1.WAV
LASRFIR2.WAV
LASRFIR3.WAV
LASRHIT1.WAV
LASRHIT2.WAV
LASRHIT3.WAV
LASRHVY1.WAV
LASRHVY2.WAV
LASRHVY3.WAV
LASRLIT1.WAV
LASRLIT2.WAV
LASRLIT3.WAV
LASRMAS1.WAV
LASRMAS2.WAV
LASRMAS3.WAV
LASRSLOW.WAV
LGHTHVY1.WAV
LOADAIR.WAV
LOADWTR1.WAV
LOADWTR2.WAV
MECHANI.WAV
METALNO1.WAV
METALNO2.WAV
METLOFF1.WAV
METLOFF2.WAV
METLON1.WAV
METLON2.WAV
METLRUN1.WAV
METLRUN2.WAV
MEXOFF1.WAV
MEXOFF2.WAV
MEXON1.WAV
MEXON2.WAV
MEXRUN1.WAV
MEXRUN2.WAV
MISICBM1.WAV
MISICBM2.WAV
MISICBM3.WAV
MISLHVY1.WAV

MISLHVY2.WAV
MISLHVY3.WAV
MISLITE1.WAV
MISLITE2.WAV
MISLITE3.WAV
MISSTAR1.WAV
MISSTAR2.WAV
MOHOFF1.WAV
MOHOFF2.WAV
MOHOON1.WAV
MOHOON2.WAV
MOHORUN1.WAV
MOHORUN2.WAV
NANLATH1.WAV
NANLATH2.WAV
PAIRACTV.WAV
PAIRWORK.WAV
PHASER.WAV
PLABACTV.WAV
PLABWORK.WAV
PLASHVY1.WAV
PLASMED1.WAV
PLASSML1.WAV
PSHPACTV.WAV
PSHPWORK.WAV
PVEHACTV.WAV
PVEHWORK.WAV
RADADDE1.WAV
RADADDE2.WAV
RADADVN1.WAV
RADADVN2.WAV
RADAR1.WAV
RADAR2.WAV
RADARDE1.WAV
RADARDE2.WAV
RADJAM1.WAV
RADJAM2.WAV
RANGE1.WAV
RANGE2.WAV
RECLAIM1.WAV
RECLAIM2.WAV
REPAIR1.WAV
REPAIR2.WAV
ROCKET.WAV
ROCKHVY1.WAV
ROCKHVY2.WAV
ROCKHVY3.WAV
ROCKLIT1.WAV
ROCKLIT2.WAV
ROCKLIT3.WAV
ROCKXPL1.WAV
ROCKXPL2.WAV
ROCKXPL3.WAV
SERVLARG.WAV
SERVLRG1.WAV
SERVLRG2.WAV
SERVLRG3.WAV
SERVLRG4.WAV
SERVMD01.WAV
SERVMD02.WAV
SERVMD03.WAV
SERVMED1.WAV
SERVMED2.WAV
SERVMED3.WAV
SERVMED4.WAV
SERVROC1.WAV
SERVSML1.WAV
SERVSML2.WAV
SERVSML3.WAV
SERVSML4.WAV
SERVSML5.WAV
SERVSML6.WAV
SERVTNY1.WAV

SERVTTY2.WAV
SHARMMOV.WAV
SHARMSEL.WAV
SHARMSTP.WAV
SHCORMOV.WAV
SHCORSEL.WAV
SHCORSTP.WAV
SING.WAV
SOLAR1.WAV
SOLAR2.WAV
SONAR1.WAV
SONAR2.WAV
SONARDE1.WAV
SONARDE2.WAV
SONARJAM.WAV
SPIDER.WAV
SPLSLRG.WAV
SPLSMED.WAV
SPLSSML.WAV
STORMTL1.WAV
STORMTL2.WAV
STORNGY1.WAV
STORNGY2.WAV
SUARMMOV.WAV
SUARMSEL.WAV
SUARMSTP.WAV
SUCORMOV.WAV
SUCORSEL.WAV
SUCORSTP.WAV
TARMMOVE.WAV
TARMSEL.WAV
TARMSTOP.WAV
TCORMOVE.WAV
TCORSEL.WAV
TCORSTOP.WAV
TIDEGEN1.WAV
TIDEGEN2.WAV
TORPEDO1.WAV
TORPEDO2.WAV
TREETED.WAV
TURLGSO1.WAV
TURLGSO2.WAV
TURLONG1.WAV
TURMED1.WAV
TURMED2.WAV
TURMED3.WAV
TWRACTV1.WAV
TWRACTV2.WAV
TWRACTV3.WAV
TWRACTV4.WAV
TWRTUR63.WAV
TWRTURN1.WAV
TWRTURN2.WAV
TWRTURN3.WAV
TWRTURN4.WAV
UNITDONE.WAV
VARMMOVE.WAV
VARMSEL.WAV
VARMSTOP.WAV
VCORMOVE.WAV
VCORSEL.WAV
VCORSTOP.WAV
VICTORY2.WAV
VICTORY4.WAV
VTOLARAC.WAV
VTOLARLD.WAV
VTOLARMV.WAV
VTOLCRAC.WAV
VTOLCRLD.WAV
VTOLCRMV.WAV
WARNING1.WAV
WARNING2.WAV
WINDGEN1.WAV

WINDGEN2.WAV
XPLOBLD1.WAV
XPLOBLD2.WAV
XPLOBLD3.WAV
XPLOCOM1.WAV
XPLODEP1.WAV
XPLODEP2.WAV
XPLODEP3.WAV
XPLOLRG1.WAV
XPLOLRG2.WAV
XPLOLRG3.WAV
XPLOLRG4.WAV
XPLOMAS1.WAV
XPLOMAS2.WAV
XPLOMAS3.WAV
XPLOMAS4.WAV
XPLOMED1.WAV
XPLOMED2.WAV
XPLOMED3.WAV
XPLOMED4.WAV
XPLONUK1.WAV
XPLONUK2.WAV
XPLONUK3.WAV
XPLONUK4.WAV
XPLOSML1.WAV
XPLOSML2.WAV
XPLOSML3.WAV
XPLOSML4.WAV

—textures

ARMBLDG.GAF
ARMCAMO.GAF
ARMSHIPS.GAF

ARMVEHIC.GAF
CORBLDG.GAF
CORCAMO.GAF
CORSHIPS.GAF
CORVEHIC.GAF
LOGOS.GAF
WRECKAGE.GAF

—unitpics

ARMAAP.PCX
ARMAC.PCX
ARMACA.PCX
ARMACK.PCX
ARMACV.PCX
ARMALAB.PCX
ARMAMD.PCX
ARMANNI.PCX
ARMAP.PCX
ARMARAD.PCX
ARMASER.PCX
ARMASP.PCX
ARMASY.PCX
ARMATLAS.PCX
ARMAVP.PCX
ARMBATS.PCX
ARMBRAWL.PCX
ARMBRTHA.PCX
ARMBULL.PCX
ARMCA.PCX
ARMCARRY.PCX
ARMCK.PCX
ARMCOM.PCX
ARMCROC.PCX
ARMCRUS.PCX
ARMCS.PCX
ARMCV.PCX
ARMDRAG.PCX
ARMESTOR.PCX

ARMPAST.PCX
ARMPAV.PCX
ARMPIDO.PCX
ARMPFIG.PCX
ARMPFLASH.PCX
ARMPFUS.PCX
ARMPGATE.PCX
ARMPGEO.PCX
ARMPGUARD.PCX
ARMPHAM.PCX
ARMPHAWK.PCX
ARMPHLT.PCX
ARMPJAM.PCX
ARMPJETH.PCX
ARMPLAB.PCX
ARMPLANCE.PCX
ARMPLLT.PCX
ARMPMAKR.PCX
ARMPMART.PCX
ARMPMMERL.PCX
ARMPMEX.PCX
ARMPMOHO.PCX
ARMPMSHIP.PCX
ARMPMSTOR.PCX
ARMPNIX.PCX
ARMPPEEP.PCX
ARMPPNIX.PCX
ARMPPT.PCX
ArmpPW.PCX
ARMPRAD.PCX
ARMPRL.PCX
ARMPROCK.PCX
ARMPROY.PCX
ARMPSAM.PCX
ARMPSEER.PCX
ARMPSILO.PCX
ARMPSOLAR.PCX
ARMPSONAR.PCX
ARMPSPID.PCX
ARMPSTUMP.PCX
ARMPSUB.PCX
ARMPSUBK.PCX
ARMPSY.PCX
ARMPTHUND.PCX
ARMPMTID.PCX
ARMPMTIDE.PCX
ARMPMTL.PCX
ARMPMTSHIP.PCX
ARMPVADER.PCX
ARMPVVP.PCX
ARMPVRAD.PCX
ARMPWIN.PCX
ARMPZEUS.PCX
CORAAP.PCX
CORAC.PCX
CORACA.PCX
CORACK.PCX
CORACV.PCX
CORAK.PCX
CORALAB.PCX
CORAP.PCX
CORAPE.PCX
CORARAD.PCX
CORASP.PCX
CORASY.PCX
CORAVP.PCX
CORBATS.PCX
CORCA.PCX
CORCAN.PCX
CORCARRY.PCX
CORCK.PCX
CORCOM.PCX
CORCRASH.PCX

CORCRUS.PCX
CORCS.PCX
CORCV.PCX
CORDOOM.PCX
CORDRAG.PCX
CORESTOR.PCX
CORETER.PCX
CORFAV.PCX
CORFING.PCX
CORFINK.PCX
CORFMD.PCX
CORFUS.PCX
CORGATE.PCX
CORGATOR.PCX
CORGEO.PCX
CORGOL.PCX
CORHLT.PCX
CORHURC.PCX
CORINT.PCX
CORLAB.PCX
CORLLT.PCX
CORMAKR.PCX
CORMART.PCX
CORMEX.PCX
CORMIST.PCX
CORMOHO.PCX
CORMSHIP.PCX
CORMSTOR.PCX
CORPT.PCX
CORPUN.PCX
CORPYRO.PCX
CORRAD.PCX
CORRAID.PCX
CORREAP.PCX
CORRL.PCX
CORROACH.PCX
CORROY.PCX
CORSEAL.PCX
CORSHAD.PCX
CORSHARK.PCX
CORSILO.PCX
CORSOLAR.PCX
CORSONAR.PCX
CORSPEC.PCX
CORSTORM.PCX
CORSUB.PCX
CORSY.PCX
CORTHUD.PCX
CORTIDE.PCX
CORTITAN.PCX
CORTL.PCX
CORTSHIP.PCX
CORVALK.PCX
CORVAMP.PCX
CORVENG.PCX
CORVP.PCX
CORVRAD.PCX
CORVROC.PCX
CORWIN.PCX

units

ALLUNITS.XLS
ARMAAP.FBI
ARMACA.FBI
ARMACK.FBI
ARMACV.FBI
ARMALAB.FBI
ARMAMD.FBI
ARMANNI.FBI
ARMAP.FBI
ARMARAD.FBI
ARMASER.FBI
ARMASP.FBI

ARMASY .FBI
ARMATLAS .FBI
ARMAVP .FBI
ARMBATS .FBI
ARMBRAWL .FBI
ARMBRTHA .FBI
ARMBULL .FBI
ARMCA .FBI
ARMCARRY .FBI
ARMCK .FBI
ARMCOM .FBI
ARMCROC .FBI
ARMCRUS .FBI
ARMCS .FBI
ARMCV .FBI
ARMDRAG .FBI
ARMESTOR .FBI
ARMPFAST .FBI
ARMPFAV .FBI
ARMPIDO .FBI
ARMPFIG .FBI
ARMPFLASH .FBI
ARMPFUS .FBI
ARMPGATE .FBI
ARMPGEO .FBI
ARMPGUARD .FBI
ARMPHAM .FBI
ARMPHAWK .FBI
ARMPHLT .FBI
ARMPJAM .FBI
ARMPJETH .FBI
ARMPLAB .FBI
ARMP Lance .FBI
ARMP LLLT .FBI
ARMPMAKR .FBI
ARMPMART .FBI
ARMPMERL .FBI
ARMPMEX .FBI
ARMPMOHO .FBI
ARMPMSHIP .FBI
ARMPMSTOR .FBI
ARMPPEEP .FBI
ARMPPNIX .FBI
ARMPPT .FBI
ARMPW .FBI
ARMPRAD .FBI
ARMPRL .FBI
ARMPROCK .FBI
ARMPROY .FBI
ARMPSAM .FBI
ARMPSEER .FBI
ARMPSILO .FBI
ARMP SOLAR .FBI
ARMPSONAR .FBI
ARMPSPID .FBI
ARMPSTUMP .FBI
ARMPSUB .FBI
ARMPSUBK .FBI
ARMPSY .FBI
ARMPTHUND .FBI
ARMP TIDE .FBI
ARMP TLL .FBI
ARMP TSHIP .FBI
ARMPVADER .FBI
ARMPVP .FBI
ARMPWIN .FBI
ARMPZEUS .FBI
BUILDALL .BAT
CORAAP .FBI
CORACA .FBI
CORACK .FBI
CORACV .FBI
CORAK .FBI

CORALAB.FBI
CORAP.FBI
CORAPE.FBI
CORARAD.FBI
CORASP.FBI
CORASY.FBI
CORAVP.FBI
CORBATS.FBI
CORBUILD.FBI
CORCA.FBI
CORCAN.FBI
CORCARRY.FBI
CORCK.FBI
CORCOM.FBI
CORCRASH.FBI
CORCRUS.FBI
CORCS.FBI
CORCV.FBI
CORDOOM.FBI
CORDRAG.FBI
CORESTOR.FBI
CORETER.FBI
CORFAV.FBI
CORFINK.FBI
CORFMD.FBI
CORFUS.FBI
CORGATE.FBI
CORGATOR.FBI
CORGEO.FBI
CORGOL.FBI
CORHLT.FBI
CORHURC.FBI
CORINT.FBI
CORLAB.FBI
CORLLT.FBI
CORMAKR.FBI
CORMART.FBI
CORMEX.FBI
CORMIST.FBI
CORMOHO.FBI
CORMSHIP.FBI
CORMSTOR.FBI
CORPT.FBI
CORPUN.FBI
CORPYRO.FBI
CORRAD.FBI
CORRAID.FBI
CORREAP.FBI
CORRL.FBI
CORROACH.FBI
CORROY.FBI
CORSEAL.FBI
CORSHAD.FBI
CORSHARK.FBI
CORSILO.FBI
CORSOLAR.FBI
CORSONAR.FBI
CORSPEC.FBI
CORSTORM.FBI
CORSUB.FBI
CORSY.FBI
CORTHUD.FBI
CORTIDE.FBI
CORTITAN.FBI
CORTL.FBI
CORTRUCK.FBI
CORTSHIP.FBI
CORVALK.FBI
CORVAMP.FBI
CORVENG.FBI
CORVP.FBI
CORVRAD.FBI
CORVROC.FBI

CORWIN.FBI
FBI2XLS.PL
MAKEXLS.BAT
More.txt
XLS2FBI.PL

weapons

CANNONS.TDF
FIRES.TDF
LASERS.TDF
METEORS.TDF
MISSILES.TDF
ROCKETS.TDF
UNITS.TDF
WEAPONS.TDF

Totala2.hpi Contents

TOTALA2.HPI

—bitmaps

ARMBKG.PCX
armguibottile.pcx
armguisidetile.pcx
armguitoatile.pcx
battleroom2.pcx
battlestart.pcx
BUTTONS2.PCX
CDLOG256.PCX
CONSOLE.PCX
COREBKG.PCX
corecamp0.PCX
corecamp1.pcx
corguibottile.pcx
corguisidetile.pcx
corguitoatile.pcx
createnew.pcx
DHELP.PCX
DLoadgame2.pcx
DLoadList.pcx
DRESTART.PCX
DSavegame2.pcx
DSaveList.pcx
DSelectmap2.pcx
DVIEWMAP.PCX
ENDCAMP.PCX
FAILURE.PCX
FRONTBG.PCX
Frontbgold.pcx
Frontend1F.PCX
FrontendX.pcx
GROMMETS.PCX
Hattfont10.PCX
Hattfont11.PCX
IGButtons.pcx
IGMBRIEF.PCX
IGOPT0X.PCX
IGOPT1X.PCX
Igoptintx.pcx
igoptionsTEMP.PCX
Igoptmusx.pcx
Igoptsoux.pcx
IgoptTEMP.pcx
Igoptvisx.pcx
IGPATCH.PCX
Installgame.pcx
InstallgameJ.pcx
Installglam.pcx
LOADBAR.PCX
Loadgame2bg.pcx
LOGOTEST.BMP
mbriefarm.pcx
mbriefcor.pcx
Mission02Win.PCX
Mission02Win2.PCX
Mission02WinBW.pcx
newcampaign4.pcx
newcampaign4x.pcx
newcamplogos.pcx
OptInterface4x.pcx
Options4x.pcx
Optmusic4x.pcx
OptSound4x.pcx
OptVisual4x.pcx
OUTCOME0.PCX
OUTCOME1.PCX

playanygame4.pcx
Playgame2.pcx
Playgame2J.pcx
pressedbone.PCX
PREVIEW.PIX
SAVEGAME.PCX
selconnect2.pcx
selectgame2x.pcx
SINGLEEBG.PCX
Skirmsetup4x.pcx
SMALLDOG.BMP
small_cavedog_logo.pcx
stagebuttons.pcx
STAR.PCX
TEMP.PCX
temptrans.pcx
TITLSCRN.PCX
UnitRestrict.pcx
UnitRestrict4x.pcx
UnitRestrict5x.pcx

maps

Anteer Strait.ota
Anteer Strait.tnt
Ashap Plateau.ota
Ashap Plateau.tnt
Caldera's Rim.ota
Caldera's Rim.tnt
Coast To Coast.ota
Coast To Coast.tnt
Dark Side.ota
Dark Side.tnt
Etorrep Glacier.ota
Etorrep Glacier.tnt
Evad River Confluence.ota
Evad River Confluence.tnt
Fox Holes.ota
Fox Holes.tnt
Full Moon.ota
Full Moon.tnt
Gods of War.ota
Gods of War.tnt
Great Divide.ota
Great Divide.tnt
Greenhaven.ota
Greenhaven.tnt
Hundred Isles.ota
Hundred Isles.tnt
Kill The Middle.ota
Kill The Middle.tnt
King of the Hill.ota
King of the Hill.tnt
Lava & Two Hills.ota
Lava & Two Hills.tnt
Lava Alley.ota
Lava Alley.tnt
Lava Highground.ota
Lava Highground.tnt
Lava Mania.ota
Lava Mania.tnt
Lava Run.ota
Lava Run.tnt
Metal Heck.ota
Metal Heck.tnt
MULTIPLAY.TDF
Over Crude Water.ota
Over Crude Water.tnt
Painted Desert.ota
Painted Desert.tnt
Pincushion.ota
Pincushion.tnt
Red Hot Lava.ota
Red Hot Lava.tnt


```

Red Planet.ota
Red Planet.tnt
Red Triangle.ota
Red Triangle.tnt
Ring Atoll.ota
Ring Atoll.tnt
Rock Alley.ota
Rock Alley.tnt
Seven Islands.ota
Seven Islands.tnt
SHERWOOD.OTA
SHERWOOD.TNT
Shore to Shore.ota
Shore to Shore.tnt
The Cold Place.ota
The Cold Place.tnt
The Desert Triad.ota
The Desert Triad.tnt
The Pass.ota
The Pass.tnt
Two Continents.ota
Two Continents.tnt
Yerrot Mountains.ota
Yerrot Mountains.tnt

```

HPI File Format Documentation

(from JoeD)

Warning: This is intended for use by people that already know what they're doing.

I'm a C programmer, so I'm doing things in C notation here, but I'll try to explain it so that those of you that don't speak C will be able to understand. If you don't understand, write me at joed@cws.org and I'll try to clear things up.

I'm also a big believer in examples, so I'll be walking you through an HPI file as I explain.

The first part of the file is a header. Except for the copyright statement at the end, this is the only unencrypted portion of the file. The header looks like this:

```

typedef struct _HPIHeader {
    long HPIMarker;          /* 'HAPI' */
    long SaveMarker;        /* 'BANK' if saved gamed */
    long DirectorySize;     /* The size of the directory */
    long HeaderKey;         /* Decrypt key */
    long Start;             /* File offset of directory */
} HPIHeader;

```

Here's a hex dump of a sample header:

```

00000000  48 41 50 49 00 00 01 00 24 02 00 00 7D 00 00 00   HAPI....
$....}...
00000010  14 00 00 00

```

Taken individually:

HPIMarker	This is just a marker. The value is always HAPI in ASCII. In hex, it's 0x49504148.	
-----------	--	--

SaveMarker	If it's a saved game, the value is BANK in ASCII, or 0x4B4E4142 in hex. Save game files are something of a special case, and I haven't done much to try to decode those. The value in normal HPI files is 0x00010000, but I have no idea if this means anything. I just check for BANK, and ignore it otherwise.	
DirectorySize	This is the size of the directory contained in the HPI file. Here, the value is 0x224, or 548 bytes. This includes the size of the header	
HeaderKey	The decryption key. Its value is 0x0000007D. More on this later.	
Start	The offset in the file where the directory starts. I have yet to see one that didn't start immediately after the header at offset 0x14, but you never know.	

Now we know enough to read the directory. But first, a small implementation note. Instead of allocating a buffer of DirectorySize bytes and then reading the directory into it, allocate a buffer of DirectorySize bytes, and read DirectorySize-Start bytes into the buffer at position Start. This is because the directory contains pointers, but the pointers are relative to the start of the file, not the start of the directory. By moving the directory down Start bytes into the buffer, we simplify the program. If we didn't do this, we'd have to subtract Start from every offset, and that would be a royal pain.

Now some of you are undoubtedly looking at an HPI file with a hex dump program, and saying "That sure doesn't look like a directory to me!" Well, you're right. That's because it's encrypted.

To decrypt it, first calculate the decryption key from the HeaderKey variable:

```
Key = NOT ((HeaderKey * 4) OR (HeaderKey >> 6))
```

Doing this on the 0x0000007D, you get FFFFFFFE0A (I think).

Here is the C code for the decryption routine. Since everything in the file is encrypted, I found it easier to combine the read and decryption functions into one.

```
int ReadAndDecrypt(int fpos, char *buff, int buffsize)
/*
Read "buffsize" bytes from the HPI file at position "fpos" into "buff",
and then decrypt it.
*/
{
    int count;
```

```

int tkey;
int result;

/* first, position the file */
fseek(HPIFile, fpos, SEEK_SET);

/* read the data into buff */
result = fread(buff, 1, buffsize, HPIFile);

/* for each character in buff... */
for (count = 0; count < buffsize; count++) {

    /* compute tkey = (fpos + count) XOR Key */
    tkey = (fpos + count) ^ Key;

    /* and then decode the character:
       buff[count] = tkey XOR (NOT buff[count]) */
    buff[count] = tkey ^ ~buff[count];
}

/* result is the number of bytes actually read in,
   and should be equal to buffsize */
return result;
}

```

Note that the position of the byte in the file (fpos+count) is used to decrypt.

And here is a decoded directory to make it easy to follow. Note that I loaded the actual directory starting at offset 0x14, so that the first 0x14 bytes are all zeros. See the implementation note above.

All numbers here are 32-bit integers, ie "longs".

```

00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000010 00 00 00 00 00 08 00 00 00 1C 00 00 00 64 00 00 .....d...
00000020 6A 00 00 00 01 97 00 00 00 A0 00 00 00 01 C6 00 j.....
00000030 00 00 CF 00 00 01 13 01 00 00 1D 01 00 00 01 .....
00000040 66 01 00 00 6E 01 00 00 01 94 01 00 00 9D 01 00 f...n.....
00000050 00 01 C3 01 00 00 C9 01 00 00 01 EF 01 00 00 F7 .....
00000060 01 00 00 01 61 6E 69 6D 73 00 01 00 00 00 72 00 ...anim.....r.
00000070 00 00 7B 00 00 00 8E 00 00 00 00 61 72 6D 66 6C ..{.....armfl
00000080 61 6B 5F 67 61 64 67 65 74 2E 67 61 66 00 24 02 ak_gadget.gaf.$
00000090 00 00 D8 2D 00 00 01 64 6F 77 6E 6C 6F 61 64 00 ...-...download.
000000A0 01 00 00 00 A8 00 00 00 B1 00 00 00 BD 00 00 00 .....
000000B0 00 41 52 4D 46 4C 41 4B 2E 54 44 46 00 61 28 00 .ARMFLAK.TDF.a(
000000C0 00 01 01 00 00 01 66 65 61 74 75 72 65 73 00 01 .....features..
000000D0 00 00 00 D7 00 00 00 E0 00 00 00 E8 00 00 00 01 .....
000000E0 63 6F 72 70 73 65 73 00 01 00 00 00 F0 00 00 00 corpses.....
000000F0 F9 00 00 00 0A 01 00 00 00 61 72 6D 66 6C 61 6B .....armflak
00000100 5F 64 65 61 64 2E 74 64 66 00 E3 28 00 00 68 02 _dead.tdf..(.h.
00000110 00 00 01 6F 62 6A 65 63 74 73 33 64 00 02 00 00 ...objects3d....
00000120 00 25 01 00 00 37 01 00 00 43 01 00 00 00 4C 01 .%...7...C...L.
00000130 00 00 5D 01 00 00 00 61 72 6D 66 6C 61 6B 2E 33 ..]...armflak.3
00000140 64 6F 00 39 2A 00 00 7B 14 00 00 01 61 72 6D 66 do.9*..{...armf
00000150 6C 61 6B 5F 64 65 61 64 2E 33 64 6F 00 C9 34 00 lak_dead.3do..4.
00000160 00 1A 11 00 00 01 73 63 72 69 70 74 73 00 01 00 .....scripts...
00000170 00 00 76 01 00 00 7F 01 00 00 8B 01 00 00 00 41 ..v.....A
00000180 52 4D 46 4C 41 4B 2E 43 4F 42 00 67 3F 00 00 E4 RMFLAK.COB.g?...
00000190 09 00 00 01 75 6E 69 74 70 69 63 73 00 01 00 00 ...unitpics....
000001A0 00 A5 01 00 00 AE 01 00 00 BA 01 00 00 00 41 52 .....AR
000001B0 4D 46 4C 41 4B 2E 50 43 58 00 B4 42 00 00 91 25 MFLAK.PCX..B...%
000001C0 00 00 01 75 6E 69 74 73 00 01 00 00 00 D1 01 00 ...units.....

```

```

000001D0 00 DA 01 00 00 E6 01 00 00 00 41 52 4D 46 4C 41 .....ARMFLA
000001E0 4B 2E 46 42 49 00 89 63 00 00 39 05 00 00 01 77 K.FBI..c..9....w
000001F0 65 61 70 6F 6E 73 00 01 00 00 00 FF 01 00 00 08 eaons.....
00000200 02 00 00 1B 02 00 00 00 61 72 6D 66 6C 61 6B 5F .....armflak_
00000210 77 65 61 70 6F 6E 2E 74 64 66 00 2D 67 00 00 42 weapon.tdf.-g..B
00000220 02 00 00 01

```

Let's get started...

```

00000010 00 00 00 00 08 00 00 00 1C 00 00 00 64 00 00 00 .....d...
                ^^^^^^^^^^^^ ^^^^^^^^^^^^

```

At offset 0x14, you see the number 0x8. This is the number of entries in the directory. Grabbing the next 32-bit number at offset 0x18, you get 0x1C. This is the offset of a list of directory entries. In this case, there are 8 entries in the list. The format of an entry is:

```

typedef struct _HPIEntry {
    long NameOffset;    /* points to the file name */
    long DirDataOffset; /* points to directory data */
    char Flag;         /* file flag */
} HPIEntry;

```

NameOffset	Pointer to the file name. This is a 0-terminated string of varying length.
DirDataOffset	Pointer to the directory data for the file. The actual data varies depending on whether it's a file or a subdirectory
Flag	If this is 1, the entry is a subdirectory. If it's 0, it's a file.

Looking at offset 0x1C, we see:

```

00000010                                64 00 00 00 .....d...
00000020 6A 00 00 00 01 97 00 00 00 A0 00 00 00 01 C6 00 j.....
00000030 00 00 CF 00 00 00 01 13 01 00 00 1D 01 00 00 01 .....
00000040 66 01 00 00 6E 01 00 00 01 94 01 00 00 9D 01 00 f...n.....
00000050 00 01 C3 01 00 00 C9 01 00 00 01 EF 01 00 00 F7 .....
00000060 01 00 00 01

```

The 8 entries are:

- 0x064, 0x06A, 1**
- 0x097, 0x0A0, 1**
- 0x0C6, 0x0CF, 1**
- 0x113, 0x11D, 1**
- 0x166, 0x16E, 1**
- 0x194, 0x19D, 1**
- 0x1C3, 0x1C9, 1**
- 0x1EF, 0x1F7, 1**

Let's look at the first entry. The Flag is 1, so it's a subdirectory. At offset 0x64, we see:

```
00000060 01 00 00 01 61 6E 69 6D 73 00 01 00 00 00 72 00 ....anims.....r.
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

or 'anims'. This is the name. Since this is a subdirectory, offset 0x6A contains the number of entries in the subdirectory, followed by a pointer to the first entry. This is exactly like the count/pointer pair at 0x14 that got us started. Think recursion.

```
00000060 01 00 00 01 61 6E 69 6D 73 00 01 00 00 00 72 00 ....anims.....r.
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
00000070 00 00 7B 00 00 00 8E 00 00 00 00 61 72 6D 66 6C ..{.....armfl
                ^^^^^
```

The number at offset 0x6A is a 1, indicating that there's only 1 file in this subdirectory. 0x6E contains the offset of the first (and only) entry in the subdirectory, which is:

0x7B, 0x8E, 0

The 0 indicates that this is a file. Looking at offset 0x7B, we see:

```
00000070 00 00 7B 00 00 00 8E 00 00 00 00 61 72 6D 66 6C ..{.....armfl
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
00000080 61 6B 5F 67 61 64 67 65 74 2E 67 61 66 00 24 02 ak_gadget.gaf.$
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

or 'armflak_gadget.gaf'. This is the name of the first (and only) file in the 'anims' subdirectory. Since this is a file, the data at offset 0x8E is a little different.

There are 3 items here instead of one:

```
typedef struct _HPIFileData {
    long DataOffset;      /* starting offset of the file */
    long FileSize;      /* size of the decompressed file */
    char Flag;          /* file flag */
} HPIEntry;
```

DataOffset	This is the offset in the HPI file that this file starts at.
FileSize	This is the decompressed file size. When you extract the file, it should be this many bytes long.
Flag	If this is 1, the file is compressed with LZ77 compression. If it's 2, it's compressed with ZLib compression. If it's 0, it's not compressed at all. This is the format used by the unit viewer

```
.
00000080 61 6B 5F 67 61 64 67 65 74 2E 67 61 66 00 24 02 ak_gadget.gaf.$
                ^^^^^
00000090 00 00 D8 2D 00 00 01 64 6F 77 6E 6C 6F 61 64 00 ...-...download.
                ^^^^^ ^^^^^^^^^^^^^ ^^^
```

Looking at offset 0x8E, we see that the three items are:

0x224, 0x2DD8, 1

If you recall, the directory size was 0x224 bytes. This says the file starts at the first offset after the directory, which makes sense and means we're interpreting things correctly. This also says that the extracted file should be 0x2DD8 (or 11,736) bytes long.

At this point, we know enough to actually traverse the directory tree in the HPI file. Here's a recursive pseudocode function to do it. The initial call to it would be 'TraverseTree(".", Header.Start)'.
to it would be 'TraverseTree(".", Header.Start)'.

TraverseTree(string ParentName, int offset)

Entries = Directory[offset]

EntryOffset = Directory[offset+4]

for count = 1 to Entries

NameOffset = Directory[EntryOffset]

DataOffset = Directory[EntryOffset+4]

Flag = Directory[EntryOffset+8]

Name = ParentName+"\\"+Directory[NameOffset]

print "Processing ",Name

if Flag = 1

TraverseTree(Name, DataOffset) <- recursion!

else

ProcessFile(Name, DataOffset)

End If

EntryOffset = EntryOffset + 9

Next Count

If you code this up in your language of choice and run it, it should print something like this: (if you haven't guessed already, the file I'm using as an example is the "Arm Flakker" unit's aflakker.ufo file)

```
.\anims
.\anims\armflak_gadget.gaf
.\download
.\download\ARMFLAK.TDF
.\features
.\features\corpses
.\features\corpses\armflak_dead.tdf
.\objects3d
.\objects3d\armflak.3do
.\objects3d\armflak_dead.3do
.\scripts
.\scripts\ARMFLAK.COB
.\unitpics
.\unitpics\ARMFLAK.PCX
.\units
```

```
.\units\ARMFLAK.FBI
.\weapons
.\weapons\armflak_weapon.tdf
```

At this point, I urge you to go look at that directory hex dump and traverse the thing by hand until it makes sense.

I can hear you now. "What the heck is that 'ProcessFile' function?" It decodes the file. I'll explain in a bit.

But first, here's a list of the various files in this HPI file, and their starting offsets.

If you don't understand where I got the starting offsets, go reread the directory hex dump until you do.

```
anims\armflak_gadget.gaf          0x0224
download\ARMFLAK.TDF              0x2861
features\corpses\armflak_dead.tdf 0x28E3
objects3d\armflak.3do             0x2A39
objects3d\armflak_dead.3do        0x34C9
scripts\ARMFLAK.COB               0x3F67
unitpics\ARMFLAK.PCX              0x42B4
units\ARMFLAK.FBI                 0x6389
weapons\armflak_weapon.tdf        0x672D
```

Because it's a short file, and because it decodes to readable text, I'm going to use the ARMFLAK.TDF file as the example.

If the file was not compressed at all, then the file is just inserted into the HPI file as one big chunk.

But if it is...

This is where the -REAL- fun begins. I'm going to take it slow here because I'm still half figuring it out myself (writing this has actually made me realize some things that I hadn't before).

When the file was compressed, it was broken up into chunks of 64K (65536) bytes each, plus one more chunk to hold anything left over. Each chunk was then compressed. Note that some chunks are larger when compressed than decompressed, which means that some compressed chunks can be larger than 64K.

The total number of chunks in the file can be obtained by the following formula:

```
chunks = Entry.FileSize / 65536
if (Entry.FileSize mod 65536) <> 0
    chunks = chunks + 1
```

The offset in the directory points to a list of 32-bit numbers that are the compressed sizes of each compressed chunk of data.

Following the list of sizes are the actual compressed chunks of data.

In this HPI file, each file has only one chunk, but the total1.hpi file contains some files with a dozen or so, and the hpi files on the CDs have files in them with over a hundred.

Going to offset 0x2861, we read in a chunk o'data and decrypt it to find this:

```

00002860      7E 00 00 00 53 51 53 48 02 01 01 6B 00 00 00      .~...SQSH...k...
00002870  01 01 00 00 FE 36 00 00 20 5B 51 49 4E 55 3C 0E      .....6.. [QINU<
00002880  64 64 94 5D 49 5D 11 14 29 7B D5 26 18 55 6E 75      dd.]I]..){.&.Unu
00002890  64 54 34 41 79 6C 9C 71 81 83 8B 3B 59 49 CB 4D      dT4Ay1.g...;YI.M
000028A0  43 D1 42 54 9A A5 A8 AA AF B0 B8 64 61 AC 6D B0      C.BT.....da.m.
000028B0  B1 82 72 34 79 B8 B0 BD DD A3 82 81 E8 86 AC 89      ..r4y.....
000028C0  98 92 C2 CF 98 EB 9D E0 56 BF A2 6F AB 5F A8 96      .....V..o._..
000028D0  B5 C3 9F B8 EB B9 BE 7D 4F C7 5F CE 2F D1 4C D1      .....}O._./.L.
000028E0  D0 D2 90

```

The decompressed file size of ARMFLAK.TDF is 257 bytes. This tells us that there's only one chunk. The size of this chunk is 0x7E bytes. The chunk itself immediately follows.

Each chunk looks like this:

```

typedef struct _HPICchunk {
    long Marker;          /* always 0x48535153 (SQSH) */
    char Unknown1;
    char CompMethod;     /* 1=LZ77, 2=ZLib */
    char Encrypt;        /* is the block encrypted? */
    long CompressedSize; /* the length of the compressed data */
    long DecompressedSize; /* the length of the decompressed data */
    long Checksum;       /* Checksum */
    char data[];         /* 'CompressedSize' bytes of data */
} HPICchunk;

```

Marker	This is the start-of-chunk marker, and is always 0x48535153 (ASCII 'SQSH').
Unknown1	I know not what this is for. It's always 0x02. Maybe some sort of version number?
CompMethod	This is the compression method. It's 1 for LZ77, 2 for ZLib.
Encrypt	This tells whether the block is encrypted a second time. See below.
CompressedSize	This is the size of the compressed data in the chunk. 0x6B bytes.
DecompressedSize	This is the size of the decompressed data in the chunk. 0x101 bytes.
Checksum	This is a checksum of the data. It's merely the sum of all the bytes of data (treated as unsigned numbers) added together
data	The actual compressed data in the

	chunk. CompressedSize (0x6B) bytes of it.
--	---

Let's look at the data.

```

00002870                20 5B 51 49 4E 55 3C 0E    ....6.. [QINU<.
00002880  64 64 94 5D 49 5D 11 14 29 7B D5 26 18 55 6E 75    dd.]I]..){.&.Unu
00002890  64 54 34 41 79 6C 9C 71 81 83 8B 3B 59 49 CB 4D    dT4Ayl.q...;YI.M
000028A0  43 D1 42 54 9A A5 A8 AA AF B0 B8 64 61 AC 6D B0    C.BT.....da.m.
000028B0  B1 82 72 34 79 B8 B0 BD DD A3 82 81 E8 86 AC 89    ..r4y.....
000028C0  98 92 C2 CF 98 EB 9D E0 56 BF A2 6F AB 5F A8 96    .....V..o._..
000028D0  B5 C3 9F B8 EB B9 BE 7D 4F C7 5F CE 2F D1 4C D1    .....}O._./..L.
000028E0  D0 D2 90

```

Doesn't look like much, does it. That's because (YOU GUESSED IT!) it's encrypted yet again! Note: the checksum is calculated BEFORE this decryption.

The 'Encrypt' field in the HPICchunk header is set to 1 to indicate that this decryption needs to be done.

To decrypt, do this (more pseudocode):

```

for x = 0 to CompressedSize-1
  data[x] = (data[x] - x) XOR x
next x

```

This gives us:

```

00002870                20 5B 4D 45 4E 55 30 00                [MENU0.
00002880  54 52 80 59 31 5D 0D 0A 09 7B D1 00 10 55 4E 49    TR.Y1]...{...UNI
00002890  54 22 00 3D 41 52 60 4D 41 43 4B 3B 11 01 83 01    T".=AR`MACK;....
000028A0  33 81 32 02 42 55 54 54 4F 4E B4 02 19 42 01 4E    3.2.BUTTON...B.N
000028B0  41 70 02 C2 01 46 4C 41 DD 23 02 7D E0 04 20 05    Ap...FLA.#.}...
000028C0  18 00 32 CF 00 D3 01 DE 56 3F 02 4F 03 5F 04 68    ..2....V?.O._.h
000028D0  05 33 1F 06 D3 01 3E 41 8F 07 9F 08 AF 09 80 0D    .3....>A.....
000028E0  00 00 4C                ..L

```

Woohoo! Look! Readable word fragments! But remember, the chunk is still compressed.

In this case, the block is compressed with LZ77, since CompMethod is 1.

The compression algorithm is a very basic sliding window compression scheme from the LZ77 family using a 4095 byte history and matches from 2 to 17 bytes long.

The first byte is kind of a "tag" byte which determines if the next eight pieces of data are literal bytes or history matches. Starting with the least-significant bit, this tag byte is scanned to figure out what to do.

When the current bit is a zero, the next byte of the input is transferred directly to the output and added to end of the history buffer.

When the current bit is a one, the next two bytes taken from the input are used as a offset/length pair. The upper 12 bits are the offset into the history buffer and the lower 4 bits are the length. If the offset is zero, the end of the input data has been reached and the decompressor simply exits.

Since we can assume that there will be no matches with a length of zero or only one byte, any match is a minimum of two bytes so we just add two to the length which extends our range from 0-15 to 2-17 bytes.

The match is then copied from the history buffer to the output and also added onto the end of the history buffer to keep it in sync with the output.

When all eight bits of the tag byte have been used, the mask is reset and the next tag byte is loaded.

Here is some decompress code:

```
int Decompress(char *out, char *in, int len)
{
/*
  Decompress buffer "in" of size "len" into buffer "out" (previously allocated) returns the
  number of decompressed bytes.
*/

  int x;
  int outbufptr;
  int mask;
  int tag;
  int inptr;
  int outptr;
  int count;
  int done;
  char Window[4096];
  int inbufptr;

  for (x = 0; x < len; x++) {
    in[x] = (in[x] - x) ^ x;
  }

  done = FALSE;

  inptr = 0;
  outptr = 0;
  outbufptr = 1;
  mask = 1;
  tag = in[inptr++];

  while (!done) {
    if ((mask & tag) == 0) {
      out[outptr++] = in[inptr];
      Window[outbufptr] = in[inptr];
      outbufptr = (outbufptr + 1) & 0xFFF;
      inptr++;
    }
    else {
      count = *((unsigned short *) (in+inptr));
      inptr += 2;
      inbufptr = count >> 4;
      if (inbufptr == 0)
        return outptr;
      else {
        count = (count & 0x0f) + 2;
        if (count >= 0) {
          for (x = 0; x < count; x++) {
            out[outptr++] = Window[inbufptr];
            Window[outbufptr] = Window[inbufptr];
            inbufptr = (inbufptr + 1) & 0xFFF;
            outbufptr = (outbufptr + 1) & 0xFFF;
          }
        }
      }
    }
  }
}
```

```

    }
  }
}
mask *= 2;
if (mask & 0x0100) {
  mask = 1;
  tag = in[inptr++];
}
}
return outptr;
}

```

When fed the data, the routine spits out:

```

00000000 5B 4D 45 4E 55 45 4E 54 52 59 31 5D 0D 0A 09 7B [MENUENTRY1]...{
00000010 0D 0A 09 55 4E 49 54 4D 45 4E 55 3D 41 52 4D 41 ...UNITMENU=ARMA
00000020 43 4B 3B 0D 0A 09 4D 45 4E 55 3D 33 3B 0D 0A 09 CK;...MENU=3;...
00000030 42 55 54 54 4F 4E 3D 33 3B 0D 0A 09 55 4E 49 54 BUTON=3;...UNIT
00000040 4E 41 4D 45 3D 41 52 4D 46 4C 41 4B 3B 0D 0A 09 NAME=ARMFLAK;...
00000050 7D 0D 0A 0D 0A 5B 4D 45 4E 55 45 4E 54 52 59 32 }...[MENUENTRY2
00000060 5D 0D 0A 09 7B 0D 0A 09 55 4E 49 54 4D 45 4E 55 ]...{...UNITMENU
00000070 3D 41 52 4D 41 43 56 3B 0D 0A 09 4D 45 4E 55 3D =ARMACV;...MENU=
00000080 33 3B 0D 0A 09 42 55 54 54 4F 4E 3D 33 3B 0D 0A 3;...BUTON=3;..
00000090 09 55 4E 49 54 4E 41 4D 45 3D 41 52 4D 46 4C 41 .UNITNAME=ARMFLA
000000A0 4B 3B 0D 0A 09 7D 0D 0A 0D 0A 5B 4D 45 4E 55 45 K;...}...[MENUE
000000B0 4E 54 52 59 33 5D 0D 0A 09 7B 0D 0A 09 55 4E 49 NTRY3]...{...UNI
000000C0 54 4D 45 4E 55 3D 41 52 4D 41 43 41 3B 0D 0A 09 TMENU=ARMACA;...
000000D0 4D 45 4E 55 3D 33 3B 0D 0A 09 42 55 54 54 4F 4E MENU=3;...BUTON
000000E0 3D 33 3B 0D 0A 09 55 4E 49 54 4E 41 4D 45 3D 41 =3;...UNITNAME=A
000000F0 52 4D 46 4C 41 4B 3B 0D 0A 09 7D 0D 0A 0D 0A 0D RMFLAK;...}.....
00000100 0A

```

Yay! Clear decoded text. Write this chunk out, and go get the next one. When there are no more chunks, close the file, and go process the next one.

To recompress, do something like the following:

```

WHILE look ahead buffer is not empty
  find a match in the window to previously output data
  IF match length > minimum match length
    output reference pair
    move the window match length to the right
  ELSE
    output window first data item
    move the window one to the right
  ENDIF
END

```

If CompMethod is 2, use ZLib compression to decompress the block. You can get the zlib source code from the zlib home page at <http://www.cdrom.com/pub/infozip/zlib/>

FBI Commands

By MartiD@worldnet.att.net

In the FBI tables, "0" means "off", "1" means "on" and a "?" denotes an unknown use or entry.

Editable Categories

Unit Name	the name of the unit
Version	?
Side	the side it is on
Objectname	I think is the internal name for the unit
Designation Name	the name you see when you put the mouse over the unit
Description	the description of the unit
Footprint X	the footprint going ne way
Footprint Z	the foot print going the over way
BuildCostEnergy	how much energy it takes to build it
BuildCostMetal	how much metal you need to build it
MaxDamage	how much damage the unit can take
MaxWaterDepth	how deep it can go in water
MaxSlope	how much of a slope it can be built on and go over?
EnergyUse	how much energy it needs to do things
BuildTime	how long it takes to build it
WorkerTime	how fast it will build some thing
Bmcode	? You probably don't want to mess with this
Builder	I think if it can build things or not
ThreeD	if its a 3D unit
ZBuffer	some thing to do with the graphics you probably don't want to mess with this
NoAutoFire	If the unit has no auto fire
SightDistance	how far it can see
RadarDistance	how far they radar will go
SoundCategory	?
EnergyStorage	how much energy that unit can hold leave it blank for the commander
MetalStorage	same as above but for metal
ExplodeAs	how big the explosion is when your unit dies
SelfDestructAs	how big the explosion is when you make the unit self destruct
Category	what type of unit it is ?
TEDClass	you probably don't want to mess with it
CopyRight	the annoying Cavedog copy right
YardMap	?
Corpse	what the wreckage looks like
GermanName	what the name of the unit is in German
GermanDescription	the description in German
UnitNumber	the number of the unit
FrenchName	the name of the unit in French
FrenchDescription	the description in French
firestandorders	?
StandingFireOrder	?
mobilestandorders	?
StandingMoveOrders	?
canmove	the unit can move
canpatrol	the unit can patrol
canstop	the unit can stop what ever its doing
canguard	the unit can guard another unit
MaxVelocity	?

BrakeRate	how fast the unit can stop
Acceleration	how fast the unit can speed up
TurnRate	how fast the unit can turn
SteeringMode	how it steers
ShootMe	?
CanFly	the unit can fly if you put a 1
crusisealt	?
Scale	?
BankScale	?
Builddistance	how far the unit can be away from the unit to start building
CanReclamate	can reclaim?
EnergyMake	how much energy the unit makes
MetalMake	how much metal the unit makes
DefaultMissionType	?
manuverleashlength	how for the unit can go when you put the orders to maneuver
MovementClass	how it moves like a Kbot?
Upright	if the unit stands up or not
standingmoveorders	?
buildangle	what angle it can build at or what angle it can be built at
Weapon1	the primary weapon
wpri_badTargetCategory	what kind of unit the primary weapon is not go at killing
BadTargetCategory	the kind of units that the unit is not go at killing
antiweapons	?
DamageModifier	if the unit will repair its self
canattack	the unit can attack another unit
ActivateWhenBuilt	the unit is activated after is built
onoffable	the unit can be turned off and on
RadarDistanceJam	the area the unit will jam
sortbias	?
IsAirBase	?
MinWaterDepth	the depth that the water has to be to go there
WaterLine	?
NoShadow	the unit wont have a shadow
TransMaxUnits	the max amount of units that it can carry
canload	the unit can load other units
transportsize	?
Weapon2	the second weapon
wsec_badTargetCategory	the group that the second weapon has trouble killing
Floater	if the unit floats?
NoChaseCategory	the type of unit that it wont chase
HoverAttack	if the unit will hover while attacking
Weapon3	the third weapon
SonarDistance	the distance of the units sonar
candgun	they unit can dgun
CloakCost	the cost of cloaking the unit
CloakCostMoving	the cost of cloaking that unit while its moving
HealTime	how long it takes to heal that unit
CanCapture	if they unit can capture or not
HideDamage	if the other players can see if that unit is about to die

ImmuneToParalyzer	immune to paralyzer
norestict	?
IsFeature	?
PigLatinName	the name in Pig Latin
SpanishName	the Spanish name
ItalianName	the name in Itlian
JapaneseName	the name in Japanese
PigLatinDescription	the description in PigLatin
MoveRatel	?
teleporter	if the unit can teleport
Stealth	if the unit is stealth
MakesMetal	if the unit makes metal
ExtractsMetal	if the unit extracts metal
altfromsealevel	?
attackrunlength	
Waterline	?
TidalGenerator	if it is a tidal generator
transportmaxunits	the max amount of units that it can transport
kamikaze	if it's a kamikaze
WindGenerator	if it's a wind generator
MobileStandOrders	?
BuildAngle	the angle that it can build at or the angle it can be built at
PitchScale	?
MoveRate2	?

ARM Abbreviations

ARMAAP	Advanced Aircraft Plant
ARMACA	Advanced Construction Aircraft
ARMACK	Advanced Construction Kbot
ARMACV	Advanced Construction Vehicle
ARMALAB	Advanced Kbot Lab
ARMAMD	Fortitude Anti-nuke
ARMANNI	Annihilator
ARMAP	Aircraft plant
ARMARAD	Advanced Radar Tower
ARMASER	Eraser
ARMASP	Air Repair Pad
ARMASY	Advanced Ship Yard
ARMATLAS	Atlas
ARMAVP	Advanced Vehicle Plant
ARMBATS	Millenium
ARMBRAWL	Brawler
ARMBRTHA	Big Bertha
ARMBULL	Bulldog
ARMCA	Construction Aircraft
ARMCARRY	Colossus
ARMCK	Construction Kbot
ARMCROC	Triton
ARMCRUS	Conqueror
ARMCS	Construction Ship
ARMCV	Construction Vehicle
ARMDRAG	Dragons Teeth

ARMESTOR	Energy Storage
ARMPFAST	Zipper
ARMPFAV	Jeffy
ARMPFIDO	Fido
ARMPFIG	Freedom Fighter
ARMPFLASH	Flash
ARMPFUS	Fusion Plant
ARMPGEO	Geothermal Plant
ARMPGUARD	Guardian
ARMPHAM	Hammer
ARMPHAWK	Hawk
ARMPHLT	Sentinel
ARMPJAM	Jammer
ARMPJETH	Jethro
ARMPLAB	Kbot Lab
ARMPLANCE	Lancet
ARMPLLT	Light Laser Tower
ARMPMAKR	Metal Maker
ARMPMART	Luger
ARMPMERL	Merl
ARMPMEX	Metal Extractor
ARMPMOHO	Moho Mine
ARMPMSHIP	Ranger
ARMPMSTOR	Metal Storage
ARMPPEEP	Peeper
ARMPPNIX	Phoenix
ARMPPT	Skeeter
ARMPW	Peewee
ARMPRAD	Radar Tower
ARMPRL	Defender
ARMPROCK	Rocko
ARMPROY	Crusader
ARMPSAM	Samson
ARMPSEER	Seer
ARMPSILO	Retaliator
ARMPSOLAR	Solar Collector
ARMPSONAR	Sonar Station
ARMPSPID	Spider
ARMPSTUMP	Stumpy
ARMPSUB	Lurker
ARMPSUBK	Piranha
ARMPSY	Ship Yard
ARMPTHUND	Thunder
ARMPTIDE	Tidal Generator
ARMPRTL	Torpedo Launcher
ARMPSHIP	Hulk
ARMPVADER	Invader
ARMPVP	Vehicle Plant
ARMPWIN	Wind Generator
ARMPZEUS	Zeus
ARMPCOM	Commander
ARMPGATE	Galactic Gate

CORE Abbreviations

CORAAP	Advanced Aircraft Plant
CORACA	Advanced Construction Aircraft
CORACK	Advanced Construction Kbot
CORACV	Advanced Construction Vehicle
CORAK	A.K.
CORALAB	Advanced Kbot Lab
CORAP	Aircraft Plant
CORAPE	Rapier
CORARAD	Advanced Radar Tower
CORASP	Air Repair Pad
CORASY	Advanced Ship Yard
CORAVP	Advanced Vehicle Plant
CORBATS	Warlord
CORCA	Construction Aircraft
CORCAN	The Can
CORCARRY	Hive
CORCK	Construction Kbot
CORCRASH	Crasher
CORCRUS	Executioner
CORCS	Construction Ship
CORCV	Construction Vehicle
CORDOOM	Doomsday Machine
CORDRAG	Dragons Teeth
CORESTOR	Energy Storage
CORETER	Deleter
CORFAV	Weasel
CORFINK	Fink
CORFMD	Fortitude Anti-nuke
CORFUS	Fusion Plant
CORGATOR	Instigator
CORGEO	Geothermal Plant
CORGOL	Goliath
CORHLT	Gaat Gun
CORHURC	Hurricane
CORINT	Intimidator
CORLAB	Kbot Lab
CORLLT	Light Laser Tower
CORMAKR	Metal Maker
CORMART	Pillager
CORMEX	Metal Extractor
CORMIST	Slasher
CORMOHO	Moho Mine
CORMSHIP	Hydra
CORMSTOR	Metal Storage
CORPT	Searcher
CORPUN	Punisher
CORPYRO	Pyro
CORRAD	Radar Tower
CORRAID	Raider
CORREAP	Reaper
CORRL	Pulverizer
CORROACH	Roach
CORROY	Enforcer
CORSEAL	Crock

CORSHAD	Shadow
CORSHARK	Shark
CORSILO	Silencer
CORSOLAR	Solar Collector
CORSONAR	Sonar Station
CORSPEC	Spectre
CORSTORM	Storm
CORSUB	Snake
CORSY	Ship Yard
CORTHUD	Thud
CORTIDE	Tidal Generator
CORTITAN	Titan
CORTL	Torpedo Launcher
CORTSHIP	Envoy
CORVALK	Valkyrie
CORVAMP	Vamp
CORVENG	Avenger
CORVP	Vehicle Plant
CORVRAD	Informer
CORVROC	Diplomat
CORWIN	Wind Generator
CORCOM	Commander
CORGATE	Galactic Gate
CORBUILD	Hydration Plant
CORTRUCK	Truck

BOS Functions

Distances are in meters and Angles are in degrees	
Cache [Object];	Disable texture animation (default = disabled)
Call-Script [Script]([Parameters]);	Call a script
Dont-Cache [Object];	Enable texture animation (default = disabled)
Dont-Shade [Object];	Disable shading (default for building = enabled; default for moving units = disabled)
Emit-Sfx [Type] from [Object];	Emit a SFX (fire or bubbles) from a point (pointbased) or from a vertex (vectorbased)
Explode [Object] [Type];	Explode an object

Get [SysVar];	Get a sysvar like BUILD_PERCENT_LEFT
Hide [Object];	Hide an object (default = all objects shown)
Move [Object] to [Axis] [Position] Speed [Speed];	Move an object along a selected axis (?-axis) with selected speed
Move [Object] to [Axis] [Position] Now;	Move an object immediatly
Return [Value];	Return to selected value
Set [SysVar] To [Value];	Set a sysvar to a value (ex : SET ARMORED TO TRUE)
Set-Signal-Mask [Signal];	Set an ID (0 , 2 , 4 , 8 , etc..) for the signal mask. this is how the script can be killed.
Shade [Object];	Enable shading (default for building = enabled; default for moving units = disabled)
Show [Object];	Show an object (default = all objects shown)
Signal [Signal];	Emit a signal (0 , 2 , 4 , 8 , etc..) to kill script with same signal-mask
Sleep [Time];	make a little pause between 2 or more events
Spin [Object] Around [Axis] Speed [Speed] Accelerate [Acceleration];	start spinning a object
Spin [Object] Around [Axis] Speed [Speed];	spin an object directly to max speed
Start-Script [Script]([Parameters]);	Start a script

Stop-Spin [Object] Around [Axis] Decelerate [Deceleration];	stop spinning an object with deceleration
Stop-Spin [Object] Around [Axis];	stop spinning an object immediatly
Turn [Object] to [Axis] [Bearing] Speed [Speed];	turn an object
Turn [Object] to [Axis] [Bearing] Now;	turn an object immediatly
Wait-For-Move [Object] Along [Axis];	Wait until selected object has finished moving along selected axis
Wait-For-Turn [Object] Around [Axis];	Wait until selected object has finished turning around selected axis

Still not tried :

Attach-Unit [Unit] To [Object];	Attach a Unit to selected object
Drop-Unit [Unit];	Drop unit
Jump [Destination];	Jump if ([Value] == False) [Destination];
Rand([Min], [Max]);	Randomize a static-var value . still not supported by the cobble

BOS Script Tutorial - TA

(written by Alakhbar)

Firstly let me break down the BOS file into its componets here. For all you C programmers, this is very much like C but in its basic form. This is a general format for the BOS file.

<pre> piece static-var #define #include Create() StartMoving() StopMoving() AimPrimary(heading,pitch) AimSecondary(heading,pitch) AimTetriary(heading,pitch) AimFromPrimary(piecenum) AimFromSecondary(piecenum) AimFromTetriary(piecenum) QueryPrimary(piecenum) QuerySecondary(piecenum) QueryTetriary(piecenum) QueryNanoPiece(piecenum) Activate() Deactivate() StartBuilding() StopBuilding() QueryBuildInfo(piecenum) Demo() QueryTransport(piecenum) BeginTransport(height) EndTransport() SweetSpot(piecenum) Killed() </pre>	<p>This lists most of the functions I have looked up in the BOS files in totala.hpi. Mainly I will be using the arm battleship.bos file for examples but I will try and cover the ones here as best as I can.</p> <p>I will also try to cover 2 topics here. The building functions and the transport functions.</p> <p>Anyways lets get started with the first one, "piece". This one tells TA which objects will be affected by the script. Such as "base" for the sweetspot() or "wake" for the bubble animations for ships and of course "beam" for the nano sprays. A general rule of thumb is any object that is used to aim with, emits a special effect (like smoke, bubbles, vtol flame, or nano spray), or creates that flash effect from cannon barrels gets put in here. The format for this is:</p> <pre> piece base, flare, wakel, wake2, turret; </pre> <p>Dont forget the colon. Its a C thing that symbolized end of statement. As you can see the names after "piece" are objects from the 3do file.</p>
---	--

static-var

This item is where you would put variables that change in the bos file. Say you wanted to alter fire between 3 barrels on a turret like how the battleship does, you would put that variable name here. That way it gets initialized and tells TA that this variable is going to store variables. Let me run through an example of the armbats.bos file for its primary turret.

<pre> static_var next_barrell; FirePrimary() { if (next_barrell==1) { } if (next_barrell==2) { } if (next_barrell==3) { } next_barrell=next_barrell+1; if (next_barrell==4) { nextbarrell=1; } } </pre>	<p>A little note on the ";"</p> <p>Notice how on most of the items here that they dont have a ";" after them? Thats because these aren't statements like the one on the bottom,</p> <pre> next_barrell=next_barrell+1;. </pre> <p>The FirePrimary() is a function and the if (next_barrell==1) is a comparison statement. They dont get a ";" at the end of them. Also notice the brackets "{" and "}". They also enclose statements. Each bracket that starts with a "{" must end with a "}" as in the example to the left.</p> <p>This is another part of C language here. You would use these in its basic form like below:</p> <pre> FirePrimary() { show flarela; sleep 150; hide flarela; } </pre> <p>where the brackets "{" and "}" enclose all the statements in the FirePrimary() function call. A more complex version is like the one to the left.</p>
---	---

The "..." just means that there was additional information there that didn't need to be typed up here.

As you can see by the "if" comparisons that its using the static-var "next_barrell" to see if it should run the statement below it. It is also incremented at the bottom by the statement "next_barrell=next_barrell+1". Also I need to note here that the difference between the "=" and the "==" thing is another C expression that means this. "=" means that a variable will have a variable of whatever you put after it and the "==" is used for comparisons like they are used in the "if" statements listed above. If you put "if (next_barrell=1)" all it will do is say "see if next_barrell is equal to 1, but first set next_barrell to be equal to 1, now compair".

static-var only holds variables. Thats all this does. TA can distinguish which variable is assigned to which unit so dont worry about having 30 battleships in the game and it being confused as to which turret is being assigned a variable and stuff.

#define

This confusing little thing is what you would use to define signals and animation variables. Yea confusing isn't it? Here's an example, again of the armbats.bos file:

```
...
#define SIG_AIM1 2
#define SIG_AIM2 4
#define SIG_WAKE 8
#define SMOKEPIECE1 base
...
aim at
```

This is going to be confusing so I will start with the first 3 items to the left. SIG is a signal which uses a value to separate them so they are independent, like how these 3 are.

What a signal does is stop a function from doing an action so that it can be restarted or stopped. The reason this is done is because say a battleship is aiming its turrets at a target, if the SIG isn't there then the turrets won't be able to re another target until the first target is dead. I believe this is what it does. I haven't tested a unit without them so I am uncertain as to if this is true.

Also the numbers next to them are what make the signals independent to one another. So say if SIG_AIM1 and SIG_AIM2 were both 2, then when the signal was sent to SIG_AIM1 to stop aiming, SIG_AIM2 would also be halted, even if it wasn't finished aiming (again I think this is what happens on overlapping signal numbers).

"SMOKEPIECE1 base" means that the object name "base" is where the smoke will emanate from when the unit is badly damaged. You can have more pieces smoke by telling it like say you want the turret to smoke also. It would be "#define SMOKEPIECE1 base" then below that "#define SMOKEPIECE2 turret1".

Another #define item is ANIM_VARIABLE TRUE. This tells TA that within the BOS file there are statements dealing with how the item is deactivated and activated. Like how the Brawler tucks in its wings when it lands or how the buildings open and close

	<p>themselves up for building stuff. Again these are just signals that allow a process to stop and be restarted.</p> <p>Notice also that the #defines dont use a ";" at the end of them.</p>
--	--

#include

Basically this just includes other scripts that are to go along with the regular bos file. The cobbler has a hard time, I think, in using path names. My suggestion is to change them to just a simple file name and keep the bos file, all the .h files and the Cobbler in the same directory. The format for this is:

```
#include "SFXtype.h"
#include "smokeunit.h"
#include "exptype.h"
```

Thats all the #include does is include files when the Cobbler compiles the BOS script. Also notice that the file names are surrounded by a " and that there is no ; at the end of these either. They are called using the statement "start-script 'function name'();" like, for example, the function inside smokeunit.h would be "start-script SmokeUnit();".

Create()

This is a function and what it does is tell TA what to do when the unit is completed. Any variable names that were used in static-var get initialized in here and any animations like the radar on top of the carrier and aircraft plants get placed in here as well. Its also a good idea to put the "start-script Smokeunit();" call in here as well so its ready to go when the unit is damaged, even while it was being built.

<pre> Create() { hide flare1a; hide flare1b; hide flare1c; hide flare2a; hide flare2b; hide flare2c; next_barrell=1; next_barrel2=1; start } script SmokeUnit(); </pre>	<p>As you can see the armbats.bos file here hides all the flares that are shown when the cannons are fired and the next_barrell=1; variables are initialized in here as well.</p> <p>Every unit must have this, even if there isn't anything done to the unit like the dragon teeth. It still needs this part.</p>
---	--

<pre> Create() { padflip = 0; start script SmokeUnit(); while(get BUILD_PERCENT_LEFT) { sleep 1000; } spin radar around y axis speed <60>; } var </pre>	<p>Take a look here. This is the Create() function for the armcarry.bos file. Again the variable padflip is initialized the start script SmokeUnit(); is called and below that a "while" condition is called. This "while" condition prevents the next line, which spins the radar, from starting until the BUILD_PERCENT_LEFT is completed. If you didnt have this, the radar would be spinning while the unit was being built.</p> <p>The "padflip = 0;" statement is just a static that is just being initialized.</p>
---	---

StartMoving()

Generally, k-bots, ships, and subs use this function to either show the wakes from ships and subs, or tell TA that the k-bot is walking. I havent

seen any use of these in vehicles or aircraft in their BOS files. Anyways for vessels a typical setup would be this:

<pre> StartMoving() { signal SIG_WAKE; set mask SIG_WAKE; while (TRUE) { emit emit sleep 300; } } </pre>	<p>See the "signal SIG_WAKE;" call?</p> <p>This tells the unit that if its changing its direction to restart the script. The signal function "StopMoving()" will have the call to stop the wake bubble animation. "set signal sfx SFXTYPE_WAKE1 from waken; sfx SFXTYPE_WAKE1 from wake2; mask" is what allows itself to be killed by another "signal" call.</p> <p>All this does is create the wake bubbles from a ship while its moving.</p>
---	--

The statement "while (TRUE)" means that if the signal hasnt been killed off, to continue with the following statements below it that are enclosed in brackets "{ }". "emit-sxf SXFTYPE_WAKE1 from waken;" is what makes the bubbles and "waken" is the object name in the 3do file. The delay, "sleep 300" means to not repeat the above statements for 300 ticks (I think) so that the bubbles arnt overlapping themselves and stuff.

For the k-bots, the only thing I've found is a boolean expression to indicate if the unit is moving or not. Its used in another function called "MotionControl()" which houses all the information for aiming certain parts like the nanobeams and such. Anyways, what k-bots use is:

<pre> static var bMoving; StartMoving() { bMoving = TRUE; } </pre>	<p>Pretty straight forward here. Its just using the static var of Moving to show the "MotionControl()" function that the unit is moving. You could pass this to another function, say the "Deactivate()" function by giving it a condition like "if (bMoving)".</p>
--	---

StopMoving()

This is just the opposite of "StartMoving()" and it is used the same way as the "StartMoving()" function call but what its used for is to send signals, like this:

<pre>StopMoving() { signal SIG_WAKE; }</pre>	<p>For ships and subs, all this is doing is killing off the animation bubbles that were started in the "StartMoving()" function.</p> <p>You dont need the "set signal mask" statement in here because its already set by the "StartMoving()" function.</p>
--	--

The k-bot script would use:

<pre>StopMoving() { bMoving = FALSE; }</pre>	<p>No signal call here, just the variable set to FALSE so the variable can be compaired in another function.</p>
--	--

AimPrimary(heading,pitch)
AimSecondary(heading,pitch)
AimTetriary(heading,pitch)

Turret aiming galore! Yup thats what this function does is aim the turret and the barrels (if any) towards their target. Turret based weapons need this. If you are unsure about needing this, check the "weapons.tdf" file and look for the item "turret=1" for the weapon you want to use. If thats true then you definitely will need this.

Using the armbats.bos file for an example, the format is:

<pre> AimPrimary(heading,pitch) { signal SIG_AIM1; set signal mask SIG_AIM1; turn turret1 to y signal axis heading speed <55>; turn turret1 to x axis (0 pitch) speed <30>; wait for turn turret1 around y axis; wait for turn turret1 around x axis; return(TRUE); } </pre>	<p>Again like in the "StartMoving()" function we have the signal calls here but for SIG_AIM1 instead. There isn't another function that stops the SIG_AIM1, its done in here because the weapon is constantly being aimed and it can be reaimed at another target because it has the "set mask" statement in here.</p> <p>I think the last statement, "return(TRUE);" is kinda like a condition where the unit wont stop aiming until it has the proper aim towards its target, then it goes on to the next phase... FIRING!!!</p>
--	--

Anyways the line "turn turret1 to y-axis heading speed <55>;". "turn", of course, means turn or rotate around an axis. "turret1" is the object as named in the 3do file. "to y-axis" means around y-axis. Thats confusing. "heading" is the number that is passed from the function "AimPrimary(heading,pitch)", its the direction of the target to "turret1". Or in naval terms, bearing. "speed <55>" is the speed setting of the object that will rotate in this statement.

A little note on "passing". This is a C thing where in a function name (such as in "AimPrimary(heading,pitch)") a value or set of values can be sent from the main program, through this function and used in the statements inside of it. Since the main program of TA already initializes and uses the variables "heading" and "pitch", you dont need to define them in the static-var part of your bos file.

You will notice here that the next line "turn turret1 to x-axis (0-pitch) speed <30>;" is using the "turret1" object to elevate the barrels of the turret along the x-axis. This I dont like because all the barrels are being aimed at the same time but I guess in the game you cant tell the difference. Everything is the same here exopt for the (0-pitch) number. It gets this number exactly like how it got the "heading" number, passed from the function name of "AimPrimary(heading,pitch)". The (0-pitch) sets the angle to where a projectile weapon will fire and since gravity is an issue in this game, it uses this format. I think this is how it will turn negative numbers into positive elevations. Again we have the speed setting by "speed <30>".

In the units .fbi file, AimPrimary is Weapon1, AimSecondary is Weapon2, and AimTetriary is Weapon3. I dont know what Weapon4 is though.

AimFromPrimary(piecenum)
AimFromSecondary(piecenum)
AimFromTertiary(piecenum)

All this function does is set which object is going to be used to aim with. The format for this is:

<pre>AimFromPrimary(piecenum) { piecenum=turret1; } AimFromSecondary(piecenum) { piecenum=turret2; }</pre>	<p>Again, turreted weapons need to have this so TA knows what object to use to get the "heading" and "pitch" angles from.</p> <p>We have the "passed" variables here but they are going the other way, "piecenum" is being defined here and in the TA main program it will have that information from this function.</p> <p>That's all this function does.</p>
---	--

QueryPrimary(piecenum)
QuerySecondary(piecenum)
QueryTertiary(piecenum)

Dont let the name fool you. These functions are what actually fires the weapon. TA asks "what object do I use to launch the weapon from?" and they are usually the objects named FLARE.

The format for this is:

<pre>QueryPrimary(piecenum) { piecenum=flare1a; } QuerySecondary(piecenum) { piecenum=flare1b; }</pre>	<p>Again like in the "AimFromPrimary()" function, its passing variables, or object names, to TA giving it the object names to use as firing positions for that particular weapon.</p>
---	---

Heres a rotating firing sequence for a triple barrel turret as like the one on the Arms Battleship:

<pre> QueryPrimary(piecenum) { if (next_barrell==1) { piecenum=flare1a; } if (next_barrell==2) { piecenum=flare1b; } if (next_barrell==3) { piecenum=flare1c; } } </pre>	<p>Back above in the "static var" section, the example shows the "next_barrell" variable being incremented and down here its being compaired to see which flare should be used. It would look really odd to have the center barrel firing all the time while the left and right barrels show their flares. Thats why this is done like this.</p> <p>Thats all this function does.</p>
--	---

FirePrimary(piecenum)

FireSecondary(piecenum)

FireTetriary(piecenum)

Yup another name mixup here. What this function does is show animation for firing. Turrets use this to move their barrels back and to show flares, like this:

<pre> FirePrimary() { if (next_barrell==1) { move barrella to z axis; axis [2.4] speed [500]; wait move barrella along z move barrella to z axis [0] speed [3.0]; show flarela; sleep 150; hide flarela; } } </pre>	<p>After the "if" comparison is done, a barrel is moved along the z axis to a position [2.4] AWAY from the center of its z axis at the speed of [500].</p> <p>Watch the brackets and where they are used. The bottom of this file has a list as to their occurrence. The "wait move" is for a pause that will wait for the specified object to actually move to its required length along a specified axis before proceeding to the next statement. The "along" is a good addition because it allows you to have multi axial movements (say a barrel that is raising out of a ground or whatever).</p> <p>The next statement is the barrel returning to the ready position but moving slowly because of the speed setting of [3.0]. Also notice that z-axis is [0] and not [2.4]. That's because its returning the object barrella to its original position. If you had said [2.4] instead of [0], the barrel would appear 2.4 units away from the center of its z-axis.</p> <p>The "show flarela;" does just that, it shows the flarela object, sleeps for 150 ticks (I think its in ticks which I think is 1/100th of a second?), and then hides the flarela object again. This gives it that flash effect.</p>
---	---

**Activate()
Deactivate()**

These are the state requests of a particular unit. The Brawler, Missile ship, MERL, Construction Yards, etc... use them. What they do is make a unit useable or un-useable until it goes into the ready status. Like how you cant fire a Nuke until its done loading it on the pad or why the MERL wont fire until the rocket is in a position to fire.

An important note here is that the #include "Statechg.h" line needs to be added twice in the bos file. Once before the #define ACTIVATECMD / DEACTIVATECMD and once after it. Let me show you an example of the arm's missile silo bos file:

<pre>....</pre>	<pre>See how "Statechg.h" was added twice</pre>
-----------------	---

```

#define ANIM_VARIABLE TRUE
#include "StateChg.h"
#include "activatescr.bos"
#include "deactivatescr.bos"
#include "smokeunit.h"
#include "exptype.h"

("Go()" and "Stop()" functions
here)
#define ACTIVATECMD call-script
Go();
#define DEACTIVATECMD call-script
Stop();
#include "StateChg.h"
-
.....
Go()
{
dont cache door1;
dont cache door2;
dont cache missile;
dont cache plate;
dont cache arm;
show missile;
call script activatescr();
ready = TRUE;
}

Stop()
{
ready = FALSE;
    script activatescr();
sleep 4000;
call script deactivatescr();
cache door1;
cache door2;
cache missile;
cache plate;
cache arm;
}

```

Here? Remember, once before and once after the "#define ACTIVATECMD" and "#define DEACTIVATECMD" definitions. Also notice after the ACTIVATECMD there is the segment "call script Go();". This is the function used to show the animation of the unit being readied to shoot. The same with the DEACTIVATECMD but to deactivate the unit. Take note of the "#define ANIM_VARIABLE TRUE" definition here as well. It will be needed.

Here is where the functions "Go()" and "Stop()" are at. I am not sure what the "don't cache" and "cache" do but by following this, a rule of thumb can be made. Use "dont cache" for objects that will be animated by the function "activatescr()" and "cache" for objects animated by the function "deactivatescr()".

You can see that "Go()" and "Stop()" are defined at the top by "#define ACTIVATECMD call script Go();" and "#define DEACTIVATECMD call script Stop();". What this says is when TA gets the ACTIVATECMD from the user (you wanting the unit to do something) to start running the function "Go()". And within that "Go()" function is a "call statement. And below that the variable "ready" which can be used for comparison in another function.

```

// Activate.bos file //
activatescr()
{
  If ( ANIM_VARIABLE )
  {
    move plate to y axis <0.00> now
    move arm to x axis <0.00> now
    move arm to z axis <0.00> now
    turn door1 around z axis <0>
now
    turn door1 to z axis [89.55]
speed <47.13>
    turn door2 around z axis <0>
now
    turn door2 to z axis [90.00]
speed <47.37>
    sleep <1900>
  }
  If ( ANIM_VARIABLE )
  {
    move door1 to y axis [6.10]
speed <3.13>
    move door2 to y axis [6.10]
speed <3.13>
    sleep <1950>
  }
  If ( ANIM_VARIABLE )
  {
    move plate to y axis [8.00]
speed <4.06>
    sleep <1970>
  }
  If ( ANIM_VARIABLE )
  {
    turn arm to x axis [90.00]
speed <45.54>
    sleep <1976>
  }
  sleep <114>
  return ( 0 )
}

```

Here is the "activatescr()" function from the arm's missile silo. Being called by the "Go()" function, and included by #include "activatescr.bos" definition, here is where the pieces are maneuvered into place that will allow the unit to fire.

You can see the "if" comparisons to see if ANIM_VARIABLE exists, which it does, so these conditions will always be true. Delays by the statements "sleep<1900>" pause the sequences here.

To me it looks like a better way of scripting this could have been done by using the "wait on move" command and removing the "if" comparisons, but I guess it has to be written like this.

At the very bottom is the "return (0)" statement which is just returning to the "Go()" function saying its done with this and can continue on with the rest of its statements.

The "Deactivate()" function is just the opposite of this "Activate()"function because it just replaces all pieces back into their original position.

Now all these functions are just setting up how the unit is to be activated. The actual call is from here:

```

// Armsilo.bos //
AimPrimary(heading,pitch)
{
    start script
RequestState( ACTIVE );

    signal SIG_AIM;
    set signal
mask SIG_AIM;
    while (!ready)
        {
            sleep( 250 );
        }
    start script
RestoreAfterDelay();

    return(TRUE);
}
// Segment from file
statechg.h //
.....
-
if (statechg_DesiredState ==
ACTIVE)
{
ACTIVATECMD
actualstate = ACTIVE;
}

```

So why is the statement "start script RequestState(ACTIVE);" the one that does it? Because in the "statechg.h" file it has an "if" comparison checking to see which state was requested. Since "ACTIVE" was passed from the "RequestState(ACTIVE);" statement, the "if" comparison is true and the statement within the brackets are executed. "ACTIVATECMD" is called and since "ACTIVATECMD" was defined to call script "Go()", we go through that function now. Walking through the "Go()" function we move to another "call script" but this one goes to the "activatescr()" function where it starts moving parts. After that's done it returns to "Go()" and then back to AimPrimary(heading,pitch)" where it then sets up the signal and goes on from there.

I know this was a big one but I had to over a very in depth example for using this function and how. If you are lost then load up the "armsilo.bos", "activate.bos", and the "statechg.h" files so you can walk through it with this text. If you want to make units that follow this, you need to get this.

**StartBuilding()
StopBuilding()**

This function just animates the construction unit or plant, depending on what it is, and can be used to call the ACTIVATECMD or state of the unit. A particular format is followed by the unit type as listed below:

<pre>StartBuilding() { spin pad around y axis speed <30>; } StopBuilding() { stop spin pad around y axis; }</pre>	<p>For plants that have a pad that spins around, they use this. Shipyards don't have a pad that spins so they don't use this at all.</p>
--	--

<pre>StartBuilding() { set INBUILDSTANCE to TRUE; } StopBuilding() { set INBUILDSTANCE to FALSE; }</pre>	<p>Construction aircraft use this one. TA has INBUILDSTANCE already defined and all this is doing is setting that variable as true or false.</p>
---	--

<pre>StartBuilding(heading) { buildheading = heading; start script RequestState(ACTIVE); } StopBuilding() { start script RequestState(INACTIVE); }</pre>	<p>C bots, C ships and C vehicles use this format. Their states are requested so that the animation of their nanolathing pieces can be opened and the variable "heading" is used in another function called TargetHeading(heading)" so that the nanolathing piece can be properly aimed.</p>
---	--

<pre> StartBuilding(heading,pitch) { bAiming = TRUE; while (NOT bCanAim) { sleep 100; } turn torso to y axis heading speed <300>; wait turn luparm to x axis (0 pitch<30>) speed <45>; wait for turn torso around y axis; for turn luparm around x axis; set INBUILDSTANCE to TRUE; } StopBuilding() { set INBUILDSTANCE to FALSE; signal SIG_AIM; set signal mask SIG_AIM ; call script RestorePosition(); } </pre>	<p>This is the layout for what the commander uses for its building process. Its using "bAiming" for comparison in other functions as well as the INBUILDSTANCE variable. The commander also uses signals because it also has regular weapons to fire with.</p> <p>You would use this format if your unit had weapons and a building capacity.</p>
--	---

TargetHeading(heading)

This is the second part for the "StartBuilding()" function that the mobile construction units (k-bots, ships, and vehicles) use. I believe this is what the nanolathing pieces use for their aiming.

<pre> TargetHeading(heading) { buildheading = 0 heading; } </pre>	
---	--

QueryNanoPiece(piecenum)

This is exactly like the "QueryPrimary()" function call. It tells TA which object piece to use for the nanolathing spray.

<pre>QueryNanoPiece(piecenum) { piecenum=beam; }</pre>	<p>This is just a simple designation for a unit that has one nanolathing piece named "beam".</p>
--	--

<pre>QueryNanoPiece(piecenum) { if(spray == 0) { piecenum=beam1; } if(spray != 0) { piecenum=beam2; } spray = !spray; }</pre>	<p>Here is a way to get 2 nanolathing pieces to fire at the same time. The example script is from the arms advanced vehicle plant. Apparently this is a way to pass 2 pieces through the "piecenum" variable by using this series of "if" comparisons. If your building unit has 2 nanolathing pieces, this may be the way to go.</p>
---	---

QueryBuildInfo(piecenum)

This function defines what object piece is used to attach the unit thats being built too. Only construction plants as well as the ship yards use this. Mobile construction units dont use this. The format is:

<pre>QueryBuildInfo(piecenum) { piecenum=pad; }</pre>	<p>Just defining what piece is used to locate the unit thats being built. I believe it attaches the unit to this piece so when the pad is spinning, the unit will spin with it.</p>
---	---

QueryTransport(piecenum)

Not sure exactly what this does but by a guess it might be the object that is used to carry the unit. Below is the fragment from the arm atlas bos file. I believe that the statement "piecenum=1;" references the first object in the list for the transport, the one named "base".

<pre> QueryTransport(piecenum) { piecenum=1; } </pre>	
---	--

BeginTransport(height)

This item sets the attachment point for the unit to be picked up by moving the object named "link" to the unit that will be picked up. The part (0-height) takes the height of the transport and turns it into a negative number and that is where "link" gets moved too. The "RequestState(ACTIVE);" handles the animation of the unit lowering because as like before in the "Activate()" function description above, it will reference its own "Activatescr()" function which will have its own statements for lowering the unit to carry it and then raising it again to its original height.

<pre> BeginTransport(height) { move link to y axis (0 height) now; start script RequestState(ACTIVE); } </pre>	
---	--

EndTransport()

This is the ending part for the "BeginTransport(height)" function. It also contains the "RequestState(INACTIVE);" call and does the opposite of the above function.

<pre> EndTransport() { start script RequestState(INACTIVE); } </pre>	
--	--

SweetSpot(piecenum)

I believe this is the object that is used to reference where the user needs to click in order to select it. It might also refer to where the opposing units aim for while targeting.

<pre> SweetSpot(piecenum) { piecenum=base; } </pre>	
---	--

Demo()

Part of CD's unit viewer call. Not all units have this but I think it allows a unit to be viewed in the unit viewer. The format for this is:

<pre>Demo() { unitviewer = TRUE; }</pre>	Also in other parts in different bos files, I have found that within the "Activate()" functions, the statement "unitviewer = FALSE;" was included so its possible to just add that statement to the "Activate()" functions of all bos files.
--	--

Killed(severity, corpsetype)

This is how the unit is destroyed and what to do in a certain condition. Mainly the condition is how much damage the unit takes. Here is the arm aircraft plants "Killed()" function.

```
Killed( severity, corpsetype )
{
if (severity <= 25)
{
    corpsetype = 1;
    explode base type BITMAPONLY | BITMAP1;
    explode beam1 type BITMAPONLY | BITMAP2;
    explode beam2 type BITMAPONLY | BITMAP3;
    explode door1 type BITMAPONLY | BITMAP4;
    explode door2 type BITMAPONLY | BITMAP5;
    explode light type BITMAPONLY | BITMAP1;
    explode nano1 type BITMAPONLY | BITMAP2;
    explode nano2 type BITMAPONLY | BITMAP3;
    explode pad type BITMAPONLY | BITMAP4;
    explode plate1 type BITMAPONLY | BITMAP5;
    explode plate2 type BITMAPONLY | BITMAP1;
    explode post1 type BITMAPONLY | BITMAP2;
    explode post2 type BITMAPONLY | BITMAP3;
    explode radar type BITMAPONLY | BITMAP4;
    return( 0 );
}
if (severity <= 50)
{
    corpsetype = 2;
    explode base type BITMAPONLY | BITMAP1;
    explode beam1 type FALL | BITMAP2;
    explode beam2 type FALL | BITMAP3;
    explode door1 type BITMAPONLY | BITMAP4;
    explode door2 type BITMAPONLY | BITMAP5;
    explode light type FALL | BITMAP1;
```

```

explode nano1 type SHATTER | BITMAP2;
explode nano2 type BITMAPONLY | BITMAP3;
explode pad type BITMAPONLY | BITMAP4;
explode platel type BITMAPONLY | BITMAP5;
explode plate2 type BITMAPONLY | BITMAP1;
explode post1 type FALL | BITMAP2;
explode post2 type FALL | BITMAP3;
explode radar type FALL | BITMAP4;
return( 0 );
}
if (severity <= 99)
{
corpsetype = 3;
explode base type BITMAPONLY | BITMAP1;
explode beam1 type FALL | SMOKE | FIRE | EXPLODE_ON_HIT | BITMAP2;
explode beam2 type FALL | SMOKE | FIRE | EXPLODE_ON_HIT | BITMAP3;
explode door1 type BITMAPONLY | BITMAP4;
explode door2 type BITMAPONLY | BITMAP5;
explode light type FALL | SMOKE | FIRE | EXPLODE_ON_HIT | BITMAP1;
explode nano1 type SHATTER | BITMAP2;
explode nano2 type BITMAPONLY | BITMAP3;
explode pad type BITMAPONLY | BITMAP4;
explode platel type BITMAPONLY | BITMAP5;
explode plate2 type BITMAPONLY | BITMAP1;
explode post1 type FALL | SMOKE | FIRE | EXPLODE_ON_HIT | BITMAP2;
explode post2 type FALL | SMOKE | FIRE | EXPLODE_ON_HIT | BITMAP3;
explode radar type FALL | SMOKE | FIRE | EXPLODE_ON_HIT | BITMAP4;
return( 0 );
}
}

```

GUI File Format

by Dark Rain

This is a description of the GUI file format used by TA to store menus layout information. It's used in build menus, dialogs etc.

All the stuff in this file was found by trial and error and some deduction, hours of fun ^_^.

Generic Infos about gui files :

Each gui files represent an interface on the screen, it can be full screen or just a small part of it. We'll call a gui file as as whole an Interface.

Each interface is divided into sub components called Gadgets, these components can be anything from a button to a scroll bar.

Here's an example of a simple gui file :

```
[GADGET0]
{
  [COMMON]
  {
    id=0;
    assoc=205;
    name=Mainmenu.GUI;
    xpos=0;
    ypos=0;
    width=640;
    height=480;
    attribs=52685;
    colorf=52685;
    colorb=52685;
    texturenumber=0;
    fontnumber=-51;
    active=1;
    commonattribs=-51;
    help=;
  }
totalgadgets=6;
[VERSION]
{
  major=-51;
  minor=-51;
  revision=-51;
}
panel=;
crdefault=;
escdefault=;
defaultfocus=SINGLE;
}
[GADGET1]
{
  [COMMON]
  {
    id=1;
    assoc=126;
    name=SINGLE;
    xpos=139;
    ypos=393;
    width=96;
    height=20;
    attribs=2;
    colorf=0;
    colorb=0;
    texturenumber=0;
    fontnumber=0;
    active=1;
    commonattribs=54;
  }
}
```

```

        help=;
    }
    status=0;
    text=SINGLE;
    quickkey=83;
    grayedout=0;
    stages=0;
}

```

Each gadget is between the

```

[GADGET0]
{
    [COMMON]
    {
        Gadget infos go here
    }
}

```

You have to note that only the [] are necessary. We have to suppose that GAGDET1, GAGDET2 etc are for the benefice of Cavedog Menu Editor, so the following is valid :

```

[]
{
    [COMMON]
    {
        Gadget infos go here
    }
}

```

Each Gadget is divided in 2 parts, the common attributes and the gadget specific attributes.

COMMON : The common attributes are attributes that are well.. common to each gadgets ^_^ . This mean that each gadget has all these attributes BUT it doesn't mean that all are effective. Some just dont apply to a type of gadget, like a button doesnt react to any change in size and width etc.

SPECIFIC : These are attributes that are needed for some type of gadgets, for example, some gadgets need a Text description or they need to be grayed out or not etc.

Another important thing, is the first gadget. This gadget represent the interface itself. It defines the interface position on the screen, it'S size, position, background graphic etc. All the following gadgets are for element of the interface in question.

```

-----
-----

```

COMMON Tag descriptions :

id :

The ID, is the most important attribute of a gadget, it defines what the gadget is. Is it a button, text field, scrollbar etc? This ID decides it.

Here are the various values the ID tag can take and their effects :

- 0 -> The ID 0 is always used in cavedog gui files to represent the first gadget that defines the interface. However, this is like Gadget#, it'S optional and probably only for their interface tool. In truth, you can give to the ID of the first Gadget in the file any value. Personaly, I would leave it to 0, it could have unpredictable results.
- 1 -> It makes the gadget a button
- 2 -> Creates a listbox.
- 3 -> Creates a textfield, it's doesnt have any borders, so you have to create them yourself in the background image of the form for example.
- 4 -> Creates a Vertical/horizontal Scroll bar.
- 5 -> It seems to makes the gadget the equivalent of a label for those familiar with VB. It allows you to place only text where you want to.
- 6 -> This creates a blank surface that will receive a picture at run time. It's used to display a small picture of the map when you're selecting a map, or to show a screenshot of your saved game.
- 7-> It is used to set the default font for labels
- 12 -> Used to display a picture box, really handy.

Most of these gadget types need other optional fields or have extra possibilites, we'll discuss it later in this text.

Name :

The name is used for two things, the first one is the graphic used for the gadget and as a target for "events" that are hard coded.

GRAPHIC :

The name you choose for it, is used to look directly in a gaf file that has the same name as your menu file. For example, MAINMENU.GUI has a corresponding file named MAINMENU.GAF. As you may or may not know, Gaf file are made of several image sequences that are packed inside of it and each sequence has a name. It will try to match that name with one inside the gaf file. Failling to do that,

it will resort to using the default graphics for it's ID and for it's size. This is why the buttons in MAINMENU.gui, which represent the main menu that appear when you open TA, use a button graphic even though there's no corresponding sequence in the gaf file.

You have to understand that a menu, any menu, be it a unit menu or the battle room menu, has access only to the graphic sequence stored in the gaf file with it's name and the sequences in commongui.gaf. This is because all the sequences in commongui.gaf are treated like global variables.

For changing those graphics individually you can either change the default graphic in commongui.gaf but this will change the button for ALL the buttons using the default graphics. This is a bit clunky. The right way to do this, would be to create a sequence inside MAINMENU.gaf called INTRO (for the Intro gadget). Create it with two frames for pressed and unpressed and voila.

This opens a lot of doors, this would allow TC to assign a different graphic to each button, sure as hell makes for more variety than the same damn default button everywhere ^_^.

Note : I thought of adding all the units Build Pictures to commongui.gaf. It would save a LOT of space for GUI based menus and MDF would be able to be done with little or no additional space compared to normal factories. Sadly, adding anything to commongui.gaf crashes TA on startup. I urge ppl to try to find a solution if you're interested but personally I'm flat out of ideas.

EVENT :

For a button to actually do something, it has to be linked to an event that's often hard coded. In MAINMENU.gui, you can see several button gadgets, one named INTRO, Credits, MULTI, SINGLE and EXIT. These 5 words are hard coded variable names for this menu. You cannot have one from another menu, as far as I can tell. So you can't really have functionalities to a TA menu, just subtract some by not using them.

There's a lot of hard coded events, pretty much a completely different new set for each menu. The only way to know them, is to look at Cavedog's original GUI files and deduce them from the buttons in it.

Something interesting to note, is that while you cannot add events, you can take out some. A use for this, is in a TC, to remove the Cavedog Logo that looks kind of ugly.

A button with a name that isn't linked to an event will be pressable but it won't have any impact on the game.

UNIVERSAL EVENT NAME :

As far as I can tell, HELPTEXT, works in every GUI. It's used as a label name, to make it display info about a gadget your cursor is on. It seems that the info is hard coded sadly, so you can't add new ones. I'm really not sure about this so if anyone else finds out gimme a call ^_^.

width/height

This one is pretty obvious, it just specify the with and height of a gadget. However, it's useless for most types of gadgets, since buttons, labels and picture boxes aren't affected by it.

xpos/ypos : Read, not so obvious detail inside.

These two tags represent the x, y coordinates of the gadget. Pretty easy, but there's a detail to know. For the first gadget, the x, y coordinates will be respected ONLY if it still allow the interface to fit completly in the screen. This is why in MAINMENU.gui, you need to reduce the interface width and height to something smaller than 640x480 before you move it around.

active :

If 1, then the gadget will be visible, if 0 then it's invisible.

fontnumber :

I found ONE use for this but I'm sure I'm missing something. When you set a custom font, it sets all the labels to this font. Well if you set fontnumber to something differant than 0, then it will use the default TA font again.

attribs :

There's only one use for it, that I've stumbled upon. For scrollbars, the value for vertical one has to be 2 and 1 for horizontal scrollbars.

assoc :

I did a lot of searching to find this one and I'm sure I'm only scratching the surface.

What it does is very simple, it assocy gadgets together (duh). For 90% of the gadgets I've seen so far, the value of assoc doesnt change anything. It gets activated only when it's used with gadgets that are

what we'll call "assoc aware". So far, I found only 2 such gadget that can communicate with each others, it's the scrollbar and the listbox.

When associated together, the listbox will tell the scrollbar how many item it has, what length the knob should have etc. The scrollbar on ther other hand, will tell the listbox when to scroll up or down the list as the user moves it around.

useless side effect :

When 2 scrollbar have the same assoc number, using the arrows to make the knob move will result in making the knob on the other scrollbar move.

I'm sure there's more to it, but I cant seem to figure it out.

As far as I can tell, the following tags do nothing :

colorf/colorb,
texturenumber,
commonattribs.

UNCOMMON Tag descriptions :

I'll describe these tags with the gadget they're used with, it'll simplify things greatly, since many have to be used in conjunction with each others.

Many of these tags have default value of 0 or an empty text but in game, the value isnt 0 or the text is different. This is because for gadgets having Event name for the current interface, TA do some special checks or in this case it modify some variables value.

One last thing, some gadget are invisble in game but you KNOW they're there and they should be visible. Things like buttons or labels. This is because TA checks for gadgets with some specific Event name and make them invisible under some circumstances.

Headers (ID 0) :

The header is the first gadget in an interface that I spoke of earlier. The special tags are :

totalgadgets -> This is used to indicate the total number of gadgets in the interface but it's a bogus tag. It doesnt have any effect on the interface, that I can see.

panel -> This is used to set the background picture of the interface. If the interface already use a pcx based background, then

leave it blank : panel=;

Setting the gaf works a bit like the name tag, it refers to the name

of a sequence in a gaf file with the same name as the gui menu or a sequence in the commongui.gaf.

crdefault -> I have no idea yet.

escdefault -> This indicate the name of the button that will be pressed when Escape is pressed.

defaultfocus -> This contain the name of the button that will be have the focus on by default when the interface open. (The focus is represented by the glowing rectangle)

Version -> This bit has to be added, after the common section. Put whatever value you want to, for the major, minor and revision. It doesnt matter.

```
[VERSION]
{
  major=-51;
  minor=-51;
  revision=-51;
}
```

EXAMPLES :

```
[GADGET0]
{
  [COMMON]
  {
    id=0;
    assoc=205;
    name=Mainmenu.GUI;
    xpos=0;
    ypos=0;
    width=640;
    height=480;
    attribs=52685;
    colorf=52685;
    colorb=52685;
    texturenumber=0;
    fontnumber=-51;
    active=1;
    commonattribs=-51;
    help=;
  }
  totalgadgets=6;
  [VERSION]
  {
    major=100;
    minor=-51;
    revision=-51;
  }
}
```

```
panel=;
crdefault=MULTI;
escdefault=IGPATCH;
defaultfocus=SINGLE;
}
```

Buttons (ID 1) :

The special tags for a button are :

status -> For simple buttons, this dictate which frame of the sequence in the gaf file the button will start on.
If you go past the max number of frames in the sequence, the button will be invisible.

For multiple stages buttons, the status has to be 0 or you'll get the wrong frame.

stages -> This is used to indicate the number of stages a button posses. For example, a On/Off button would have 2 stages.

text -> This is just the text of the button, type whatever you want to. For simple buttons, you just type your text, however, for multiple stages button, you have to use pipes to separate the text of each stages of the button.
It works as follow (3 stages) : text=Continues|Ends|Deathmatch;

quickkey -> This is a shortcut key that you can set for the gadget.
The key is an ascii number. Consult your nearest ASCII character table for more infos ^_^.

grayedout -> Tells if the button is grayed out not. It makes the button disabled but visible, unlike setting active to 0.

EXAMPLES :

Normal button ->

```
[GADGET4]
{
  [COMMON]
  {
    id=1;
    assoc=126;
    name=INTRO;
    xpos=409;
    ypos=393;
    width=96;
    height=20;
    attribs=2;
    colorf=0;
    colorb=0;
    texturenumber=0;
  }
}
```



```

        fontnumber=0;
        active=1;
        commonattribs=54;
        help=;
    }
    status=0;
    text=INTRO;
    quickkey=73;
    grayedout=0;
    stages=0;
}

```

Multi Stage Button (2 stages) ->

```

[GADGET6]
{
    [COMMON]
    {
        id=1;
        assoc=0;
        name=ARMPREV;
        xpos=100;
        ypos=283;
        width=300;
        height=40;
        attribs=32;
        colorf=240;
        colorb=0;
        texturenumber=0;
        fontnumber=0;
        active=1;
        commonattribs=0;
        help=;
    }
    status=0;
    text=On|Off;
    quickkey=0;
    stages=2;
}

```

Note that the text is aligned in the middle for simple buttons and it's aligned to the right for multiple stages buttons.

Listbox (ID 2) :

Well sadly, there's no special tags for it, so I'll just give an example :

```

[GADGET1]
{
    [COMMON]
    {

```

```
        id=2;
        assoc=1;
        name=GAMES;
        xpos=10;
        ypos=10;
        width=242;
        height=176;
        attribs=1;
        colorf=15;
        colorb=0;
        texturenumber=0;
        fontnumber=0;
        active=1;
        commonattribs=54;
        help=;
    }
}
```

Textbox (ID 3) :

Pretty simple gadget, just indicate it's size and position.

maxchars -> The maximum number of character in the textbox.

EXAMPLE :

```
[GADGET9]
{
    [COMMON]
    {
        id=3;
        assoc=0;
        name=ADDRESS;
        xpos=53;
        ypos=61;
        width=250;
        height=26;
        attribs=0;
        colorf=100;
        colorb=0;
        texturenumber=0;
        fontnumber=0;
        active=1;
        commonattribs=-125;
        help=;
    }
    maxchars=30;
}
```

Scrollbar (ID 4) :

Whether it's a vertical or horizontal scrollbar is decided by the width and height. One with say a width of 16 and a height of 184 will be a vertical scrollbar, where one with a width of 184 and a height of 16 will be an horizontal scrollbar.

With the height and width, you'll get the graphic for the orientation you want but to make it fully operational, you need the right attribs value : 1 for horizontal and 2 for vertical.

range -> This is the number of item the scroll bar contains. This isn't important, because it seems TA will always change it for scrollbar that have Event name and have an associate, beside, the result isn't visible.

thick -> ? Doesn't seem to affect anything.

knobpos -> I assume it's a position within the range of the number in the "range" tag. Sadly, you can't pre-set it, it doesn't do anything.

knobsize -> This, is obviously the size of the knob on the scrollbar. This too is usually changed by TA when the scrollbar has an associate.

EXAMPLES :

Vertical ->

```
[GADGET1]
{
  [COMMON]
  {
    id=4;
    assoc=243;
    name=KNOB;
    xpos=17;
    ypos=30;
    width=13;
    height=150;
    attribs=2;
    colorf=4;
    colorb=0;
    texturenumber=0;
    fontnumber=0;
    active=1;
    commonattribs=0;
  }
  range=151;
  thick=50;
  knobpos=0;
  knobsize=10;
}
```

Horizontal ->

```

[GADGET4]
{
  [COMMON]
  {
    id=4;
    assoc=243;
    name=SLIDER;
    xpos=317;
    ypos=63;
    width=150;
    height=13;
    attribs=1;
    colorf=4;
    colorb=0;
    texturenumber=107;
    fontnumber=0;
    active=1;
    commonattribs=10;
  }
  range=151;
  thick=100;
  knobpos=0;
  knobsize=10;
}

```

Labels (ID 5) :

Labels are used to display text, you can place them anywhere inside the interface and make them say whatever you want to.

text -> This is the text that the label will display on the screen. Like scrollbar, if the label name is an even name, then the text you picked might get changed by TA at run time.

link -> This is used to link a label to a button. What will happen is that when you click on the label instead of the button, it'll act just as if you clicked on the button. To make it work, just give the name of the target button as the link value.

EXAMPLE :

```

[GADGET3]
{
  [COMMON]
  {
    id=5;
    assoc=243;
    name=TEXT;
    xpos=476;
    ypos=132;
    width=117;
    height=13;
    attribs=17;
    colorf=0;
  }
}

```

```
        colorb=0;
        texturenumber=0;
        fontnumber=0;
        active=1;
        commonattribs=0;
        help=;
    }
text=Location;
link=StartLocation;
}
```

Blank Surfaces (ID 6) :

As I mentioned earlier, blank surfaces, are dynamic picture box. They're used in TA mainly to display the Mini map, as a preview when you're selecting a map, to show the screenshot of the minimap of your saved games etc. Just make sure it has the right event name so that TA can fill it with a nice picture.. Oooooooooooh a map. (I'm tired I think ^_^;)

hotornot -> This is used to enable the selection rectangle visible on button and such to be visible on the Blank Surface. If hotornot = 1 then it's visible, if it's equal to 0 then the focus rectangle is invisible.

EXAMPLE :

```
[GADGET2]
{
  [COMMON]
  {
    id=6;
    assoc=0;
    name=MAPPIC;
    xpos=328;
    ypos=78;
    width=252;
    height=252;
    attribs=0;
    colorf=15;
    colorb=0;
    texturenumber=0;
    fontnumber=0;
    active=1;
    commonattribs=0;
    help=;
  }
hotornot=0;
}
```

Fonts (ID 7) :

This sets the default font for labels for the whole interface. Sadly, I havent been able to change the font color or size yet, so you're stuck with black colored fonts.

filename -> This is the name of the font you want to use, check the fonts directory to know the names of the font. It's very important that you only use the name of the font in filename and not include the extension. So, for example, the files SMLFONT.FNT becomes filename=SMLFONT; .

EXAMPLE :

```
[GADGET1]
{
  [COMMON]
  {
    id=7;
    assoc=243;
    name=FONT2;
    xpos=600;
    ypos=321;
    width=26;
    height=66;
    attribs=13;
    colorf=15;
    colorb=0;
    texturenumber=100;
    fontnumber=9;
    active=1;
    commonattribs=-125;
    help=;
  }
  filename=SMLFONT;
}
```

Picture Box (ID 12) :

This allow you to display a picture of a gaf sequence. As usual, the name correspond to the sequence in the gaf file that you want the picture to display.

```
[GADGET13]
{
  [COMMON]
  {
    id=12;
    assoc=0;
    name=IGPATCH;
    xpos=0;
  }
}
```

```

        ypos=0;
        width=300;
        height=40;
        attribs=2;
        colorf=15;
        colorb=0;
        texturenumber=0;
        fontnumber=0;
        active=1;
        commonattribs=86;
        help=;
    }
}

```

Well that's it for now. I'll probably update when I find what the 3 mystery tags do ^_^ . For more infos or questions email me at rochdenis@hotmail.com

GAF Format

This is a description of the GAF file format used by TA to store all kinds of graphic elements, including animations, static pictures, user interface elements, etc.

Credits:

This document builds on the original by Saruman and Bobban.

In addition, I got much helpful info from Bizmut, Kinboat, and Manu.

I even figured a couple of things out myself, but the people listed above did most of it.

Warning: This is intended for use by people that already know what they're doing.

I'm a C programmer, so I'm doing things in C notation here, but I'll try to explain it so that those of you that don't speak C will be able to understand. If you don't understand, write me at joed@cws.org and I'll try to clear things up.

I'm also a big believer in examples, so I'll be walking you through a GAF file (Archipelago.GAF) as I explain.

The first part of the file is the header, which looks like this:

```

typedef struct _GAFHEADER {
    long IDVersion; /* Version stamp - always 0x00010100 */
    long Entries; /* Number of items contained in this file */
    long Unknown1; /* Always 0 */
} GAFHEADER;

```

Let's look at a sample header:

```
00000000 00 01 01 00 39 00 00 00 00 00 00 00
```

IDVersion is 0x00010100 like we expect. Entries is 0x39, indicating that there are 57 items contained in this file.

Immediately following the header is a list of pointers, one for each entry.

The list of pointers here looks like:

```
00000000                                68 EA 04 00
00000010 98 EA 04 00 C8 EA 04 00 F8 EA 04 00 78 EB 04 00
00000020 A0 EB 04 00 C0 ED 04 00 40 EE 04 00 68 EE 04 00
00000030 98 EE 04 00 C8 EE 04 00 F8 EE 04 00 78 EF 04 00
00000040 A8 EF 04 00 C8 F1 04 00 48 F2 04 00 78 F2 04 00
00000050 A8 F2 04 00 D8 F2 04 00 08 F3 04 00 88 F3 04 00
00000060 A8 F5 04 00 28 F6 04 00 58 F6 04 00 88 F6 04 00
00000070 B8 F6 04 00 E8 F6 04 00 18 F7 04 00 B8 F7 04 00
00000080 E8 F7 04 00 20 FB 04 00 C0 FB 04 00 F0 FB 04 00
00000090 20 FC 04 00 50 FC 04 00 80 FC 04 00 20 FD 04 00
000000A0 68 00 05 00 08 01 05 00 38 01 05 00 68 01 05 00
000000B0 98 01 05 00 C8 01 05 00 68 02 05 00 98 02 05 00
000000C0 F0 05 05 00 90 06 05 00 C0 06 05 00 F0 06 05 00
000000D0 68 07 05 00 28 08 05 00 58 08 05 00 88 08 05 00
000000E0 B8 08 05 00 E8 08 05 00 18 09 05 00 48 09 05 00
```

The next byte after the pointer list is at offset F0. Remember this.

Each pointer points to a structure that looks like this:

```
typedef struct _GAFENTRY {
    short Frames;      /* Number of frames in this entry */
    short Unknown1;   /* Unknown - always 1 */
    long Unknown2;    /* Unknown - always 0 */
    char Name[32];    /* Name of the entry */
} GAFENTRY;
```

The first pointer is directs us to location 0x04EA68. Going there, we find:

```
0004EA60                                01 00 01 00 00 00 00 00    t.....
0004EA70 46 72 6F 6E 64 30 31 00 00 00 00 00 00 00 00    Frond01.....
0004EA80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    .....
```

This entry has 1 frame, and is called 'Fron01'.

Following each entry is another list of structures, one for each frame.

```
typedef struct _GAFFRAMEENTRY {
    long PtrFrameTable; /* Pointer to frame data */
    long Unknown1;      /* Unknown - varies */
} GAFFRAMEENTRY;
```

The frame entry looks like this:

```
0004EA90 78 D5 03 00 02 00 00 00
```

PtrFrameTable is 0x03D578. This points us to a structure containing data about the first frame in this entry. It looks like this:

```
typedef struct _GAFFRAMEDATA {
    short Width;      /* Width of the frame */
    short Height;     /* Height of the frame */
    short XPos;       /* X offset */
}
```



```

short YPos;          /* Y offset */
char Unknown1;      /* Unknown - always 9 */
char Compressed;    /* Compression flag */
short FramePointers; /* Count of subframes */
long Unknown2;      /* Unknown - always 0 */
long PtrFrameData;  /* Pointer to pixels or subframes */
long Unknown3;      /* Unknown - value varies */
} GAFFRAMEDATA;

```

Here's the data:

```

0003D570          31 00 1F 00 15 00 0F 00
0003D580 09 01 00 00 00 00 00 00 F0 00 00 00 00 00 00

```

Width and height are (duh) the width and height of the frame. This frame is 0x31 by 0x1F pixels (49 x 31).

XPos and YPos are actually offsets that give the displacement of the frame from the entry's actual position on the map. So if the entry itself was placed at position 100,100, the frame would be at position 100-XPos,100-YPos. These offset can be negative. Here, they place the frame at 21 pixels left, and 15 pixels above the initial placement of the item on the map.

Unknown1 is always 9. No idea what it really means.

Compressed is the compression flag. If it's 0, the image is not compressed.

This image is compressed. More on this in a bit.

FramePointers. This is where it gets a little weird. If FramePointers is 0, then PtrFrameData points to pixel data. If it isn't, then PtrFrameData points to a list of that many more pointers to GAFFRAMEDATA structures. Each of these subframes is collectively treated as one frame. More in this in a bit.

Unknown2 is always 0.

PtrFrameData points to the pixel data or to more GAFFRAMEDATA structures, depending on the value of FramePointers. Here it's pointing to offset 0xF0.

If you remember, this is the first byte after the list of entry pointers way back at the start of the file.

Unknown3 is a mystery. Sometimes the value is 0. Sometimes it isn't. No idea what it means or how to calculate it.

Ok. Now we have this frame entry. Since FramePointers is 0, PtrFrameData points to pixel data. If the frame were not compressed, it'd just be the raw pixels, 31 chunks of 49 bytes each, corresponding to each line. This frame is compressed, so things are a little different.

Let's look at the data:

```

000000F0 07 00 29 00 44 17 00 45 21 0E 00 1B 00 44 17 04
00000100 44 B3 07 00 45 09 00 44 1B 10 00 1B 00 A3 0D 00

```

```

00000110 44 07 00 A2 03 00 A2 0F 00 44 1D 16 00 13 00 45
00000120 07 00 44 03 00 45 13 00 44 03 00 45 09 00 45 03
00000130 00 A3 1D 1F 00 15 00 45 07 04 44 A3 07 00 45 03
00000140 00 44 03 00 44 03 00 A3 03 00 45 03 00 45 03 08
00000150 44 B3 A2 1F 1D 00 09 04 45 44 0B 04 44 45 07 08
00000160 44 A3 44 03 00 A2 03 04 A2 B3 05 00 45 05 08 44
00000170 A3 B3 23 25 00 0D 00 45 03 04 B3 44 05 00 45 07
00000180 08 A3 44 B3 03 04 44 5B 05 06 45 05 00 44 03 04
00000190 A2 45 09 04 44 45 07 00 44 0F 2B 00 0F 00 A2 03
000001A0 00 45 03 00 45 03 00 44 03 00 45 03 04 B4 35 03
000001B0 00 B3 05 08 45 B3 44 03 04 A2 46 03 08 B4 44 A2
000001C0 05 00 44 0D 00 44 0D 2B 00 09 00 45 05 34 45 44
000001D0 A3 45 A2 44 B4 45 45 B3 36 45 45 35 03 0C A2 46
000001E0 A3 45 03 04 45 46 03 00 A2 03 04 B3 45 03 00 45
000001F0 0B 00 44 0F 28 00 0B 04 45 A2 05 38 B2 A3 B4 A2
00000200 A2 B4 45 A2 45 45 A3 46 44 46 44 03 14 A2 46 46
00000210 44 B3 37 03 00 35 03 04 A2 45 05 06 45 13 2B 00
00000220 04 A3 44 03 00 44 07 34 B4 A2 44 45 46 45 46 A2
00000230 A2 46 A2 B4 46 A2 03 04 B3 44 03 24 B2 A2 44 A2
00000240 36 A2 45 B4 A2 46 03 04 46 44 17 2A 00 09 08 44
00000250 A3 44 05 08 A3 45 36 03 60 A2 45 B2 45 45 A2 A2
00000260 B4 44 A3 35 45 A3 44 B4 A2 46 B2 45 A2 A2 B2 46
00000270 44 A3 07 04 A2 45 0D 29 00 0B 10 44 45 45 B3 45
00000280 03 60 46 44 45 46 A2 45 B3 45 B2 35 A3 44 B3 A3
00000290 B4 A1 46 45 45 A2 46 A2 B4 45 45 03 0C 46 44 A2
000002A0 45 11 2F 00 00 44 03 0C 44 A3 45 B3 03 04 45 35
000002B0 03 00 44 03 0C 45 B4 45 B4 03 58 46 A2 45 B3 44
000002C0 A3 44 45 A2 46 44 45 45 46 44 44 45 A2 44 45 B4
000002D0 A2 46 13 28 00 00 A3 05 00 45 07 00 46 03 00 45
000002E0 03 04 45 A3 0E 45 48 A2 46 45 A2 46 B4 A2 45 46
000002F0 B3 46 45 46 46 45 A2 A2 B4 A3 0A 45 15 31 00 0B
00000300 20 A2 45 45 35 B3 44 46 45 45 03 06 45 00 46 03
00000310 5C A2 46 44 A2 45 B3 45 45 46 36 B3 A1 A3 44 B2
00000320 A2 45 B4 A2 A2 B3 45 44 45 03 04 A3 44 05 00 44
00000330 2F 00 07 00 45 03 04 45 A3 03 14 B3 36 A3 B2 45
00000340 45 03 70 45 A3 B3 36 45 46 B3 A2 44 A3 45 A3 44
00000350 46 35 B4 45 B3 35 46 A2 36 45 B4 36 A2 A3 B2 45
00000360 0B 2C 00 09 00 45 07 0C A3 45 45 36 03 06 45 20
00000370 B4 45 A2 5B 46 36 45 B4 45 03 24 46 44 46 B3 A2
00000380 45 45 A3 A1 B4 05 0C 46 A2 B4 5A 07 00 45 09 2A
00000390 00 07 00 45 03 04 44 45 03 3C 45 B3 A2 45 45 38
000003A0 A2 45 44 45 45 46 45 45 A2 45 03 28 37 B4 45 A2
000003B0 46 44 46 45 46 45 45 07 00 45 13 26 00 09 28 A3
000003C0 45 B3 A2 46 45 A2 B2 A3 44 38 03 04 B3 45 03 00
000003D0 45 03 00 45 09 18 45 46 A1 36 B4 A2 46 05 08 B3
000003E0 45 45 17 27 00 0F 00 46 0A 45 08 A3 45 46 03 06
000003F0 5B 00 A3 03 0C 5B 45 45 37 05 04 45 A2 03 04 45
00000400 B4 03 04 45 A2 07 00 45 05 00 45 13 24 00 09 06
00000410 45 14 46 44 A3 44 B3 45 03 00 38 05 00 5A 05 04
00000420 44 A3 07 14 46 45 B2 35 45 A2 05 04 44 45 0D 06
00000430 45 0F 23 00 07 00 45 09 04 B3 37 03 00 45 05 00
00000440 36 03 00 44 03 04 45 B3 05 20 45 44 A2 46 44 A3
00000450 B3 44 45 0B 00 44 19 1B 00 0D 0C 45 A2 5A 37 0B
00000460 04 44 A3 03 04 46 A3 03 08 35 B3 45 07 04 B3 45
00000470 0F 00 A3 19 1C 00 0B 04 44 A2 07 00 46 07 08 44
00000480 46 44 07 06 45 03 00 45 03 00 45 03 00 35 03 00
00000490 A2 27 15 00 09 00 45 11 08 44 45 A3 09 00 44 05
000004A0 00 A2 0B 00 46 0F 00 44 17 0B 00 1B 00 A3 11 04
000004B0 B3 44 0D 00 44 25 08 00 19 04 B3 44 0B 00 44 3B
000004C0 07 00 19 00 36 0D 00 45 3B 07 00 27 00 A2 05 00
000004D0 A2 35 04 00 2B 00 44 37

```

PtrFrameData points to a short integer that is a count of bytes for the first line. Skip ahead that many bytes, and you get to a count for the second line, etc, etc. The first line is 7 bytes long, and consists of 29 00 44 17 00 45 21. The Height parameter tells you how many lines there are, in this case, 31. Broken into lines, minus the length data, we get:

```
Line 0 29 00 44 17 00 45 21
```

```

Line 1 1B 00 44 17 04 44 B3 07 00 45 09 00 44 1B
Line 2 1B 00 A3 0D 00 44 07 00 A2 03 00 A2 0F 00 44 1D
Line 3 13 00 45 07 00 44 03 00 45 13 00 44 03 00 45 09 00 45 03 00 A3 1D
Line 4 15 00 45 07 04 44 A3 07 00 45 03 00 44 03 00 44 03 00 A3 03 00 45 03 00 45 03 08 44
B3 A2 1F
Line 5 09 04 45 44 0B 04 44 45 07 08 44 A3 44 03 00 A2 03 04 A2 B3 05 00 45 05 08 44 A3 B3
23
Line 6 0D 00 45 03 04 B3 44 05 00 45 07 08 A3 44 B3 03 04 44 5B 05 06 45 05 00 44 03 04 A2
45 09 04 44 45 07 00 44 0F
Line 7 0F 00 A2 03 00 45 03 00 45 03 00 44 03 00 45 03 04 B4 35 03 00 B3 05 08 45 B3 44 03
04 A2 46 03 08 B4 44 A2 05 00 44 0D 00 44 0D
Line 8 09 00 45 05 34 45 44 A3 45 A2 44 B4 45 45 B3 36 45 45 35 03 0C A2 46 A3 45 03 04 45
46 03 00 A2 03 04 B3 45 03 00 45 0B 00 44 0F
Line 9 0B 04 45 A2 05 38 B2 A3 B4 A2 A2 B4 45 A2 45 45 A3 46 44 46 44 03 14 A2 46 46 44 B3
37 03 00 35 03 04 A2 45 05 06 45 13
Line 10 04 A3 44 03 00 44 07 34 B4 A2 44 45 46 45 46 A2 A2 46 A2 B4 46 A2 03 04 B3 44 03 24
B2 A2 44 A2 36 A2 45 B4 A2 46 03 04 46 44 17
Line 11 09 08 44 A3 44 05 08 A3 45 36 03 60 A2 45 B2 45 45 A2 A2 B4 44 A3 35 45 A3 44 B4 A2
46 B2 45 A2 A2 B2 46 44 A3 07 04 A2 45 0D
Line 12 0B 10 44 45 45 B3 45 03 60 46 44 45 46 A2 45 B3 45 B2 35 A3 44 B3 A3 B4 A1 46 45 45
A2 46 A2 B4 45 45 03 0C 46 44 A2 45 11
Line 13 00 44 03 0C 44 A3 45 B3 03 04 45 35 03 00 44 03 0C 45 B4 45 B4 03 58 46 A2 45 B3 44
A3 44 45 A2 46 44 45 45 46 44 44 45 A2 44 45 B4 A2 46 13
Line 14 00 A3 05 00 45 07 00 46 03 00 45 03 04 45 A3 0E 45 48 A2 46 45 A2 46 B4 A2 45 46 B3
46 45 46 46 45 A2 A2 B4 A3 0A 45 15
Line 15 0B 20 A2 45 45 35 B3 44 46 45 45 03 06 45 00 46 03 5C A2 46 44 A2 45 B3 45 45 46 36
B3 A1 A3 44 B2 A2 45 B4 A2 A2 B3 45 44 45 03 04 A3 44 05 00 44
Line 16 07 00 45 03 04 45 A3 03 14 B3 36 A3 B2 45 45 03 70 45 A3 B3 36 45 46 B3 A2 44 A3 45
A3 44 46 35 B4 45 B3 35 46 A2 36 45 B4 36 A2 A3 B2 45 0B
Line 17 09 00 45 07 0C A3 45 45 36 03 06 45 20 B4 45 A2 5B 46 36 45 B4 45 03 24 46 44 46 B3
A2 45 45 A3 A1 B4 05 0C 46 A2 B4 5A 07 00 45 09
Line 18 07 00 45 03 04 44 45 03 3C 45 B3 A2 45 45 38 A2 45 44 45 45 46 45 45 A2 45 03 28 37
B4 45 A2 46 44 46 45 46 45 07 00 45 13
Line 19 09 28 A3 45 B3 A2 46 45 A2 B2 A3 44 38 03 04 B3 45 03 00 45 03 00 45 09 18 45 46 A1
36 B4 A2 46 05 08 B3 45 45 17
Line 20 0F 00 46 0A 45 08 A3 45 46 03 06 5B 00 A3 03 0C 5B 45 45 37 05 04 45 A2 03 04 45 B4
03 04 45 A2 07 00 45 05 00 45 13
Line 21 09 06 45 14 46 44 A3 44 B3 45 03 00 38 05 00 5A 05 04 44 A3 07 14 46 45 B2 35 45 A2
05 04 44 45 0D 06 45 0F
Line 22 07 00 45 09 04 B3 37 03 00 45 05 00 36 03 00 44 03 04 45 B3 05 20 45 44 A2 46 44 A3
B3 44 45 0B 00 44 19
Line 23 0D 0C 45 A2 5A 37 0B 04 44 A3 03 04 46 A3 03 08 35 B3 45 07 04 B3 45 0F 00 A3 19
Line 24 0B 04 44 A2 07 00 46 07 08 44 46 44 07 06 45 03 00 45 03 00 45 03 00 35 03 00 A2 27
Line 25 09 00 45 11 08 44 45 A3 09 00 44 05 00 A2 0B 00 46 0F 00 44 17
Line 26 1B 00 A3 11 04 B3 44 0D 00 44 25
Line 27 19 04 B3 44 0B 00 44 3B
Line 28 19 00 36 0D 00 45 3B
Line 29 27 00 A2 05 00 A2 35
Line 30 2B 00 44 37

```

To decode the line, to the following:

1. Read a byte. This is a mask.
2. If (mask & 0x01) = 0x01
 - skip ahead (mask >> 1) pixels. This is transparency, allowing whatever was under the frame to show through.
 - else if (mask & 0x02) = 0x02
 - copy the next byte (mask >> 2)+1 times to output.
 - else
 - copy the next (mask & 0x02)+1 bytes to output.
3. go back to 1, until there are no more bytes left in the line.

A C code fragment to do this is:

```
char *data; // points to pixel data
```

```

for (y = 0; y < FrameData.Height; y++) {
    bytes = *((short *) data);
    data += sizeof(short);
    count = 0;
    x = 0;
    while (count < bytes) {
        mask = (unsigned char) data[count++];
        if ((mask & 0x01) == 0x01) {
            // transparent
            x += (mask >> 1);
        }
        else if ((mask & 0x02) == 0x02) {
            // repeat next byte
            repeat = (mask >> 2) + 1;
            while (repeat-- > 0)
                putpixel(x++, y, data[count]);
            count++;
        }
        else {
            repeat = (mask >> 2) + 1;
            while (repeat-- > 0)
                putpixel(x++, y, data[count++]);
        }
    }
    data += bytes; // point to next line
}

```

We do this to the above mess of data, and we get:

```

Line 0
45
Line 1
45
Line 2
44
Line 3
45
Line 4
45
Line 5
44
Line 6
44
Line 7
46
Line 8
46
Line 9
44
Line 10
A2
Line 11
A2
Line 12
45
Line 13
45
Line 14
45
Line 15
36
Line 16
46

```

```

Line 17          45          A3 45 45 36    45 45 B4 45 A2 5B 46 36 45 B4 45    46 44 46
B3 A2 45 45 A3 A1 B4    46 A2 B4 5A    45    *
Line 18          45    44 45    45 B3 A2 45 45 38 A2 45 44 45 45 46 45 45 A2 45    37 B4 45
A2 46 44 46 45 46 45 45    45    *
Line 19          A3 45 B3 A2 46 45 A2 B2 A3 44 38    B3 45    45    45    45 46
A1 36 B4 A2 46    B3 45 45    *
Line 20          46 45 45 45 A3 45 46    5B 5B A3    5B 45 45 37    45 A2
45 B4    45 A2    45    45    *
Line 21          45 45 46 44 A3 44 B3 45    38    5A    44 A3    46 45 B2 35
45 A2    44 45    45 45    *
Line 22          45    B3 37    45    36    44    45 B3    45 44 A2 46 44 A3
B3 44 45    44    *
Line 23          45 A2 5A 37    44 A3    46 A3    35 B3 45    B3
45    A3    *
Line 24          44 A2    46    44 46 44    45 45    45    45    35
A2    *
Line 25          45    44 45 A3    44    A2
46    44    *
Line 26          A3    B3 44
44    *
Line 27          B3 44    44
*
Line 28          36    45
*
Line 29          A2    A2
*
Line 30          44
*

```

This is essentially a big green splat, used to represent a patch of reclaimable foliage.

This entry is very simple. One frame. No subframes. If there were multiple frames, you'd display each frame in sequence. If the FramePointers member were not 0, then instead of pixels, PtrFrameData would point to a list of pointers that had that many entries. Each pointer would point to another GAFFRAMEDATA entry. When you are assembling that frame, you would paint all the subframes in order, and treat the whole thing as one single frame for animation purposes.

I think I've covered everything. If you have any questions, let me know.

SCT Format

Written by Kinboat, 1998.

**** Note ****

In order for these sections to work in TAE, their dimensions must be a multiple of 128.

**** File Header (28 bytes) ***

```

typedef struct _SCTHEADER {
    long Version;           // SCT Version. Set to 3.
    long PtrMinimap;       // Offset to the section's minimap.
    long NumTiles;         // The number of tiles in the section.
    long PtrTiles;         // Offset to the section's tiles.
    long Width;            // Width of the section.

```

```

    long Height;           // Height of the section.
    long PtrData;         // Offset to the section's data.
} SCTHEADER;

** TileData (PtrTiles) **
Each tile is 32x32 pixels.

BYTE TileData[NumTiles * 1024];

** SectionData (PtrSectionData) **
short SectionData[SectionWidth*SectionHeight];
    This array is an array of index values pointing to tiles stored in
    the TileData.

typedef struct _HEIGHTDATA {
    BYTE Height;           // Height value.
    short Constant1;      // Always -1.
    BYTE Constant2;       // Always 0.
} HEIGHTDATA;

HEIGHTDATA HeightData[SectionWidth * SectionHeight * 4]
    This is an array to the 3D height data for the section. Each tile
    pointed to by SectionData has 4 height values.

** Minimap (PtrMinimap) **
BYTE Minimap[128 * 128];

```

OTA Format

Initial Mission Commands

The Initial Mission selection is the true heart of mission creation. Most missions created do not rely on the AI to handle the battle, rather, everything is carefully planned ahead. Having units wait until they are attacked, having groups of units attack at the same time, having them target specific units, even roving patrols, are all commands that are set here. Giving all (or most) of your enemy units a series of commands that interact and mesh to make an appropriate defense (and offense) is what mission creation is all about. If a Unit is given no initial commands, or runs out of commands, the AI will take over, if it is a Computer controlled Unit (anything other than Player 1). If it is a Player controlled Unit and you want the Player to be able to interrupt the initial commands, insert a "s" (to make it selectable) in the initial command list.

What follows is a list of the available commands; use a comma to separate commands. Remember that all unit references should be to the unit's name, not the unit's description (e.g. CORTL not Core Torpedo Launcher).

m X Y

Tells the Unit to move to coordinates X and Y as measured in pixels; see "Location" on the Status Bar at the bottom of the program window.

p X Y

Tells the unit to patrol from the coordinate at which it is presently to the coordinate that follows the "p" and back again. If only one coordinate is given the computer will assume the starting point to be the second point in the command list. If you give two points (p X Y X Y), the unit will patrol between those two points. If you give three points, the Unit will patrol between those three points in the order given, etc. You can specify any number of points in this way. The patrol command should be at the end of the command list, since once on patrol, it will ignore all later commands.

a UNITNAME

Tells the Unit to attack the nearest specified Unit. This could be either a Unit Type (for example "a ARMCOM") or a unique identifier (for example "a FRED"). If there are no units of the specified Type, the AI will take control of the unit.

w SECONDS

Tells Unit to wait at its present location for the specified number of seconds before continuing with the command list. You can specify fractional seconds (e.g. "w 0.5").

b UNITTYPE X Y

Mobile Construction Units only. Tells the Construction Unit (or Commander) to build a Unit of the specified Type at location X Y.

b UNITTYPE n

Construction Building. Tells the Construction Building to build n Units of type UNITTYPE on its construction pad.

d

Destroys the Unit instantly.

s

Makes the Unit selectable even though the initial command list isn't completed yet, allowing the AI or player to take control of the Unit.

i UNITNAME

Place this Unit in the specified transport. (e.g. "i ARMSHIP" or if there is more than one transport give the transport a unique identifier e.g. "i JAKE").

u XY

You can then tell the transport where to unload the unit that it carries by designating u XY.

wa

In the initial mission list will wait until the unit is attacked.

"wa otherunit" will wait until otherunit is attacked. This works like the "i" mission--otherunit identifies a specific unit, not a general type of unit. In order for this command to work, 'otherunit' must be given a unique name.

Example: Let's say for a moment, that you're in the middle of creating an Arm mission, for which the goal is to get your squad of Zeus into the enemy's base and take out their fusion plant. While setting up the defenses, you might program a fleet of Rapier gunships to wait until the Arm lightning-chuckers have destroyed a radar tower abandoned in the middle of a valley, move to co-ordinates 513, 700, followed by a run on the Arm Commander, nestled in it's camp. To spring your little trap, you would need to do the following setup.

-Place radar tower with unique name (e.g. "TRAP")
-Place Rapier units, each with the following initial command
string:

```
wa TRAP, a ARMZEUS, m 513 700, a ARMCOM
```

-It's very important to make sure that you've provided the player with at least one Zeus unit to start with. If there is no Zeus units on the map, the Rapiers will simply skip that command and go straight after the Commander. Upon destruction of the radar tower by any other unit than a Zeus, the Rapiers will still search out the Zeus and try to kill it.

BugFix Information

By Switeck

Unit ID Number Changes:

18 Cavedog units share the same unit id #:

Unit	Name	Description	id#	new id#
ARMCA	Construction Aircraft	Tech Level 1	18	18
ARMCSA	Construction Seaplane	Tech Level 1	18 *CONFLICT: ARMCA	400 *
CORCA	Construction Aircraft	Tech Level 1	90	90
CORCSA	Construction Seaplane	Tech Level 1	90 *CONFLICT: CORCSA	403 *
ARMDECOM	Decoy Commander	Decoy Commander	232	232
CORDECOM	Decoy Commander	Decoy Commander	232 *CONFLICT: ARMDECOM	404 *
ARMPLAT	Seaplane Platform	Builds Seaplanes	242	242
CORPLAT	Seaplane Platform	Builds Seaplanes	242 *CONFLICT: ARMPLAT	405 *
ARMMANNI	Penetrator	Mobile Energy Weapon	269	401 *
CORSUMO	Sumo	Adv. Armored Assault Kbot	269 *CONFLICT: ARMMANNI	269
CORSS	Sea Serpent	Indigenous Lifeform	277	406 *
CORSSUB	Leviathan	Battle Sub	277 *CONFLICT: CORSS	277
CORUWES	Underwater Energy Storage	Increases Energy Storage	293	293
CORUWMS	Underwater Metal Storage	Increases Metal Storage	293 *CONFLICT: CORUWES	407 *

ARMUWES	Underwater Energy Storage	Increases Energy Storage	298		298
ARMUWMS	Underwater Metal Storage	Increases Metal Storage	298	*CONFLICT: ARMUWES	402 *
ARMEMP	Stunner	EMP Missile Launcher	322		
CORMABM	Hedgehog	Mobile Missile Defense	322	*CONFLICT: ARMEMP	408 *
CORPLAS	Immolator	Plasma Tower	124	*CONFLICT: CORPUN	409 *

The * (asterick) means changed.

ALL FIXED -- by moving the conflicting units to different unit id #'s, starting at 400 and going to 409. This should be safe so long as Cavedog doesn't create nearly 100 new units. According to Cavedog, this bug has no effect on gameplay - but since I'm removing other bugs, I'm removing this bug as well!

Weapon ID Number Changes:

All of the Cavedog weapon id's (in numerical order) are in 1 MASTER weapons.tdf file in the weapons dir (in REV31.GP3) instead of divided up among many little files of only a weapon or 2 each. The other files are of zero-length to force TA to forget all the incorrect and redundant entries in ccdata.ccx and total1.hpi.

Although there is only 255 weapon id #'s to work with, Cavedog has quite a few weapons id #'s that have unused weapons in them. Plus there is quite a few duplicates among Cavedog's weapons. I don't mean just similar - I mean identical down to the picture, speed, turn rate, and even acceleration speed of the weapon. Although it's a lot of work, it's just going through the unit FBI files and referring them to the same weapon id and going through weapons id #'s and deleting the unused ones. The problem is, getting TA both to recognize the changes and ignore the old id's. To do that, I have to change around a lot of stuff in REV31.GP3 -- it's the only file that overrides other files containing identical FBI's and weapons id's. For old weapons id #'s to be ignored, I have made empty files of the same name and put the whole weapon id #'s list in a single WEAPONS.TDF file.

With my tests so far, it all seems to work -- even units with 2 identical weapons such as the Seaplanes and Stealth Fighters seem to work correctly.

Cavedog IDENTICAL and UNUSED weapon id's!

Weapon ID	Name	Range	Damage	Area of Effect	Cost Metal	Cost Energy	Reload Path	Reload Time	
7	MINDGUN	500	100	16	-	-	L 1	1	unused! *
8	SBMISSILE	300	80	16	-	-	L 1.6	1.6	unused! *
19	VTOL_EMG2 (unused!)	510	12	8	-	-	L .6	.6	burst=3 burst rate=.1 *
22	ARM_DISINTEGRATOR	240	30000	48	-	400	L 1.2	1.2	
23	CORE_DISINTEGRATOR	240	30000	48	-	400	L 1.2	1.2	*
30	AMD_ROCKET	32000	500	96	200	10000	L 120	120	
117	ARMSCAB_WEAPON	27000	500	96	200	10000	L 120	120	* -- see NOTE #1
32	FMD_ROCKET	32000	500	96	200	10000	L 120	120	
118	CORMABM_WEAPON	22000	500	96	200	10000	L 120	120	* -- see NOTE #1
38	ARMAMPH_WEAPON2	700	46(97)	48	-	-	L 2	2	*
106	ARMRL_MISSILE	700	46(97)	48	-	-	L 2	2	
43	CORFLAK_GUN	750	146	125	-	-	B .7	.7	Flak
79	ARMAAS_WEAPON3	750	146	125	-	-	B .7	.7	Flak *

180	CORARCH_WEAPON3	750	146	125	-	-	B	.7	Flak	*
50	ARM_LIGHTCANNON	240	50	32	-	-	B	1.53		
51	CORE_LIGHTCANNON	240	50	32	-	-	B	1.53		*
55	CORE_THUD	230	80	48	-	-	B	1.9		
159	CORAMPH_WEAPON1	230	80	48	-	-	B	1.9		*
75	ARMSFIG_WEAPON	510	50(140)	48	-	-	L	3		
76	CORSFIG_WEAPON	510	50(140)	48	-	-	L	3		*
96	ARMSFIG_WEAPON2	510	50(140)	48	-	-	L	3		*
97	CORSFIG_WEAPON2	510	50(140)	48	-	-	L	3		*
77	ARMAAS_WEAPON1	710	57(108)	48	-	-	L	2		-- see NOTE
#2										
78	ARMAAS_WEAPON2	710	57(108)	48	-	-	L	2		*
103	ARMSHIP_MISSILE	710	57(108)	48	-	-	L	2		
178	CORARCH_WEAPON1	710	57(108)	48	-	-	L	2		*
179	CORARCH_WEAPON2	710	57(108)	48	-	-	L	2		*
82	ARM_LASER	180	30	8	-	-	L	.865		*
88	CORE_LASER	180	30	8	-	-	L	.865		
95	ARMSEAP_WEAPON3	510	44(130)	48	-	-	L	3		*
98	CORSEAP_WEAPON3	510	44(130)	48	-	-	L	3		*
108	ARMVTOL_MISSILE	510	44(130)	48	-	-	L	3		
107	CORRL_MISSILE	700	45(95)	48	-	-	L	2		
148	CORFRT_MISSILE	700	45(95)	48	-	-	L	2		*
112	ARMVTOL_ADVMISSILE	659	70(150)	48	-	-	L	3		
115	ARMVTOL_ADVMISSILE2	659	70(150)	48	-	-	L	3		* -- see NOTE
#3										
113	CORVTOL_ADVMISSILE	650	68(155)	48	-	-	L	3		
114	CORVTOL_ADVMISSILE2	650	68(155)	48	-	-	L	3		*
146	ARMSEAP_WEAPON1	300	400	16	-	-	L	6		
152	CORSEAP_WEAPON1	300	400	16	-	-	L	6		*
193	ARMSEAP_WEAPON2	300	400	16	-	-	L	6		*
194	CORSEAP_WEAPON2	300	400	16	-	-	L	6		*
145	ARMSCORP_WEAPON	320	400	8	-	-	L	1.2		*
181	CORSCORP_WEAPON	320	400	8	-	-	L	1.2		
143	ARMFRT_MISSILE	700	46(97)	48	-	-	L	2	unused!	*
157	ARMFARK_WEAPON	320	112	48	-	-	B	1.8	unused!	*
158	ARMSCRAM_WEAPON	320	112	48	-	-	B	1.8	unused!	*
162	CORHUNT_WEAPON	320	112	48	-	-	B	1.8	unused!	*
164	ARMSEHAK_WEAPON	320	112	48	-	-	B	1.8	unused!	*

The * (asterick) shows the weapon id #'s that have been freed up with the removal of that weapon.

This patch frees up 32 weapons id #'s.

These are the weapon ID#'s removed in numerical order:

7,8,19,23,38,51,76,78,79,82,95,96,97,98,114,115,117,118,143,145,148,152,157,158,159,162,164,178,179,180,193,194

7,8,19,143,157,158,162,164 are never used even in the original game!

All of Cavedog's units use 197 weapon id #'s BEFORE this patch. (counting unused weapons id's in the game.)

All of Cavedog's units use 165 weapon id #'s AFTER this patch.

NOTE #1:

Although the 4 antinuke weapons have VASTLY different maximum ranges, ALL the antinukes WILL shoot down nukes fired into their protected zone from beyond their maximum range. The only thing the antinuke weapon range

changes is how soon the antinukes fire after the nuke is fired. The shortest-ranged antinuke weapon, CORMABM_WEAPON (Core Hedgehog's weapon) has a weapon range of 22,000 pixels -- even this could cross from one side to the other of a 40 x 40 map. (The largest non-3rd party map in TA is only 40 x 40!) Since the ranges all correspond to an incredibly huge distance, and because any of the antinuke weapons WILL shoot down nukes fired from beyond their max range, I have removed ARMSCAB_WEAPON and CORMABM_WEAPON and used the original antinuke weapons (AMD_ROCKET and FMD_ROCKET) instead on the mobile antinuke vehicles.

NOTE #2:

The missile weapon used on Arm Flakker ships (and also reused on Core's Flakker ships as well) has been modified to make them a 2-weapon unit instead of a 3-weapon unit. This was done by halving the reload time for the weapon, thus making it roughly equal to the original 2 weapons it replaces. The scripting model for the AA ships also had to be changed so the single missile weapon appeared to be firing from different turrets.

NOTE #3:

The 2 different Hawk missiles (ARMVTOL_ADVMISSILE and ARMVTOL_ADVMISSILE2) have a tiny difference in top speed (559 vs 554 -- less than a 1% difference), I removed the 2nd missile because for all practical purposes they are identical. Many things in the game are rounded down to the nearest multiple of 32 (the tile size) and speed is probably one of them.

3rd party units don't work with Bugfix?

Some 3rd-party units may not work with this patch without changes *IF* they use weapons removed by TA Bugfix. Like if a unit used MINDGUN as a weapon. I tried not to take "sides" when deleting shared weapons id #'s. If equal numbers of units used different weapons, the oldest (as in the case of ARMRL_MISSILE) weapons id # was kept or the lowest weapons id # was kept in the case of ARM_DISINTEGRATOR.

To make these 3rd-party units work with TA Bugfix, place the old_weapons.ufo file in the totala directory. This will ADD BACK all the removed weapons with corrections to them as well. This should allow almost ANY 3rd party unit to work with TA Bugfix. (And if it doesn't please email me.)

Speed Benifit:

Another side benifit of removing all duplicate files is an increase in gamespeed - at least one speed notch. Also, the game will load a new mission faster because it shuffles through fewer files.

Because of the decrease of EMG lag as well as the removal of many categories for units, internet games and LAN games should lag less.

AI fixes:

Another bug fixed by this patch is making map OTA's use a better ai profile. Original pre-TA:CC maps were particularly bad about their choice of ai profiles that they used. This should make the ai's at least slightly more effective on those maps than originally.

Map OTA -----	Did use -----	Now uses -----	
Acid Foursome	Acid	Default	
Acid Pools	Hover	Wind	
Checker Ponds	Acid	Default	
Crystal Cracked	SeaBattle	Hover	
Etorrep Glacier	Default	Wind	
Evad River Confluence aircraft-only maps	AirBattle	Hover	*AirBattle is strictly
Gasplant Plain	Default	Wind	
Hundred Isles	Default	SeaBattle	
Kill the Middle	AirBattle	Wind	
Lava Mania	Default	Wind	
Metal Heck	Default	Metal	
Metal Isles	Default	SeaMetal	
Over Crude Water	AirBattle	Krogoth	
Ring Atoll	Default	Hover	
Seven Islands	Default	SeaMetal	
Shore to Shore	Seabattle	Hover	
Show Down	Default	Wind	
Surface Meltdown	Acid	Default	
The Cold Place	Default	Wind	
The Pass	Default	ThePass	
Trout Farm	Default	Urban	
Two Continents aircraft-only maps	AirBattle	Hover	*AirBattle is strictly

Descriptions of AI Profile types:

(this is what they probably should be, not what they are...)

Acid.txt

Presumes that each player is separated by acid, so ships and ground troops cannot (safely) get from one island to another. This means the ai should build aircraft and hovercraft (and Pelicans).

AirBattle.txt

Presumes that each player is separated by impassable terrain (clouds, lava, etc) that even hovercraft cannot cross. This means the ai should only build aircraft for attack.

Default.txt

Presumes a big land map that all units can cross. Presumably no water is on the map, so making ships is pointless. So the ai should build everything but ships.

Hover.txt

Presumes a water map with lots of land to build many factories on. This allows the ai to build many aircraft plants as well as hovercraft platforms. Ships aren't neglected, but have low limits.

Krogoth.txt

For a metal map that has water as well as land. Ground troops presumably can travel from almost any part of the map to any other part. Ships aren't neglected, but have low limits.

Metal.txt

Strictly a metal map with little or no water and no obstructions to ground units. So the ai should build everything but ships and metal makers.

SeaBattle.txt

Presumes a water map with limited land to build factories on. Ships are the main form of attack or defense, but small numbers of air plants and hovercraft platforms might be built.

SeaMetal.txt (Does not normally exist in the game!)

A metal water map, presumably with lots of land to build many factories on. This allows the ai to build many aircraft plants as well as hovercraft platforms. Ships are still a major form of attack but by no means the only one. Actually, this is just a metal form of the Hover.txt ai -- with the exception that the ai shouldn't make metal makers.

ThePass.txt (Does not normally exist in the game!)

A special map ai, designed specifically for The Pass map. It builds very few factories or base defenses to conserve its very limited build space, nor does it build many metal extractors.

Waterwrld.txt

Presumes an entirely water map. Ships are the only form of attack or defense, with seaplane platforms as the only other factory type. The ai should not try to build any land-based factories.

Wind.txt (Does not normally exist in the game!)

Presumes a big land map that all units can cross. Presumably no water is on the map, so making ships is pointless. So the ai should build everything but ships and solars. However the ai's primary early energy source is wind generators instead of solars.

Urban.txt

Presumes a water map with lots of land to build many factories on. This allows the ai to build many aircraft plants as well as hovercraft platforms. Ships aren't neglected, but have low limits. Also, this map is low on resources but may have reclaimable objects (buildings, rocks, and trees) to make up for this lack. Also presumes there is underwater metal spots.

SIDEDATA.TDF changes:

The ai's build menus are pretty messed up:
(this is stored in the SIDEDATA.TDF file in the GAMEDATA dir.)

FIXED: Core is unable to build seaplane platforms and advanced construction subs!

FIXED: Arm's seaplane platforms sometimes quit building for no reason!

ADDED: All the TA:CC units that were missing, plus the 6 post-v3.1 patch units.

I left the ability of the ai's construction hovercraft to build advanced vehicle plants and the ability of the ai's construction seaplanes to build

advanced k-bot labs in, because it is only a small advantage for the ai -- and the ai needs that (and more) to compete with good players.

Ai profile changes:

There are NO ai profiles in][BF14.ufo! TA Bugfix uses the ai profile files found in the old Rev31.gp3 file (renamed Rev31.ccx or incorporated into totala1.hpi) instead. I did not include my ai profiles (that is found on other websites) with this patch. Also, ai profiles are something that typically get changed often like maps do - which is much easier to do if the ai files are in an AI directory instead of buried inside of a ufo file.

Single-player Mission changes:

Some single-player maps have been altered so they can be used as multiplayer maps as well, HOWEVER they *MUST* be installed to the harddrive in order to run (the big totala4.hpi file, that is).

Also, a small and currently incomplete Arm campaign has been added as well. It takes place at about the same time as the missions in TA and TA:CC.

The missions will play better if ai profiles are added to augment them! The special ai profiles they use are:
SeaMMiss.txt for Sea Metal maps
Sea_Miss.txt for Sea maps

The Krogoth Campaign has been altered so the original hard setting is now only the medium setting. The new hard setting adds only 1 enemy unit to the ai and removes much of the waittimes placed on the enemy units. This makes the ai attack earlier and with greater strength, and resupplies its attack force quicker. I personally have beaten it, but it is very hard. For those that find either version too much, an easy version has been added as well - but even it is no cakewalk.

There is a major (in my opinion!) mission bug in every mission included in TA:

The ai will not automatically build MOST TA units added after the very first v1.0 release! Strangely, there are a handful of units that don't have this problem. These are:
armscorp, cormabm, cornecro, corplas, corscorp, cortruck

Probably can be built by the ai as well:
armscab, armss, armgate, corgate, corbuild (but only the ai would be able to build it!), and both the arm and core beac and dev1.

NOTE: Normally most of these units aren't buildable, but once a download tdf file is added to build them the ai WILL build them!

I am hard at work to make it so the ai can autobuild *ANY* units in TA in missions, even if they're 3rd party units. This is *IF* the ai is allowed to build them at all. (via the useonly files.) So, if the unit is grayed out the ai cannot build it.

Minor Map changes:

I moved around a few of the starting locations on some maps because of conflicts -- either the commander starts out "stuck" on/in the scenery, or starts very close to other commanders.

I did this on these maps:

Painted Desert (w/ more than a 4 player start, players 1 and 5 were close together. Player 5 is now located in the center.)

Town and Country (one player starts out embedded in a car)

Evad River Confluence (tidal strength has been increased from 0 to 15 -- this is supposedly a river isn't it?)

New control options!

These new CTRL hotkey shortcuts are designed to eliminate some of the micromanagement in controlling all your units:

CTRL+G - ground units (armed ground units only)

CTRL+H - hovercraft (armed hovercraft only) (Hovercraft are *NOT* selected with CTRL+G)

CTRL+L - Laser towers + other fixed defenses (but not AA towers!)

CTRL+N - Naval units (armed ships only)

CTRL+Q - AA towers (Missile Towers, Flakkers, Naval Missile Towers)

CTRL+T - Torpedo Launchers (fixed versions only)

CTRL+X - superweapons such as antinukes, nuke silos, and berthas (LRPC's)

NOTE: Pelicans are selecttable with both CTRL+G and CTRL+H.

NOTE: The crawling bombs cannot be selected with CTRL+G, because their "weapon" is their death.

NOTE: Decoy commanders are not selecttable with either CTRL+G or CTRL+W, but are selecttable with CTRL+B.

NOTE: Minelayers, Arm FARK, and Core Necro are selecttable with CTRL+B.

The "X" key is now a hotkey shortcut for nuke silos and antinuke silos build menus. It will repeat the last mouse build command for that silo - to either build 1 missile or to remove 1 missile from the build que. It can be used with the shift key to build/remove 5 missiles at a time - hold down shift+"X" to queue up many missiles quickly.

The Arm Annihilator and Core Doomsday Machine now have an On/Off switch. When set to "Off", the turrets act as normal - opening as necessary when an enemy comes into range. When set to "On", the turret remains in an always open state. Changing from "On" to "Off" causes the turret to immediately begin its close-up sequence regardless of what the turret is doing. The turrets need to be set to "HOLD FIRE" to remain closed and "OFF".

(Sadly, we're still having problems with the Annihilator and the Doomsday Machine -- when set to "OFF" and fire at will and something comes into range, they turn themselves "ON" and remain "ON".

Also added to the Core Doomsday Machine is the "D-Gun" option. This does NOT fire a D-gun but instead fires the Core Doomsday Machine's 3rd weapon -- its red (LLT) laser. By using "Attack" in combination with the "D" key

(using shift), the DDM can fire at multiple targets at once using different weapons on each.

The Core Krogoth also has the "D-Gun" option to fire its 3rd weapon, its arm-mounted Gauss Guns.

Category changes in unit FBI files:

Morty is now treated as a LEVEL2 unit in its FBI file.
Both the Arm Samson and Core Slasher are now LEVEL1 units -- not LEVEL2 like they were.
Thuds can no longer be selected with CTRL+P.
Most/all units have ALTERED category information in their FBI.
Any units with radar, sonar, or jamming ability are now selectable with CTRL+R.
The Targetting Facility can now be selected with CTRL+R.
ALL Underwater units now have the UNDERWATER category in their FBI.
These categories have been totally removed:
ANTISUB, BEACON, BOMB, CARRY, defensive, DEV, extractor, HOVER, JAM, KAMIKAZE, KBOT, LEVEL, LEVEL10, MINE, MINELAYER, PARAL, PHIB, RAD, REPAIRPAD, SCORP, SHIP, SONAR, SPECIAL, SPY, STEALTH, STORAGE, strategic, TANK, TPORT, VTOL

Although this seems excessive, the categories tell the ai what category each unit is in. Since this information was not used by the ai (or did not appear to be used), it was removed.

These categories have been added: CTRL_G, CTRL_H, CTRL_L, CTRL_N, CTRL_Q, CTRL_T, CTRL_X, NOTSTRUCTURE
CONSTR and PLANT were added back because ai profiles DESPERATELY needs this information!

TORP now takes the place of ANTISUB
CTRL_H now takes the place of HOVER
CTRL_G now takes the place of KBOT -- selects all ground units with weapons
CTRL_G now takes the place of TANK -- selects all ground units with weapons
CTRL_L now takes the place of DEFENSIVE -- selects all Laser towers + other fixed defenses
CTRL_M now takes the place of MINE
CTRL_N now takes the place of SHIP -- selects all naval ships with weapons
CTRL_Q now takes the place of SPECIAL -- selects all AA towers
CTRL_R now takes the place of JAM
CTRL_R now takes the place of RAD
CTRL_R now takes the place of SONAR
CTRL_T now takes the place of TORP -- selects torpedo launchers
CTRL_V now takes the place of VTOL
CTRL_X now takes the place of strategic -- and selects antinukes, nuke silos, and bertha weapons.
Because the remaining categories make decent substitutions, the originals are not needed.

Units ai behavior is improved with appropriate badtarget lines in their FBI.

Both Light Carriers have an added "ENERGY" line in their categories in their FBI file.

The Geothermal Plant, Energy Storage, Underwater Energy Storage, Fusion Reactor, and Underwater Fusion Reactor have an added "METAL" line in their categories in their FBI file. Although this is incorrect, it should encourage the ai to build them more often and will have no effect on the player. Also, the Moho Metal Makers no longer have the "METAL" line in their categories in their FBI file. This is to partially overcome the ai's tendency to build moho metal makers when it has nothing but solars to power it and will also have no effect on the player.

Core Slinger (CORAH) had BadTargetCategory=VTOL;
Arm Swatter (ARMAH) had BadTargetCategory=VTOL;
even though they're ANTI-Aircraft hovercraft!

Core Thunderbolt, Floating Heavy Laser Tower (CORFHLT) had
BadTargetCategory=NOTAIR;
but it's HARDLY an AA weapon!

Core Shadow (CORSHAD) had NoChaseCategory=UNDERWATER;
but it's very unlikely that it'd ever find an underwater target to chase
in the first place.
(All bombers have this line!)

Note that +SHOOTALL overrides the bad target settings, allowing units to autotarget resources and the like.
Also, bad target setting have been changed so units don't shoot at c-units and factories by default.

The BadTargetCategory= line seems to be "overall" for *ALL* the weapons on the unit. (But is sometimes ignored!)
The NoChaseCategory= line seems to be for whether the unit will chase a unit it's told to attack if it moves out of range.
The wpri_BadTargetCategory= line seems to disallow the FIRST weapon of that unit from targetting certain enemy types. It is the GREATEST deciding factor on what the unit will target quickly and what it will ignore if other units are around.

For missile weapons with AA uses, what seemed to work best was:
wpri_BadTargetCategory=NOTAIR;
BadTargetCategory=NOWEAPON;
This means firstly, it would ignore units not flying (NOTAIR) and out of the flying units it would prefer units with weapons. The other way around made them too likely to shoot at anything with a weapon -- whether it was flying or not!

COB script file changes:

The Core Spy no longer "runs" while being built.
Both sides' Construction Kbots (both Basic and Advanced), Core Necro, and Arm FARK should almost never have guarding and building problems now.

Almost *ALL* buildings that close when damaged were not becoming armored when closed - DESPITE having a damage modifier line in their FBI file that says that they should be! This happens to the Arm Annihilator, Arm Advanced Radar (although it lacks a damagemodifier line, since it's closed it should be slightly tougher - so I added a damamgmodifier of 0.5, this

will have little effect because it is so fragile), the Core Doomsday Machine, the Moho Metal Makers, both side's Targeting Facilities, and both side's antinukes were *NOT* more resistant to damage while in a closed state. Antinuke silos are fragile and easy to kill because of this bug. THIS has been fixed!

To make matters worse about this failure to armor bug, the Arm Advanced Radar, both side's solars, both side's sonars, and both side's moho metal makers incorrectly OPEN and activate when shot -- even when not COMPLETELY BUILT! This makes them easier to kill. This has been fixed as well.

The Core Viper pop-up HLT had a scripting error that caused it to not leave a corpse, this has been corrected. The corpse metal value has also been reduced to less than the metal cost of the Viper.

A few multi-weapon units have an either-or approach to using their weapon. The Arm Warrior for instance will fire either its cannon OR its EMG but seldom both.

The fixed Arm Warrior attempts to use both weapons at once, if in range. Even with the original unit, this could be done by using "Hold Position" and "Fire at Will" and ordering the Warrior to move near an enemy instead of ordering it to attack the enemy - this fix just removes the micromanagement involved. This was done with a COB fix.

Both side's Flakker ships have lost their second missile weapon, leaving them with 2 weapons -

weapon 1: their first missile weapon

weapon 3: their flakker weapon

The flakker weapon had to be made weapon 3 instead of weapon 2, or it would be force-fireable (which would make it able to destroy almost any unit not strictly immune very quickly.)

This means the flakker ships should shoot their flakkers more often. This was done with a COB fix and an FBI change. The reload time on the remaining missile weapon used on Arm Flakker ships (and also reused on Core's Flakker ships as well) has been halved, thus making it roughly equal to the 2 original missile weapons it replaces.

The Core amphibious K-Bot, the Gimp, should now use both of its weapons at the same time. Just like the Arm Warrior, the Gimp could use both of its weapons at once - but only if not ordered to attack a specific target. This should help make the Core Gimp more effective against the Arm Pelican without requiring changes on either of their stats. This was done with a COB fix.

The Arm Maverick and Arm Zeus will now take considerably longer (about 10 seconds) to put their guns away -- this will decrease the likelihood that they are "disarmed" during intermittent fighting.

The Core Battleship's laser turret animation has been corrected so the 3rd barrel rotates into position.

The Arm Wind Generator's blades have now been changed to spin in opposite direction from the originals. This better matches the wind direction as seen by which way smoke blows.

The Arm Jethro, Arm Pelican, Arm Phalanx, Arm Samson, Core Copperhead, and Core Slasher have modified unit COB scripts to make them less likely to ignore enemy aircraft flying past them.

The Arm and Core Scorpion has a simplified COB script.

Almost all Hovercraft have modified unit COB scripts, so they won't leave blocking wreckage in the water. Instead, they will leave passable wreckage. While on land, they can still leave blocking wreckage. SOME of the hovercraft scripts also have corrected explosions.

Arm Stunner EMP silo and Core Nuke silo both have a COB script modification to prevent the problem the ai has when using this silo. Previously when the ai used these launchers, conditions could occur which caused the missile to explode inside the silo instead of launching!

If I included a COB file in TA Bugfix, it definitely makes some changes over the original -- with the exceptions of the 6 extra Cavedog units, their COBs were needed to work.

Core Necro changes:

Changes made from the original Necro:

- 1.Name change from "Ressurrection Kbot" to "Necro".
- 2.Added `builddistance=30;` to the `necro.tdf` file, so it doesn't have to be touching units and wreckage to repair them.
- 3.Removed bogus attack button information in the `necro.tdf` file.
- 4.Fixed necro's unit script so that the guard factory bug (where the necro gets "STUCK" repairing units built by the plant even after they've left the plant) happens less frequently. (Total elimination of this bug is nearly impossible though, because what triggers the bug seems to be the interaction of many scripts at once: factory, necro, and unit being built.) This bug *ALSO* affects any repetitious nanolathe animations, even back-to-back resurrections that don't require movement.

Why I made these changes to the Necro:

- 1.The Necro, which was obviously found and released as it, has the bug of not even being called the Necro.
- 2.The Necro could not reliably repair units because of a missing `"builddistance=###;"` in its `cornecro.fbi` file. Any number between 20 and 100 could be used for `builddistance`, but I chose 30 after talking with Boneyards sysops about how the Core Necro affects game balance. If I gave it a value of 60, like the Arm FARK, it could be abused by making it possible to guard a single adv. air plant with 10 or more Necros!
- 3.And to further confuse matters, the Necro has bogus attack information in its `cornecro.fbi` file!
- 4.The Core Necro has the same build/assist scripting problems as the Core Adv. C-Kbot -- which means other than the Decoy Commander (which is a very weak construction unit) Core does not have a reliable construction unit that is built by its Advanced K-Bot lab.

The results of these changes:

With a too-large `builddistance`, the Necro would be able to do everything the Arm FARK could AND resurrect units as well! I tried to avoid this by making the Necro's `builddistance` tiny -- even shorter than most Level 1 construction units (which normally have a `builddistance` of 40.)

Core's Necro is still only a so-so Arm FARK counter. Although it costs less metal than the FARK, it costs 3 times as much energy, is built 40% slower, has far less armor (paper thin, in fact), doesn't make any resources (the FARK makes +0.5m/s and +17e/s), would assist factories at a slower rate than the FARK, cannot be used to MASS-assist a single factory like the FARK can, and has a considerably slower top speed. But at least with this fix, it can resurrect and repair correctly unlike the original. And if you don't have any wreckage around to rez, it can even reclaim rocks and trees (while on Patrol) and repair already-living units (while on Patrol or ordered to repair) -- which is something it previously had great trouble doing! Only 1-3 Necros can get close enough to a single factory to assist in building mobile units, and then they often block the factory's exit. After all these changes, the Necro goes from being a micromanagement pain to use (because it was always failing to complete orders and would sit idle) to easy and FUN to use. This gives Core a workable but LESSER counter for the Arm FARK. (in my opinion)

Antinuke silo changes:

I've added a weapons line "toaironly=1;" to the antinuke missiles for the Arm Protector and Scarab and Core FMD and Hedgehog so that they will not open and attempt to shoot at anything other than aircraft that hit them. This negated their 0.5 damage modifier they gained for being closed and made them TWICE as easy to destroy. Sadly, they still open stupidly when aircraft shoot them - but I could find no weapons-modifying line that could tell them to ignore attacks by aircraft. So I settled for the partial fix instead of none at all. I also made it so antinuke silos should only stay open for 10 seconds after firing a missile. This keeps the "window of opportunity" to destroy them while they're open and more vulnerable reasonably small without interfering with their standard operations.

Originally, if the Core Antinuke silo (CORFMD) is ever nuke-flooded -- and the silo survives -- the silo locks up and refuses to ever fire at nuke missiles again even if it has missiles in it.

BEFORE my changes, ALL antinuke silos can temporarily overload and quit firing if 5 or more nukes are fired at them at once. (This is what "nuke-flooded" means.) Once ALL nukes have impacted or are shot down, they might resume firing. The antinuke's animation scripts cannot keep up with how fast the antinuke fires, causing the antinuke to refuse to fire until there are no more incoming nukes and resets its variables. A "workaround" for the problem is to put the antinuke on hold fire, resetting its script, then back on fire at will -- which should cause it to resume firing.

To get the antinuke silo's animations to match their actions, I had to slow down their rate-of-fire slightly and both speed up and simplify their animations, especially for Arm's antinuke.

Arm's antinuke silo's rotary launcher was much too slow for the rate-of-fire needed in the event of a nuke-flood, and did not match its apparent rate-of-fire. Originally, it ignored the position of the rotary launcher and fired multiple antinukes almost instantaneously, while the rotary launcher would continue to rotate erratically -- sometimes even reversing direction! Now, Arm's antinuke silo loads 3 antinuke missiles on 3 separate launch rails (in a triangular configuration) and has to close and

rearm after every 3 shots. It does this quickly enough to seldom cause problems.

Core's antinuke silo has to close and rearm after EVERY shot, but it rearms faster than Arm's antinuke silo so both should be balanced yet still retain originality.

The mobile antinukes needed only a firerate reduction to remove most of their problems.

After all the antinuke changes, the fixed antinuke silos can fire slightly quicker than their mobile counterparts. This should make fixed antinukes worthwhile versus their mobile counterparts which cost the same.

Arm Stunner EMP silo changes:

The Arm EMP Silo's paralyzation was changed so ANY units not mentioned in its weapon damage file are affected by it - previously, they were immune. This will affect almost all 3rd party units and any new Core and Arm units not specifically mentioned as immune in the damage lists. Unless many 3rd party units are used, this should have little effect on gameplay. I didn't do the same to the Core Neutron Silo because it affects only Arm mobile units - not ALL of Arm's units. Having the Neutron damage ALL 3rd party units would be imbalancing.

Krogoth changes:

Krogoth Duplication Bug:

When a healthy veteran krogoth is captured, self-d'ed, or traded, the krogoth (being captured, self-d'ed, or traded) has 30,000 damage done to it which is LESS than its total health. So the original krogoth (that was captured, self-d'ed, or traded) lives, barely; and a NEW healthy duplicate is created as well. By repeating this cheat, krogoths can be created for free! This fix reduces the krogoth's health to half (to 14,959 which is half of 29,918) and adds a `damagemodifier=0.5;` line in its FBI file, which makes the krogoth have 1/2 the health but take 1/2 damage all the time. A COB file fix was also needed which set the Krogoth to the always armored state.

When the modified krogoth is captured, self-d'ed, or traded the 30,000 damage ignores the `damagemodifier=0.5;` in its FBI file. So, the original is removed when the copy is created. This prevents the appearance of extra, but badly damaged, krogoths. One small but extra benefit is Krogoths will be repairable at double their original rate - so it'll take a construction aircraft about 5 minutes to repair a badly damaged krogoth instead of 10 minutes.

Also to maximize the krogoth's damage potential in combat, its 3 weapons have been moved around. Since the krogoth tends to use its primary weapon over its secondary weapons, it makes good sense to make its primary weapon one it can use in MOST situations.

The weapons are swapped around to:
Weapon1=CORKROG_HEAD;
Weapon2=CORKROG_ROCKET;
Weapon3=CORKROG_FIRE;

The bad target categories are changed to:

wpri_BadTargetCategory=LEVEL1;
wsec_BadTargetCategory=NOTAIR;
wspe_BadTargetCategory=CTRL_V;
BadTargetCategory=NOWEAPON;
NoChaseCategory=CTRL_V;

The weapons will still fire at bad targets, but only if ordered or if there are no other targets available.

Also added to the Krogoth is the "D-Gun" option. This does NOT fire a D-gun but instead fires the Krogoth's Gauss guns on its arms. When "Attack" is used in conjunction with the "D-Gun" option (via the shift key), the Krogoth will fire all its weapons at once -- possibly even at different targets! When using "Attack" alone, the Krogoth will only try to get within range of its head-mounted Blue Laser (700 pixel range) from the target.

The Krogoth can now be selected to built in the Krogoth Gantry using the "K" key when the Gantry is selected and the Krogoth is showing on the menu. Page 3 of the Krogoth Gantry also has a LARGE Krogoth build pic. It is linked to the normal Krogoth build pic on page 2, so clicking on one is the same as clicking on the other.

Landmines Changes:

Minelayers are now selectable with CTRL+B (as Constructors) and now have patrol, guard, and reclaim abilities! They are still considerably slower than C-vehicles, nearly as costly, do not make resources on their own, plus they have only landmines on their build menu - but now at least they can clean up the messes their minefields can make!

Landmines can now all have their fire orders changed to HOLD FIRE, RETURN FIRE, and FIRE AT WILL. Why a player would want to is questionable, but since they qualify as a weapon - it makes sense that they have a "safety" feature too. The Nuclear Mines default to HOLD FIRE, but all others default to FIRE AT WILL. While a landmine is set on Hold Fire, it can be manually detonated for maximum damage effect. Or, landmines can be used as cloaked spotters. (Their poor LOS range, cloaking cost, and lack of armor makes landmines poor spotters.)

Originally, landmines were only resistant to damage caused by landmines of the exact same kind. This has been changed so that all landmines of one side are resistant to other mine's damages of the same side. The only exception is the nuclear mine's explosion which kills ALL landmines nearby -- and just about everything else too! Arm mines affect Core mines and vice-versa just fine. This is done to encourage the use of dissimilar mine types in a tightly packed minefield.

Previously, the Arm Precision Mine (ARMMINE5) had 2,000 armor while ALL other mines had only 100 or 200 armor. Because the mine is extremely tough for such a cheap price (which is unbalanced), I presume an extra unintended 0 was inadvertently put there. The armor has been reduced to 200 -- still TWICE as much as all other Arm mines. (Sorry Arm players, there goes your "killer" strat.)

Both side's Nuclear Mines can now be built underwater. However, since the minelayers can drive only into shallow water, a naval construction unit is required to build them in deep water. So, Nuclear Mines have been added to the Advanced Construction Sub's build menu. (Page 3, position 5) This should be useful as a naval defense against Fibbed subs and Leviathans, which previously lacked good counters. Nuclear Mines have a tiny LOS radius, no sonar, damage both friendly and enemy units, are one-shots, and cost a considerable amount of resources -- making this change difficult to abuse.

LRPC Changes:

All the LRPC's (Arm Big Bertha, Arm Vulcan, Core Buzzsaw, Core Intimidator) have been modified so the direction they point when built isn't always straight down. Now, they can be pointed to the left, down, or to the right. I tried making them so they could point up also, but something caused them to be unable to fire when they were built facing up.

I added an On/Off switch to the Arm Vulcan. Consider it a single-shot/full auto fire switch. While On, the Vulcan fires at its normal superfast drain-your-energy-dry rate. While Off, the Vulcan fires at the same rate as an Arm Bertha -- making it likely to be mistaken as one by its rate-of-fire and its sound. It also doesn't require 8+ fusions to sustain fire when firing while Off.

The Core Buzzsaw doesn't get this On/Off feature because I couldn't get its unit script to work with it. Plus, it costs almost 10,000 metal cheaper than the Arm Vulcan so it's "balanced" not for it to get this minor ability.

Bugfix for the Arm Pelican:

Despite the many and much heated debates about the Pelican being too powerful in TA, I still prefer to think of the Pelican (and Flash as well, but that's a different story) as a legitimate unit in TA. However, missiles aimed at the pelican when it is on the water tend to hit the water instead of the Pelican unless fired from nearly point-blank range. This is a bug or is at least unintentional on the designers part. All ships are targetted correctly by missiles, SO if pelicans are in fact meant to be a ship (but then why the Floater=0; line? -- all ships have a Floater=1; line!), subs should be able to target them. However, subs cannot target them. (But subs can force-fire past them, hitting the Pelicans. The same trick can be done to all Hovers.) Pelicans also have a canhover=1; line which ALL other hovercraft have as well - which proves that Pelicans are hovercraft and not ships. Pelicans currently use waterline=9; which is ODD because all hovercraft lack a waterline=9; in

their unit FBI file -- which due to its absence defaults to 0. So obviously, part of the Pelican is hiding below the waterline -- which is where missiles are aiming in their attempt to hit them. However 1 other fact remains which causes the behaviour of even partially underwater Pelicans to not make any sense: Pelicans take NO DAMAGE from acid on acid maps! As such, there is a simple fix for the WHOLE problem: set the pelican's waterline to 0 so it isn't partially underwater.

Pelicans may appear to be hovering over the water with this fix when viewed at 640 x 480. I couldn't tell any difference myself except that missiles hit the modified Pelicans more reliably. HOWEVER, any missiles fired from near or at max range that might miss an ordinary hovercraft could well miss the Pelican as well. This change does nothing else to the Pelican's unit stats, nor does it change how any other weapon is fired at them. It only allows missiles and direct-fire rockets to hit Pelicans as they would any ship or hovercraft in the same conditions.

Bugfix for ALL Hovercraft:

(This idea was taken straight from BSR's UH.)
Hovercraft leave passabled debris when destroyed on the water. (That is IF they leave debris!) The debris created is also left on land IF the hovercraft is badly destroyed. The debris is roughly half as valuable and half as tough as the "shell" wreckage is.

Arm Anaconda's corpse has been increased in value from 105metal to 205metal -- this is a probable typo, because Core's equivalent hovercraft, the Core Snapper, has a corpse value of 224metal! Normally, corpses are worth roughly 60-80% the metal value of the unit it came from -- but this corpse originally was only worth 38.6% the value of the original Arm Anaconda.

This required a new COB script for every hovercraft in TA, as well as changes to the 4 corpse files.

Bugfix for Naval Defensive Structures:

Both side's floating HLT's and Core's floating missile tower's waterlines have been changed from 3 and 4 respectively to 0.3 -- this will allow them to better shoot over naval dragon's teeth while still allowing subs to target them. It will also allow guided missiles targetted at them to hit them from a longer range, previously the missiles would hit the water like the Pelican bug (see above).

Bugfix for the Naval Dragons Teeth:

Naval Dragons Teeth are made to sit deeper in the water to allow naval HLT's and missile towers to shoot over them. Originally, naval HLT's and missile towers could not do so reliably -- which makes countering Pelicans all the more difficult! This would also act as slightly better torpedo netting against subs.

Bugfix for the Arm Penetrator:

Originally, Penetrators could recenter their turret quicker than the turret turns to aim! It's also the slowest-turning turret in the game. Why not aim the turret as fast as it can recenter in the first place? It won't be around to recenter if it can't acquire the target! So, the Penetrator's turret has been changed to a faster-turning (but still slow) turret with the same turret speed as its larger counterpart, the Arm Annihilator. The Penetrator's weapon lacks the range and damage of the Arm Annihilator and costs nearly as much energy to fire -- these characteristics I have not changed.

EMG weapon changes:

I edited the Flash tank's, Brawler's, and Warrior's EMG weapons to not make any impact noise (soundhit=; in their weapon file) -- this should decrease lag caused by these weapons immensely. However, you should still hear them firing *JUST* fine.

I changed their sound behavior with:

```
soundtrigger=0;
```

which makes only 1 firing sound per burst of shots.

An altered sound file was made which plays the sound of an entire burst of shots as a single wave.

In the 3 EMG weapons (Flash/PW, Brawler, Warrior) files, I replaced:

```
startsmoke=1;
```

WITH:

```
startsmoke=0;
```

```
endsmoke=0;
```

Those tiny smoke puffs were very hard to see on 640 x 480 - on anything higher, nearly impossible.

Also, I removed the wave for weapon impact (changing it to):

```
soundhit=;
```

All the impacting pictures for EMG have been removed as well, they were also a major source of lag. Unfortunately, the impact smoke animation is still visible -- that can only be removed by using +SFX while ingame.

That is all I have done for the lag on emg units.

Core Pyro changes:

I removed the wave for weapon impact (changing it to):

```
soundhit=;
```

These shoot 17 times every 1.2 seconds - which in theory would make them a bigger lag-producer than EMG weapons!

This has been changed to firing a burst of 5 shots over almost exactly the same time period (0.7 sec instead of 0.68 sec) and doing the same total damage of 170 as the original to all units but pyros, but doing 1 extra point of damage to them for the whole burst - 35 damage instead of 34 like the original.

Crawling Bomb changes:

Crawling bombs have their self-d countdown reduced to 2, the same as landmines. This will *NOT* speed up the countdown of a flying bomb so it should have little effect on the game. This allows crawling bombs on the ground to be a slightly more viable tactic without changing a variety of their statistics. Just as before, a crawling bomb can be told to attack the ground just in front of it to cause an instant explosion without the 2 (or 5) second self-d delay!

Arm Fibber changes:

Not much else can be done about the Arm Fibber, it's just one unfair unit because of the way sonar jamming works. Sonar jamming gives the equivalent to underwater cloaking on ALL units within the jamming radius, and in the case of the Fibber this costs NO energy! (Even mincloakdistance=100; does not negate this cloaking-like effect.) The ONLY change done to the Arm Fibber is if the Fibber somehow takes damage, its sonar jamming effect is temporarily disabled for about 5 seconds. This is no different than a radar that turns off briefly when hit, and should help slightly in "Fibber war" games.

Core Leviathan Super-Sub changes:

SoundCategory=ARM_SUB;
has been changed to:
SoundCategory=CORE_SUB;
since it is after all a *CORE* sub!

It turns out that this sub partially sticks up out of the water if it moves into water that's too shallow. This will not be changed, and should be good reason to keep your Leviathan subs in DEEP water!

Cruiser and Destroyer Changes:

On ALL ships with Depth-Charge, (This is the Arm and Core Cruisers and Destroyers -- 4 ships in all) the depth-charge weapon has been moved from weapon 2 to weapon 3. This will allow these ships to hold position and fire their guns at range when told to attack a distant target, without them attempting to move within depth-charge range. The depth-charge will

still automatically shoot at subs in range, but can also be manually targetted at legitimate targets. Unfortunately, Fire at Will *AND* +shootall has to be activated before they will shoot at weaponless underwater buildings. This is how it's always been, so there's no real change except Destroyers and Cruisers will no longer "charge" the shore trying to get in depth-charge range of inland targets.

Missile Frigate Changes:

These units have less than 1/3 the armor of the cheaper cruisers made by the same factory, yet prove less useful in almost every case. The only semi-useful task they can perform is base bombardment, but because they are terribly slow, unmanueverable, and poorly armored the only base they're likely to bombard is your own when they attempt to shoot down enemy aircraft with their heavy rockets.

To combat the lack of armor, I did not change the ship's armor stat but rather looked at the ship's design. Because the heavy rocket launcher occupies about 1/2 the ship's length, I gave the missile frigates a 1/2 damage modifier when it is closed and not exposing the ship's innards and live heavy rockets on the launcher. This makes them just over half as tough as a cruiser when closed.

To combat the friendly fire problems caused by the heavy rockets, I added an On/Off switch to these ships which controls whether or not the heavy rockets will fire. On Core's Missile Frigate, this also has the unfortunate effect of turning off its radar, so I made the radar quit spinning while off as well. While set to the "Off" state, the missile frigates will still fire their AA missiles at enemies if any are in range.

Advanced bomber changes:

All the impacting pictures for the advanced bombers laser have been removed, they were a major source of lag when 10+ advanced bombers attacked at once.

Anti-Aircraft missiles changed:

Despite the obvious problem of all anti-aircraft missiles not doing extra damage to all aircraft, especially 3rd party ones, this is not really a bug. The Seaplanes found in TA:CC are an example of how Cavedog tried to work around this problem. They take less damage from anti-aircraft missiles than other Cavedog aircraft, but they have less health so it averages out. The problem is, it looks like Cavedog intended to add them to the special damage lists (and added them to the stealth fighters' missiles) but not to anything else. This is incredibly imbalancing because stealth fighters can destroy them easily - often with only 1 missile hit, while they need 4 to 5 hits to kill 1 stealth. However, they still have a weakness to flakker weapons which do full damage to them. All other weapon types kill them easier too.

I removed the special damage lines for ARMSEAP, ARMSFIG, ARMSEHAK, CORHUNT, CORSEAP, CORSFIG on these weapons:

ARMVTOL_ADVMISSILE	(ID=112)
CORVTOL_ADVMISSILE	(ID=113)
CORAH_WEAPON	(ID=199)

Flakker changes:

It's been pointed out that flakkers are often as dangerous TO your base as it is to enemy aircraft. As a fix, I'm adding a "unitonly=1;" line to all flakker weapons. This will prevent the flakker weapon from detonating when it hits the ground or when it hits DT. Zero-damage lines have also been added to the tall buildings that were typically affected by the flakkers - now if for some reason the flakkers do hit them, the damage the buildings take is zero. The flakker weapon names and weapon id numbers are:

ARMFLAK_GUN	ID=42
CORFLAK_GUN	ID=43
ARMYork_GUN	ID=116
CORSENT_GUN	ID=133

(Note: The other flakker weapons in the game have been absorbed into the above 4 weapons, so all flakkers used in the game are affected by this change.)

This may not be considered a bug, but is a big game balance issue - because what's the point of making an anti-aircraft weapon that destroys your base when it misses? This is like an antinuke that nukes your base if it misses an incoming nuke!

Zero-tolerance bug:

MANY units in the game have a terrible target-tracking problem because of an overprecise TOLERANCE=###; line in their weapons file. More specifically, for many original TA units (but not for any of the downloadable add-on units) there is no TOLERANCE=###; line at all! This means the turret/torso of the unit had to be precisely aimed on the center of the target before it would fire its weapon. A good example of this is the commanders' lasers, Arm Zeus, or Core AK. They would track a target but seldom fire at it if the target moved by them quickly. I would NOT have considered this a bug, except for the fact that Cavedog has included a TOLERANCE=###; line for all but 1 of the downloadable units released AFTER the original game came out. I added a low tolerance to all the tolerance-lacking weapons.

Weapons Changes:

COMMANDER_BLAST has been altered to be a nuclear missile weapon that has a 5 minute buildtime with a massive cost - 2x the metal and 3x the energy as

the regular nuke weapon. This will have NO effect on the game, but allows for the design of 3rd party units with this "new" oddball weapon. CRAWL_BLAST has been altered to be a medium-ranged mini nuclear missile weapon with a cost slightly greater than the Arm Stunner missile weapon. This will have NO effect on the game, but allows for the design of 3rd party units with this "new" oddball weapon. ARM_DISINTEGRATOR will no longer start fires (firestarter=0;) -- the D-gun should destroy trees, not burn them! Both the COMMANDER_BLAST and the CRAWL_BLAST have been set so they will destroy trees (firestarter=0;) instead of setting them on fire. This will make the Crawling Bombs good for clearing trees -- and almost nothing should survive a commander explosion!

5 explosion weapons have been turned into true weapons:

Weapon ID	Name	Range	Damage	Area of Effect	Cost Metal	Cost Energy	Path	Reload Time	Special note
Used by:									
BEFORE:									
37	CORMINE3(M-303) Core Mine #3	480	700(45)	160	-	-	B	3.6	10% edge effect
110	MEDIUM_UNIT Medium Unit self-d	480	250	95	-	-	B	3.6	
185	ARMINE2(Area) Mine #2	480	600(30)	200	-	-	B	3.6	5% edge effect Arm
204	SMALL_UNIT Small Unit self-d	480	200	75	-	-	B	3.6	
207	MEDIUM_BUILDINGEX Medium Building explosion	480	100	105	-	-	B	3.6	
213	COMMANDER_BLAST Commander explosion	380	9999	950	-	-	B	3.6	75% edge effect
214	CRAWL_BLAST Crawling Bomb self-d	480	2500	556	-	-	B	3.6	
AFTER:									
37	CORMINE3(M-303) TAWF010b_weapon	780	700(45)	160	-	-	L	6.48	10% edge effect
110	MEDIUM_UNIT Dragon	2400	250	95	-	-	L	2	
185	ARMINE2(Area) ADVSAM	2100	600(30)	200	-	-	L	10	5% edge effect
204	SMALL_UNIT ARM_MORTAR	850	200	75	-	-	B	2.8	
207	MEDIUM_BUILDINGEX dart	600	100	105	-	-	L	1	
213	COMMANDER_BLAST Super-nuke?	32700	9999	950	6000	600000	V	300	75% edge effect
214	CRAWL_BLAST nukes? (V= Vertical Launch)	4000	2500	556	1300	52000	V	130	Medium-range Mini-nukes?

These weapons will still retain their original ability as explosions, but can also be used as weapons on 3rd party units.

ADDED 3do and sound files:

```
[ARM_MORTAR]
    model=mortarshell;
    soundstart=Mortar1;
[Dragon]
    model=Dragon_miss;

[ADVSAM]
    model=ADVSAM;
    soundstart=launch;
```

```
[dart]
    model=dart;

[TAWF010b_weapon]
    model=rpack;
    soundstart=shoot;
[CRAWL_BLAST]
    model=advmiss;
```

After I add the appropriate sounds and 3do files to TA Bugfix, everything needed to use these weapons has been included.

Admittedly, it may seem a little strange for a unit to fire SMALL_UNIT, MEDIUM_UNIT, MEDIUM_BUILDINGEX, ARMMINE2, or CORMINE3. But confusion is the price to pay for saving 7 weapons id's.

The 7 weapons conversions plus extra 3do and sound files adds less than 100kb to the total filesize.

Accumulating scars bug:

After many hours of battling on a map, many of the map features are either removed or destroyed - such as trees or recycleable rocks - leaving behind a "damaged terrain" scar. This is also true of dead units and dragon's teeth which are recycled. The accumulation of many of these scars causes some slowdown in the game. In the case of dragon's teeth, the slowdown could well be excessive - plus revealing where dragon's teeth walls once were to all players. Currently, I've left the other scars but removed the "dead" dragon teeth scars.

Also, it was strange that although there is no way to destroy a scar - the scar has a scar corpse as well! This unnecessary scar corpse has been removed. ALL the remaining scars do NOT have a corpse.

Corpse Changes:

All corpses and heaps (in alphabetical order by unit abbreviation) are in 4 master files (Arm corpses, Arm heaps, Core corpses, Core heaps) -- this would also aid in discovering unit conflicts. The other files are of zero-length to force TA to forget all the incorrect and redundant entries in cdata.ccx and total1.hpi.

FIXED: The Core Fortitude Missile Defense (CORFMD) incorrectly uses Arm's Anti Missile Defense's (AMD's) corpse (armamd_dead) instead of its own corpse (corfmd_dead) which is in TA. This isn't too big a deal, unless you're using resurrector units - in which case corpses don't always turn back into the unit that they came from!

FIXED: The Core Goliath (CORGOL) uses the Core Diplomat's (corvroc_dead) corpse. This was why previously a dead goliath was worth less metal than a dead Reaper! This has been replaced by the considerably more valuable goliath corpse (corgol_dead), which is also in the game.

FIXED: The Arm Retaliator nuke silo's corpse was worth almost 2,000 more metal than it cost to build the silo. This has been changed to about 800 metal -- roughly the same value as Core's nuke silo's corpse.

FIXED: The Arm Seer (ARMSEER) and the Core Informer (CORVRAD) used the same corpse as the Arm Jammer, this has been changed to ARMSEER's corpse and CORVRAD's corpse, respectfully.

Unit Footprints, Corpses, and Heaps have been changed so that they are the same size, what few units this affects have had their corpses reduced by a small amount. The only effect this may have on the game is slightly more corpses can be packed into an area without overlap -- meaning that a few extra corpses will be left behind after a battle that would otherwise have been covered up and unreclaimable. (This makes the Core Necro and Arm FARK more useful for when they go on resurrection and reclaim patrols!)

Build Menu Changes:

Both Arm and Core's AWACs (Arm Eagle and Core Vulture) have been moved from page 3, position 0 of the Adv. Air Plant to page 2, position 5. This puts ALL Cavedog advanced aircraft on one build menu.

The Arm Panther has been moved from page 4, position 0 of the Arm Adv. Vehicle plant to page 3, position 5. Previously, it was needlessly on a menu page by itself.

Nuclear Mines (which can now be built underwater) have been added to the Advanced Construction Sub build menu. (Page 3, position 5)

The Seaplane Platform is now buildable on the Advanced Construction Aircraft build menu at Page 4, position 4. (This is the same build position used by the Core Advanced Construction Kbot to build the Krogoth Gantry, another Level 3 factory.) Because the Seaplane Platform is an advanced factory that builds aircraft, some type of construction aircraft should be able to build it. (This follows the general build rule that advanced factories are built by less-advanced construction units of the same type.) Since the Seaplane Platform is a Level 3 factory, the Advanced Construction Aircraft (a Level 2 unit) should build it because the Advanced Construction Aircraft is the only construction aircraft capable of building Level 3 buildings.

Unit Name Changes:

The Decoy Commanders have been renamed "Commander" so you don't see "Decoy Commander" when you move your cursor over top them in skirmish. You can still use F1 to see a difference. I did not change the Commander build cost to the same as the Decoy Commanders because of a gamecheat which makes Commanders buildable - if Commanders cost no more than Decoy Commanders, it would be more tempting to use that cheat.

The Core Mobile Artillery has been renamed "Pillager".

The Core Missile Frigate has been renamed "Hydra".

The Core Resurrection Kbot has been renamed "Necro".

Unit Changes:

Both the downloadable=1; and the ovradjust=1; lines have been removed from *EVERY* Cavedog unit in TA -- they do nothing so have no effect on the game except possibly slowing it down a bit.

Land-based metal maker corpses have been made worth 1 metal so c-units on patrol will automatically reclaim them if low on metal -- previously, their wreckage would block passage but would never be automatically reclaimed. 1 metal is a negligible amount and not exploitable as a cheat because the energy to metal conversion ratio of this "cheat" is nearly 700e to 1m! (Over 10 times worse than using an equivalent amount of energy to power metal makers.)

The units Torpedo Seaplane/s (ARMSEAP and CORSEAP), Seaplane Fighter/s (ARMSFIG and CORFIG), Arm Escort (ARMSJAM), Core Phantom (CORSJAM), Core Adv. Torpedo Launcher (CORATL), Core Leveler (CORLEVL) all use:

```
ExplodeAs=MEDIUM_UNITEX;
```

But MEDIUM_UNITEX is a nonexistent weapon and does no damage to nearby units when they die.

This has been changed to: ExplodeAs=BIG_UNITEX;

The replaced explosion does 1/5th the damage in nearly the same radius as the self-destruct for the above units.

Arm's Moho Metal Maker (ARMMMKR) and Core's Moho Metal Maker (CORMMKR) uses:

```
ExplodeAs=BIG_BUILDINGEX;
```

```
SelfDestructAs=BIG_BUILDING;
```

But both are nonexistent weapons and do no damage to nearby units when they die.

(Because these units have nonexistent weapons for their explosions, a weapon of the same name could be created and used to cheat!)

This has been changed to:

```
ExplodeAs=BIG_UNITEX;
```

```
SelfDestructAs=BIG_UNIT;
```

Core Toaster (CORTOAST): ExplodeAs=LARGE_BUILDINGEX;
(probably still works, but just to be safe...)

armthover.fbi typo:

ARMTHOVER has a brakerate of 0.0018, BUT CORTHOVER has a brakerate of 0.017!

Therefore, ARMTHOVER has a typo in it - an extra 0 that wasn't intended.

armaser.fbi typo:

ARMASER has a brakerate of 1, BUT the Core equivalent has a brakerate of 0.12!

Therefore, ARMASER almost certainly has a typo in it - so I reduced its brakerate to 0.2!

The missions-only unit ARMSCORP has a goof-up in its FBI file:

```
ItalianDescription=;Scorpione
```

This has been changed to:

```
ItalianDescription=Scorpione;
```


The Arm Energy Storage has an error in its yardmap. It is a 4 x 4 unit with a yardmap of:

```
YardMap=0;
```

instead of:

```
YardMap=oooooooooooooooooooo;
```

like Arm Metal Storage - which is also a 4 x 4 unit. This was a simple cut-and-paste fix. ;)

Arm's Hawk's 2nd missile has an error in its data, it has:

```
weaponacceleration=130;
```

instead of:

```
weaponacceleration=130;
```

(notice the "0" instead of a "0")

This has been corrected (but now only affects the old_weapons.ufo file.)

All the aircraft plants (basic and advanced) have had their yardmaps changed so ground c-units don't block production as badly (but can still occur!) while guarding them.

Individual unit _gadget.gaf build menu picture files have been added to the Amins directory for many units that lacked them.

Cursor.gaf in the Amins directory now have a better unload animation.

(Thank BSR for this one!)

Help menu pictures (gotten by pressing F1 while the cursor is on them)

have been added to the Arm Scorpion, the Core Sea Serpent, and the Core Hydration Plant.

Mobility changes on units:

The Arm Triton (ARMCROC) and Core Crock (CORSEAL) now has climbing ability out of the water (maxslope=30) equal to their in-water climbing ability (maxunderwaterslope=30) -- this is for metal maps like Over Crude Water, where they would often get stuck going in and out of the water.

The Arm Zipper and Core Freaker now has climbing ability identical to most other kbots. (maxslope=15) Previously they could climb only gentle slopes.

The Core Krogoth has been given climbing ability identical to the Arm Bulldog tank. This will allow it to wade into shallow pools of water but will not let it snorkle accross deep bodies of water like fully amphibous units.

Fire Standing Orders and Fire Moving Orders for mobile units have been changed so most mobile units are set to Fire at Will MOST of the time.

GAMEDATA dir (in REV31.GP3) file changes:

Category.tdf file has been altered to reflect the reduced number of different Categories actually used in the game.

Help.TDF is the help file viewed from inside the game. Some of the new commands added are now listed in the help menu. Page 3's last line should tell if TA Bugfix is installed or not.

The LOS.TDF (the Line of Sight tables) file has been altered to include entries out to 640 pixels -- so now far-seeing units such as the Leviathan will reveal what is in the outer edges of their sight radius.

Meteor.tdf (in the Gamedata dir, NOT in the weapons dir) has been replaced with modified meteor file. This is the data used when the +METEOR cheat is used in skirmish.

Moveinfo.TDF contains the mobility attributes of all the units in the game.

The Sidedata.tdf file stores the ai's build menus, which has been corrected. (See AI fixes section for more details.)

Sound.TDF stores the list of soundfiles used by various units. Some unused entries have been deleted.

Weapons.tdf (in the Gamedata dir, NOT in the weapons dir) has been replaced with a zero-length file to mask the one in the old Gamedata dir.

Unfinished work:

These are game bugs that I'd like to address/fix but do NOT know how.

1.AI antinukes that work in skirmish have not been made. I've made an antinuke silo that doesn't use the stockpile=1; line in its weapon data (it is NOT included in this patch), but it still seems to require the gui file and the gaf file (for the antinuke missile itself). Also, the antinuke must build 1 fake stockpiled antinuke missile (on the build menu) before the antinuke will work - even though the stockpile=1; line is missing. The fake stockpiled antinuke missile is never deducted when the antinuke fires.

2.BURSTED BOMBER BUG: Bombers could not have their bombs converted to a burst weapon -- to remove the long-string-of-bombs bug that is overused by many players who think it's a legitimate tactic. Bombers are powerful enough without this "tactic" especially when lag is affecting them. I'm not talking about the multiple-targetting micromanagement strategy where bombers are ordered to bomb another target downrange after their first -- this is little more than selecting multiple targets with shift except it removes the "stupid" stage from bombers where they attempt to either stop and land or fly in a straight line.

Bombers on patrol and fire at will, don't attack targets as they come into the bombers maneuver range. The fix I tried never worked right.

3.The Domsday Machine doesn't always point it's big-blue-laser turret in the direction that the big-blue-laser fires..

4.I tried changing ALL landmines found in the game from yardmap=0; to yardmap=y; - because even units that didn't know the mines were there "knew" to drive around them. The reason I changed it back was because

yardmap=y; causes the landmines to be near-impossible to hit with laser weapons. Only weapons which affect an area when they impact would be likely to damage the mines.

5.The Arm Spider and Arm EMP Stunner Silo problem is that paralysis does not act as a "STOP" command on units that are hit by them. And what's worse, with micromanagement paralysis can be negated entirely on mobile units. A fix for this requires an EXE change which I cannot do!

6.Also, the Core Leviathan may be partially stuck out of the water and trackable like any other ship when it's in shallow water. This I have confirmed but am unable to fix.

7.With the removal of the "commander" category in the ARMCOM.FBI and CORCOM.FBI files, the Commanders are still selectable with CTRL+C, but CTRL+C no longer centers the viewscreen on them. However, the viewscreen can still be centered on them with the "T" key. Sometimes you don't want the viewscreen centered on the commander when you use CTRL+C. It has since turned out that the "commander" category in the ARMCOM.FBI and CORCOM.FBI files has another use as well -- it prevents players from giving their allies (or enemies) their commander in multiplayer games - especially commander dies = ends games! Because this is an important limitation, I had to add the "commander" category back to the ARMCOM.FBI and the CORCOM.FBI files.

Map Tutorials

Terragen Map Tutorial

By Caer

OK, first off I want to make something clear - this is NOT a tutorial for making tilesets with Terragen. It is a tutorial to make an entire map in one go.

WHAT YOU WILL NEED

- The registered version of Terragen if you want to make maps bigger than 2x2. This is because the unregistered version only allows images up to 1280x960 to be rendered.
- Adobe Photoshop 4 or 5, Paint Shop Pro, or basically any image editor that has crop, resize and reduce colour depth tools. Basic stuff, really. Oh yeah - from now on, for simplicity's sake I'm going to say 'Photoshop' when I actually mean any image editor that meets the requirements.
- The Total Annihilation colour palette, or the Photoshop .ACT file.
- Annihilator 1.5. It's so much better than TAE, you wouldn't believe it. Besides, I don't think TAE lets you import an image to use as a map, although I may be wrong. I've only used it about three times.
- A program to read Zip files, for example WinZip, for opening the palette and World files for this tutorial.
- A lot of RAM - 128MB is the recommended minimum, but you can get away with less if you're prepared to sit around waiting for long periods of time.

Now that's sorted, we can begin.

PART ONE - SETTING UP TERRAGEN

For this part, I'm going to assume you have Terragen installed and at least know the basics. If not, install it (duh), and go to the Terragen website and look for some tutorials. The URL is <http://www.planetside.co.uk/terrigen/>. Alternatively, just have a mess about with the various settings and see what happens - most of it is self-explanatory.

The first step is to download and open the basic Terragen world file. This is important, because it sets up the camera and sun position correctly. If you don't want to download a 1.4K file, first turn off both 'Fixed height above...' options, and use the following settings:

```
Position   x: 128   y: -1265.337   z: 5200
Target     x: 128   y: 128         z: 0
```

One thing that you must remember is that these are **Terragen units**, not metres.

Zoom: maximum (32)

Sun

heading: 225

altitude: 30 - 50 (doesn't matter too much)

Disable 'Terrain casts shadows', too. Although shadows make the map look nicer, it looks odd when units move into a 'shadow' area and don't get darker. You may as well disable the 'Clouds cast shadows' option too, although we won't be rendering the sky anyway. Another thing you should consider for later is the shadow and ambient light setting - you don't really want sunset-red shadows on a snow map, do you?

Lastly, you need to open the Atmosphere window and set the 'Density'/'Decay' sliders to minimum, otherwise it will look like your map is being viewed from from 470,000 feet (which it is, actually).

Right then, now that we have that sorted, on to part 2.

PART TWO - DESIGNING YOUR MAP

This part can be a bit complicated, or not depending on how much work you want to put in to your map. This is because it involves a fundamental part of map design - terrain. You really should have an idea of what type of map you want to make by now.

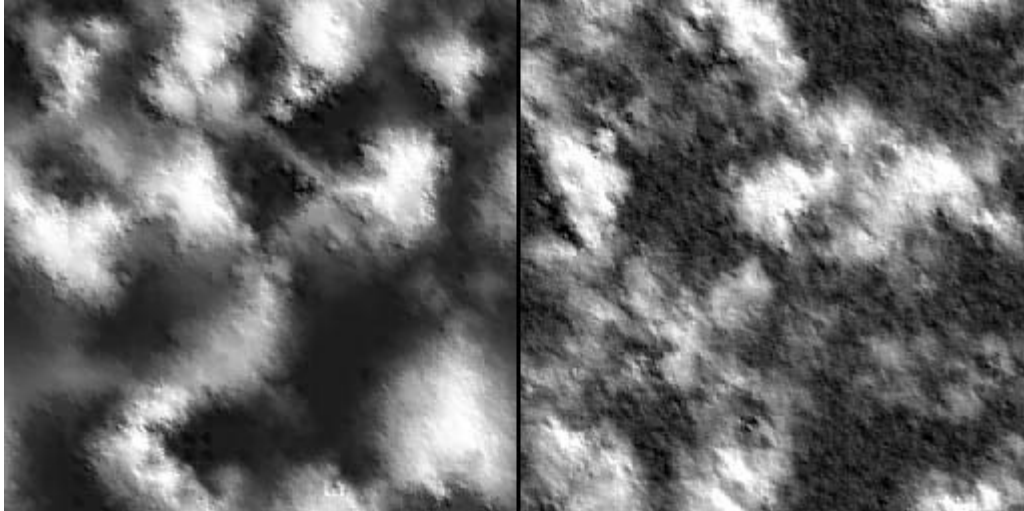
Basically, there are a number of ways to design a terrain:

- Using Terragen's various built-in generation tools
- Using an image editor
- Using another terrain generator, like Bryce
- A combination of the above

Personally I think Terragen generates very realistic terrain, although it can be difficult to get what you want. This is where the import/export option comes in - you can generate a terrain that kind of resembles what you want, export it as a .RAW file, tweak it in Photoshop, then import it back in to Terragen. Alternatively, you could sketch out a rough sample of what you want in Photoshop, import it into Terragen and the generate a terrain based on the existing one (look in the terrain generation dialog).

Before I go on, I had better explain the functions of the various controls in the terrain generation dialog:

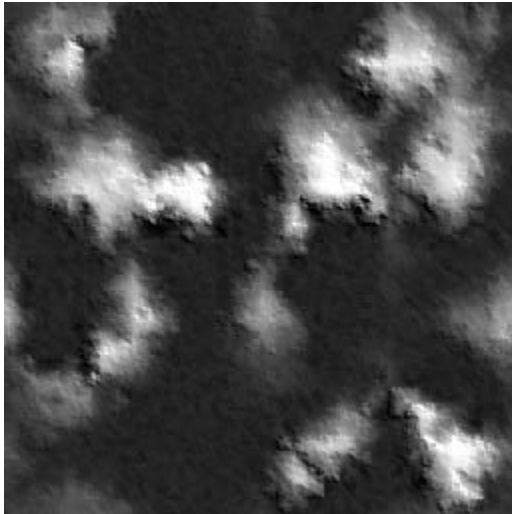
- Method: simply, what algorithm is used to generate the base terrain. Try the different options, and see what happens.
- Action: easy - generate a brand new terrain, or base it on what you have now?
- Status: well, er, it's kind of obvious what this is.
- Realism: kind of complicated. Basically it stops small bumps interfering with the basic shape of the terrain. For example, try setting it to zero and cick 'Generate'. You'll see a very rough looking terrain with lots of variations in height and no real overall shape. Set it to maximum and you get large hills with smoothish edges. This image should explain things a bit:



The first image is with realism set very high.

- Smoothness: this controls how smooth edges are allowed to be. When set to a low value, the resulting terrain has lots of sharp edges, making it look kind of like folded paper. Set this high for nice rolling hills.
- Glaciation: easy one, this. It simply controls how flat the bottoms of valleys are. When set quite high, you will end up with large flat areas and steep mountain sides.

Here's an example of a terrain with the glaciation set quite high:



- Canyonism: another simple one, this controls how flat the tops of mountains/hills are. When canyonism and glaciation are both set to maximum, you get a kind of cliff effect.
- Size of features: pretty obvious - it controls how large your hills, mountains and general landscape features are. One thing you must remember is when you import a terrain, its vertical (altitude) constraints will be from 0.25 to 63.75. This is too high, so adjust the range to go from 0 to 30 (using the 'Modify Terrain' button).

And that's that. Oh yeah, before I continue, I'd like to plug my Terrain View program. Sorry. Terrain View 3D is a really simple program that lets you open a .TER (terrain) file and preview it in 3D (surprise surprise). You can view it with lighting, or using the heightmap as a 'texture', or you can see a wireframe view, and you can even adjust the water level! Get

it from the Delphi Programs section of my website:
<http://caer.cjb.net/delphi/delphi.html>

The next stage in desinging your map is to decide what it will actually look like: will it be a snowy wasteland, lush green fields, barren desert, bright red and blue DRUHgland (;o), or something else? This is where the surface map editor comes in.

Terragen has a very powerful surface map editor that can create very realistic-looking ground (or not, if you so desire). The surface map is represented by a tree control, and the top layer is the parent surface. This parent surface can not be removed, as it makes up the base of your terrain. Without it, you would not see any terrain at all. Under that you add child surfaces, which in turn can have child surfaces. Surfaces lower down on the tree get rendered on top of ones above, so if you have a 'grass' layer, then a 'snow' layer lower down, the snow will be rendered on top of the grass (as it should be).

The key part fo the surface map bit is the editor. Click the Edit button and you are presented with a whole host of distribution options (you may have to click the 'Advanced Distribution' tab first). Most of these are pretty self-explanatory, although one or two may need explaining. These are:

- Depth/scale: this controls the 'size' of the surface. In other words, setting it really low would produce a surface that appeared in small patches on the the terrain, while setting it high would make it appear in large areas.
- Sharpness: these sliders affect how sharp the boundary between one constraint and the other is. For example, setting the max altitude sharpness very high would produce a surface that would go all the way up to the specified altitude and stop very quickly. This sort of effect could be used to create a waterline, for example. Oh, and I reccomend you give each surface its own name, otherwise you'll be left with a whole lot of layers called '[New Surface]'

At this point you may be asking 'how do I make a surface use custom textures?'. The answer is you need a (free) plugin, called SoPack, by Sean O'Malley. You can get this plugin from <http://www.geocities.com/~ffrog>. It allows you, among other things, to create a surface that uses a texture map of your choosing, and use a greyscale image to control the distribution of a surface. So, if you wanted to create your very own custom Core Prime map, you could make a texture from a screenshot of something, and set it to repeat over the whole terrain. Voila - instant metal! Take a look at the SoPack documentation for more details.

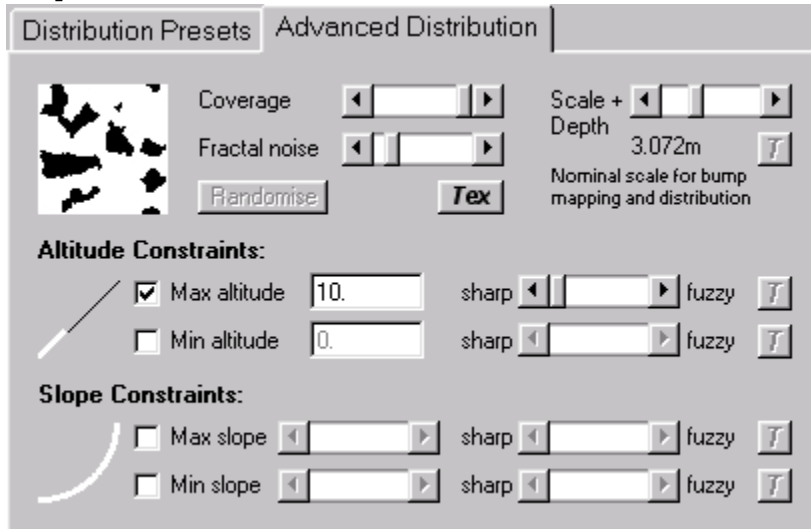
If you really can't be bothered making your own surfaces, Terragen comes with a selection of ready-made ones that look pretty good.

Another thing you may be asking is 'what about water?'. Well, water can either be really simple, or not quite as simple (and a bit more time-consuming).

The easy way to make water is using Terragen's built-in water, and SoPack to get the transparency effect. Unfortunately, this method results in water that doesn't look that nice at all.

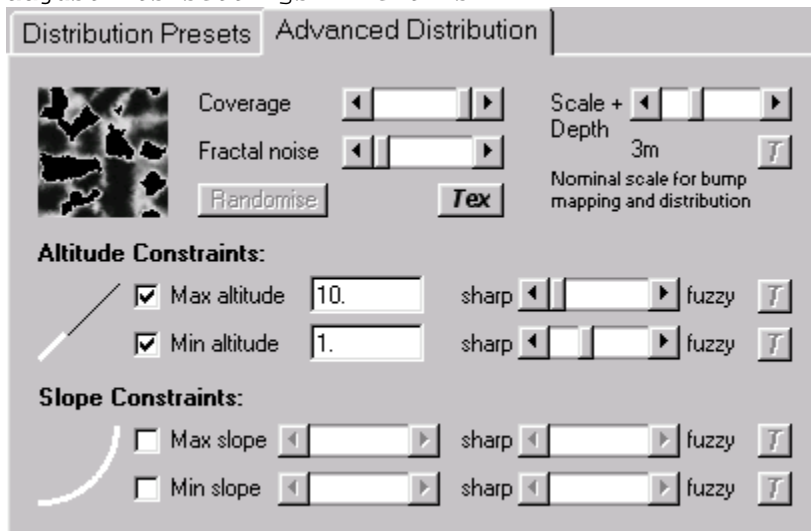
The alternative is to use a bit sleight-of-hand and use surface maps that *look* like water.

To do this, add two child surfaces to the main surface. Select the first one, and open the surface editor. Rename this surface to something like 'Deep Water'. Now, change its colour to a nice dark blue, so it looks like deep water, and set its bumpiness pretty low. Next, adjust the settings so they look like this:



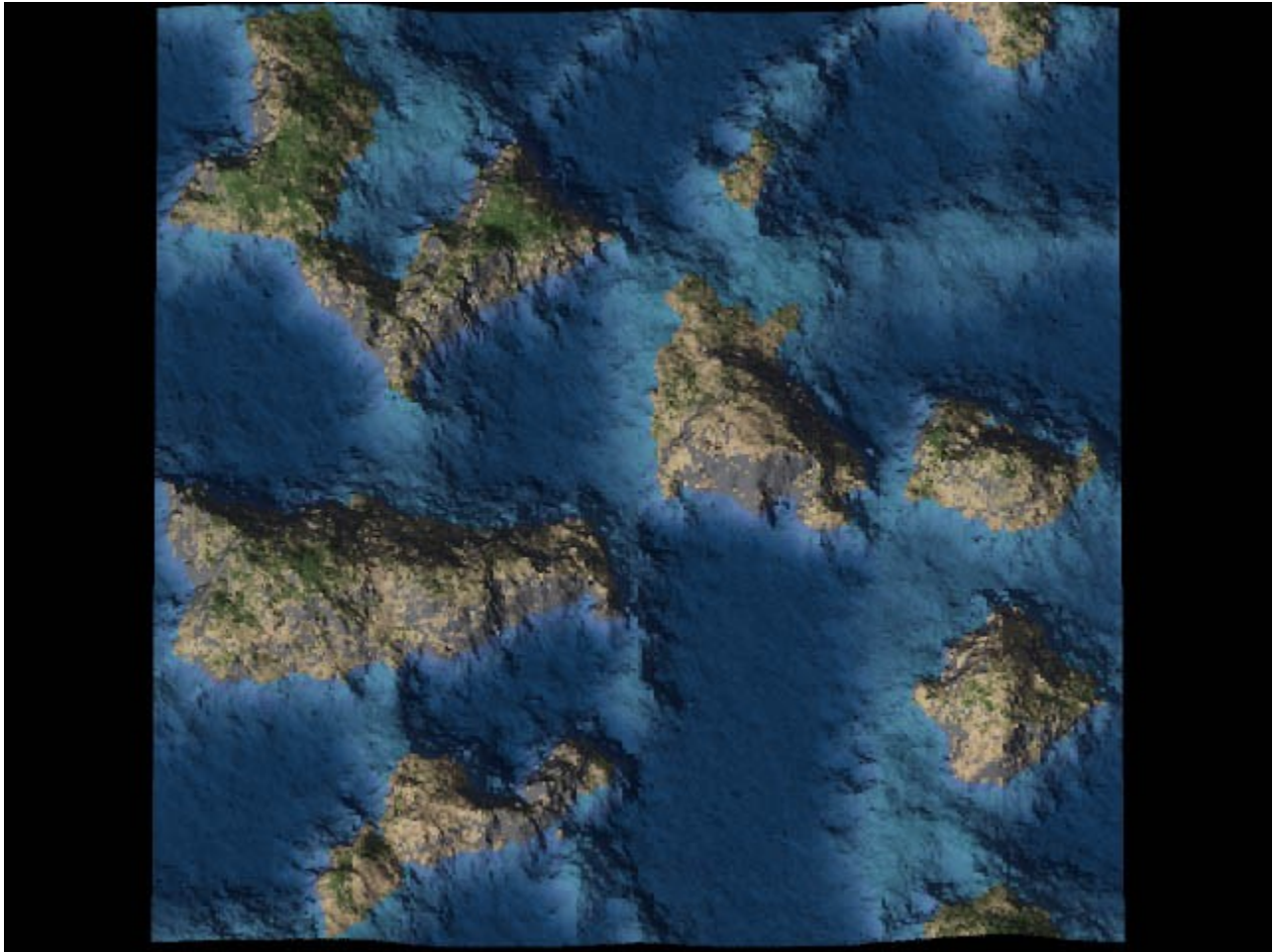
Note that the maximum altitude setting is scaled to the 0 - 30 range of the terrain. The range TA uses is 0 - 255, so you will have to adjust the desired level to fit in the 0 - 30 range.

Next, select the other new surface and call it 'Shallow Water', then adjust its settings like this:



This will create the appearance of shallow water near the shore - you should adjust the colour to look like your base surface with water on top of it.

When you render, you should end up with something looking like this:

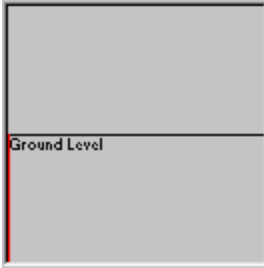


If you want to be really clever, you could render two images - one with SoPack-water and one without, then combine them in Photoshop. You'll need a lot of patience for this though, as working with huge images will be very slow.

Now, although the image above is a pretty good approximation, it can be improved. This can be achieved even without the aid of SoPack - first, save your current World file, as we'll be making some pretty drastic changes now.

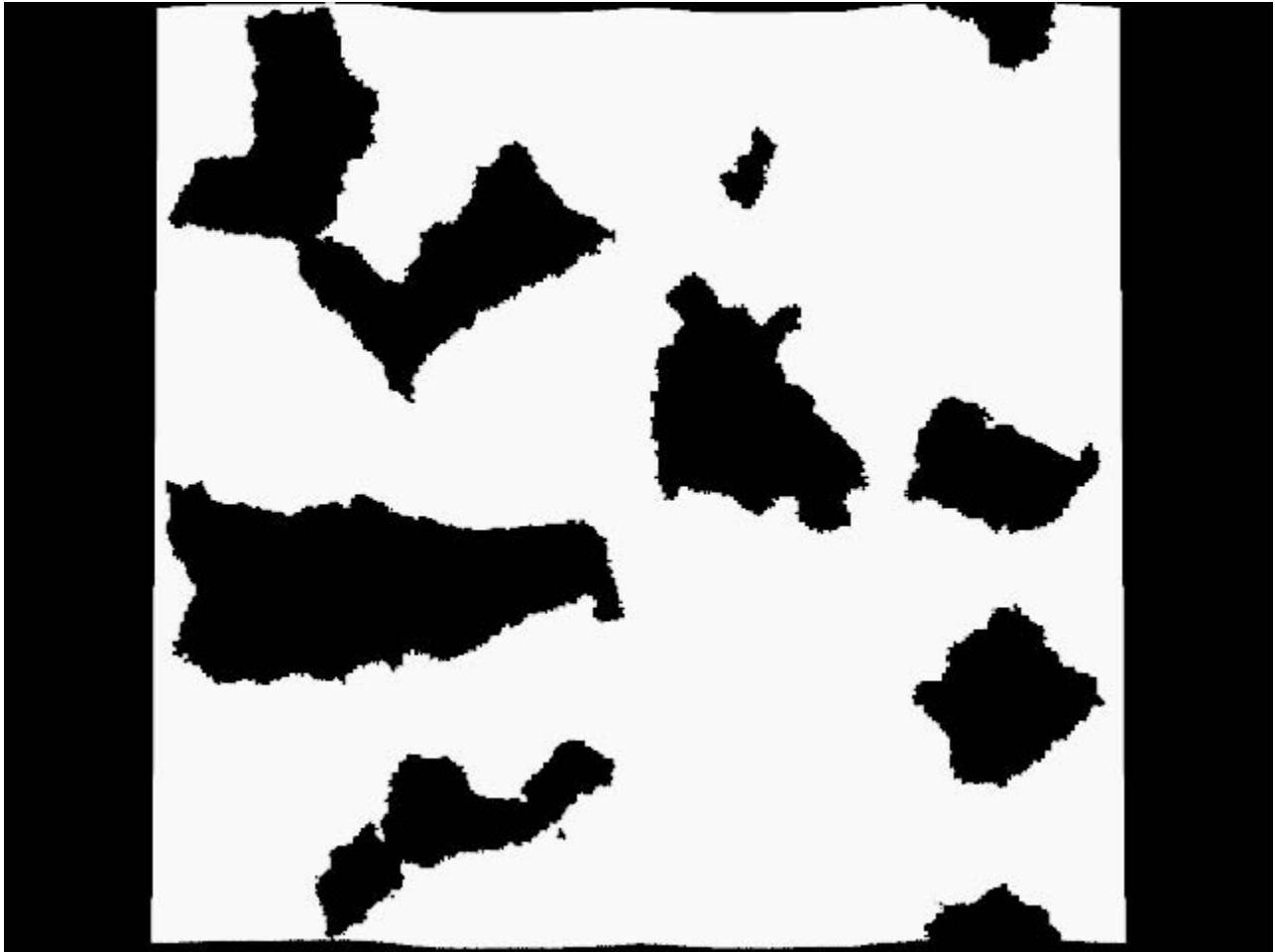
Once that's done, make a note of the maximum height of the water - you'll need this value later. Next, delete all child surfaces, leaving just the main one. Click on it, and then click the edit button. When the editor appears, set the surface's colour to pure white, bumpiness to zero, and coverage to maximum (i.e. with no height/slope constraints - the surface should cover the entire terrain). The next step is to add a single child surface, and make it pure black, again with no bumps. With this surface, set the coverage to full and with no slope constraints, but with the maximum altitude set to the desired water altitude, with sharpness set to maximum.

Next, you need to open the Sun control window, and set the sun altitude to -90. In other words, the sun must be pointing straight up, underneath the terrain, like this:



Now, click the 'Background Light' tab, and set the shadow lightness to 100, and the shadow colour to white. Make sure the 'Single Colour Shadow Lighting' option is selected.

Now, when you render your image (at the same size as the original), you should end up with something like this:



Save this image as 'watermask.bmp' - you'll need it later.

That's as much as can be done at the moment regarding map design. Now all you have to do is decide how big your map should be. The map size is based on how big you render your image, but you must make sure the width and height are in a 4:3 ratio. In other words, the width must be $1.33333 \times$ the height. If it isn't, the rendered image will have bits missing, making lining up the heightmap a real hassle.

Make sure the 'Detail' slider is set all the way to the right, unless you want a very pixelly map, then click 'Render Image'. And wait. Depending on the size of the map and how many surface layers you have, and the speed of your computer etc. etc., this can easily take 6 hours or more. For the record, my Mallodden Plain map took 26 hours to render on my ancient P133.

Once the image has been rendered, you can save it. This is annoying, as I'd much prefer it if you could specify a filename beforehand and render directly to the file. I should warn you that for a 12x11 map, you will need approximately 400MB of free space to account for the swap file and actual image. A 6400x4800 24-bit bitmap takes up 88MB! This is where having the odd half-gigabyte of memory comes in handy (I only have 64MB).

PART THREE - MAKING IT A MAP

Now that you have your enormous bitmap (or two, if you're going to include water), you need to open it in Photoshop.

You can skip the next two paragraphs if you aren't going to have water in your map.

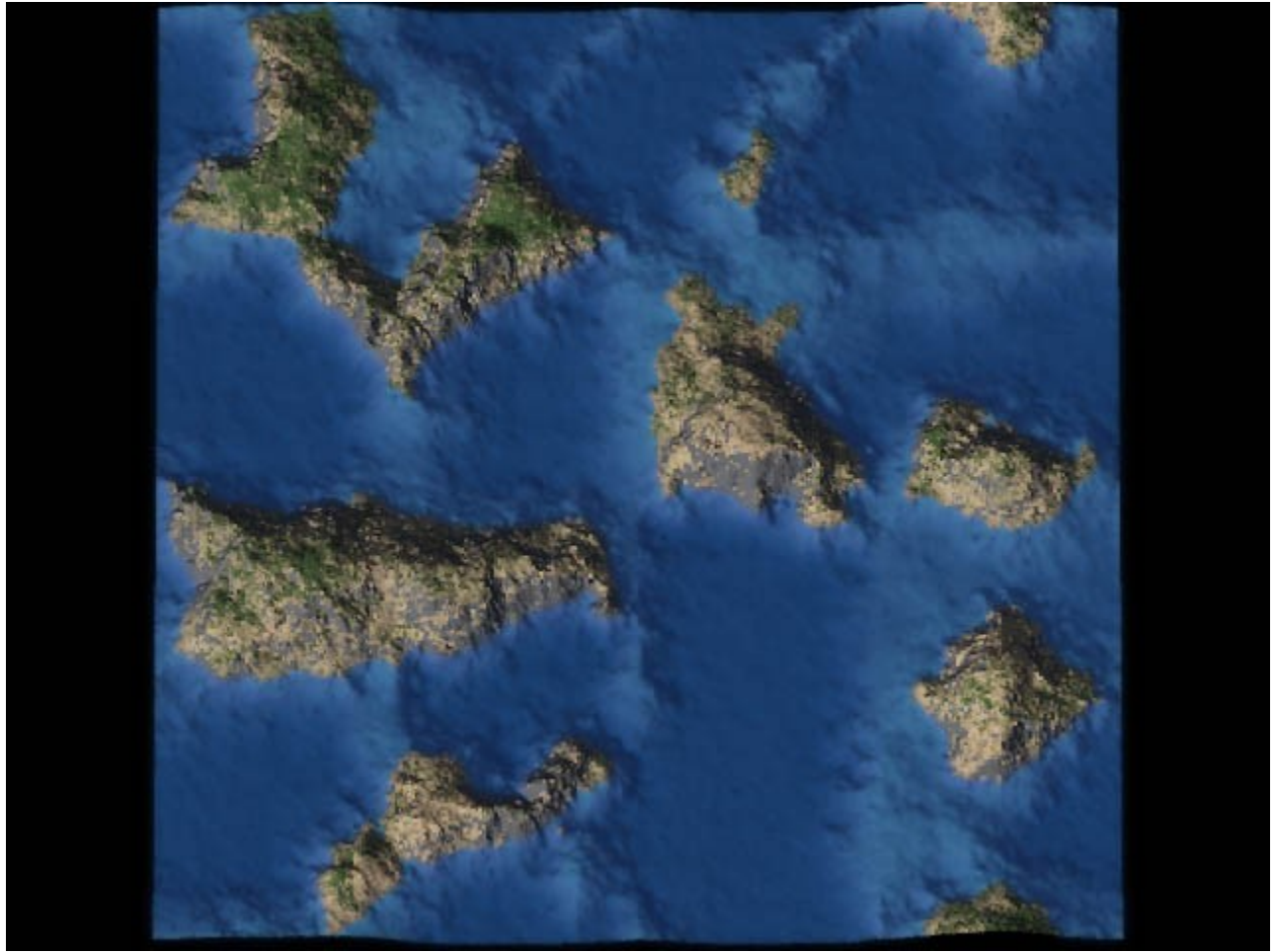
Select the 'water mask' image. Before we can do anything with the image, it should be converted to Bitmap mode (2 colour), to speed up processing and save memory (24x less memory needed!). Once that's done, copy the entire image to the clipboard.

Next select the main image, add a new layer above the background, and fill it with a colour that sort of matches the basic 'water' colour. Making sure the new layer is selected, go to the Layer menu and add a layer mask. Open the Channels tool palette, and paste the clipboard (which contains the water mask) in to the layer mask 'channel'.



You will then end up with a new layer that covers only the 'water', although at the moment it looks a bit nasty. To fix this you need to adjust the overall transparency of the layer - around 40% looks quite good. To further improve the effect, you could try adding noise, then blurring it, or create a wave effect using the 'Lighting Conditions' filter. Another thing you can do is use the 'water mask' to create a selection, then use this to apply a Gaussian blur with a level of around 0.8 to the main image, to blur the underwater areas a bit. You could also try blurring the mask for the water layer slightly. Finally, you may need to adjust the colour of the water, as you may not get the results you

expect. In the end, you should have an image which looks like this:



You can now crop away the black areas. HOWEVER, there is a particular method to this, which is best explained by this image:

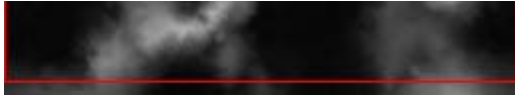


Now you can reduce the colour depth of the image, using the TA palette (you need to flatten the layers first though - you may want to save a backup .PSD file at this point). You must remember to use the 'dither' method, otherwise Photoshop will match each colour to its nearest corresponding colour in the TA palette, resulting in a hideous mess of a map. This reduction in colour depth will also take quite a while (35 minutes on my P133. Yes, I enjoy moaning about my computer's speed, or lack of it). Now save the bitmap.

Next you need to export the terrain as a .RAW file in Terragen, then open it in Photoshop. You will also need to flip it vertically, because Terragen writes the file backwards. There's probably a good reason for this.

Once you have opened the heightmap, you need to crop it according to how much was cut off the main image. The reference used to crop the heightmap must be based on a 'low' part of the main image (like the bottom of the sea). This should explain it better:






Of course, on a larger main image you will be cropping off quite a large number of pixels, but you must still crop the same amount from the heightmap, i.e. just because you're cropping a large amount from the main image, it doesn't mean you should crop a larger amount from the heightmap. If you zoom out from the main image, and crop at a low zoom level (like 16%), it will be easier to judge.

Once this is done, you need to resize the heightmap so it is 1/16th the width of the main image, and (1/16th) + 8 the height of the main image. For example, if the main image was 512x512, the heightmap would be 32x40. Now you need to crop the image so the height is 1/16 the height of main image; in other words cut off the bottom 4 rows. The reason for doing this is that Annihilator (and maybe TAE) cut off the bottom part of the main image, so the heightmap doesn't have to go all the way to the bottom. That's not a very good explanation, but it's the best I can think of. Sorry. Once you're done resizing and cropping, make sure the heightmap is in greyscale mode, and save it as a .BMP (Windows bitmap) file, called heightmap.bmp or something.

Now comes the awkward(ish) bit - importing into Annihilator. First, obviously, you need to load Annihilator. Next, click the 'New' button, then click the 'Bitmap' tab. Enter the filenames as appropriate, or use the '...' buttons, then click OK. Again depending on the size of your map, the speed of your computer, the amount of memory in it, and the phase of the moon, this may take a while.

When Annihilator has imported your map, the first thing you should do, if your map has no water present, is click the 'Height Editing' button () and move the 'Sea level' slider all the way to the left. Now, switch on the contour map (the button), and check the contours match the terrain. If everything has gone right, they will. If not, things can get sticky. If the contours appear mis-aligned, it may simply be a case of re-opening the .RAW heightmap (the original), and then resizing and cropping it slightly differently. Because of a bug in Annihilator, you will not be able to save the new version of the heightmap over the old one, so you'll have to call it 'heightmap2.bmp' or something. If things continue to not look right, you may well end up with 4 or 5 different heightmap bitmaps.

If you have decided to include water in your map, you should simply adjust the Sea Level slider until the red contour line matches the shoreline on your map image.

Once you have the heightmap/contour bit sorted out, it's time to set the map settings. This is up to you, and beyond the scope of this tutorial. Once you've done that, you need to save your map and close it, because Annihilator won't let you place any features on your new map yet.

Next just re-open the map, and start placing features!

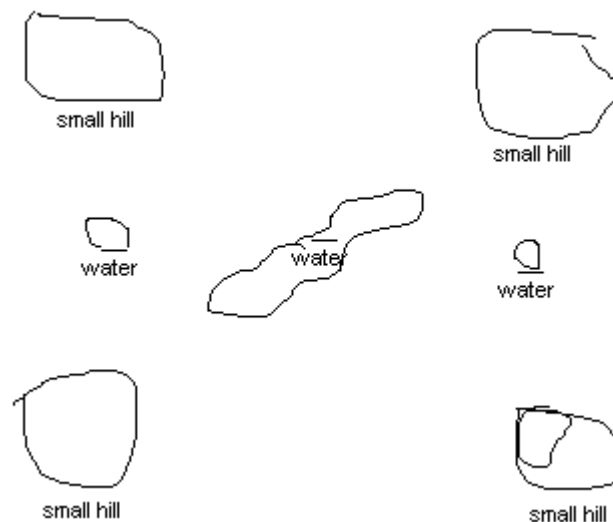
Creating Custom Tilesets

By C_A_P

- This tutorial will walk you through , step by step , my technique for creating unique tilesets for TA maps. You will need the following tools to use this torial to the fullest:
- PhotoShop 4.0
- Eyecandy (plug-in for PhotoShop)
- Paint Shop Pro
- TABuilder
- Annihilator .21

Create a concept

1. Its always a good idea to draw a simple concept of you map first, before you delve into the graphics. A program such as PAINT will do the job just fine.
2. The map we will be designing will be very small, but enough to get you started.
3. The following is a very simple concept sketch I whipped up in PAINT in about 1 min.



4. Now that we have our basic concept, we can begin with creating the graphics.

Design Your Texture and create a template.

1. There are a couple of things you will want to keep in mind while designing your texture. The most important, is that you want it to be tilable, and seamless. Also, with the current version of Annihilator, you cannot import a .bmp that is larger than 512 x 768 in size. This will create a nice 24 x 16 section .
2. So, lets get started. You will want to create a template first. You can experiment here with your own ideas, or use the one that came with this tutorial. (template.bmp). You will want to keep this template at 512 x 768 in pixel size. So fire up PhotoShop, or your favorite paint program.

You will understand where im going with this concept as we progress.



This is the main texture of our little map. We will use this section to spawn other sections.

3. Now that we have our basic texture done , save this as template.bmp. This will be the template that we use to create the terrain features.

Create the Terrain Features **Create a simple platform**

1. We will use our template file as a base for all of our terrain features. The first feature will be just a simple small hill. Open the template.bmp file in PhotoShop and select an area. I've just selected a square area to create a very simple platform.



2. Under the filters menu in PhotoShop, select Eyecandy and then Outer Bevel.
3. These are the settings I used to create the simple platform
 - Bevel Width: 54
 - Shape: Flat
 - Smoothness: 8
 - Shadow Depth: 83
 - Highlight Brightness: 44
 - Highlight Sharpness: 24
 - Lighting Direction: 128
 - Lighting Inclination: 73



A nice simple little platform that looks fairly convincing

1. You will actually want to perform the above procedure TWICE. The second time, change the lighting direction parameter to 330. Experiment around with the settings a bit to suit your fancy.
2. Voila!! We have just created our first simple platform, and it doesn't look too shabby. Save this as plat1.bmp

Create A Pool of water

Now we will create a little pool of water.

1. Start by opening up the template.bmp file in PhotoShop again. (see where im going with the template thing....this retains the tileability of each section)
2. Make a nice jaggedly, rounded selection

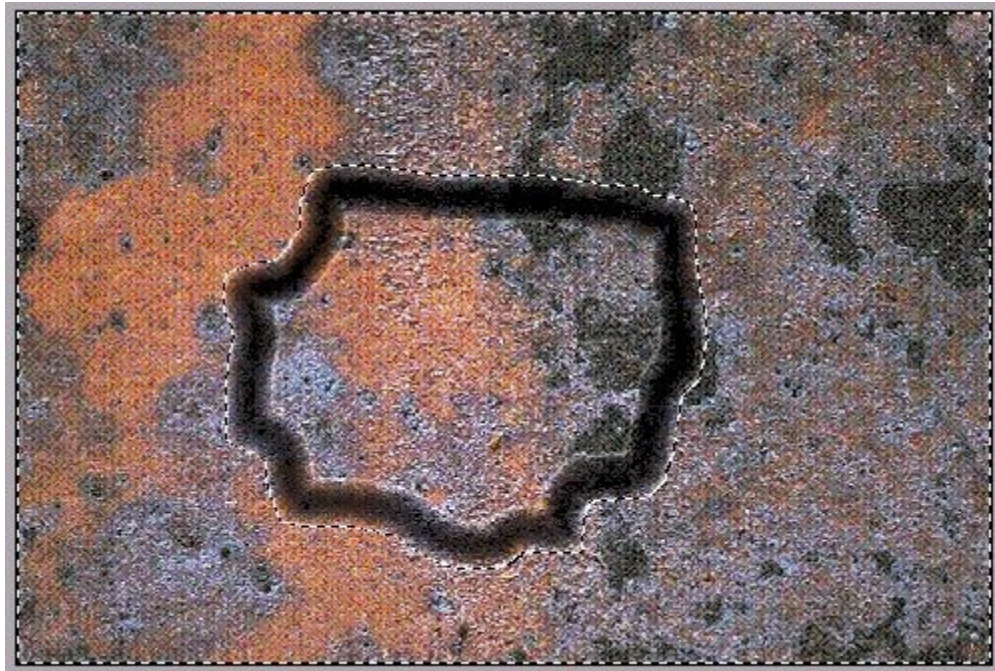


3. Now go to the selection menu and choose invert from the menu.
4. Go to Filters, select Eyecandy and choose Outer Bevel again. Settings for the tutorial are as follows:

- Bevel Width: 45
- Bevel Shape: Rounded
- Smoothness: 8
- Shadow Depth: 83
- Highlight Brightens: 44
- Highlight Sharpness: 24
- Lighting Direction: 128
- Lighting Inclination: 73

1. Again, you will want to repeat this process twice, changing the lighting direction from around 128 to about 330. Play around with these settings to get the 3D effect looking nice....

So far, it should look something like the picture below



2. Now, go back to the selection menu and choose invert again. Now choose contract and enter 15 as the value. Your selection should be right in the middle of slope, making a nice level for some cool looking rusy water texture.

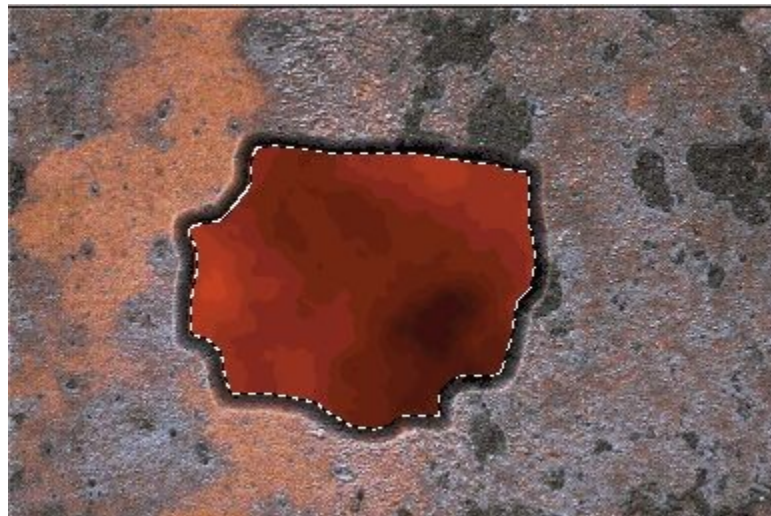
Create the acid

Now that we have our selection, its time to create the acid texture, so it looks like our little crevice is filled with some kind of corrosive substance.

1. Ok, now for some cool looking corrosive goo to go with our texture. For this tutorial we'll go with an almost lava type texture that looks very nasty.....
2. Click on the Foreground color and make it solid black.
3. Click on the Background color and change the RGB values to:

- Red: 190
- Green:48
- Blue: 25

1. Now select Filter and then Render and then Clouds
2. It should look something like the picture below:



3. Lets add a little texture to the liquid . Select Filter then Eyecandy and Glass
4. I used the following settings, but feel free to experiment:

- Bevel width: 9
- Bevel Shape: Button
- Flaw Spacing: 7
- Flaw Thickness: 18
- Opacity: 75
- Refraction: 46
- Glass Color (Solid White)
- Highlight Brightness: 0
- Highlight Sharpness: 22
- Lighting Direction: 135
- Inclination 85

Now save this as Acidpool.bmp and we're ready to move on to the next section.

Create a River of Acid

This will walk you through creating a river of acid, using two sections.

While creating my maps, I ran into an annoying bug in the current version of Annihilator (.21). If I tried to import AND place a bitmap as a section that was larger than 512 x 768 I got a script9 error. So ... I used the following method to create larger sections....

1. Start again, by opening up the template.bmp texture in PhotoShop. Go to Select and choose all
2. Then go to the edit menu and choose copy
3. Now go to Image and select Canvas Size
4. Orient to the top left, and resize it to 1536 x 768
5. Now you should see something like the following picture



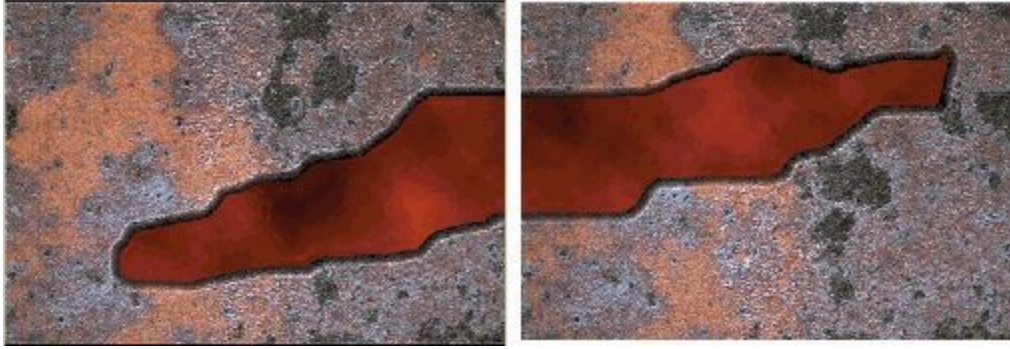
6. Now go to the edit menu and choose paste
7. Zoom in nice and tight and align the pasted portion exactly with the other side.



8. Now select a river looking area and perform the same operations as you did with the acid pool. When you are done, you should come up with something that looks something like the picture below:



9. Save this as river.bmp and lets move on.
10. Now to work around that bug that lurks in Annihilator
11. We have to split the river in half, each as their own section. To do this, go to Canvas Size and orient the picture top left and change the size to 768 x 512. You will get a warning that the image will be clipped. Don't worry, this is what we want!!!
12. Save this as River-l.bmp
13. Open River.bmp again. Do the same thing, except align right before you resize it. Save this as river-r.bmp
14. You should have 2 sections like the ones below:



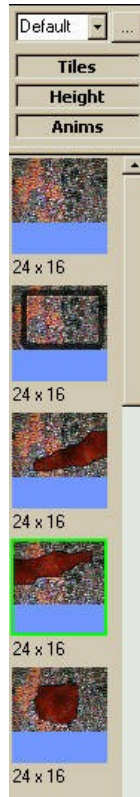
Converting the sections to the TA color palette

The last portion of the graphics tutorial is converting these to the TA color palette so that they can be imported as sections in Annihilator.

1. I have included the TA palette for Paint Shop Pro in this tutorial. Open each of the bitmaps (except for River.bmp) in Paint Shop Pro
2. select 1 of the images and choose color and load palette (make sure you have error diffusion turned on), select TA.pal and re-save it.
3. Do this for each of the sections; template.bmp , plat1.bmp, acidpool.bmp, river-l.bmp, and river-r.bmp

Importing the bitmaps

1. In the last step of Part 1, we converted the bitmaps to the TA color palette. We will now import those bitmaps into Annihilator .21 as sections. So, fire up Annihilator and lets get goin....
2. Click on file and then new map and lets create a simple map to place our sections on. Select Tiles tab and choose tutorial.tnt for both the TILES and ANIMATIONS values. Leave the default height at 75 and sea level at 50. Map size of 100 x 100 is fine. Enter whatever you desire as map description and mission description.
3. Go to Sections and choose Load BMP as Section
4. Now go to the directory that contains the bitmaps and select template.bmp and click OK. Notice how it appears on the right side as a 16 x 24 section. Repeat this process for each of the bitmaps. When you're done it should look like this



5. At this point, it's a good idea to save these off as a section. Do this by selecting sections then save sections and choose a name and location. These sections are included with this tutorial. Get them [Here](#).
6. Now, click on the height button to disable it, and place the first one on the map about 3 tiles down from the top and 1 or 2 tiles in from the left. This will make more sense later....
7. Repeat the process, separating each section by 1 or 2 tiles. When your done, your map should look something like this. This will make more sense as we go on...



Ok, now its time to do the fun stuff like height editing and creating the terrain archives

Editing Height

This will walk you through editing height and saving a terrain archive so we can create our map

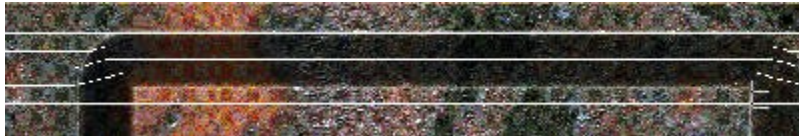
1. OK. Now that the easy part is over, lets move on to the more time consuming part of making maps. Height editing. Switch to height mode..
2. Ok, the template is for the most part finished already, it will be a flat section of ground, so we will leave that piece alone.
3. Lets move on to the section that has the platform. For this tutorial, we will make the platform tall enough to give it some advantage, but not steep enough so that units cant get up the sides. So we want all units to be able to get up the sides and onto the platforms, since it doesn't LOOK to be very tall.
4. In the Height Interval section, press the spinner until its value is 5
5. So now we have to edit the height to make it appear as though it matches the graphics. Ok..lets start at the top left. Just at the base of the platform click once with your LMB. You will see a little portion go up at a slight angle. Move the pointer to the right a hair and click the RMB again. You will see it start to pull up the line at the bottom of the platform. Get into this rhythm..move click....move click....move click....should look like this when your done.



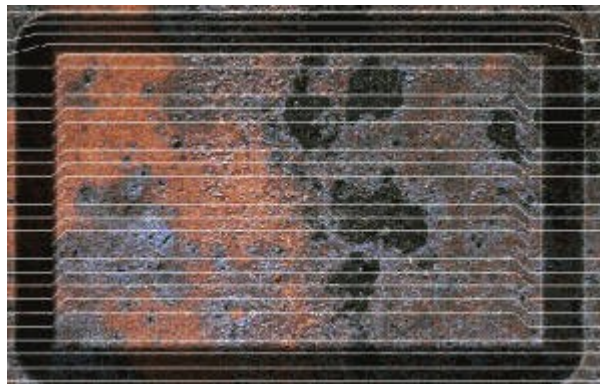
6. Ok. You can probably tell that this is going to take some time. Next step is to move up to the next line and start again. This time click the LMB 3x so it creates a nice even slope. You might find it easier to adjust the height to 15 units here, so you don't have to click 3x each time. When you're done, it should look like this...;



7. Now we want to create the next one. Increase your height interval to 15 this time and click on the next line up and do the move..click...move thing. Now we want to smooth out the edges a bit, so adjust your height interval back to 5 and go along the edge to make a nice smooth slopeit should look like this.



8. Ok. Now adjust your height interval up to 50 and increase the aperture area up to 5. To save some time, we are going to raise the platform up to its full height 125 and worry about the edges afterwards. So, move the aperture to the edge of the platform and click. You should notice that a much larger section moves up this time....do this until the majority of the platform is raised. It should look something like this when your done...



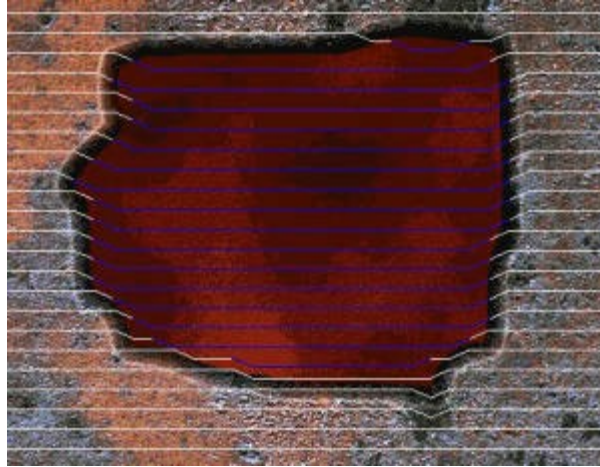
9. Ok. Now finish it by going around the edges and making a nice slope on all 4 sides. Spend some time to make it nice and tight around the edges. The ground should be at 75 and the top of the platform should be 125.



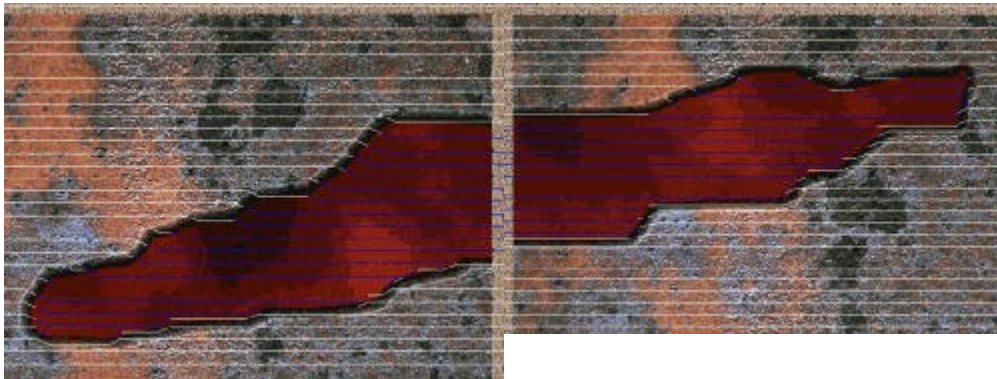
10. Time to move on to the acid pool next...save your work at this point either as a .tnt file or as a .ann file.

Finishing Touches

- When editing sections that contain parts that are going to be below sea level, its important to know what you want your map to behave like. The TA engine is limited in what it can do. You cannot have one part of it be 'lava' and another part be 'water'. For the purposes of this tutorial, I've elected to make the acid pool just shallow enough for the units to easily enter in and cross, but not deep enough to allow a shipyard to be built. And for an added twist, since im sure most of you have TA:CC by now, I will show you how to make these sections cause damage to the vehicles that enter them....
1. Start by entering height editing mode and changing the height interval to 15. Click around the edges of the pool **ONCE**. This will create a nice lip around the rim that the units will have no problem going over. Once you have the whole rim created, change the height interval to 30 and click just inside the first area you did. You will notice the lines starting to turn blue at this point. This is because the height has fallen below sea level (should be at 45, and sea level is at 50). This is just deep enough to make our acid pool do its job. Finish it up, and it should look like this when its completed...again..save your work as you go along!!!



2. Now move on to the Left and Right river sections. By now you should have a decent understanding of how height editing works in Annihilator .21. The two sections , when finished should look like the pictures below. Once all your sections have been edited to your satisfaction for height, its time to move on to the next thing...

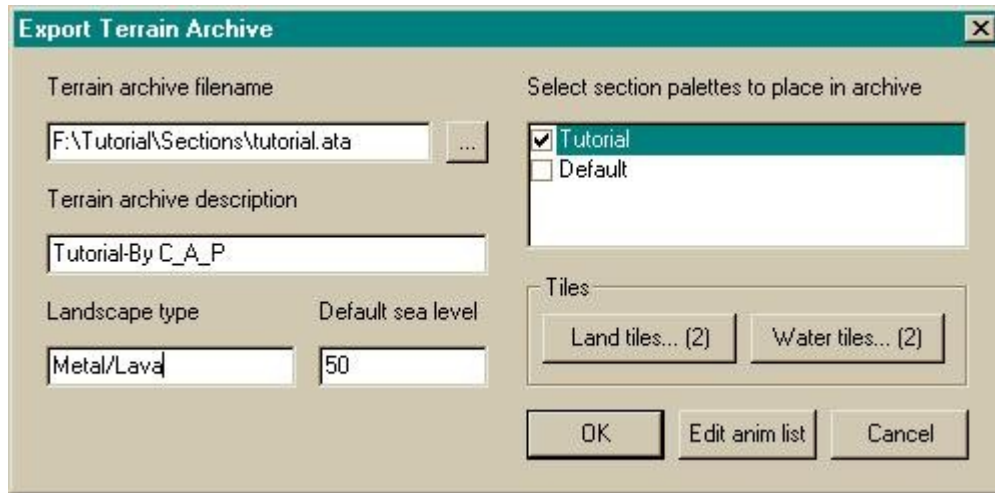


Now its time to create a Terrain Archive and make your map!

Our little map is coming along nicely, we have all of our pieces created, now its time to put them to use!. To make the job of making a map easier, we should create a Terrain Archive. This isn't NECESSARY, but makes it easier. It also enables you to be able to distribute your tileset to other mapmakers in the TA community.

1. Press the button that looks like three dots. This will create a new, blank selection palette. Highlight the word default and press backspace enough times to clear the word and replace it with Tutorial
2. Position your cursor on the top left side of the template section. Press and hold down the RMB and drag a rubber band around the section. **MAKE SURE** you select only the section. It should be exactly

- 24 x 16. Let go of the mouse button and the selection will remain. Press ctrl+c to copy the selection, with its newly created height info , to the selection palette.
- Repeat this process until you have all the sections re-copied over with their new height info.
 - Go to the sections menu and choose export terrain archive. The following window will come up.



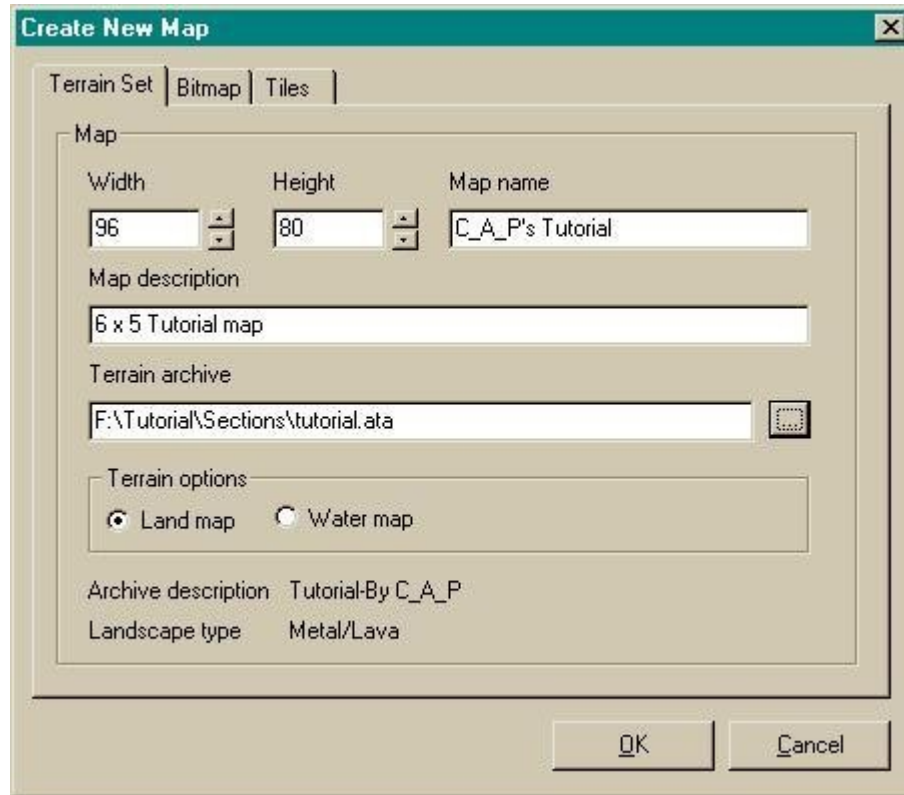
- Check the tutorial box, so that it will only export the selection palette that includes the height

- Give it a name
- Give it a description
- Give it a Type
- Leave the sea level at 50 for this
- Press the land tiles button and choose 1 or 2 tiles for land
- Press the water tiles button and select 1 or 2 tiles for water
- Press OK (don't worry about the edit anim list one for now)
- Done, now we just have to study our original design and decide how to make the pieces fit...

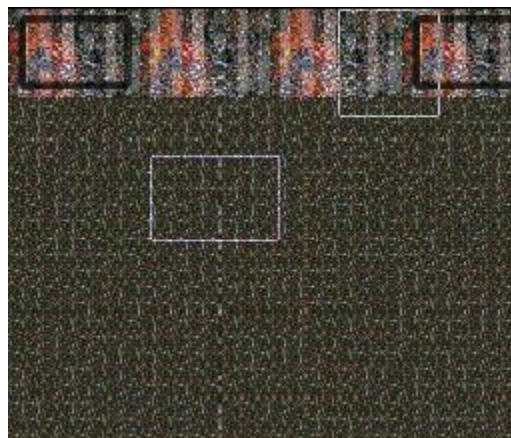
Make your Map

Now its time to put it all together.

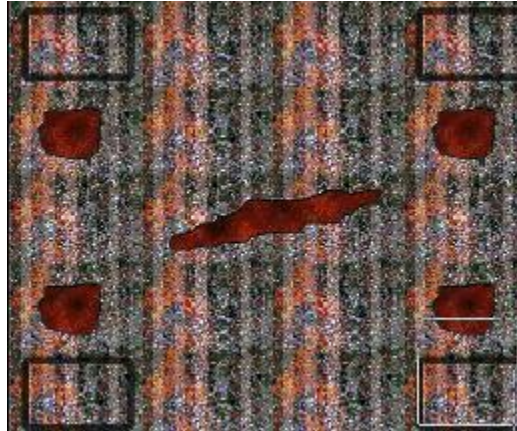
- The first thing to do is press file and new map and it will open the following dialog.



2. Enter in whatever information you'd like, **but the width and height should be kept the same for the purposes of this tutorial**. And of course, the locations of various files will be different. Press OK and give it a few seconds...and presto! You should have a fairly nasty looking map with all your sections, including height values, on the right hand side.
3. Now, begin to place each section, starting from the top left.
4. Place the Platform section in the very top left. Place a ground section to the left of the platform. Then one more ground section, then another Platform section. It should look like this so far.... (minimap view). Notice how it all fits together seamlessly...



1. Now, just work your way down the map. The next row goes like: Acid pool, ground , ground, Acid pool.
2. The next row is: ground, river left, river right, ground.
3. The next row is: Acid pool, ground, ground, Acid pool.
4. The last section is: platform, ground, ground, platform. Voila!! Our map is nearly complete! It should look like this:



- Ok!! We have created a reasonable facsimile of our original concept drawing in tutorial 1. (Notice I added some extra acid pools that weren't in the concept map).
- Now, all we have to do is place the starting points, features, and set the map properties, and we're ready to test it out!
- I will let you dictate where you want the features to be placed, and the starting points.
- Now, save the map as a .tnt file. It will also create a .ota file that contains all the various settings of your map. We will want to open this .ota file in a program such as notepad. Add these two lines in the section that reads [GlobalHeader]

- waterdoesdamage=1;
- waterdamage=100;

This will make it so that when the units go through our little 'acid' pools, they will be hurt....badly. You might want to adjust this to a lower setting, but I think its funny to send a few flashes through the acid and watch it smoke and blow up in a few seconds....

It should look like this:



```
tutorial.ota - Notepad
File Edit Search Help
GlobalHeader]
{
missionname=C_A_P's Tutorial Map
missiondescription=6 x 5 -Tutori
planet=Acid;
missionhint=;
brief=;
narration=;
glamour=;
lineofsight=0;
mapping=0;
tidalstrength=10;
solarstrength=20;
lavaworld=0;
killmul=50;
timemul=;
minwindspeed=0;
maxwindspeed=3000;
waterdoesdamage=1;
waterdamage=100;
gravity=112;
```

AI TWEAKING GUIDE

BY Switeck

What is the ai?

AI stands for artificial intelligence. It refers to the computer-controlled players in skirmish games AND multiplayer games with computer players added.

Cavedog's ai, and its problems:

"From time to time, people complain about the weather. But no one ever does a darned thing about it." -- Mark Twain

The ai in Total Annihilation was quite dismal in versions 1.0 and 1.1, and only slightly better in v1.2b1. With the v2.0b1 patch, the ai was improved but still not much of a threat in a 1 vs 1 ai game against a good player. Cavedog has considered improving the ai to be a low-priority goal in patches, and it's looking like the v2.0b1 patch may be the best ai that cavedog makes.

In the earlier ai's (before v2.0b1), what made the ai dismal was that it was limited to only 5 factories and 6 construction units. It poorly managed its resources and was often unable to constantly run even that pathetic amount of production. It would try to attack with ground units on a sea/island map. Or it'd build metal makers on all-metal maps. Or fill its base up with crawling bombs that never moved.

In later versions the ai was not quite so bad, but still shared many of the same problems. With v2.0b1, it looked like Cavedog *almost* knew what they was doing. Finally they'd made an ai that could build a huge base in a reasonable amount of time AND build a large attacking force. No longer did it build many of the absolutely useless units like lots of radars and crawling bombs. I even lost a game to it on Metal Heck while playing with 2 people vs 3 ai's over the internet.

Despite its many improvements, the v2.0b1 patch added NEW problems to the ai. Since that patch allowed cavedog's add-on units (and even 3rd party units) to be used by the ai, the ai would build many of the new units without limit. Although this was ok with units such as the Core Sumo, this was worthless with the Arm Marky - causing the area around the advanced k-bot labs to become so crowded that the labs could no longer make any more units. With the fixed base units, this was even worse: The ai didn't know when to quit building naval metal makers, naval missile towers, and naval laser towers.

Cavedog's built-in AI is simply shameful in v3.0 and v3.1 of TA:CC. One would expect that with each patch to the game that the ai would slowly get better. But the ai went from decent in v2.0 to terrible in v3.0 and almost as awful in v3.1. The v3.0 patch was the first patch to use Core Contingency which introduced many new units into the game that were specialty units that were almost all unuseable by the ai.

A major complaint of the ai in v3.0 and v3.1 is that it builds a base and lots of construction units but very few attackers. It's like the ai wants to play a game of sim-city instead of a wargame. With the addition of TA:CC, all the new units that CAN be built tend to "water down" the ai. A BIG problem is the unit count, with 70+ new units it is impossible for the ai to build a reasonable number of any particular unit.

General problems with all the ai's made by Cavedog:

All the ai versions would (on very rare occassion) build a nuke silo or an antinuke silo. But never would a nuke be fired nor would the antinukes shoot down nukes shot in its direction. A hacked unit that acts SIMILAR to the real unit is required for the ai to actually shoot nukes and possibly antinukes.

The AI plays the map, not the other player. Metal spots (mostly) determine where it builds metal extractors and thermal vents are where it sometimes gets sane and builds a geothermal plant. Its base ends up being built around whatever features are on the map, with no consideration for where its enemy/s are. It doesn't consider what the player is doing to decide what to build next. It looks for the nearest enemy unit to its attack force, that's almost all it knows to do about the enemy player. If you do something unexpected, it falls apart. (The "Solitaire" bug or "Sim-City" bug)

All the ai versions also have their attacking ground units "sleep" after destroying all nearby targets. The ground units quit moving for a couple of seconds, making them easy pickings for things like guardians. In my opinion, this is probably the ai's BIGGEST weakness - but it is one that is unchangeable by ai profiles. (The "Coffee Break" bug. Maybe it's a union thing...)

All the ai versions would have their ground forces concentrate their firepower on 1 unit. Often, the ai ends up shooting at resources (like a closed solar, which takes quite a lot of damage to kill) while nearby base defenses pound away on them. (The "Tunnel-vision"/"Target Fixation" bug)

Sometimes, because of the terrain or DT, the unit the ai is trying to destroy cannot be hit - but the ai doesn't quit trying to kill it. A good example of that is when peepers are landed in the middle of the square columns on Metal Heck before the ai's ground force arrives. Until a Merl or similar unit that can shoot over hills arrives, the peeper is reasonably safe. Another good example is a sub close to the shore of the enemy base - ground units will wade into the shallow water trying to get to the sub, but that only makes them vulnerable to torpedoes. (The "Why won't it die?" bug)

If the unit the ai is trying to destroy runs away, the ai's ground forces will chase that 1 unit around the map until it either gets destroyed or it gets about 2 screen lengths away from the ground force. Note: flash tanks are really good for doing this. (The "Pied-Piper" bug)

Any of the ai's units that are not already shooting at something sometimes fire on whatever enemy unit gets into their range. (The "Free Will" bug) Whoops! Maybe that's not a bug. Probably good the game has that bug! ;)

If a friendly unit is near where another friendly's weapon hits, they often take splash damage. Although this is not a bug in itself, the ai doesn't try to stay out of the line-of-fire. Construction units are especially bad about this - maybe they're suicidal... (The "Friendly Fire" bug)

If the ai's commander is damaged, it stops where it is for up to 15 seconds. This makes it very likely to be hit again, since it's not trying to get away. Worse still, if the ai's commander is being hit very quickly over and over again by multiple units, it doesn't even shoot its wimpy laser at the target it just twitches until it's no longer being fired at or (more likely) it dies. 3 flash tanks shooting the ai's commander at one time are good at making it wiggle helplessly till it dies. (This is just a "Gone Stupid" bug.)

After many attacks and retreats, the ai's ground force gets smaller and smaller. Makes you wonder if the ai is becoming pacifistic. This usually happens after the ai's base has been left alone and it's had hours to build up without its base getting bombed. This is because the ai is at or near the unit max. The lower the unit max, the worse this problem gets and the quicker it occurs. The ai doesn't avoid building a base containing so many base buildings that it cannot build any ground forces - this is because the ai doesn't know what the unit max is! It could be only 100 units max, and the ai will still try to build a large base. ("Make peace not war" bug.)

Massed aircraft are considered by many to be the most powerful attacking force in the game, yet rarely does the ai have even 3 aircraft working

together. The ai's aircraft wander alone randomly around the map. Only if the aircraft stumble across an undefended unit are they likely to successfully destroy it and survive. Never is more than a modest air defense needed to prevent the ai's airforce from destroying a base.

The ai doesn't build ENOUGH resources quickly enough to keep up with its rate of consumption - often it has few solars and metal extractors to run 3 or more factories and 8 construction units.

A key AI problem is that some units aren't useful to the ai. The ai will build multiple radars right next to each other, or build metal makers when out of both metal AND energy. This compounds the ai's weakness in resource-gathering ability - already LOW on resources, the ai generally wastes what it does have on units that it doesn't know how to use.

But that can be changed!

Without modifying Total Annihilation (other than OFFICIAL game patches), if you wanted the ai to give you a good fight in a skirmish game, you either have to take on 3 ai's vs you (or worse odds) or restrict yourself to give the ai a chance (no nukes, berthas, dragons teeth, aircraft, etc, etc).

Now, there's an alternative: because Cavedog made TA easy to be changed and added to, the ai can be changed as well! In the absence of an official Cavedog fix to the Ai problems, we can design our own AI profiles which influences many things that the computer does. Just by eliminating the useless units (to the ai only!) the ai can be improved over what Cavedog has made. And by balancing the useful but expensive units, the ai can be made even better.

I've discovered that almost every time I eliminate units so the ai no longer builds them, the ai generally gets better. Or, if I add in whole groups of units the ai tends to become slower in attacking and base-building.

My ai profiles attempt to fix the problem by changing what the ai can build and how often it can build it. This doesn't permanently modify the game but it does fix some of ai problems in the game. The patch is totally compatible with TA, and can be used in multiplayer games. What's more, if you put an ai into the multiplayer game it is controlled by YOUR ai profiles, not the host computer's! What this means is your ai may be better than anyone else's in the game.

My ai patch tries to fix the problems not caused by the underlying ai engine hard-coded into the EXE. There are still problems in the game that prevent the ai from playing a good game EVERY time it plays, like when it builds its vehicle plants with the exits blocked by walls - because the ai engine controls WHERE it builds, my ai profiles only control WHAT it builds. I do not claim my ai is unbeatable, but players will find skirmish games against the ai more of a challenge.

What are ai's and Where do I put them?

The ai's are simple text files that can be viewed and edited with almost any word processor, including Wordpad and Notepad. All the ai's go in the

AI directory under TOTALA. If the standard install for TA was done, all the ai files are in the C:\Cavedog\Totala\Ai directory. Cavedog DOES NOT create this directory for you nor will there be any ai files in it unless you place them there, this is the directory you have to create to change the ai.

In the (unlikely) event you wanted to remove the ai profiles so the game uses the original built-in ai profiles, either delete the files in the \Ai directory *or* rename them or move them to another directory.

How does the computer know which ai profile to use?

The map files themselves determine which ai profile that the computer will use on each map. This is a single line in the OTA file of the map:
aiprofile=default;

Note: if the maps are stored in UFO file format (or even HPI or CCX), they *still* contain OTA files. The only difference is that the OTA files are compressed inside the UFO file.

In TA:CC there are many standard ai profiles that a map author can use for their maps: Acid, AirBattle, Default, Hover, Krogoth, Metal, Missions, SeaBattle, Urban, and Waterwrld. All are ordinary text files that contain what the ai is allowed to build for those map settings. Most maps will work ok with one of the above ai profiles, but putting in the wrong one can be worse than none at all.

There's nothing worse for the ai than being told that the map is a green land map (DEFAULT.TXT) in the OTA file when there's water on it and it should probably use AirBattle or SeaBattle ai file instead. Trying to flash-rush from one island to another just doesn't work. ;)

ANY map can benefit from an ai profile designed specifically for it or for its type (water/land/metal/wind). A few maps need something more for the ai to attack before the 30 minute mark - that's where a modified ai is really needed!

How do I change the ai?

The ai text files use simple commands to tell the ai what it can and cannot build. Most such commands only affect 1 specific unit, but there are a few universal values that can be used to affect groups of similar units or units all on one side. Sadly, the universal values aren't as useful as they might seem - because they often affect TOO much.

Units are referred to by abbreviations made by Cavedog or 3rd party unitmakers. A table of all the units and abbreviations is a good thing to have to figure out what an abbreviation is for a certain unit or vice versa. The abbreviation's first 3 letters almost always (except in the case of a few 3rd party units) tell which side the unit is made by - either ARM or COR. And the total length of the unit abbreviation is 8 letters. Usually, the name of the unit becomes the abbreviation for the unit - such as ARMFLASH or CORCAN. Other times, the abbreviation comes from the unit's function such as ARMHLT (the Sentinel laser turret). The rest of the time, you just have to wonder what the people at Cavedog were smoking: ARMCROC (I thought Crocs were Core units!), ARMTHOVR (a throw-

over?), CORAPE (Core can build apes?), CORGATOR (sounds amphibious?), CORSEAL (I assume this is a naval unit?), CORSS (nazis?), CORTHOVR (whatever it is, Core's got one too!), CORUWMEX (underwear mexico?), CORVROC (well, roc is a mythological flying bird so this is some kind of flying unit?), CORWIN (but not CORLOSE?).

Limit tells how many of a particular unit to build and always has the form: LIMIT unitname # (with a number from 0 to at least 99)

- example:

```
Limit ARMMAKR 0
```

means the ai is not allowed to make any Arm metal makers.

(Note this isn't the same as the Arm Moho Metal Maker or Core metal maker!)

The effects weight has is a little harder to understand. If the weight command isn't used then I presume the game uses a default weight of 1. (I'm *REALLY* not sure on the default weight, so I try to weight everything!) The Weight command increases or decreases the probability that a certain unit will be built NEXT and always has the form:

WEIGHT unitname # (the smallest usable number is 0.05, the largest is probably 255)

-example:

```
Weight ARMFLASH 9
```

means Flash tanks would make the vehicle plants seem like flash tank only plants.

(this might be part of a flash-rush ai.)

Another "command" seen in many ai text files is 2 divided-by signs together "//" on the start of a line. It tells the ai to not do whatever's on this line and is just info to read if you're trying to understand what the ai does.

example:

```
// Nuke by the 10 minute mark!
```

(doesn't do anything, but sure looks good. :)

There's even divider commands to separate different parts of the ai text file for easy setting, medium setting, and hard setting. These are the "PLAN diff" commands. If you want to include parts of an ai file that are common to all difficulty settings, that can be done by putting the commands BEFORE the plan sections. example:

```
weight ARMFLASH 9
```

```
plan easy
```

```
// total pushover, doesn't build anything it might accidentally hurt you with.
```

```
limit ARMFLASH 0
```

```
plan medium
```

```
// it might try to attack you...
```

```
limit ARMFLASH 10
```

```
plan hard
```

```
// Plan to lose!
```

```
limit ARMFLASH 99
```

(So, for the above example, all difficulties weight flashes to 9. But the different skill levels only allow the ai to build so many flash tanks at one time: 0 for easy, 10 for medium, and 99 for hard.)

If for any reason a similar command is repeated twice in an ai profile, like:

```
limit ARMFLASH 0
limit ARMFLASH 99
```

the TOP command would be the command the ai uses (UNLESS they're under different SKILL settings) - and no flashes would be built. Although you'd be a little crazy to have that kind of repeated lines in your ai, universal commands such as:

```
limit plant 10
```

(Limits TOTAL number of factories/plants the ai can make to 10.)

could override later commands like:

```
limit ARMAAP 12
```

(Limits Arm's advanced aircraft plants to 12.)

That's all the commands I have. There's nothing there about not building something until the ai has enough energy/metal. There's also nothing there about NOT building something if the ai's under attack. The original Cavedog-made ai profiles do not use reasonable weights OR limits on units. This is why I say the ai can be improved over Cavedog's but only so much. Because the ai will always be a little random in what, when, and where it builds. And it'll almost never build something right when it needs it most.

However, if something is weighted VERY low (I.E. Weight ARMGUARD 0.05) then that item will most likely only get built if the construction units/buildings have nothing else they are allowed to build. So when all the resources are finished, THEN my ai is allowed to build guardians which are very expensive metal-wise.

Weights seem to work better when placed at the top of the ai profile. Since the ai profile is read from top to bottom, if 2 commands change the same thing then the TOP command would be the one the game uses.

All this is not enough information to make a good ai, however. That is best done by trial and error, as mine has over the last 6 months.

Then what does make a good ai?

I ask myself that question a lot:
Is it an ai that attacks early?
Or an early berth-builder?
Or an ai that builds a well-defended base?
Are its ground forces mostly tanks, kbots, or both?
Does it build lots of aircraft?
What about an ai that will regularly commander-rush you?

These things can all make an ai BETTER, but NONE of those are what makes a good ai!!!

A good ai is one that can get resources quickly and uses all its resources to further increase its growth AND attack the enemy. The way it attacks the enemy is just the details! If you don't have the resources to run your war machine, your war machine doesn't run.

An ai can be made to take almost all of its starting resources and make flash tanks out of them. This could make for an ai that flash rushes quite

well, but its later attacks would dwindle in strength or at least not grow very quickly. After finding out how to defeat it once (hint: try building some HLT's or LLT's with DT around them...), it'll quickly become less and less of a challenge - meaning that it's not a good ai.

So, a good ai should build nothing but resources?

That's what the ai in Cavedog's v3.1 patch seems to do... ;)

But that doesn't work either, because it's very hard to attack the enemy with solar panels, metal extractors, and construction units. The ai must slow down its rate of growth to build defensive and offensive units as well. A good ai must use its resources wisely!

Obviously wasted resources are bad?

If an ai is wasting metal, that's part of a potential attack unit that *COULD* have been made to attack the enemy. If an ai is wasting lots of energy (like +200 energy a second or more), that *COULD* have been turned into metal makers that *COULD* make the metal to make a potential attack unit that *COULD* have been made to attack the enemy. Even having a construction unit or factory not doing anything is wasteful - because if they weren't needed, more attack units *COULD* have been made instead of those worthless units.

It's not supposed to do too much at one time, but is ALSO supposed to do everything?

Yes, and that's the problem. If the ai could build a huge super-defended base while launching massive ground attacks, large air attacks, and firing berthas and nukes - all in 20 minutes - it'd definitely be a kick-ass, uh... I meant good, ai. So, unless we make an ai that cheats, (well more than it *ALREADY* does...) we have to settle for less. Maybe it can do all that by the 1 hour mark instead of the 20 minute mark. But if it builds a tiny airforce instead, the massive ground attack, berthas, and nukes might be ready by the 40 minute mark. If nukes are left out, that time might drop to 30 minutes. And if all that is wanted is a massive ground attack, the ai can pull that off in 25 minutes. If you settle for a smaller ground attack made up of just tanks and no kbots, that just might occur in under 20 minutes. This doesn't mean that the ai won't do all those other good things, just that they won't happen until later.

So optimally, the ai should have 0 metal and 0 energy in storage?

You'd think so, but that sort of balancing act is both impossible (as long as the ai is still alive anyway) and unhealthy. If the ai needs its production to run constantly at peak output, all you'd have to do to drastically slow it down is kill a single solar panel. The ai would then require more energy than it was producing. Since it had no energy stored away for a rainy day, metal extractors would quit making metal and factories and construction units would be slow to complete their build tasks for lack of metal AND energy. The resulting imbalances that these

cause are extremely erratic - going from 0 energy often up to max energy, and metal increasing and decreasing rapidly as factories finish units. Such complex behavior can be described as chaos theory based on the butterfly effect, and is best left for calculations on a Cray supercomputer. In short, it's too difficult to do *AND* it's not useful.

(This is starting to sound like the difficulty of riding a unicycle on a tightwire in an earthquake WHILE someone's shooting at you...)

How much resources should the ai be using to stay balanced?

There are no magic numbers here. But hitting 0 metal isn't good and hitting 0 energy is MUCH worse.

Early on, the ai's metal should rapidly decrease to just above 0. After that, it's personal preference and what you want the ai to do - so long as stored metal stays just above 0 and just below maximum.

If the ai does hit 0 metal, it should not stay there very long. A example of staying at 0 metal too long is what happens when the ai tries to build multiple guardians at once - that can keep the ai from building anything else for a long, LONG time (30+ minutes even). If it were working on only 1 guardian at a time, the 1st guardian could be finished and be useable before the other guardians are built. A base with 1 finished guardian is much better than a base with 4 unfinished guardians when enemy ground units come knocking. (An unhatched egg is no chicken!)

For the same reason, energy should be used at a rate so as not to use up all the energy in storage. Later on, some energy surplus should be made to cover possible energy "spikes" caused by d-gunning, laser weapons firing (including bertha-like weapons), and metal makers turning on.

If your ai's strategy relies heavily on lasers, MUCH more excess energy is needed than if your ai is almost laser-free. Since Core leans towards lots of units with lasers that need energy, Core should produce more energy than an equivalent Arm.

In the LONG term, the ai should use about as much metal as it's making and use LESS energy than it's making.

It's like the ai is totally ignoring my ai profile!

Cavedog installed some fail-safe ai routines in their executable so that if (more like WHEN...) the ai runs out of metal or energy then it builds whatever metal or energy producers that it can. This is actually quite bad for us ai profilers, because while the fail-safe routines are in use the ai is ignoring the ai profile weights on units. So a low energy fail-safe tells the ai to go build some solars, windmills, or tidal generators. A low metal fail-safe tells the ai to build metal extractors or metal makers. If the ai has already reached its max limits on energy providers or metal providers, it pretty much just gets stupid if the fail-safe modes kick in.

And if the ai is out of energy, having 10+ construction units all trying to build 10+ solars at once won't help. Another problem is that 0 energy causes the metal extractors and metal makers to quit making metal but NOT

quit draining energy. Once the metal reaches 0, the ai often mistakes the energy crisis for a metal crisis and builds more metal extractors (or worse yet, metal makers) to "fix" the problem, when it really needs a couple solars.

It starts building lots of solars even though it has max energy!

Another problem is the fail-safe routines don't always shut down when no longer needed, the ai may make solar after solar after solar even though it has max energy and much higher energy production than it is using. This could also mean you've weighted solars too high in your ai profile! A similar problem can happen with the metal extractors.

Why is the ai slow building a fusion reactor?

I believe there is a routine in the executable that checks to see if the ai has the metal and energy to build a particular unit - if not, it often doesn't build the unit -- even if it is the unit that it desperately needs. My offering of proof that such a routine exists is that the ai is "reluctant" to build high-cost items like a fusion reactor, geothermal plant, or advanced factories when it is low on resources. I haven't determined exactly how the ai decides that it doesn't have enough resources, but think it's resources it has stored plus the resources its making. Whatever calculation it uses, it doesn't seem to do a good job because it almost always ends up running out of resources. Perhaps that's because the ai doesn't consider what the other resource-consumers are doing (or WILL do!) and instead assumes that the one resource-consumer can have ALL the resources.

Despite this looking like a minor problem that can be corrected with a good ai profile, this can cause the most frustration.

How come the ai built 5 Advanced Aircraft Plants - I LIMITED it to just 3!

The ai doesn't count a unit towards its LIMIT maximum unless that unit is completed. So, if 2 advanced aircraft plants are complete, many more advanced aircraft plants may be started up to the limit of the ai's current basic construction aircraft. If the ai has 4 basic construction aircraft at the time, the probable maximum of advanced aircraft plants is 6.

I call this the "OVERBUILD BUG", but you'd never know about it unless you're messing with the ai profiles in the first place - that's why I didn't mention this problem earlier. I personally don't see this as much of a problem, EXCEPT with the moho metal makers on nonmetal maps - 1 of them is PLENTY enough, 2 or more is disaster (that's why I weight them VERY low...).

Otherwise, having the ai overbuild its base - exceeding the normal limits - is probably ok. The ai would most likely only do that if it already had a good resource base and probably would be able to support the extra

resource strain. If the ai wasn't doing well, the chances of this happening are unlikely at best - so it won't hurt the ai much then.

The difficult MATH side of ai profiles:

Now, you can plug numbers into your ai profile for weeks on end and not know why you don't get the desired results. Although the numbers the ai uses are only "suggestions" to it, over very long periods of time calculations can be done to determine the results. Even if you don't know the exact value, it's helpful to have an idea what the chances of a particular unit being made is. To do this, I introduce the term build or production probability - a percentage calculated by weights on each unit produced by a factory.

For instance, the Arm Vehicle plant has:
Construction Vehicles (weight=3 limit=4),
Flashes (weight=9 limit=no limit!),
Jeffys (weight=1 limit=0),
Samsons (weight=1.5 limit=no limit!),
and Stumpys (weight=0.5 limit=no limit!).

The total weights add up to 15, BUT since Jeffys are never made (limit=0), the actual weight total is 14. Build probability will be calculated by taking the weight of each unit and dividing it by the ACTUAL WEIGHT TOTAL.

The Construction Vehicle will have a 21.4% build probability.
The Flash will have a 64.3% build probability.
The Jeffy will have a 0% build probability, since the limit is 0 on them!
The Samson will have a 10.7% build probability.
The Stumpy will have a 3.6% build probability.

If you add up 21.4% + 64.3% + 10.7% + 3.6%, the total is 100% as it should be.

After the limit is reached on Construction Vehicles, the production probability changes. Total weight is now only 11 since Construction Vehicles are no longer a consideration.
The Flash will have a 81.8% build probability.
The Samson will have a 13.6% build probability.
The Stumpy will have a 4.5% build probability.

Adding 81.8% + 13.6% + 4.5% = 99.9% (with a rounding error to account for the missing 0.1%)
So, this also adds up to 100%.

What these numbers mean is that until the construction vehicles max out, the vehicle plant should make 6 Flashes, 2 Construction Vehicles, 1 Samson, and maybe a Stumpy - or another flash for every 10 units the vehicle plant makes.

It's very hard to tell if an ai will make more bulldogs than stumpies, because those come out of different plant types. It also gets complicated if there's 4 of one plant type but only 1 of another and 2 of a third type - to predict what the unit percent composition made by all the factories will be. However, the unit build probability for each factory type can be quite helpful in creating a more balanced attack force.

This same build probability can be applied to ALL types of construction units, even the commander. The reason to check construction unit build probabilities is an ai profile usually sucks because its RESOURCES and FACTORIES are not weighted at balanced levels. The basic construction units need the MOST careful build probability balancing. They are responsible for building much of a base.

UNFORTUNATELY, the ai fail-safe routines prevent the weights you put on the construction units from being anywhere close to accurate. This is where you have to use ridiculous numbers to get semi-acceptable results: like weight 9 (or more!) on geothermals.

The ai's clockwork build patterns:

Firstly, I tell you that the ai's build pattern is random. Now, I'm telling you it's actually clockwork. When, where, and what the ai builds is based on the map, what the ai has already built, and what the ai's current resources are. Even extreme weights on units, both very low (like 0.05) and very high (like 9) are ignored by the game if any overriding condition occurs (like fail-safe ai routines). If the ai is ALLOWED to build it AND the executable tells the ai that it NEEDS to build it NOW, that's what the ai will build regardless of how seldom it's SUPPOSED to be building it.

Anyway, this is the basic pattern that the build units follow:

Commander:

1. build at least 1 metal extractor (or metal maker if not possible)
2. randomly alternates between metal producers and energy producers
3. build process becomes a matter of ability - if energy and metal are in good supply, typically factories are built, otherwise mostly resources are built. (it's only during THIS step that the commander is likely to build anything it can.)
4. After at least 1 construction unit is build, the commander MOVEs back to the "center" of the base (somewhere near where it started often)
5. Following the MOVE, the commander is on repair patrol for the remainder of the game - if there's any construction units left.
6. If there's few construction units left (and few factories), the commander resumes step 3.

Basic Construction units:

1. If resources aren't high, randomly alternates between metal producers and energy producers.
2. build process is a matter of ability - if energy and metal are in good supply, typically factories are built, otherwise mostly resources are built.
3. Some random time later, the construction unit goes on repair patrol.
4. If much of the base is destroyed, the construction unit resumes step 2.

Advanced Construction units:

1. Almost always tries to build a moho mine or moho metal maker first -- by this point in the game, the ai is almost certainly low on metal.
2. Superweapons like berthas are often built next, even in preference over fusion reactors.
3. build process becomes random, but based slightly on need - moho mines/moho metal makers if low on metal, otherwise almost anything.

4. Some random time later, the advanced construction unit goes on repair patrol.
5. If much of the base is destroyed, the advanced construction unit resumes step 1.

How come my ARM ai *SMOKES* my CORE ai?

OR, how come my CORE ai smokes my ARM ai really bad?

ARM's units, as a whole, move quicker than similar (if there are any!) CORE units. This applies to construction units as well. One possible reason why the ARM ai seems to have an advantage over a CORE ai is CORE takes just a little longer between each building project - therefore, after 10 minutes of being a little slower each time - CORE ends up behind in production. Note: That's only my theory. :P

A simpler reason is that you assumed ARM and CORE could be told to build alike with the same results. CORE's units are often more expensive, so CORE may end up out of both energy and metal while ARM has plenty of both - despite almost identical metal/energy incomes.

If you spend more time "tweaking" one side than another, you *SHOULD* expect this result. If you play ARM more, you'd probably know what a good attack force for ARM is but might not for CORE -- so although the CORE builds lots of attacking units, because it doesn't build a good MIX of attacking units, it seems to lose more battles than it should. If you really like one side over the other, you may unknowingly "stack the deck" in its favor.

Another reason for the imbalance is what I call the "Coffee-Break" bug, where units stop moving after killing all immediate enemy units. If the attack force is under attack by a guardian or punisher, it's likely to take heavier losses if ARM than CORE because CORE's units *generally* have more armor.

Ok, I caught all of that. But my ai still seems slow to buildup!

There's something subtle going on with the ai profiles, despite their apparent simplicity. It's not so simple that you can weight flashes and vehicle plants really high and end up with a (good) flash-rushing ai. Almost every change affects something else! If vehicle plants are weighted to 9, that will mean that the construction units will build these in greater preference over resource providers. So the ai might end up with low resources meaning it'll have very few vehicle plants built (since it had little resources to build with) - even by the 20 minute mark.

The computer isn't slowed down much by a LONG ai profile - like one over 20kb long. But it probably doesn't hurt to keep the ai profile as short as possible, leaving comments about how the ai works out of the ai profile itself. You could make an ai template laying out all the unit limits, and copy and paste from it into your ai profile. That way you don't have to memorize unit abbreviations.

Weights seem to be more effective if they're put near the top of an ai

profile. Even Cavedog's ai profiles seem to follow this logic. Perhaps the ai is more likely to act on the first thing it reads in the ai profile!

testing, Testing, TESTING!

You cannot know if an ai will be any good unless you test it. Playing against it over and over again may seem like a decent testing method, but it's almost impossible to play the same way twice. If your testing is to determine if minute changes result in an improvement for the ai, that will most likely be too well hidden in random variables introduced by such a testing method. Plus, if you're playing it - you're not spending enough time watching what it is doing. On the other hand, the ai may play BETTER against another ai than a player - who won't do what it's expecting. So even if an ai is good versus other ai's, it may suck against people! Get the ai good versus another ai before testing against people, or they may just laugh at you.

Once the ai gets good versus the ai, then test on live victims - uh, I mean people... ;) Don't be surprised if a good player sends you back to the drawing board when your ai proves less successful than you hoped, and don't be afraid to ask people what the ai should've done to be even better -- even if they lost to it! BTW, treat people kindly for helping you test your ai and always try to thank them for their time. (Otherwise, it may get harder and harder to find lab rats. ;)

Generally, I test on big maps that can handle a lot of players on it at once. For my default.txt ai profile I use Greenhaven most because it is both big and loads fast.

I have quite a few ways to test the ai - and most are unbiased:

1. Rather than have multiple computers set up, I can play skirmish on my own computer with 9 AI's running around. I am allied with 4 and the other 5 are allied as well. Often I play 4 Arm vs. 5 Core, to see the disadvantages against the two. This only lets me test 1 ai at a time, but gives me a good baseline for when the ai launches an attack, how well it builds resources, how quickly it builds base defenses, and how well it fights off attacks.

2. I crossbreed ai's together - so I can play my Arm ai vs. someone else's Core ai or vice versa. This is not a very scientific test - because of the differences of ARM and CORE - but is most informative. ;)

3. I play with my ai vs. a different ai over the internet with somebody. We are teamed with an ai (typically mine, so we can see the battle through to the end :P) and stand behind the ai's base and hardly build anything - because it would increase lag. We even have cheats enabled so we can use the +VIEW # command to see how the ai's resources are doing.

4. On a Lan, I don't normally play just 1 ai vs 1 ai - I team 2 ai's of the same kind vs 2 ai's of the same kind - can't do more, because the game won't handle over 10 players (even if they're watchers they count as players). All but 1 or 2 players are watchers. If ALL players are watchers, the mission will immediately end thinking the human players have all died.

I'm doing a lot of testing, but what am I looking for to know what to change?

Running a lot of tests may help you improve an ai profile, but if you don't know what you're looking for (or AT, as the case may be) the tests are probably a waste of time. Heavy useage of the +view # cheat is a MUST - otherwise you can only guess what the energy and metal state of an ai is. Factories quit producing mobile units when less than 100 metal is in storage. If metal makers are blinking it usually means metal consumption is above metal production. An attack force that runs around in circles seems to be the result of metal/energy shortages for that ai. An ai that does great because it always starts with 10k resources might be really wimpy with 1k resources - ai's usually have a resource range that is "optimal" for them, normally this is near the 10k mark. Large increases in energy consumption often means the ai either fired a berth, turned on a moho metal maker, or started building energy-intensive unit/s. Likewise, large metal consumption increases often means something like a guardian or advanced plant is just being started. Temporary increases in metal or energy production almost always means the ai is reclaiming trees, rocks, or debris on the map. Large, sudden permanent increases in energy means a geothermal or fusion reactor was just completed. It's notable that the ai can turn on and off a few units that has an on/offswitch even WHILE the unit is being built. So the moho metal maker can start providing extra metal (and tremendous energy drain) the VERY moment it is started - long before it is finished. It's one of the ways the ai cheats, but ai profiles alone won't stop that.

If you're in a skirmish game with 9 ai's (all ARM, for instance) and NONE of them build a advanced vehicle plant before the 20 minute mark - and you WANT them to, you'll probably need to increase the weight on the advanced vehicle plant. The case may be that it's because all 9 ai's stayed out of (or very low on) metal and/or energy and could have nothing to do with the weights on the advanced vehicle plant.

For the default.txt ai profile, whenever the ai builds too many solars and metal extractors before its first factory it usually loses to one that didn't. Also, if its first factory is an aircraft plant it usually loses as well.

Important ai information to record:

- 1.What the ai's overall metal/energy production for a 10-minute game was - with SPECIAL attention to how much it wasted. Resources are probably the most important - if the ai could get another 1k of metal out of the ground in 10 minutes, it'd be much better. Energy is probably in the 60k to 100k range in 10 minutes, with metal in the 4k to 7k range on a nonmetal map (Greenhaven, in this case.) Naturally, if the ai was attacked and lost outlying metal extractors its metal production would be considerably lower.
- 1.WHEN the ai's launched their first attack (notice I said launched instead of when the attack arrives at its destination - which varies GREATLY with the size of the map and proximity to an enemy base). An ai that can launch an attack in 5 minutes is possible - with 7 minutes still being ok, even with low resources (1k and nonmetal map). On a metal map with 10k resources, this time can shrink down to as low as 3 minutes - but 5 minutes is still more common.
- 2.How quickly level 2 units are added to the attack forces. Typically, this occurs in the 10 minute to 20 minute timeframe.
- 3.How quickly the the ai starts building level 2 and level 3 base

buildings. Commonly, this is in the 10 to 30 minute timeframe.

4.How built-up the ai's base defenses were by the 10, 20, and 30 minute mark - and even later. The ai can be made to build guardians and sentinels ASAP, but that will badly hurt resource production.

6.How quickly can the ai reach the unit max. For a unit max of 250, the winning ai in a test can reach that amount in as little as 25 minutes - even on a nonmetal map. More commonly though, unit max is reached later - or even never if you set your ai profile limits very low. Reaching unit maximum too early can result in the ai having lots of level 1 units but few level 2 or level 3 units. Just how GOOD is 100+ flash tanks anyway? Wouldn't 30 flash tanks (or fewer) at a time be just as effective?

7.At what point does the winning ai manage to reach a mature level 3 economy - with redundant factories, fusions, base defenses, and metal producers. This is the point where even nuking the ai may not slow it down -- if it was at unit max, nuking it will give it more unit slots to build more mobile attack units! The earliest I've seen an ai reach this point was just under the 30 minute mark, but even the 1 HOUR mark is acceptable. If the ai doesn't use nuke silos, lots of laser towers, berthas, or moho metal makers it doesn't need 1000+ energy production to produce good ground-force attacks. Metal is vital, but if the ai uses cheaper but effective units the lack of resources will go entirely hidden from whoever's on the RECEIVING end. ;)

8.How early does the ai get a "STEAMROLLER" attack force that isn't stopped even by base defenses like guardians and sentinels. The attack force should be resupplied with new units quickly (like 1 every 15 seconds) to be effective in the meat-grinder of close combat. Often, it's not the SIZE of the attack force that causes it to win so much as how long its units last and how quickly they are replaced. ARM should have quicker unit replacements than CORE simply because ARM's units generally have lower armor (and cost less) and tend to die quicker.

9.What's missing? What could the ai add to its base or attack force that would save the day.

10.What isn't helping as much as it should? Maybe lowering the weight and limit on that unit will make the ai more effective. Try eliminating that unit and see if the ai is better without it.

It's hard to tell if an ai profile is improving just by testing it against itself. So, test one side at a time against the other side WITHOUT changing the other side. This can be done by copying and pasting all ARM entries from 1 ai and all CORE entries from another ai into a single ai file. It really pads the ego to watch your CORE ai totally mow through Cavedog's original (v3.1 patch even) ARM ai. Whoever said CORE can't rush effectively probably hasn't seen a Storm and/or Instigator rush! If your 4 CORE ai's can kill 5 original ARM ai's in under 20 minutes on Greenhaven with only 1k resources, consider your ai at least decent! (Mine can *sometimes* do that.) BTW, It's wise to team up with the (probable) winning side.

What kind of ai profile *DOES* a map need?

Most maps fall under the standard general categories of Default, SeaBattle, AirBattle, and Metal - but some need specific ai profiles made just for them. An extreme example is an all-water map with very little land. Seabattle assumes that there is land to build a good sized land base, particularly solars, metal extractors, hovercraft and aircraft plants. Lacking that, the ai will build very slowly - if at all - and will be easily beaten. The basic ai profiles categories are too vague to cover

many types of maps. So things to consider when making an ai profile are:

- 1.How large is the map? (can't flash rush effectively on a 30 x 30 land map)
- 2.How much water (by percent) is the map? (possibly a naval ai map?)
- 3.Is the land and water alternating making either form of attack difficult or impossible?
- 4.How plentiful is metal on this map?
- 5.Is there any geothermal vents on the map?
- 6.How flat/hilly is the map?
- 7.Are trees so thick that they must be reclaimed to have room to build?
- 8.Is there lots of reclaimable rocks around?
- 9.Does the map have consistantly high winds or gusty winds?
- 10.Because of some feature on the map, does one particular unit (or type) have a significant advantage over the others? (such as narrow land passes, steep slopes, or thick trees)

1.An example of a large map that isn't a typical default ai profile is Greenhaven. It's TOO big for a flash rush to reach an enemy base early enough to make winning easy - not in a 1 vs 1 game, anyway. Such a map may require the ai to build slightly more base defenses than usual and rely on long-ranged attack types (Merls, Berthas, aircraft) to effectively destroy the enemy base.

2.Gods of War is a good example of a some-land, lot-water map that's well suited for a naval ai.

3.The best examples of this is Over Crude Water (a metal map) or Evad River Confluence. Both maps have water acting as a barrier to large land attacks. Neither a land attack or naval attack can easily reach the enemy base. This forces relying on hovercraft and aircraft to strike the enemy base. So they're both air battle ai's.

4.Some maps have so little metal - such as Painted Desert - that the ai either must spread out all over the map or build metal makers. And the ai cannot manage metal makers very well, always wanting to build way more metal makers than it has energy to run. Also, its construction units will suicidally wander into enemy bases trying to build a metal extractor on unused metal spots.

5.Geothermal plants are vital stepping stones to getting fusion reactors for players, but for the ai who shouldn't build such an energy-hungry base they are even MORE vital. An ai can have a large base totaling 50 buildings and only need 15 solars and 2 geo's to run it. On nonmetal maps, where only 1 geo can be built on a vent, the ai should have a max limit of 2 to 6 geothermal plants (depending on the total number of vents on the map) so that it doesn't wastefully and suicidally send its construction units into the enemy territory to build a geo on thermal vents there. On metal maps, the max limit on geos can be as high as 10 without any problems, because up to 4 geothermal plants can be built on each vent.

6.A very hilly map makes for lots of killzones and choke points that by the middle of the game render ground forces useless. Plus a hilly map may have very little room to build a base. A few short-ranged LLT's and guardians are best as base defenses because line-of-sight is so restrictive. The guardians can shoot over some of the hills, and the LLT's can guard passes and enterances to bases.

7.Maps with wall-to-wall trees make big bulky units less useful, so often the level 1 kbots and aircraft are the only units that can get around effectively.

8.Unless there's so many rocks that they are in the way, this has little effect on an ai. However once the commander goes into repair and reclaim patrol mode, the rocks may be the only thing that keeps the ai's factories producing more units.

9.Wind generators should be used instead of solars if the wind is

consistently strong, or half solars half wind generators if the wind gusts are strong enough (like +20 or more energy).

10. Metal Isles has large islands with very tall defensive walls. Although a huge naval force may take control of the water around the island, it will be unable to shoot into the middle of the island with cruisers and battleships because of the size of the island and the high walls. Missile frigates are more useful for attacking such bases, but cannot be made the bulk of the naval force or it will surely lose to a naval force made primarily of cruisers and battleships. Also, since the map is rather large, significant amounts of anti-aircraft ships are needed with the naval force.

The flash tank excels on Metal Heck - it is fast and cheap, allowing for early base attacks. Plus it is small, making it hard to hit and better able to move around debris.

About MY ai...

My ai has no balance of base defense vs attacking force - it is almost pure attacking force. It builds a token base defense consisting of a couple berthas, a very few guardians, some flackers, and quite a few defenders. Only the defenders are built early on - they are a holdover from the long-ago popular strat of c-napping the commander before he could build much. I figure defenders are cheap enough that they won't slow production down to a crawl while still providing some defense. It might seem that the base is very vulnerable to an early rush, but that's probably when it's the most densely defended by mobile units. Later on, when the ai's forces are attacking elsewhere the ai's base can be pretty easily stormed with ground forces.

Getting a well-defended base early on is of low priority for an ai. To do so will slow or even stop its exponential growth for many minutes. Let the mobile units defend the base early on, and let the construction units build good base defenses only if they have nothing better to do.

My ai's weights go from 0.05 to 9, where a weight of 1 is assumed to be average. A weight of 9 is used to almost force the ai to build that unit instead of anything else and build it quick. A weight of 0.05 is for stuff I don't want the ai to build until AFTER other things are built. When the unit LIMITS are reached for high-weight (1-9) items, then and only then should the very low-weight (0.05-0.2) items be built. I tried a weight of 0.01 for units, but they won't get built at all - the ai rounds weights that low down to 0. The same weight and limit tricks can be used on attacking units to assure that you get frontline units quickly but not too many - and that you later get supporting units in decent quantity once the ai's attack force is large enough to defend them. An example of this is Flashes for frontline units and Merls for supporting units. Flashes are good, but too many just get in the way of one another. Merls are also good, but putting them on the frontline will make them seem useless.

Putting an upper limit on every base building is a must, otherwise once the ai reaches unit max and after each attack the ai will replace some or even all of its attacking units that got killed with base buildings. This results in the ai's attacking force growing smaller and smaller. At that point, nuking some of the ai's base literally pisses it off because it is freed up to make more attacking units! For the same reason, even the attacking units themselves need an upper limit to prevent overpopulation. More than 30 flash tanks just get in each other's way being unable to fire

on targets. For my ai, I presume that the game's unit max is somewhere in the 150 to 300 range. If the unit max is lower, the ai will become pacifistic (not attacking with ground forces) much earlier. On the other hand, if the unit max was 500, the ai would never really utilize the extra unit slots to make its attack force OR base huge.

My rule of thumb for the ai is simple - for ordinary nonmetal maps, the default.txt ai will suffice. But if there's anything unusual in the map, the ai needs to be modified to take advantage of it - be it high winds, narrow passes, lots of water, scarcity of metal, even a change in unit max. Any map can benefit from using a modified ai profile instead of the original ai profiles in TA:CC, but some maps almost demand a special ai profile for the ai to be effective at all!

TA Weapons Creation

unknown

I like to explain how a weapon file in TA is build up, and I hope it is of any use to any new unit designer. I took all of my information from the original Cave Dog weapons file.

The actual weapon file in the weapon directory of the unit (ufo)

Weapons.tdf - Here you place the miscellaneous weapon types

Types of weapons

There are different weapons but any of the weapons must fit in one of these basic categories.

ballistic	Weapon is fired according to a ballistic path using gravity
lineofsight	Weapon is fired in a straight line, gravity does not effect path
dropped	Weapon is dropped in order to use it, typically a bomb but could be a chemical

Important values and behaviour of the weapon.

ID	A unique value in the range 0-255 which identifies the weapon
range	Coverage in pixels what the protection umbrella is for weapons that shoot other weapons
noexplode	No explosion when weapon impacts target
reloadtime	Seconds between shots (floating point allowed)
energypershot	Energy consumed per shot, most use none
weapontimer	How long weapon is active in seconds (floating point allowed), trajectory weapons use 0 so it is calculated
noautorange	When set the weapon will not detonate at range automatically, used mostly for heavy rockets

weaponvelocity	Maximum attainable weapon velocity in pixels/second
weaponacceleration	Expressed in pixels/second/second
turnrate	Used for guided weapons, is in angular units (0-64k)/second
areaofeffect	The total area that receives that damage, one impact per unit in the area
edgeeffectiveness	The percentage (1.0 = 100%) of the damage that is inflicted at the edge of the area of effect. Used for drop-off

Weapon Characteristics

turret	Weapon must be deployed from a turret with a 360 deg rotation and pitch
firestarter	Weapon will cause fires, expressed as a %, where 100% guarantees a fire
unitsonly	Weapon will only detonate on enemy units as opposed to obstructing terrain
burst	How many repeat times a weapon fires in one burst, ie. Flamethrower
burstrate	The time delay when in burst mode between events
sprayangle	Maximum deviation from the straight line to the target the weapon strays, used for burst weapons
randomdecay	Maximum time delta that burst weapon will randomly decay at end of path
groundbounce	Weapon will not detonate with the ground but instead bounce
flighttime	The time the unit will fly for after it enters the second phase of operation, used for starburst missiles
selfprop	Weapon is self propelled with a burn time described by flighttime
twophase	Indicates weapon operates in two phases
weapontype2	Describes another weapon that the weapon turns into in the second phase
guidance	Indicates that weapon is guided and uses the turn rate above to track enemy unit
tracks	When set the weapon will track a moving target after a weapon conversion
waterweapon	Weapon is meant to travel through water
burnblow	Weapon will detonate when it comes to the end of its range
accuracy	Amount of accuracy in 64K deg that weapon is good for, 0=100%
tolerance	Amount of accuracy weapon will use when aiming, most are default 0
aimrate	How fast (on average) the weapon aims, in 64K deg / sec. Used by UnitView.
startvelocity	Weapon will start at this velocity instead of 0
minbarrelangle	The minimum angle (in degrees) the barrels can point, used in ballistic calculations

Special weapon stuff

paralyzer	Weapon will stun the enemy for a length of time described in the damage field, time=ticks.
-----------	--

Looks of a weapon

model	3D model to use as this weapon
color	Color of beam weapon from the game palette
color2	Color to use on the beam weapon to make it look better and cooler
smoketrail	Indicates whether or not a weapon will emit a smoke trail
smokedelay	Smoke dispersal interval expressed in seconds
startsmoke	Draw a puff of smoke when the weapon fires
endsmoke	Draw a puff of smoke when weapon terminates
rendertype	Type of rendering system to use, 3D model, bitmap, etc.
beamweapon	Weapon is a straight beam weapon like a laser
explosiongaf	.GAF file that the explosion art is in
explosionart	name of animation sequence for explosion
waterexplosiongaf	.GAF file that the water explosion art is in
waterexplosionart	name of animation sequence for water explosion
propeller	if the model has a propeller that spins

The sounds it makes

soundstart	Sound to make when the weapon fires
soundhit	Sound to make when the weapon detonates (if the weapon detonates)
soundwater	Sound to make when the weapon hits the water
soundtrigger	Make the weapon sound when the weapon fires in burst mode

Weapon Controls

commandfire	This weapon will need to be expressly fired by the user each and every time it is used, i.e. a Dgun
-------------	---

Needed resources

Here you describe the amount of metal and energy it takes to fire the weapon if applicable

energy	Amount of energy needed
metal	Amount of metal needed

Total Annihilation: Kingdoms

Directory Structure and File Formats

Directory Structure

```
D\CAVEDOG
 00000407.256
 00000409.256
 0000040a.256
 0000040c.256
 00000410.256
 binkw32.dll
 boneyards.hpi
 boneyards2.hpi
 bymaia.dll
 Cartographer.exe
 ChooseRenderer.exe
 clcd16.dll
 clcd32.dll
 clokspl.exe
 data.hpi
 dplayerx.dll
 drvmgt.dll
 english.hpi
 INSTALL.LOG
 Jersey.hpi
 Keys.TDF
 kingdoms.esk
 Kingdoms.exe
 KINGDOMS.icd
 KINGDOMS.ID
 Kingdoms.key
 kingdoms.mmz
 maps.hpi
 missions.hpi
 MP3DEC.ASI
 mss16.dll
 mss32.dll
 mssb16.tsk
 patchw32.dll
 ReadMe.htm
 ReadMe.txt
 RendererHelp.txt
 restart.exe
 Rover.dll
 secdrv.sys
 sections.hpi
 Startup.tdf
 terrain.hpi
 Uninst.isu
 upgrade.dll
 V2Rocket.hpi
 v3readme.txt
```

V3Rocket.hpi

Boneyards
server.byd

Help

Help.def
Help1.htm
Help10.htm
Help11.htm
Help12.htm
Help13.htm
Help14.htm
Help144.htm
Help146.htm
Help147.htm
Help15.htm
Help16.htm
Help17.htm
Help18.htm
Help19.htm
Help20.htm
Help21.htm
Help22.htm
Help23.htm
Help24.htm
Help25.htm
Help26.htm
Help31.htm
Help36.htm
Help37.htm
Help39.htm
Help4.htm
Help41.htm
Help42.htm
Help43.htm
Help44.htm
Help5.htm
Help6.htm
Help7.htm
Help8.htm
Help9.htm
_Help.htm

Images

a1.png
a2.png
a3.png
a4.png
a5.png
a6.png
a7.png
a8.png
ButtonBattles.png
ButtonBoneyards.png
ButtonCancel.png
ButtonCrusades.png
ButtonEditHouse.png
ButtonEditProfile.png
ButtonGathering.png
ButtonHelp.png
ButtonHostBattle.png
ButtonHostGathering.png

ButtonHouse.png
ButtonLadder.png
ButtonLocator.png
ButtonMetaGame.png
ButtonNews.png
ButtonNewsNInfo.png
ButtonOK.png
ButtonPagePeople.png
ButtonPeople.png

ButtonProfile.png
ButtonQuickplay.png
ButtonRegister.png
ButtonSearch.png
ButtonWarRoom.png
CrusadesMap.png
CrusadesRoom.png
CrusadesRoomButton.png
end_k2.png
Help.png
Login.png
MovieButton.png
NnIEditProfile.png
NnIEditRegister.png
NnILadders.png
NnINews.png
NnIProfile.png
NnISearchLadder.png
NnISearchNews.png
OverviewButton.png
PlayerPage.png
PlayerPageSent.png
PlayerProfile.png
ReconButton.png
Register.png
SearchHelp.png
SystemMessage.png
t1.png
t2.png
t3.png
t4.png
t5.png
t6.png
t7.png
t8.png
Update.png
v1.png
v2.png
v3.png
v4.png
v5.png
v6.png
v7.png
v8.png
WarBattle.png
WarBattleForming.png
WarBattleInProgress.png
WarCreateBattle.png
WarCreateGath.png
WarGath.png
WarGatheringInfo.png
WarLocator.png
WarPagePeople.png
WarPeople.png
WarPlayerInfo.png
WarRoom.png
WarSearchLocator.png
z1.png
z2.png
z3.png
z4.png
z5.png
z6.png
z7.png
z8.png

—Ladders

Ladders.def

—Metagame

Aramon.png
aramonShield.png
Borders.png

- ContestedMap.png
- CrusadeProf1.png
- CrusadeProf2.png
- CrusadesStandard1.png
- CrusadesStandard2.png
- Darien.def
- HonorMap.png
- MetaMask.png
- PreInit.jje
- Taros.png
- tarosShield.png
- TerrorMap.png
- Veruna.png
- wdhit.jje
- Zhon.png

Profile

- Achievements.htm
- MGPersonal.htm
- MGPersonal_NoName.htm
- MGProfile.def
- personal.htm
- Personal_NoName.htm
- Profile.def
- TAK_CRUSADES.htm
- TAK_CRUSADES_NOJOIN.htm
- TAK_DUEL.htm
- TAK_history.htm
- TAK_NATIONS.htm
- TAK_NATIONS_ARAMON.htm
- TAK_NATIONS_TAROS.htm
- TAK_NATIONS_VERUNA.htm
- TAK_NATIONS_ZHON.htm
- TAK_overview.htm
- Tak_rec0.htm
- TAK_reconhistory.htm
- TAK_WTA.htm

Templates

- gameform.htm
- gameprog.htm
- gathering.htm
- MGgameform.htm
- MGgameprog.htm

docs

- AFlyBuild.txt
- ASiege.txt
- TKamRat.txt
- TRictus.txt
- VLightHs.txt
- ZSWolf.txt

GC

- Free Month of Internet .lnk
- Frontier.ico
- index.html
- Left.html
- Main.html
- Top-Left.html
- Top-Right.html

images

- backgrounds
 - Back00.jpg
 - Left.jpg
 - Top-Left.jpg

- banners
 - FGCLogo.gif

- buttons
 - Dev-0.gif


```
Dev-U.gif
Fgc-O.gif
Fgc-U.gif
Gsn-O.gif
Gsn-U.gif
Pl-O.gif
Pl-U.gif
Tak-O.gif
Tak-U.gif

Maps
Movies
  Gui
    GIRL4.BIK
    GIRL5.BIK
    GIRL6.BIK
    GIRL7.BIK
    KNIGHT4.BIK
    KNIGHT5.BIK
    KNIGHT6.BIK
    KNIGHT7.BIK
    Loadscreen.bik
    machine4.bik
    machine5.bik
    machine6.bik
    machine7.bik
    SNORT4.BIK
    SNORT5.BIK
    SNORT6.BIK
    SNORT7.BIK

MPlayer
  mplaynow.exe
  MPLAYNOW.INI
  Readme.doc
  MPLAYNOW
    MPNETSUE.EXE
    RNAPH.DLL
    URL.DLL
  MPLAYER
    setup.exe

Music
  track1.wav
  track12.wav
  track13.wav
  track14.wav
  track15.wav
  track16.wav
  track17.wav
  track20.wav

  track3.wav
  track4.wav
  track5.wav
  track6.wav
  track7.wav
  track8.wav
  track9.wav
```

V2 Rocket.hpi Contents

D:\V2ROCKET.HPI

- Anims
 - TitleScreen.jpg
 - TitleScreen.tsf
- Guis
 - Mainmenu1.gui
 - VisualOptions1.gui
- maps
 - Abnar's Terrace.crt
 - Abnar's Terrace.ota
 - Abnar's Terrace.tnt
 - Abnar's Terrace.txt
 - Loch Brynn.crt
 - Loch Brynn.ota
 - Loch Brynn.tnt
 - Loch Brynn.txt
- Objects3D
 - verball.3do
- Scripts
 - VERBALL.COB
- translate
 - abnar's terrace.tdf
 - loch brynn.tdf
- units
 - ARAAT.FBI
 - ARACAN.FBI
 - arapult.fbi
 - LIFDEER.FBI
 - Lifdeer2.fbi
 - tarship.fbi
 - VERBALL.FBI
 - VERGOD.FBI
 - VERMORT.FBI
 - VERPULT.FBI
 - ZONFLIES.FBI
 - ZONGOD.FBI

V3Rocket.hpi Contents

V3ROCKET.HPI

```
├── ai
│   └── default.txt
├── anims
│   ├── bigsmoke_1555.taf
│   ├── bigsmoke_4444.taf
│   ├── bluefire_1555.taf
│   ├── bluefire_4444.taf
│   ├── dieselflame_1555.taf
│   ├── dieselflame_4444.taf
│   ├── manabomb_1555.taf
│   ├── manabomb_4444.taf
│   ├── steam_1555.taf
│   ├── steam_4444.taf
│   ├── titlescreen.jpg
│   └── titlescreen.tsf
├── BuildPic
│   ├── Arafly.jpg
│   ├── Aragren.jpg
│   ├── ARASIEGE.JPG
│   ├── aratrans.jpg
│   ├── tarang.jpg
│   ├── TARCAN.JPG
│   ├── TARHAND.JPG
│   ├── Tarkam.jpg
│   ├── verbal.jpg
│   ├── vercen.jpg
│   ├── verlight.jpg
│   ├── vermer.jpg
│   ├── zonamoe.jpg
│   ├── zonbar.jpg
│   ├── zonswamp.JPG
│   └── Zonwolf.jpg
├── gafs for release
│   └── loadingc.pcx
├── WeaponPic
│   ├── deathswordpu.jpg
│   ├── deathswordsb.jpg
│   ├── deathswordsbh.jpg
│   ├── FireBallGPU.jpg
│   ├── FireBallGSB.jpg
│   ├── FireBallGSBh.jpg
│   ├── GrenadeConcussionPU.jpg
│   ├── GrenadeConcussionSB.jpg
│   ├── GrenadeConcussionSBh.jpg
│   ├── GrenadeExplosivePU.jpg
│   ├── GrenadeExplosiveSB.jpg
│   ├── GrenadeExplosiveSBh.jpg
│   ├── GrenadeIncendiaryPU.jpg
│   ├── GrenadeIncendiarySB.jpg
│   ├── GrenadeIncendiarySBh.jpg
│   ├── LightHouseStunPU.jpg
│   ├── LightHouseStunSB.jpg
│   └── LightHouseStunSBh.jpg
├── bitmaps
│   └── initscreen.pcx
├── CanBuild
│   └── arabuild
│       ├── araat.tdf
│       ├── aracastl.tdf
│       ├── arakeep.tdf
│       └── aralode.tdf
```

arassh.tdf
aratre.tdf
arawall.tdf
arawar.tdf

—AraCastl

AraBow.tdf
AraBroad.tdf
AraBuild.tdf
AraCan.tdf
AraPal.tdf
AraPries.tdf
AraSiege.tdf
AraSmith.tdf
AraSpy.tdf

—AraFly

AraAt.tdf
AraCastl.tdf
AraKeep.tdf
AraLode.tdf
Arassh.tdf
AraTre.tdf
AraWall.tdf
ARAWAR.TDF

—arakeep

araarch.tdf
arabuild.tdf
arafast.tdf
araknigh.tdf
arapult.tdf
arasword.tdf

—araking

araat.tdf
aragod.tdf
arakeep.tdf
aralode.tdf
arangate.tdf
aratrans.tdf
arawall.tdf

—AraPries

AraDrag.tdf
AraFly.tdf
ARAGREN.TDF
AraMana.tdf

—tarcastl

tarblack.tdf
targarg.tdf
tarship.tdf
tartb.tdf
tartroop.tdf
tarzom.tdf

—TarDung

TarArch.tdf
TarBeak.tdf
TarCan.tdf
TarFire.tdf
TarHand.tdf
TarTb.tdf
TarWitch.tdf

—TarHell

TarDemon.tdf
TARKAM.TDF
TarKnigh.tdf
TarLich.tdf
TarMage.tdf

TarMind.tdf
TarPries.tdf
TarSpout.tdf
TarTb.tdf

—tarnecro

tarcage.tdf
tarcastl.tdf
tardung.tdf
targod.tdf
tarhell.tdf
tarlode.tdf
tarngate.tdf
tarwall.tdf

—tarprie2

npctemp.tdf

—tarpries

tarang.tdf
tardrag.tdf
tarmana.tdf

—tartb

tarcage.tdf
tarcastl.tdf
tardung.tdf
tarhell.tdf
tarlode.tdf
tarsh.tdf
tarwall.tdf

—verasy

verflag.tdf
verharp.tdf
verman.tdf
verscout.tdf
vertrans.tdf
vertre.tdf

—vercastl

verball.tdf
verbers.tdf
vercen.tdf
vercrus.tdf
verknigh.tdf
verliege.tdf
verlihr.tdf
vermusk.tdf

—verflag

verasy.tdf
verfltwr.tdf
verkeep.tdf
verlode.tdf

—VerKeep

VerArch.tdf
VerLiege.tdf
VerMer.tdf
Verpar.tdf
VerPult.tdf
VerSword.tdf

—VerLiege

VERASY.TDF
VerAt.TDF
VerCastl.TDF
VerKeep.TDF
VerLight.tdf
VerLode.TDF
VerMort.tdf
VerTower.TDF

- VerWall.tdf
- verlihr
 - verbal.tdf
 - verdrag.tdf
 - vermana.tdf
 - verpill.tdf
- vermage
 - verasy.tdf
 - verat.tdf
 - vergod.tdf
 - verkeep.tdf
 - verlude.tdf
 - verngate.tdf
 - verwall.tdf
- ZonHand
 - ZonBat.tdf
 - ZonFire.tdf
 - ZonGob.tdf
 - ZonLode.tdf
 - ZonSwamp.TDF
 - ZONTER.TDF
 - ZonTrain.tdf
 - ZonTroll.tdf
- zohunt
 - zonfire.tdf
 - zonglyph.tdf
 - zongod.tdf
 - zonhand.tdf
 - zonlode.tdf
- zohurt
 - zonfire.tdf
 - zonglyph.tdf
 - zonhand.tdf
 - zonlode.tdf
- zonlord
 - zondrake.tdf
 - zonflies.tdf
 - zongiant.tdf
 - zonlode.tdf
 - zonorc.tdf
 - zonroc.tdf
 - zonsham.tdf
 - zontrain.tdf
- zonsham
 - zonamoe.tdf
 - zonbar.tdf
 - zondrag.tdf
 - zonmana.tdf
- ZonTrain
 - ZONBASIL.TDF
 - ZONGLYPH.TDF
 - ZONGRYP.TDF
 - ZONHAND.TDF
 - ZONHARP.TDF
 - ZONKRAK.TDF
 - ZONLODE.TDF
 - ZONLORD.TDF
 - ZONWOLF.TDF
- CanBuildCB
 - AraBuild
 - AraAt.tdf
 - AraCastl.tdf
 - AraKeep.tdf
 - ARALODE.TDF

ARASSH.TDF
ARATRE.TDF
AraWall.tdf
ARAWAR.TDF

AraCastl

AraBow.tdf
ARABROAD.TDF
AraBuild.tdf
ARACAN.TDF
AraGren.tdf
ARAPAL.TDF
AraPries.tdf
AraSiege.tdf
ARASMITH.TDF
ARASPY.TDF

AraFly

ARAAT.TDF
ARACASTL.TDF
ARAKEEP.TDF
ARALODE.TDF
AraSiege.tdf
ARASSH.TDF
ARATRE.TDF
ARAWALL.TDF
ARAWAR.TDF

AraKeep

ARAARCH.TDF
AraBuild.tdf
AraFast.tdf
AraKnigh.tdf
AraPult.tdf
Arasword.tdf

AraKing

ARAAT.TDF
ARAKEEP.TDF
AraLode.TDF
AraNGate.TDF
ARATRANS.TDF
AraWall.TDF

AraPries

AraDrag.tdf
AraFly.tdf
AraGod.TDF
AraMana.tdf

AraTrans

ARAARCH.TDF
AraFast.tdf
Arasword.tdf

TarCastl

TARBLACK.TDF
TarGarg.tdf
TARSHIP.TDF
TarTb.tdf
TARTROOP.TDF
TarZom.tdf

TarDung

TarArch.tdf
TARBEAK.TDF
TarCan.tdf
TarFire.tdf
TarHand.tdf
TarTb.tdf
TARWITCH.TDF

TarHell

TARDEMON.TDF
TARKAM.TDF
TarKnigh.tdf
TARLICH.TDF
TARMAGE.TDF
TARMIND.TDF
TarPries.tdf
TARSPOUT.TDF
TarTb.tdf

—TarNecro

TarCage.tdf
TarCastl.tdf
TarDung.tdf
TarHell.tdf
TARLODE.TDF
TARNGATE.TDF
TarWall.tdf

—TarPrie2

NPCTEMP.TDF

—TarPries

TARANG.TDF
TarDrag.tdf
TARGOD.TDF
TarMana.tdf

—TarTb

TarCage.tdf
TarCastl.tdf
TarDung.tdf
TarHell.tdf
TarLode.tdf
TarSh.tdf
TarWall.tdf

—VERASY

VerFlag.tdf
VerHarp.tdf
VERMAN.TDF
VERSCOUT.TDF
VERTRANS.TDF
VERTRE.TDF

—VerCastl

VERBAL.TDF
VERBALL.TDF
VERBERS.TDF
VERCEN.TDF
VERCRUS.TDF
VERKNIGH.TDF
VerLiege.tdf
VerLihr.tdf
VERMUSK.TDF

—VerFlag

VERASY.TDF
VERFLTWR.TDF
VerKeep.tdf
VerLight.tdf
Verlode.tdf

—VerKeep

VERARCH.TDF
VerLiege.tdf
VerMer.tdf
Verpar.tdf
VerPult.tdf
VERSWORD.TDF

—VerLiege

VERASY.TDF

VERAT.TDF
VerCastl.TDF
VerKeep.TDF
VerLight.tdf
VERLODE.TDF
VERMORT.TDF
VerTower.TDF
VerWall.tdf

—VerLihr

VerDrag.tdf
VERGOD.TDF
VerMana.tdf
VERPILL.TDF

—VerMage

VERASY.TDF
VERAT.TDF
VerKeep.tdf
VERLODE.TDF
VERNGATE.TDF
VerWall.tdf

—ZonHand

ZONBAT.TDF
ZonFire.tdf
ZONGOB.TDF
ZONLODE.TDF
ZONTER.TDF
ZonTrain.tdf
ZONTROLL.TDF

—ZonHunt

ZonFire.tdf
ZonGlyph.tdf
ZONHAND.TDF
ZonLode.tdf

—ZonHurt

ZonFire.tdf
ZonGlyph.tdf
ZONHAND.TDF
ZonLode.tdf

—ZonLord

ZONDRAKE.TDF
ZonFlies.tdf
ZONGIANT.TDF
ZonLode.tdf
ZONORC.TDF
ZonRoc.tdf
ZonSham.tdf
ZONTRAIN.TDF

—ZonSham

ZonDrag.tdf
ZONGOD.TDF
ZonMana.tdf

—ZonTrain

ZonAmoe.tdf
ZonBar.tdf
ZonBasil.tdf
ZonGlyph.tdf
ZONGRYP.TDF
ZONHAND.TDF
ZonHarp.tdf
ZonKrak.tdf
ZonLode.tdf
ZonLord.tdf
ZonSpide.tdf
ZonSwamp.TDF
ZonWolf.tdf

Features

—All Worlds

araarch_frozen.tdf
araarch_stone.tdf
arabow_frozen.tdf
arabow_stone.tdf
arabroad_frozen.tdf
arabroad_stone.tdf
arabuild_frozen.tdf
arabuild_stone.tdf
aracan_stone.tdf
aradrag_stone.tdf
arafast_stone.tdf
arafly_stone.tdf
aragren_frozen.tdf
aragren_stone.tdf
araknigh_stone.tdf
arapal_stone.tdf
araprie2_frozen.tdf
araprie2_stone.tdf
arapries_frozen.tdf
arapries_stone.tdf
arapult_frozen.tdf
arapult_stone.tdf
arasmith_frozen.tdf
arasmith_stone.tdf
araspy_frozen.tdf
araspy_stone.tdf
arasword_stone.tdf
aratre_stone.tdf
arawall.tdf
lifbird_stone.tdf
lifcow_frozen.tdf
lifcow_stone.tdf
lifdeer2_frozen.tdf
lifdeer2_stone.tdf
lifdeer_frozen.tdf
lifdeer_stone.tdf
lifsaber_frozen.tdf
lifsaber_stone.tdf
lifwolf_frozen.tdf
lifwolf_stone.tdf
monboar_frozen.tdf
monboar_stone.tdf
mondev_frozen.tdf
mondev_stone.tdf
monghoul_frozen.tdf
monghoul_stone.tdf
monpiran_stone.tdf
npcalch_frozen.tdf
npcalch_stone.tdf
npcayla_frozen.tdf
npcayla_stone.tdf
npcbeg2_frozen.tdf
npcbeg2_stone.tdf
npcbeg_frozen.tdf
npcbeg_stone.tdf
npcburi_stone.tdf
npcdern_stone.tdf
npcduma_stone.tdf
npcemen_stone.tdf
npcfarm2_frozen.tdf
npcfarm2_stone.tdf
npcfarm_frozen.tdf
npcfarm_stone.tdf
npcheket_frozen.tdf
npcheket_stone.tdf
npchunt_frozen.tdf
npchunt_stone.tdf
npcjor2_stone.tdf
npcjor_stone.tdf
npcleim_frozen.tdf

npcleim_stone.tdf
npcpeas2_frozen.tdf
npcpeas2_stone.tdf
npcpeas_frozen.tdf
npcpeas_stone.tdf
npcpow1_frozen.tdf
npcpow1_stone.tdf
npcpow2_frozen.tdf
npcpow2_stone.tdf
npcref_frozen.tdf
npcref_stone.tdf
npcsail_frozen.tdf
npcsail_stone.tdf
npcshop_frozen.tdf
npcshop_stone.tdf
npctribe_frozen.tdf
npctribe_stone.tdf
npcwagon_stone.tdf
tararch_frozen.tdf
tarbeak_stone.tdf
tarblack_stone.tdf
tarcen_frozen.tdf
tarcen_stone.tdf
tardemon_frozen.tdf
tardemon_stone.tdf
tardrag_stone.tdf
tarfire_frozen.tdf
tarfire_stone.tdf
targarg_stone.tdf
tarhand_stone.tdf
tarkam_frozen.tdf
tarkam_stone.tdf
tarknigh_stone.tdf
tarlich_frozen.tdf
tarmage_frozen.tdf
tarmage_stone.tdf
tarminde_frozen.tdf
tarminde_stone.tdf
tarprrie2_stone.tdf
tarprries_stone.tdf
tarspout_frozen.tdf
tarspout_stone.tdf
tartb_frozen.tdf
tartb_stone.tdf
tartroop_stone.tdf
tarwall.tdf
tarwitch_frozen.tdf
tarwitch_stone.tdf
tarzom_frozen.tdf
verarch_frozen.tdf
verarch_stone.tdf
verbal_frozen.tdf
verbal_stone.tdf
verbers_frozen.tdf
verbers_stone.tdf
vercen_frozen.tdf
vercen_stone.tdf
vercrus_stone.tdf
verdrag_stone.tdf
verknigh_stone.tdf
verliege_frozen.tdf
verliege_stone.tdf
verlihr_frozen.tdf
verlihr_stone.tdf
vermer_frozen.tdf
vermer_stone.tdf
vermort_stone.tdf
vermusk_frozen.tdf
vermusk_stone.tdf
verpar_stone.tdf
verpult_frozen.tdf
verpult_stone.tdf
versword_stone.tdf

verwall.tdf
zonamoe_frozen.tdf
zonamoe_stone.tdf
zonbar_stone.tdf
zonbasil_frozen.tdf
zonbat_stone.tdf
zondrag_stone.tdf
zondrake_stone.tdf
zonflies_stone.tdf
zongob_frozen.tdf
zongob_stone.tdf
zongryp_stone.tdf
zonhand_frozen.tdf
zonhand_stone.tdf
zonharp_stone.tdf
zonkrak_stone.tdf
zonlord_frozen.tdf
zonlord_stone.tdf
zonorc_frozen.tdf
zonorc_stone.tdf
zonroc_stone.tdf
zonsham_frozen.tdf
zonsham_stone.tdf
zonswamp_frozen.tdf
zonswamp_stone.tdf
zonter_frozen.tdf
zonter_stone.tdf
zontrain_frozen.tdf
zontrain_stone.tdf
zontroll_stone.tdf
zonwolf_frozen.tdf
zonwolf_stone.tdf

Corpses

arafly_dead.tdf
aragren_dead.tdf
arapult_dead.tdf
arasiege_dead.tdf
aratrans_dead.tdf
monpiran_dead.tdf
tarcan_dead.tdf
tarhand_dead.tdf
verbal_dead.tdf
vercen_dead.tdf
verlight_dead.tdf
vermer_dead.tdf
zonamoe_dead.tdf
zonbar_dead.tdf
zonswamp_dead.tdf
zonwolf_dead.tdf

fonts

bodfontbody.gaf
bodfontbody.pcx
bodfontbody.tdf
bodfontdecor.gaf
bodfontdecor.pcx
boneyard_heading.gaf
boneyard_heading.pcx
boneyard_heading.tdf
bywarconsole.gaf
b_times new roman (100).gaf
b_times new roman (100).pcx
b_times new roman (100).tdf
b_times new roman (100b).gaf
b_times new roman (100b).pcx
b_times new roman (100b).tdf
decorativesm.gaf
decorativesm.pcx
decorativesm.tdf
font48.gaf
font48.pcx
font48.tdf

```
ig_times new roman (100).gaf
ig_times new roman (100).pcx
ig_times new roman (100).tdf
lombardic (cd).gaf
lombardic (cd).pcx
lombardic (cd).tdf
roman10.fnt
times new roman (100).gaf
times new roman (100).tdf
times new roman (100b).gaf
times new roman (100b).tdf
```

GameData

```
MOVEINFO.TDF
effects
  effects.tdf
explosions
  explosions.tdf
soundclasses
  AraFly.tdf
  AraGren.tdf
  AraSiege.tdf
  aratrans.tdf
  soundclasses.tdf
  tarang.tdf
  TarCan.tdf
  TarHand.tdf
  TarKam.tdf
  verbal.tdf
  vercen.tdf
  Verlight.tdf
  VerMer.tdf
  zonamoe.tdf
  zonbar.tdf
  ZonSwamp.tdf
  ZonWolf.tdf
```

guis

```
battlemenumulti.gui
battlemenusingle.gui
creingame.gui
creingameold.gui
creingametest.gui
creonigold.gui
hostgame.gui
mainmenul.gui
playercampaigndialogue.gui
selectgame.gui
victorycre.gui
visualoptions1.gui
```

Objects3D

```
AraBolt.3do
AraFly.3do
arafly_dead.3do
aragren.3do
AraGren1.3do
AraGren1_vet.3do
aragren_dead.3do
AraSiege.3do
arasiege_dead.3do
aratrans.3do
aratrans_dead.3do
tarang.3do
tarcan.3do
tarcan_dead.3do
tarhand.3do
tarhand_dead.3do
tarkam.3do
verbal.3do
```

verball.3do
verball_vet.3do
verbal_dead.3do
vercen.3do
vercen_dead.3do
verlight.3do
VerLight1.3do
VerLight2.3do
VerLight3.3do
verlight_dead.3do
vermer.3do
vermer_dead.3do
zonamoe.3do
zonamoe_dead.3do
zonbar.3do
zonbar_dead.3do
zonswamp.3do
zonswamp_dead.3do
zonwolf.3do
ZonWolf2.3do
zonwolf_dead.3do

Scripts

ARAFLY.COB
ARAGREN.COB
ARASIEGE.COB
aratrans.cob
tarang.cob
TARCAN.COB
TARHAND.COB
TARKAM.COB
verbal.cob
vercen.cob
VERLIGHT.COB
VERMER.COB
zonamoe.cob
zonbar.cob
ZONSWAMP.COB
ZONWOLF.COB
ZONWOLF2.COB

Sounds

ARAFLYSEL1.wav
ARAGRENDIE1.wav
ARAGRENMOV1.wav
ARAGRENSEL1.wav
ARASIEGEDIE1.wav
ARASIEGEMOV1.WAV
ARASIEGESEL1.WAV
aratransmov1.wav
aratranssell.wav
lightbeam.wav
tarangdiel.wav
tarangmov1.wav
tarangsell.wav
TARCANDIE1.wav
TARCANFIRE.wav
TARCANMOV1.wav
TARCANSEL1.wav
TARHANDDIE1.WAV
TARHANDMOV1.WAV
TARHANDSEL1.WAV
TARKAMDIE1.wav
TARKAMMOV1.wav
TARKAMSEL1.wav
verbaldiel.wav
verbalmov1.wav
verbalsell.wav
vercendiel.wav
vercenmov1.wav
vercensell.wav
VERLIGHTDIE1.wav
VERLIGHTSEL1.wav

VERMERDIE1.wav
VERMERMOV1.wav
VERMERSEL1.wav
zonoemoedi1.wav
zonoemoev1.wav
zonoemoes1.wav
zonoemose1.wav
ZONSWAMPDIE1.wav
ZONSWAMPMOV1.wav
ZONSWAMPSEL1.wav
ZONWOLFDIE1.wav
ZONWOLFMOV1.wav
ZONWOLFSEL1.wav

—Sounds-French

ARAGRENMOV1.WAV
ARAGRENSEL1.WAV
TARHANDSEL1.WAV
VERMERDIE1.WAV
VERMERMOV1.WAV
VERMERSEL1.WAV

—Sounds-German

ARAGRENMOV1.WAV
ARAGRENSEL1.WAV
TARHANDSEL1.WAV
VERMERDIE1.WAV
VERMERMOV1.WAV
VERMERSEL1.WAV

—Sounds-Italian

ARAGRENMOV1.wav
ARAGRENSEL1.wav
TARHANDSEL1.wav
VERMERDIE1.wav
VERMERMOV1.wav
VERMERSEL1.wav

—Sounds-Spanish

ARAGRENMOV1.wav
TARHANDMOV1.WAV
TARHANDSEL1.WAV
VERMERDIE1.WAV
VERMERMOV1.WAV
VERMERSEL1.WAV

—Translate

arafly.tdf
aragren.tdf
arasiege.tdf
aratrans.tdf
boneyards.tdf
Crusades.tdf
customkeys.tdf
death.tdf
features.tdf
guiexpansion.tdf
gui_text.tdf
leaderboard.tdf
load_screen.tdf
maps.tdf
messages.tdf
missions.tdf
networking.tdf
npcs.tdf
shortcutkeys.tdf
startup.tdf
tarcan.tdf
tarhand.tdf
unit chatter.tdf
unitmissions.tdf
unitnames.tdf
verbal.tdf

vercent.tdf
verlight.tdf
vermer.tdf
zonamoe.tdf
zonbar.tdf
zonswamp.tdf
zonwolf.tdf

units

araarch.fbi
araat.fbi
arabow.fbi
arabroad.fbi
arabuild.fbi
aracan.fbi
aracastl.fbi
aradrag.fbi
arafast.fbi
arafly.fbi
aragod.fbi
aragren.fbi
arakeep.fbi
araking.fbi
aralnigh.fbi
aralode.fbi
aramana.fbi
arangate.fbi
aranull.fbi
arapal.fbi
araprie2.fbi
arapries.fbi
arapult.fbi
arasiege.fbi
arasmith.fbi
araspy.fbi
arassh.fbi
arasword.fbi
aratrans.fbi
aratre.fbi
arawall.fbi
arawar.fbi
lifbird.fbi
lifcow.fbi
lifdeer.fbi
lifdeer2.fbi
lifsaber.fbi
lifwolf.fbi
monboar.fbi
mondev.fbi
monghoul.fbi
monpiran.fbi
npcalch.fbi
npcayla.fbi
npcbeg.fbi
npcbeg2.fbi
npcbotl.fbi
npcburi.fbi
npcdern.fbi
npcduma.fbi
npcemen.fbi
npcfarm.fbi
npcfarm2.fbi
npcflag.fbi
npcchet.fbi
npchunt.fbi
npcjor.fbi
npcjor2.fbi
npcleim.fbi
npcpeas.fbi
npcpeas2.fbi
npcpow1.fbi
npcpow2.fbi
npcref.fbi

npcrixx.fbi
npcsail.fbi
npcshop.fbi
npctemp.fbi
npctemp2.fbi
npcthesh.fbi
npctribe.fbi
npcwagon.fbi
tarang.fbi
tararch.fbi
tarbeak.fbi
tarblack.fbi
tarcage.fbi
tarcen.fbi
tarcastl.fbi
tardemon.fbi
tardrag.fbi
tardung.fbi
tarfire.fbi
targarg.fbi
targod.fbi
tarhand.fbi
tarhell.fbi
tarkam.fbi
tarknigh.fbi
tarlich.fbi
tarlode.fbi
tarmage.fbi
tarmana.fbi
tarvind.fbi
tarnecro.fbi
tarngate.fbi
tarprrie2.fbi
tarprries.fbi
tarsh.fbi
tarship.fbi
tarspout.fbi
tartb.fbi
tartroop.fbi
tarwall.fbi
tarwitch.fbi
tarzom.fbi
verarch.fbi
verasy.fbi
verat.fbi
verbal.fbi
verball.fbi
verbers.fbi
vercastl.fbi
vercen.fbi
vercrus.fbi
verdrag.fbi
verflag.fbi
verfltwr.fbi
vergod.fbi
verharp.fbi
verkeep.fbi
verknigh.fbi
verliege.fbi
verlight.fbi
verlihr.fbi
verlode.fbi
vermage.fbi
verman.fbi
vermana.fbi
vermer.fbi
vermort.fbi
vermusk.fbi
verngate.fbi
verpar.fbi
verpill.fbi
verpult.fbi
verscout.fbi

versword.fbi
vertower.fbi
vertrans.fbi
vertre.fbi
verwall.fbi
zonamoe.fbi
zonbar.fbi
zonbasil.fbi
zonbat.fbi
zondrag.fbi
zondrake.fbi
zonfire.fbi
zonflies.fbi
zongiant.fbi
zonglyph.fbi
zongob.fbi
zongod.fbi
zongryp.fbi
zonhand.fbi
zonharp.fbi
zonhunt.fbi
zonhurt.fbi
zonkrak.fbi
zonlode.fbi
zonlord.fbi
zonmana.fbi
zonorc.fbi
zonroc.fbi
zonsham.fbi
zonswamp.fbi
zonter.fbi
zontrain.fbi
zontroll.fbi
zonwolf.fbi
zonwolf2.fbi

unitscb

araarch.fbi
araat.fbi
arabow.fbi
arabroad.fbi
arabuild.fbi
aracan.fbi
aracastl.fbi
aradrag.fbi
arafast.fbi
arafly.fbi
aragod.fbi
aragren.fbi
arakeep.fbi
araking.fbi
araknigh.fbi
aralode.fbi
aramana.fbi
arangate.fbi
aranull.fbi
arapal.fbi
araprie2.fbi
arapries.fbi
arapult.fbi
arasiege.fbi
arasmith.fbi
araspy.fbi
arassh.fbi
arasword.fbi
aratrans.fbi
aratre.fbi
arawall.fbi
arawar.fbi
lifbird.fbi
lifcow.fbi
lifdeer.fbi
lifdeer2.fbi

lifsaber.fbi
lifwolf.fbi
monboar.fbi
mondev.fbi
monghoul.fbi
monpiran.fbi
npcalch.fbi
npcayla.fbi
npcbeg.fbi
npcbeg2.fbi
npcbot1.fbi
npcburi.fbi
npcdern.fbi
npcduma.fbi
npcemen.fbi
npcfarm.fbi
npcfarm2.fbi
npcflag.fbi
npcheket.fbi
npchunt.fbi
npcjor.fbi
npcjor2.fbi
npcleim.fbi
npcpeas.fbi
npcpeas2.fbi
npcpow1.fbi
npcpow2.fbi
npcref.fbi
npcrix.fbi
npcsail.fbi
npcshop.fbi
npctemp.fbi
npctemp2.fbi
npcthesh.fbi
npctribe.fbi
npcwagon.fbi
tarang.fbi
tararch.fbi
tarbeak.fbi
tarblack.fbi
tarcage.fbi
tarcana.fbi
tarcast1.fbi
tardemon.fbi
tardrag.fbi
tardung.fbi
tarfire.fbi
targarg.fbi
targod.fbi
tarhand.fbi
tarhell.fbi
tarkam.fbi
tarknigh.fbi
tarlich.fbi
tarlode.fbi
tarmage.fbi
tarmana.fbi
tarmind.fbi
tarnecro.fbi
tarngate.fbi
tarprrie2.fbi
tarprries.fbi
tarsh.fbi
tarship.fbi
tarspout.fbi
tartb.fbi
tartroop.fbi
tarwall.fbi
tarwitch.fbi
tarzom.fbi
verarch.fbi
verasy.fbi
verat.fbi

verbal.fbi
verball.fbi
verbers.fbi
vercastl.fbi
vercen.fbi
vercrus.fbi
verdrag.fbi
verflag.fbi
verfltwr.fbi
vergod.fbi
verharp.fbi
verkeep.fbi
verknigh.fbi
verliege.fbi
verlight.fbi
verlihr.fbi
verlode.fbi
vermage.fbi
verman.fbi
vermana.fbi
vermer.fbi
vermort.fbi
vermusk.fbi
verngate.fbi
verpar.fbi
verpill.fbi
verpult.fbi
verscout.fbi
versword.fbi
vertower.fbi
vertrans.fbi
vertre.fbi
verwall.fbi
zonamoe.fbi
zonbar.fbi
zonbasil.fbi
zonbat.fbi
zondrag.fbi
zondrake.fbi
zonfire.fbi
zonflies.fbi
zongiant.fbi
zonglyph.fbi
zongob.fbi
zongod.fbi
zongryp.fbi
zonhand.fbi
zonharp.fbi
zonhunt.fbi
zonhurt.fbi
zonkrak.fbi
zonlode.fbi
zonlord.fbi
zonmana.fbi
zonorc.fbi
zonroc.fbi
zonsham.fbi
zonswamp.fbi
zonter.fbi
zontrain.fbi
zontroll.fbi
zonwolf.fbi
zonwolf2.fbi

FBI Functions

by Hansolo

The purpose of this document is to explain the different functions of the unit files for TAK. These files are the ones in the HAPI compressed file, in the sub-directory /UNITS/. Their file extension is FBI, the same as it was in TA.

The first notable differences with TAK unit files; is that they now contain weapon data as well, which were previously in separate TDF files. This is probably due to the fact that there are no longer a finite amount of Weapon ID's, so any weapons can be duplicated without ID-using difficulties.

All right then; let's get started. Commands are now listed in alphabetical order, not that it makes any difference, but it allows specific commands to be found easier. I will be referring to the unit instructions ONLY - not the weapons ones, which appear in a different section of the file. All values marked with a '(B)', are Boolean values, where 0 = negative, 1 = positive.

FBI TABLE

acceleration = 10;	This is the rate at which a unit will accelerate
activatewhenbuilt	Whether the unit is active (turned on), when first built. (B?)
admultiplier = ?;	Unknown
animatetype = ?;	Unknown
attackrunlength = 150;	How far the flying unit will move, before turning around for another pass.
attractsgods = 1;	This unit has to be built for your side's god to appear. (Pre-3.0???)
bankscale = 0.4;	Not sure, to do with the 'banking' of flying units.
bloodcolor1 = 160 35 0; bloodcolor2 = 170 40 0; bloodcolor3 = 180 30 5;	I'm not sure about the significance of the 3 different colors, but they are the RGB values of the blood. (RED BLUE GREEN, 0-255).
bmcode = 1;	Unknown
bodytype = flesh;	I presume that this will generate the appropriate sounds and damage; yet to be confirmed.
brakerate = 10;	This is the rate at which a unit will decelerate.
Buildangle	The angle the unit is built at, not compulsory.

buildcost = 325;	This is the cost of the unit in Mana, replaces the 'buildcostenergy and buildcostmetal' of TA.
builder = 1;	Whether the unit can build units. (B)
builderlimited = 1;	Whether the unit can aid in other units building. As far as I am aware, it is only Monarchs that have a value of 0. (B)
buildtime = 125;	This is the time it takes for the unit to be built, directly linked with buildcost; ratio determines Mana depletion.
cananimate = 1;	Whether the unit can animate (turn a corpse into a ghoul). (B)
canattack = 1;	Whether the unit can attack. (B)
canbuild = 1;	Not sure if this is used by the game, would appear to have same affect as 'builder'. (B)
cancapture = 1;	Whether the unit can capture. (B)
cancloak = 1;	Whether the unit can cloak. (B)
canfly = 1;	Whether the unit can fly. (B)
canguard = 1;	Whether the unit can guard. (B)
canhover = 1;	Whether the unit hovers. (B)
canload = 1;	Whether the unit can load. (B)
canmove = 1;	Whether the unit can move. (B)
canpatrol = 1;	Whether the unit can patrol. (B)
canreclaim = 1;	Whether the unit can reclaim, or clear as it is know in game. (B)
canresurrect = 1;	Whether the unit can bring units back to life. (B)
canstop = 1;	Whether the unit can stop. (B)
cantbecaptured = 1;	Whether the unit cannot be captured. (B)
cantbestoned = 1;	Whether the unit can be turned to stone, similar to the cantbeparalyzed command in TA. (B)
cantbetransported = 1;	Whether the unit cannot be transported. (B)
cantransport = 1;	Whether the unit can transport. (B)

category = ARA BALLISTIC ATTACK;	This defines the categories the unit falls into.
	In this example, it is an Aramon unit (ARA), has a ballistic weapon, and can attack. I'm not sure how much of an effect this has on the game, but I think it may be used by the AI.
cloakcost = 14;	How much Mana the unit uses when cloaked.
cloakcostmoving = 21;	How much Mana the unit uses when cloaked and moving.
commander = 1;	Whether the unit is a commander/monarch (used when working out if you're Monarch is dead, not which unit you start off with; that's in the gamedata/sidedata.tdf).
copyright = Copyright 1999 Humongous Entertainment. All rights reserved.;	CD's little reminder who made the game. In TA v3 and above, this line was necessary. Yet to be confirmed whether the line is needed, but I presume it is.
corpse = araarch_dead;	The corpse that the unit uses. This refers to the matching record in the features/corpses/ and not the dead_3do as may be thought.
corpseadjustx = ?;	Unknown. Possibly something to do with the corpse of the unit being a different size to the building?
corpseadjustz = ?;	Unknown. Possibly something to do with the corpse of the unit being a different size to the building?
cruisealt = 200;	The distance from the ground the unit flies at.
damagecategory = Human;	I presume that this is related to the kind of damage it takes (i.e. Zombies take less damage from arrows than humans...)
defaultmissiontype = Standby;	This defines what the unit does when it is first built. Most (if not all) units are Standby, with flying units using VTOL_Standby.
description = Aramon;	Not sure whether this appears in the game, in TA it was under the name in the build menu. It appears that this simply states the side that the unit appears for.
economybonus = 30;	Unknown. Surely something to do with

	Mogrium, or Mana income, but not sure of it's relevance.
experiencepoints = 7;	The number of experience points the unit gives its killer when killed. The bigger and better the unit, the higher the points. To be confirmed.
fireatwillrandom = 1;	I'm not sure whether this is a Boolean value or not. It only seems to appear in units with ballistic weapons; and could be an indicator of whether they will fire at enemy before being attacked, if in defensive mode. (B?)
floater = 1;	Whether the unit floats on water. (B)
footprintx = 2;	The width of the unit, used to stop overlapping 3do's. Buildings only, mobile unit's footprints are defined in the movementclass.
footprintz = 2;	The depth of the unit, used to stop overlapping 3do's. Buildings only, mobile unit's footprints are defined in the movementclass.
gate = 1;	Whether the unit is a gate. (B)
ghost = 1;	Whether the unit is a ghost; and can move through features/corpses? (B)
healtime = 0.520833333;	How quickly the unit will heal. I think that this value indicates a damage unit recovered every half a second.
hoverattack = 1;	Whether the unit hovers and attacks, or makes sweeping runs. (B)
hoverattackaltitude = 150;	I presume this is only used if the altitude is different to 'cruisealt'. The altitude the hover- attacking unit is from the ground.
hoverattackdistance = 150;	How far the unit hovers from it's target.
isfeature = 1;	Used by the wall. Defines whether the unit becomes a feature once built. (B)
manarecharge rate = 10;	How quickly the unit's Mana personal storage regenerates.
maneuverleashlength = 500;	This is how far the unit will move from it's movement path. This is useful when patrolling or moving in formation. This is how far the unit will go off on a whim, before returning to it's set course.

<code>maxdamage = 1100;</code>	The health of a unit. Cannot be exceeded, despite healttime.
<code>maxslope = 15;</code>	The maximum slope the unit can be built on. Buildings only, mobile unit's maxslopes are defined in the movementclass.
<code>maxmana = 1000;</code>	How much personal Mana the unit has. Cannot be exceeded, regardless of manarecharge rate.
<code>maxvelocity = 1.25;</code>	The maximum speed of a unit. I think this is measured in the number of in-game units per second.
<code>maxwaterdepth = 0;</code>	The maximum depth the unit can be built on. Buildings only, mobile unit's maxwaterdepths are defined in the movementclass.
<code>mincloakdistance = 80;</code>	The distance at which cloaked units will be spotted.
<code>mogriumincome = 0;</code>	The amount of Mana the unit generates.
<code>mogriumstorage = 0;</code>	The amount of Mana the unit can store.
<code>movementclass = GROUND2;</code>	This refers to the movement classes in the 'moveinfo.tdf' in the gamedata directory. This defines the slopes that the unit can traverse, and the water depths. This seems to replace the 'maxslope' and 'maxwaterdepth' from TA, and goes with the general values defined in the TDF.
<code>moverate1 = 8;</code>	This value is called from the script, and I'm pretty sure it defines the speed it hover attacking units move from side to side.
<code>moverate2 = 8;</code>	See moverate1.
<code>name = Archer;</code>	The name of the unit, as it appears in the game.
<code>nochasecategory = FLY;</code>	The category that the unit will not chase.
<code>noshadow = 1;</code>	Whether the unit has no shadow. (B)
<code>notargetcategory = FLY;</code>	The category that the unit will not target.
<code>noveteran = 1;</code>	Whether the unit cannot become a Veteran (i.e. Gods). (B)
<code>objectname = ARAARCH;</code>	This is the name of the unit 3do.

<code>onoffable = 1;</code>	Whether the unit can be turned on/off. (B)
<code>pitchscale = 1.5;</code>	Unknown.
<code>radardistance = 550;</code>	How far the units radar extends.
<code>roadmultiplier = 1.21;</code>	This is the value that the unit's maxvelocity is multiplied by, when it is travelling over roads and smooth surfaces. (i.e. The Aramon Archer goes at a speed of [1.21x1.25=1.5125] when travelling on roads.)
<code>roadmultiplier = 1.21;</code>	Typo on Cavedog's part, unknown if it works the same as roadmultiplier. Only included for completion's sake.
<code>shadowart = shadow03;</code>	I think that this refers to the kind of shadow to use, but due to current Utility incompatibilities, this is unconfirmable.
<code>shadowgaf = shadows;</code>	See shadowart.
<code>shootme = 1;</code>	Whether enemy units will target the unit. (B)
<code>side = ARA;</code>	This is the side the unit fights for: ARA = Aramon TAR = Taros VER = Veruna ZON = Zhon LIF = Lifeforms NPC = Non-Player-Characters MON = Wandering Monsters
<code>sightdistance = 180;</code>	How far the unit can see.
<code>sonardistance = 100;</code>	The distance the unit's sonar extends.
<code>soundcategory = ARAARCH;</code>	The sound category that the unit uses. This unit therefore uses the sounds ARAARCHDIE1, ARAARCHMOV1 and ARAARCHSEL1.
<code>soundclass = ARAARCH;</code>	Unknown, appears to be the same as soundcategory.
<code>standingmoveorder = 1;</code>	What the unit's movement order is when built. Possibly used to say whether they are guarding, patrolling or moving, but I can't see how it would work.
<code>standingunitorder = 1;</code>	What the unit's order is when it is built. I presume that (0=Attacking, 1=Defensive and 2=Passive).
<code>stone = araarch_stone;</code>	What textures the unit uses when turned to stone.

<code>tedclass = Aramon;</code>	Used by the AI.
<code>totalallowed = 1;</code>	The number of the units allowed at one time. Can limit ultra-powerful units like the Gold Dragon, or units with many faces, such as the Aramon Chariot.
<code>transmaxunits = 1;</code>	I 'think' that this value is for flying transports only, as it appears only in the FBI of the Zhon Roc as far as I can tell. This is the same as 'transportcapacity' in this case...
<code>transportcapacity = 16;</code>	How many units the unit can transport.
<code>transportdistance = 300;</code>	How far away the unit has to be to load/unload.
<code>transportsize = 9;</code>	The maximum size of a unit that can be transported.
<code>transportsizecapacity = 64;</code>	The total capacity in unit size that the unit can carry. This might mean that it cannot reach it's full unit capacity.
<code>turninplacerate = 2400;</code>	How long it takes for the unit to turn on the spot.
<code>turnrate = 2400;</code>	How long it takes the unit to turn.
<code>unitname = ARAARCH;</code>	The name of the unit's FBI, Buildpic. Usually will match Objectname.
<code>unitnumber = 1;</code>	'Should' have no purpose other than for identification. In TA v1, it defined the unit, and could cause conflicts. TAK allegedly has unlimited unit ID's.
<code>unitstandorders = 0;</code>	A building's version of standingunitorders.
<code>upright = 1;</code>	Whether the unit is upright. Affects whether the unit will tilt on uneven terrain. (B)
<code>version = 1;</code>	The version of the unit. In TA, the FBI with the higher version, overwrote one with a lower version of the same name. Useful when updating units in updates.
<code>waterline = 1;</code>	Where the water comes up to on the unit.
<code>watermultiplier = 0.81;</code>	How the unit's speed is affected in it's waterdepth. Works in the same way as roadmultiplier.
<code>watermultipliser = 0.81;</code>	That's two typos for Cavedog, this is quite

	blatently a mis-typed watermultiplier. Only included for completion's sake.
weaponswitching = 1;	Whether the units will switch weapons independently (i.e. when it runs out of Mana, it will switch to a lower weapon.)
wind = 1;	I'm not sure about this one. I thought that it may be to do with the fact that the unit is partially propelled by the wind, seeing that it appears in the boat's FBIs. I am not aware of any wind in TAK though, although it's quite possible that it is there, but what purpose it plays is unknown. The curious thing is, that this line appears in the Lodestone's FBI. This leads to the thinking that it may be to do with the wind, moving mana around; although this is not apparent in game. I remember reading in early previews of TAK, that mana was rich in certain areas, and perhaps the wind originally moved it around. Still, as far as I can tell; this command has no use; but this cannot be tested as of yet. (B?)
workertime = 10;	How quickly the unit builds and repairs/reclaims.
wpri_badtargetcategory = FLY;	The category of units that the unit is bad at targeting; possibly only for VTOL units...
yardmap = S;	Which parts of the unit can be moved through, details unknown.

Some units have exceptions to these rules. One example is the ARASMITH or Aramon Titan. This unit gives surrounding allies an armour bonus. Other units have the ability to adjust Attack (preumably damage), and Joy (likelihood to fight to the death?) These are portrayed in the FBI in separate sections within the unit details:

[AdjustArmor/Attack/Joy]

{

adjustment = 1.4; The bonus the unit provides. I presume that the units maxdamage is multiplied by this value, but I am not sure whether the bonus is permanent, or only applies when in the radius if a Titan. Therefore, rather than modifying the unit's maxdamage, it might be the damage coming in that is divided. Also, I am not sure whether multiple Titans can 'add up' bonuses, allowing for some super-tough units. I am also not sure whether Titans can give the bonus to other Titans.

affectsenemy = 0; Whether the 'bonus' affects enemy units. (B)

```
edgeeffectiveness = 0.6;
```

```
    The effectiveness of the bonus for units not  
    completely in the radius.
```

```
radius = 200;
```

```
    The radius of the 'bonus'.
```

```
}
```

HPI Format Documentation- TA:K

ZLib compression and decompression by Jean-loup Gailly (compression) and Mark Adler (decompression). For more info, see the zlib Home Page at <http://www.cdrom.com/pub/infozip/zlib/> .

Warning: This is intended for use by people that already know what they're doing.

I'm a C programmer, so I'm doing things in C notation here, but I'll try to explain it so that those of you that don't speak C will be able to understand. If you don't understand, write me at joed@cws.org and I'll try to clear things up.

The first part of the file is a version header, followed by a file header.

The version header looks like this:

```
typedef struct _HPIVersion {
    long HPIMarker;          /* 'HAPI' */
    long Version;           /* 'BANK' if saved gamed */
} HPIVersion;
```

HPIMarker	This is just a marker. The value is always HAPI in ASCII. In hex, it's 0x49504148
Version	If it's a TA saved game, the value is BANK in ASCII, or 0x4B4E4142 in hex. If the value is 0x00010000, then it's a Total Annihilation archive. See the HPI-FMT document. If it's 0x00020000, then it's a TA:Kingdoms archive.

Immediately following the version header is the file header. It looks like this:

```
typedef struct _HPIHEADER2 {
    long DirBlock;
    long DirSize;
    long NameBlock;
    long NameSize;
    long Data;
    long Last78;
```

```
} HPHEADER2;
```

DirBlock	This is a pointer to the file directory. This block may or may not be compressed. If it starts with 'SQSH' (hex 0x48535153), it's compressed. If it doesn't, it's not. Decompress like any other SQSH block, described below.
DirSize	This is the size of the directory block pointed to above.
NameBlock	This is a pointer to the actual file names. This block may or may not be compressed. If it starts with 'SQSH' (hex 0x48535153), it's compressed. If it doesn't, it's not. Decompress like any other SQSH block, described below.
NameSize	This is the size of the file name block pointed to above.
Data	This is the start of the actual file data, as near as I can figure. It always seems to be set to 0x20.
Last78	This is either 0, or it points to the last 78 bytes of the file. There is some data at that point, but I haven't figured out more than the copyright information.

HOW THE DIRECTORY WORKS

Read in the DirBlock and NameBlock. Decompress if necessary.

The NameBlock is merely a list of null-terminated file names of varying lengths. It also starts with a null.

The directory block consists of either file or directory entries.

The first entry in the DirBlock will be the directory entry for the root directory of the file.

A directory entry looks like this:

```
typedef struct _HPIDIR2 {  
    long NamePtr;  
    long FirstSubDir;  
    long SubCount;
```

```

    long FirstFile;
    long FileCount;
} HPIDIR2;

```

NamePtr	This points to the name of the directory in the NameBlock.
FirstSubdir	This points to the directory entry in the DirBlock of the first subdirectory of this directory. Subsequent subdirectory entries follow.
SubCount	How many subdirectories there are in this directory.
FirstFile	This points to the file entry in the DirBlock of the first file of this directory. Subsequent file entries follow.
FileCount	How many file entries there are in this directory

A file entry looks like this:

```

typedef struct _HPIENTRY2 {
    long NamePtr;
    long Start;
    long DecompressedSize;
    long CompressedSize; /* 0 = no compression */
    long Date; /* date in time_t format */
    long Checksum;
} HPIENTRY2;

```

NamePtr	Points to the file name in the NameBlock.
Start	Points to the start of the file in the hpi archive.
DecompressedSize	The final decompressed size of the file.
CompressedSize	The total compressed size of the file. If this is 0, the file is not compressed.
Date	The file date in time_t format. This is the number of seconds since Jan 1, 1970, GMT.

Checksum	A checksum. This isn't really a single checksum - it's actually 4 checksums in one, but it's easier to manipulate it if you treat it as a single long. More on checksum calculation below
----------	---

CHECKSUM CALCULATION

The checksum in each HPIENTRY2 is calculated from the uncompressed file data.

It's actually four checksums in one, each 8-bit byte being one of the checksums.

The first one is the sum of all the unsigned bytes in the file.

The second one is the cumulative XOR of all the unsigned bytes in the file.

The third one is the sum of all the unsigned bytes in the file, each byte XOR'd with its offset before being added.

The fourth one is the cumulative XOR of all the unsigned bytes in the file, each byte XOR'd with its offset before being XOR'd.

Here's some C code to calculate it:

```
int CheckCalc(long *cs, char *buff, long size)
{
    int count;
    unsigned int c;
    unsigned char *check = (unsigned char *) cs;

    for (count = 0; count < size; count++) {
        c = (unsigned char) buff[count];
        check[0] += c;
        check[1] ^= c;
        check[2] += (c ^ ((unsigned char) (count & 0x000000FF)));
        check[3] ^= (c + ((unsigned char) (count & 0x000000FF)));
    }
    return *cs;
}
```

DECOMPRESSION OF BLOCKS AND FILES

If the block was not compressed at all, then it is just inserted into the HPI file as one big chunk.

Each chunk looks like this:

```
typedef struct _HPIChunk {
```

```

long Marker;          /* always 0x48535153 (SQSH) */
char Unknown1;
char CompMethod;     /* 1=LZ77, 2=ZLib */
char Encrypt;        /* is the block encrypted? */
long CompressedSize; /* the length of the compressed data */
long DecompressedSize; /* the length of the decompressed data */
long Checksum;       /* Checksum */
char data[];         /* 'CompressedSize' bytes of data */
} HPICchunk;

```

Marker	This is the start-of-chunk marker, and is always 0x48535153 (ASCII 'SQSH').
Unknown1	I know not what this is for. It's always 0x02. Maybe some sort of version number
CompMethod	This is the compression method. It's 1 for LZ77, 2 for ZLib. I don't know if TAK does LZ77 - all the files distributed with the game use ZLib.
Encrypt	This tells whether the block is encrypted. See below
CompressedSize	This is the size of the compressed data in the chunk
DecompressedSize	This is the size of the decompressed data in the chunk.
Checksum	This is a checksum of the data. It's merely the sum of all the bytes of data (treated as unsigned numbers) added together
data	The actual compressed data in the chunk

The 'Encrypt' field in the HPICchunk header is set to 1 to indicate that this decryption needs to be done.

To decrypt, do this:

```

for x = 0 to CompressedSize-1
  data[x] = (data[x] - x) XOR x
next x

```

If CompMethod is 2, use ZLib compression to decompress the block. You can get the zlib source code from the zlib home page at <http://www.cdrom.com/pub/infozip/zlib/>

TNT Format and Conversion

By: C_A_P

Index:

1. Using the Exporter

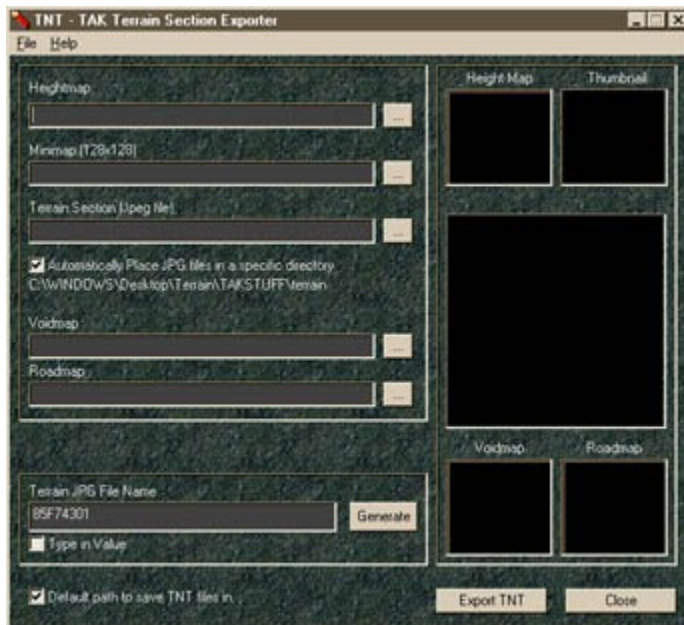
- • Heightmap
- • Minimap
- • Voidmap
- • Roadmap
- • Jpeg Key
- • The "Auto" functions

2. Using the Terrain Reader

- • Exporting the Heightmap
- • Jpeg Key Value (Hex Keys and Hex Values)

Using the Exporter:

The TAK terrain section exporter is able to create a TAK compatible terrain Section TNT file, along with organizing and auto-naming the JPG terrain pictures.



Heightmap:

The heightmap is a special bitmap that is used by the TAK engine to model the terrain mesh. You can view the terrain mesh in -game by pressing the Enter key, and typing:

"+Contour +6" (you can replace 6 with any number from 1 - 9)
You would then type "+contour +0" to turn the mesh lines off...

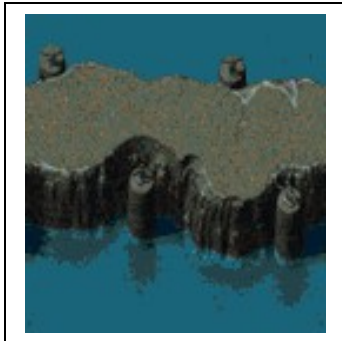
The heightmap is a 256 color grayscale bitmap, and needs to be sized to 1/16 the size of your terrain image. Below is what a grayscale map may look like for a 512x512 section (the heightmap is 32x32 pixels):



The Minimap:

The minimap is the thumbnail picture that Cartographer displays when you are browsing through the terrain sections. You must size this image to 128x128 pixels, and it should be converted over to the correct color palette. (each world in TAK has its own terrain color palette, which can be found in the palettes folder in the data1.hpi file)

Below is the minimap picture to go along with the heightmap above:



The Voidmap:

TAK allows you to create a map, much like the heightmap, that tells the game where to not allow units to travel. The voidmap works by using White pixels to indicate that the area is voided, and Black pixels to indicate the area is not voided. It needs to be the same size as the heightmap.



The Roadmap:

The Road map is exactly like the voidmap except that instead of indicating where units cannot go, it indicates where there are 'roads' or pathways. There are normally used on terrain sections where there are stone paths,

or roads. These areas cause the units to move faster than normal when they walk on a 'road' ...

The Jpeg Key

The Jpeg Key is how TAK knows which terrain graphic goes with the .tnt file. In TAK the terrain graphics were actually embedded in the .tnt files, but in TAK they are separate. This allows for much smaller map sizes...

There is a checkbox next to this that says "Type in Value" this allows you to specify a particular jpg, instead of randomly generating a number... However, you do NOT type in the actual Hex number, it needs to be the numeric equivalent of the hex code.

For example: The Hex value of: 7F413CFF
Has a numeric value of: 2134981887
See more on this in the "Hex Keys and Hex Values' section"

The Auto Functions:

The 'auto' functions are simply those two checkboxes that are in the exporter screen. One is labeled "Automatically copy JPG files to a specific folder", and the other is labeled "Default path to save TNT files in"

The 'JPG' option will automatically name and place your jpg files into a specific folder (you will need and want this functionality when you being to build your terrain sections and need to pack them up into a file that can be used by the game).

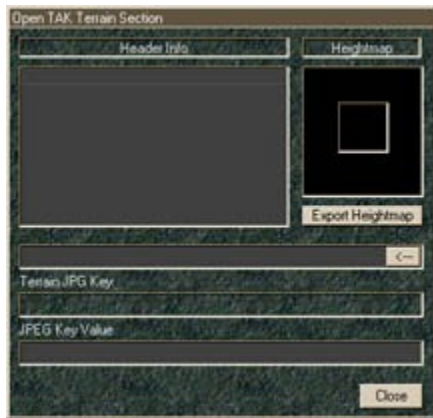
Your JPG files MUST correspond with the hex value embedded in the .tnt files you create. This is done automatically for you if this box is checked...

The "Default path for TNT' option is simply so the program knows where you want to save your .tnt files and will automatically select this as the default directory to display for you when you save your TNT files. Of course you can browse and put it somewhere else if you want.

The Terrain Reader

The terrain reader is simply a utility to extract the greyscale heightmaps from the terrain .tnt files. Its main purpose it to allow people to extract the heightmaps from the tnt section files so that they may blend them all together eventually!!! Mooohooahahahaha ;)

Simply press the export button to save off the terrain section greyscale bitmap...



Hex Keys and Hex Values

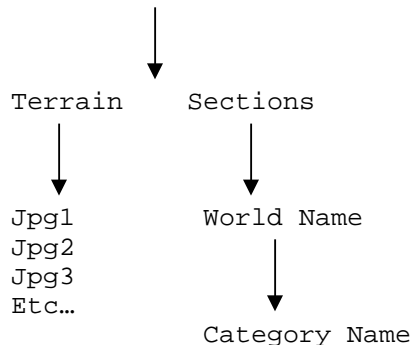
The hex keys are actually the names of the .jpg files. If you open up the file called "terrain.hpi" using HPI View, then you will see all the strangely named .jpg files. These names are encoded into the TNT files so TAK know which terrain pieces go with each other. When you open a .tnt section in TNT, it will show you the terrain .jpg key so you will know which terrain graphic goes with it!!

Also, you get the 'value' which you can use if you need to alter an existing TNT and keep the same JPG reference.....only advanced users should deal with this...

Package and Deployment!

To make your creations TAK compatible, you will need to get the TAK Compatible version of HPI Pack. Use the following example as a guide for how to build your directory structure:

Root Directory



Note: you can also open up my HPI file in HPI View and check out the directory structure within that file as well!

Have fun!!

Conversion Tutorial

By Jerry60000

Here in this tutorial I will try to explain how to use the TAK TNT. The other part of this tutorial is C_A_P's official one. I am just trying to help explain the few things that seem to confuse people the most.

Things you will need to work with making Kingdoms tile sets:

- Good paint utility
- The proper TAK palettes
- To make your own tiles you will need Bryce3D or equivalent.
- Lots of patience.
- TNT program from C_A_P, and TA: K, of course.
- HPI View and HPI pack

Other things that make life easier are a DrawPad. It is way easier to make small details and adjustments using a pen instead of a mouse.

Now here's a rundown of the files used.

Height Map : A grayscale bmp that defines the terrain's characteristics. It makes mountains act like mountains, water act like water.

Void Map : Also a grayscale bmp. Will be identical to the **Height map** except that the places you don't want units to move will be marked. For example would be a lava terrain, you would open the **Height map** with your Paint Shop. Then in the places that contain lava you would select with your brush the color white. Just plain old white (Hex=FFFFFF) anything less won't work. This makes the lava voided to units. i.e. impassable.

Road Map : The opposite of the **Voidmap**. Another grayscale bmp. Used for when you are making road tiles and bridge tiles. Select the color white and just paint over the road part of the image.

JPEG Key : This is a jpg that is the full size of the tile in the game. Can be any size you want. But remember that the Kingdoms Cartographer is a little unstable to handle big tiles.

Units Editor Tutorial - TDF Edit

INTRODUCTION

One of the qualities who made TA such a successful game, is with no doubt the possibility players have to make their own units. That allowed the

game to be constantly evolving. This possibility also exists in TA:Kingdoms, and this tutorial is here to explain you how to create your own units.

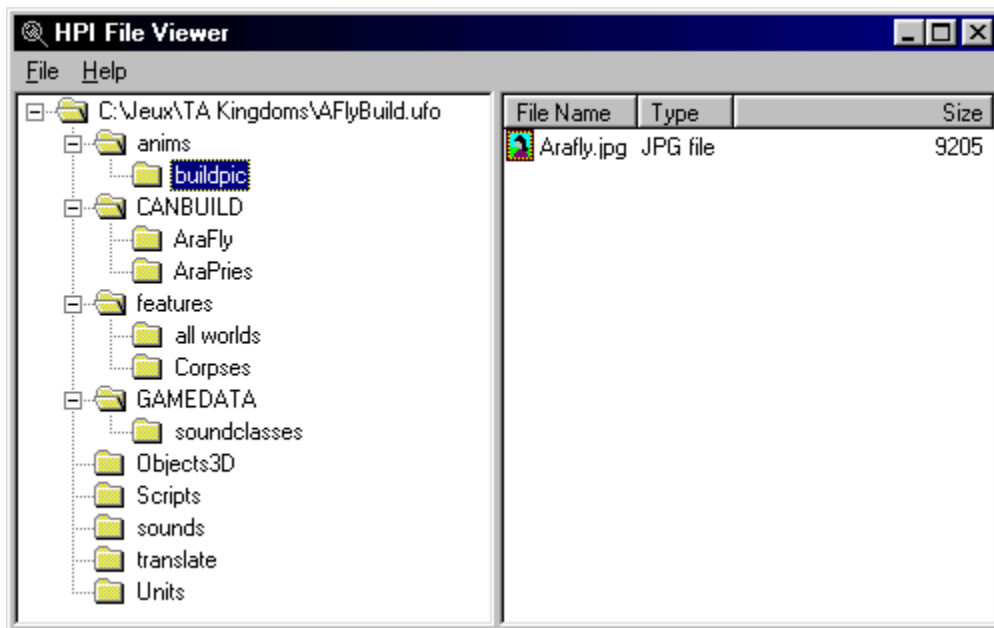
The softwares you need to make your own units are the followings :

- HPI View 1.9.1
- HPI Pack 1.7
- 3Do Builder + 2.0
- A 3D soft, able to save under .LWO format file
- TDF Edit

You can download them here : www.gfruitrene.com or there www.multimania.com/takingdoms.(sorry it's french sites).

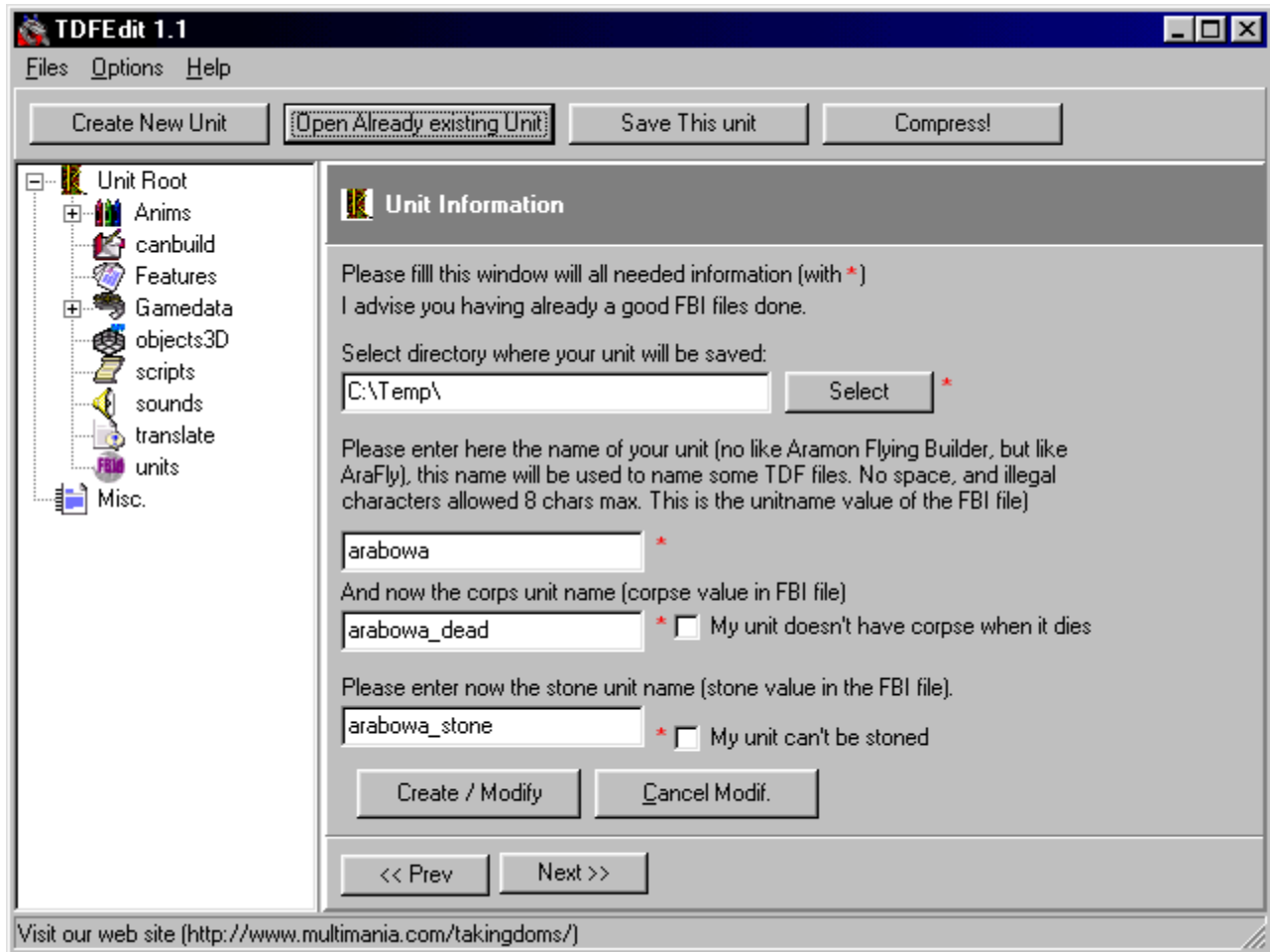
DESCRIPTION OF A TA:K UNIT

A TA:Kingdom unit is a .UFO file. This file is a compression (with HPI Pack) of all the files and folders a unit needs to work. One can see all these files / folders with HPI View. You can also use this soft to extract one or all these files.



File Organization of Aramon Flying Builder

Each file or folder has a very specific role. To help with the creation of these files / folders, we will use TDF Edit, a utility that helps unit creation.



TDF Edit Program

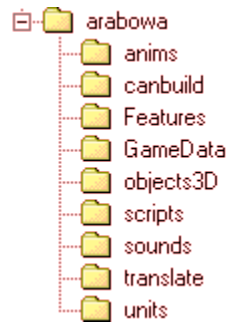
Important Note:

You probably noticed that all the files have the same name, with a different extension : ARAFLY for the Aramon Flying Builder. This name is the short name of the unit. The first three letters give the camp : ARA for Aramon, TAR for Taros, VER for Veruna, and ZON for, guess what ? Zhon ! The last five letters describe the unit. The short name must be unique among all the TA:K units, and all the files you are going to create must have this name and be in the same folder. You can check all the [short names](#) of all TA:K units [here](#).

HOW TO USE THIS TUTORIAL ?

This tutorial is divided up in parts that match each folder of a typical unit. Each part explains what is found in this folder, how to edit it with TDF Edit, gives additional informations on this folder, and how to edit it by hand. This last method is needed to create precise units, and enables you to quickly change a parameter without launching TDFedit. Personally I would advise you to use TDF Edit as much as you can, as it greatly reduces all the typos you can make while manually editing a file. For those who

want to make things without TDF Edit, here is the tree they must first create :



You can also use a mixed method : create the tree and the main files with TDF Edit and tweak things by hand afterwards. Dont forget to name all the files with the short name you chose for your units.

HOW TO USE TDF EDIT

When you first launch TDF Edit, you must [configure](#) the programm. One this is done, you can : - create a new unit (Create new unit) or - modify an existing unit (Open already existing unit). If you chose to create a new unit, you will have to choose a folder where TDF Edit will save your work.

Unit Information

Please fill this window with all needed information (with *)
I advise you having already a good FBI files done.

Select directory where your unit will be saved:
C:\Temp\ * *Folder where your unit will be saved*

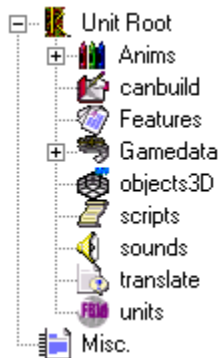
Please enter here the name of your unit (no like Aramon Flying Builder, but like AraFly), this name will be used to name some TDF files. No space, and illegal characters allowed 8 chars max. This is the unitname value of the FBI file)
arabowa * *Short name of your unit*

And now the corps unit name (corpse value in FBI file)
arabowa_dead * My unit doesn't have corpse when it dies
Name of the dead corpse

Please enter now the stone unit name (stone value in the FBI file).
arabowa_stone * My unit can't be stoned *Name of the stoned corpse*

The New Unit Creation Window. Here, the short name is ARABOWA, an Aramon archer. The unit will be saved id C:\temp.

Then, on the left, you will find a tree that holds all the sections needed by the unit. Their names match the names of the folders you would have to create by hand. Let us have a closer look at what is to be done in each section.



ANIMS

Description

In this folder there is a subfolder "Buildpics".



The pic of the unit that will appear in the construction menu is here. It is a JPG image, of size 64x48, 16 bits.

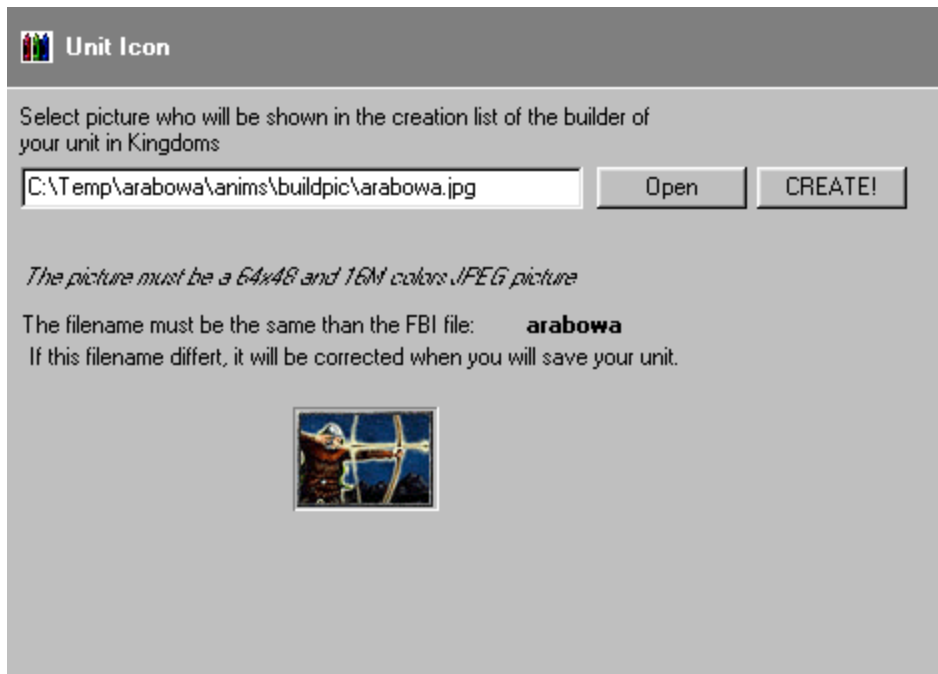


Example : Aramon Flying Builder.

Action

1 : Create an JPG image, of size 64x48, 16 bits. We will use it to represent your new unit in the construction menus.

2 : Launch TDF Edit, click Open and open the image that you just created. It should appear in the window.



The selected image... Ok, its not an original one !

Info

I: Manual Edition of this folder

1: Save your image in the subfolder buildpic, created in the folder anims.

CANBUILD

Description

You will find in this folder the informations needed in the game to build the unit, and its construction capacity if it is a builder.

This folder contains some subfolders:

- If one of these subfolder has the same name of the unit (short name), that means that this unit is a builder, and this subfodler will contain TDF files of all the units it can build.
- If it has the same name as an existing unit (the [short name](#) of the existing unit), then this subfolder will hold the TDF file of the unit you are creating.

Action

1: First, you need to know if your unit is a builder or not. Check "Yeh, my unit is a builder" accordingly.

Yeh, my unit is a builder!

Is your unit a builder ? Here: No!

2: You have to decide which units in TA:K will be able to build your unit.

3: To tell TA:K the builders that can build your units, you must give the [short name](#) of the builders in the "required" list, and give a number. This number gives the place where the pic of your unit will appear in the build menu of the builder. If you want to put it between the 5th and the 6th image, then you can give 5.5.



The builder will be Aramin Keep, and the icon will be placed between the 5th and the 6th.



Here is the result: the buildpic of my unit (mage archer) between the catapult (n. 5) and the architect (n. 6).

4: If you want your unit to be build by other builders, repeat step 3 with "Optionnal Builder #2, #3,...".

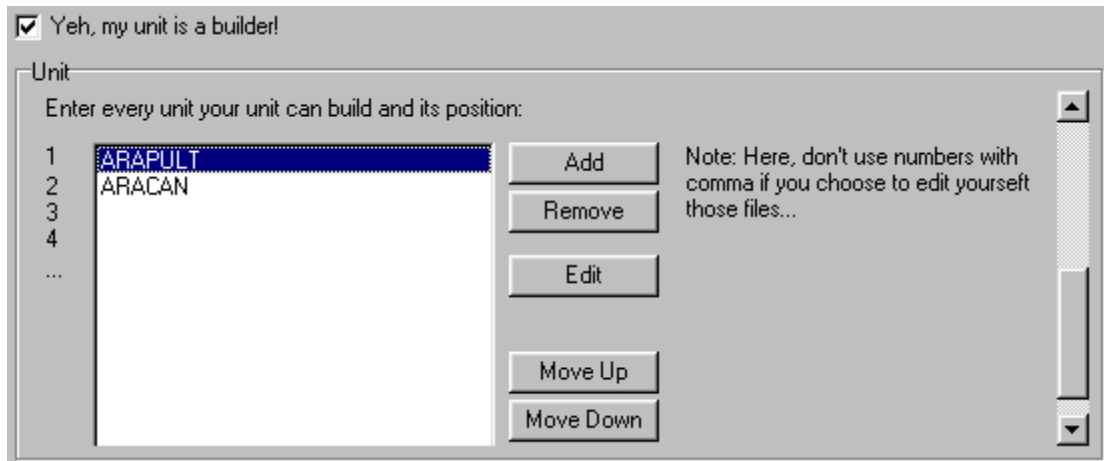
5: If your unit is a buider, continue to step

6: If not, you can go to [Features](#).

6: You are still here, good ! You have checked the "Yeh, my unit is a builder" didnt you ?

7: Now you must choose which units can be built. [Short names](#) are required.

8: click on Add and select the first unit. It will appear in the first place in the build menu. Repeat for all the units you need. Use Move up / down to change their order.



The first unit we will be able to build is ARAPULT on this example. The second is ARACAN.

Info

1: Create files manually

1 : first, you have to create a folder with the short name of the unit that will build yours, in the canbuid fodler.

2 : Use the notepad to type this :

```
[Menu]
{
Priority = X;
}
```

X is the number where the build pic will appear in the build menu.

3 : save this file under the name shortname.tdf, in the shortname subfolder.

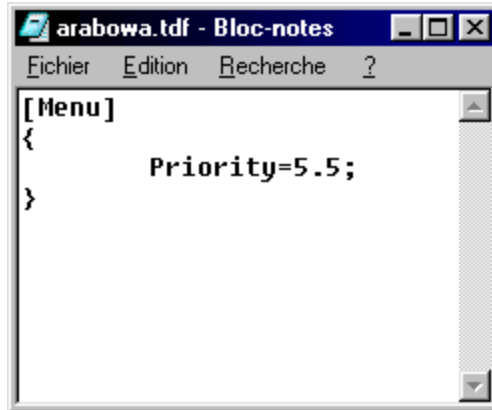
4 : if your unit is a builder, create a folder with the [short_name](#) of your unit.

5: create as much .tdf files as your unit can build :

```
[Menu]
{
Priority = X;
}
```

X is the number where the build pic will appear in the build menu. 1 for the first, 2 for the second...

6 : Save all these files under shortname.tdf in the folder you just created.



An example this file: my example unit (arabowa) is between the 5th and the 6th picture.

FEATURES

Description

This folder contains two other folders : "All worlds" and "Corpses".



1 : Corpses

In this folder you will find a .tdf file, named with the short name of your unit, followed by "_dead" (ex: arabowa_dead.tdf). This file holds the informations about the corpse of your unit.

2 : All Worlds

In this folder is a .tdf file, named with the short name of your unit, followed by "_stone" (ex: arabowa_stone.tdf). This file contains the informations about the petrified body of your unit. If your unit do not leave any corpse when it dies, like the Monarch, you should not fill the following.

Action

1 : Dead Unit (eq. to Corpses)

You just have to fill all the boxes. Here is their meaning.

animatable	Tells whether this corpse can be brought back to life, in the form of a Ghost ou a Ghoul. (1 = yes, 0 = no)
damage	Amount of damage points the corpse can stand before disappearing
description	English description : what is put on the screen when you will point the corpse with the mouse

objects	Name of the .3DO object that will be used to render the corpse. Usually, it is the short name followed by _dead
footprintx	Size of the corpse along the x axis, when the unit faces the bottom of the screen
footprintz	Size of the corpse along the y axis, when the unit faces the bottom of the screen
height	Height of the unit's corpse
noshadow	Does this corpse drop any shadow ? 0 = yes, 1 = no
blocking	Does this corpse block the movement of the other units ? 1 = yes, 0 = no
decomposetime	Amount of time it takes for the corpse to disappear
resurrectable	Is this unit resurrectable ? 1 = yes, 2 = no
reclaimable	Can this corpse be swept by other units ? 1 = yes, 0 = no
world	Leave "all worlds"
hitdensity	Probability (%) that a direct shoot goes through the corpse. 100% : the corpse blocks everything, 0% nothing.

2 : Stone Unit (eq. to All worlds)

world	Same as dead unit.
damage	Same as dead unit. Most of the unit have 4000 here.
description	Same as dead unit.
object	Same as dead unit.
footprintx	Same as dead unit.
footprintz	Same as dead unit.
height	Same as dead unit.
noshadow	Same as dead unit, 0 in all the cases.
blocking	Same as dead unit, 1 in all the cases.
hitdensity	Same as dead unit, 100% in all the cases.
resurrectable	Same as dead unit.
reclaimable	Same as dead unit.
isstone	1 all the time.

If your unit does not leave any corpse when it dies, or when it cant be petrified, leave all these blank.

Info

I: How to edit this folder manually

It allows to add variables that are not accessible from TDF Edit. There are some.

1 : In the folder "features", create two new folders : "all worlds" and "corpses"

2 : Create a file named shortname_stone.tdf in the folder "all worlds" and a file "shortname_dead.tdf" in the folder corpses. Dont do anything if the unit does not leave anything when it dies.

3 : Write this in the "stone" file :

```
[shortname_stone]
{
variable1= X;
variable2= X;
...
}
```

Where shortname stands for the short name of the unit. Replace _stone by _dead for the other file. variable1, ... are variable names that you can choose among the followings, and X are the values you want to give to these variables. You can do that by copy / paste with existing files, taken from existing .hpi.

Variable	Description	Valeur possible
animatable	Leave 1	0 or 1 (<code>animatable=1;</code>)
blocking	Does this corpse block the mouvement of the other units ? 1 = yes, 0 = no.	0 or 1 (<code>blocking=1;</code>)
damage	Amount of damage points the corpse can stand before disappearing.	Positive integer (<code>damage=4000;</code>)
decomposetime	Amount of time it takes for the corpse to disappear.	Positive integer (<code>decomposetime=30;</code>)
description	English description : what is put on the screen when you will point the corpse with the mouse.	English name (<code>description=Archer Stone;</code>)
footprintx	Size of the corpse along the x axis, when the unit faces the bottom of the screen.	Positive integer (<code>footprintx=2;</code>)
footprintz	Size of the corpse along the y axis, when the unit faces the bottom of the screen.	Positive integer (<code>footprintz=2;</code>)
height	Height of the unit's corpse.	Positive integer (<code>height=0;</code>)
hitdensity	Probability (%) that a direct shoot goes through the corpse. 100% : the corpse blocks everything, 0% nothing.	Integer between 1 and 100 (<code>hitdensity=100;</code>)
isstone	Is it a petrified corpse (1) or not (0)	0 or 1 (<code>isstone=1;</code>)
noshadow	Does this corpse drop any shadow ? 0 = yes, 1 = no.	0 or 1 (<code>noshadow=0;</code>)
object	Name of the .3DO object that will be used to render the corpse. Usually, it is the short name followed by _dead for a corpse and _stone for a petrified unit.	Name of the .3DO objetc, without the extension (<code>object=arabowa_stone;</code>)
reclaimable	Can this corpse be swept by other units ? 1 = yes, 0 = no.	0 or 1 (<code>reclaimable=1</code>)
resurrectable	Is this unit resurectable ? 1 = yes, 2	0 or 1

	= no.	(resurrectable=1;)
world	World in which all this take place. Used in Cartographer.	Always "All worlds" (world=All worlds;)

4 : Save these in the right folder. .

```

[arabowa_dead]
{
    animatable=1;
    damage=100;
    description=dead;
    object=arabowa_dead;
    footprintx=2;
    footprintz=2;
    height=0;
    noshadow=1;
    blocking=0;
    decomposetime=30;
    resurrectable=1;
    reclaimable=1;
    world=All Worlds;
    hitdensity=100;
}

```

Example of such a file, here is the one of the "arabowa".

GAMEDATA

Description

When referring to a simple unit, this folder does not contain any .tdf file. But in the case of the Rictus, there is a moveinfo.tdf file in it. This file defines a new way of walking. It is not recommended to edit this kind of file, as this new moveinfo.tdf replaces the previous one. If all the unit makers add their own things, all these informations are going to override each other, resulting in a terrible mess. Ta:Kingdoms takes into account only the most recent version of this file. So we will not explain how to edit this file.

But the Gamedata folder holds a "soundclasses" sub-folder, that holds all the informations about the sounds the unit will emit. This subfolder is the same as the [sounds](#) folder, which holds the sounds your unit will emit.



Action

1: you have to choose a sound (see step 2) for each action your unit can perform. That is : attack, default, guard, move, patrol, select. If you want to add a custom sound, please refer to section [sounds](#).

To listen to existing sounds, open the file "english.hpi" (or the one that matches your language) with HPIView, and double-click on the file.

2: After having chosen a sound (or 2) for each action, you have to select this sound in the list under the action it is linked to. If you want to add a second one, check "hum, i'd like to add one more sound for this action" , and select a second sound in the new list.

3: You have to choose the priority of the sounds. If there is only one sound, the priority of that sound is 1. Simple. If you selected two sounds, out 1 for the rare sound, and a higher number for the other one. If you put 1 and 4, that means that the second sounds will be played 4x more than the first one.

4: You then have to select the priority for all the sounds of the unit. When this unit is in a group, this priority will decide which unit one will hear. Usually we find here a 1, apart for the dragons and the monarchs.

Attack:
Select sound, or enter new one (New sound files have to be added in Sounds tabs)
Toneara **First sound**
Select priority for this sound: 1 **Priority of the first sound**
Click here
 Hum, i'd like to add one more sound for this action
to add a second sound for the action Select sound, or enter new one (New sound files have to be added in Sounds tabs)
arabuildmov3 **second sound**
Select priority for this sound: 1 **Priority of the second sound**

In our example, for the action Attack, the first sound is Tonearan, and the second one is arabuildmov3.

The frequency for both is the same.

Info

I: The bongs that one can hear in TA:K are :

Toneara for Aramon,

Tonetar for Taros,

Tonever for Veruna,

Tonezon for Zhon,

Tonenpc for the non-player characters.

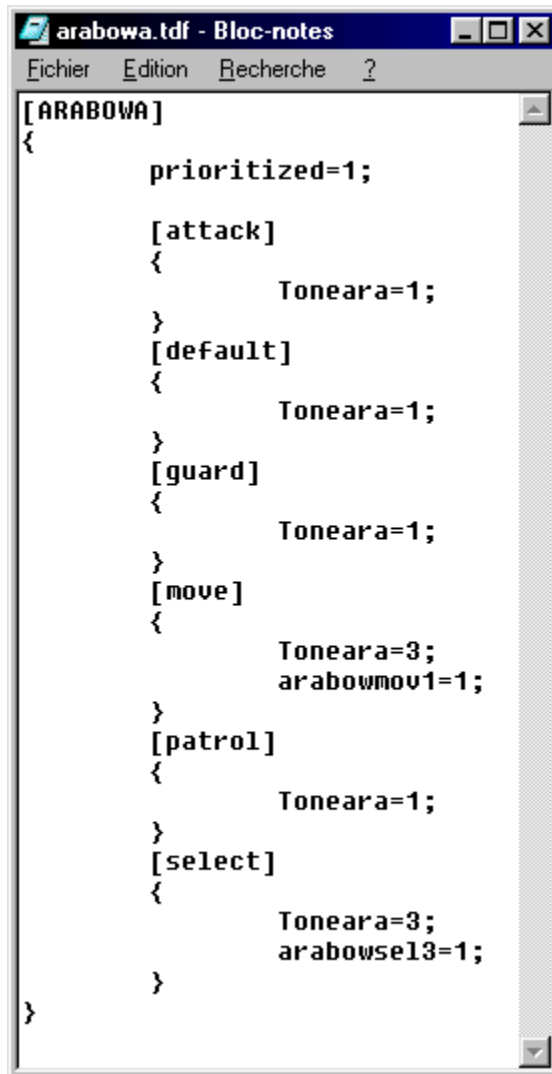
II: Manually editing this file

1 : In the folder Gamedata, create a subfolder "soundclasses". Type this in a notepad :

```
[NOMCOURT]
{
prioritized=1;
[attack]
{
son1=X;
son2=Y;
}
[default]
{
son1=X;
son2=Y;
}
[guard]
{
son1=X;
son2=Y;
}
[move]
{
son1=X;
son2=Y;
}
[patrol]
{
son1=X;
son2=Y;
}
[select]
{
son1=X;
son2=Y;
}
}
```

2: Replace SHORTNAME by the short name of your unit, "sound1" by the sound you want to hear with this action, and X by the priority of that sound. If there is a second sound, use sound2 and Y.

3: Save this file in the subfolder soundclasses with the name SHORTNAME.tdf.



```
[ARABOWA]
{
    prioritized=1;

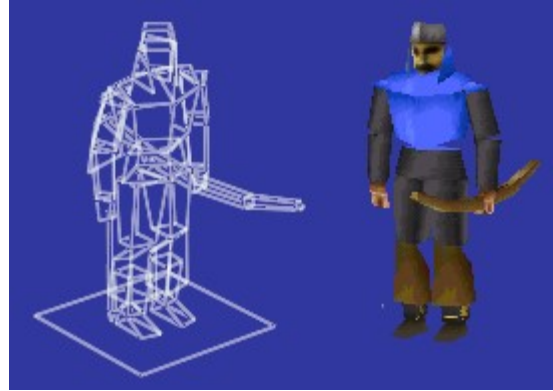
    [attack]
    {
        Toneara=1;
    }
    [default]
    {
        Toneara=1;
    }
    [guard]
    {
        Toneara=1;
    }
    [move]
    {
        Toneara=3;
        arabowmov1=1;
    }
    [patrol]
    {
        Toneara=1;
    }
    [select]
    {
        Toneara=3;
        arabowse13=1;
    }
}
```

The sound file "arabowa.tdf" of our example "arabowa".

OBJECTS3D

Description

This folder contains the different 3D models of your unit. There are up to three : one for the unit itself, one for the corpse, one for the petrified version. These models are made with 3DO Builder +, which adds together models from .LWO files and textures. To make a unit, you need a 3D soft.



The 3D model of arabowa

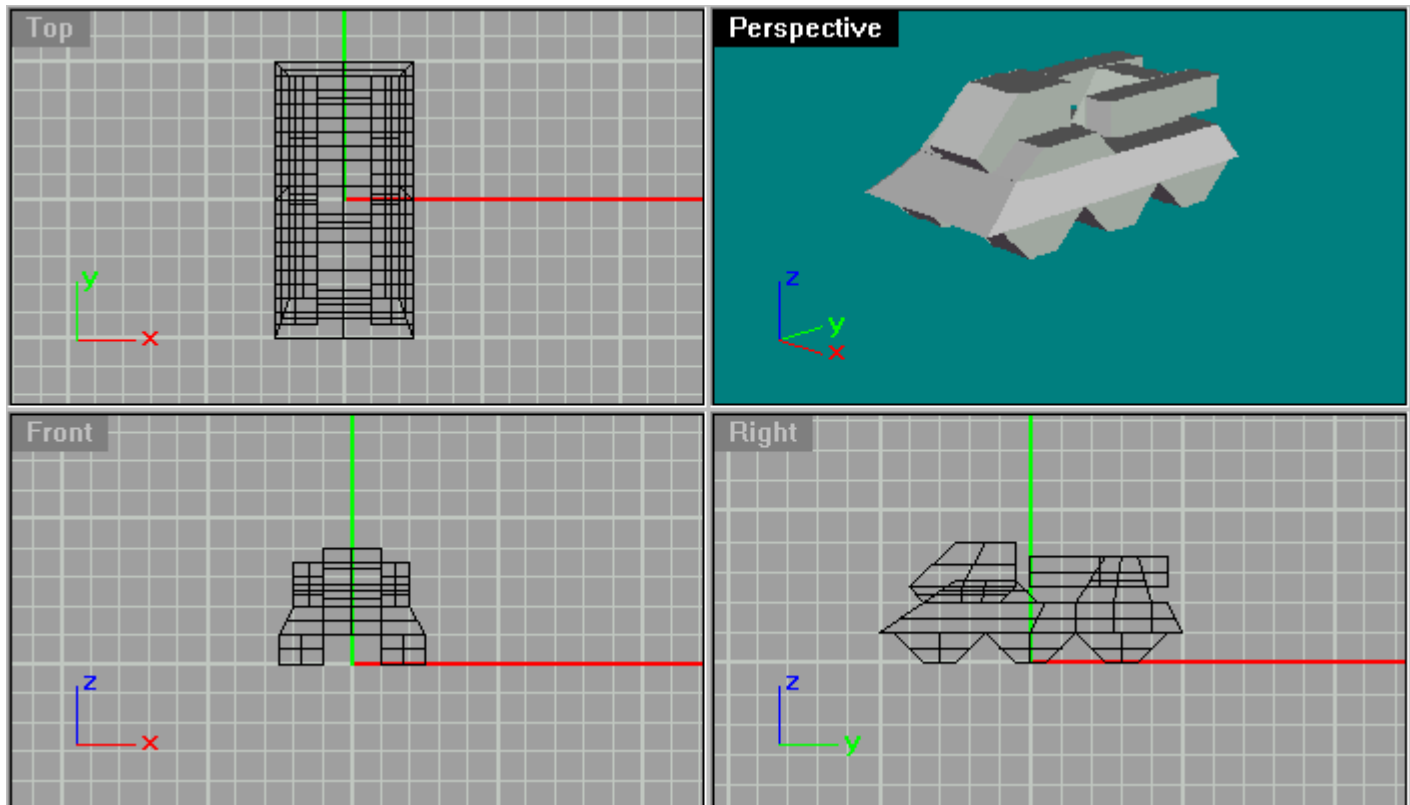
Action

I: In your 3D soft

1: First, you have to create two 3D models for your unit. The first will be used for the unit and its petrified version, the other one for its corpse.

2: Center your 3D models, the feet at level 0 on the vertical axis, and the center on the 0 along the X and the Y axis.

3: Once it is done, to export it to 3D0 Builder +, read below.

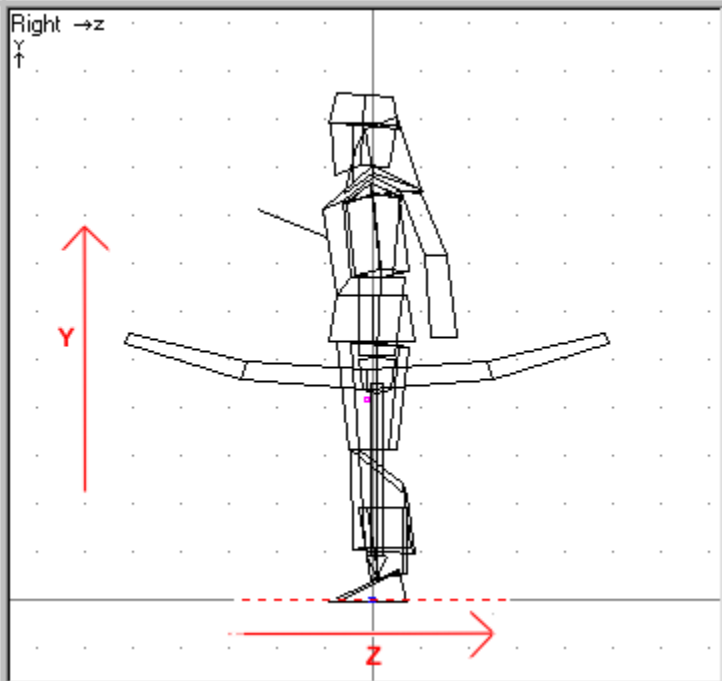
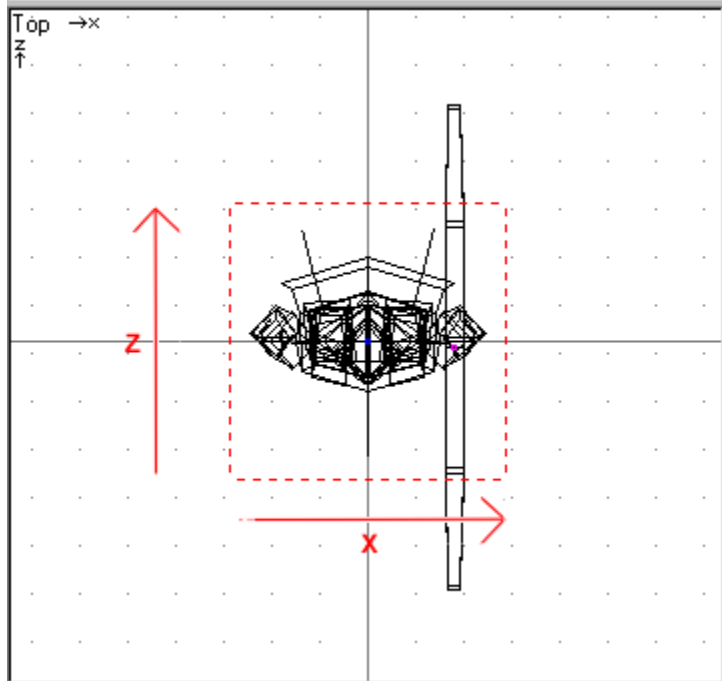
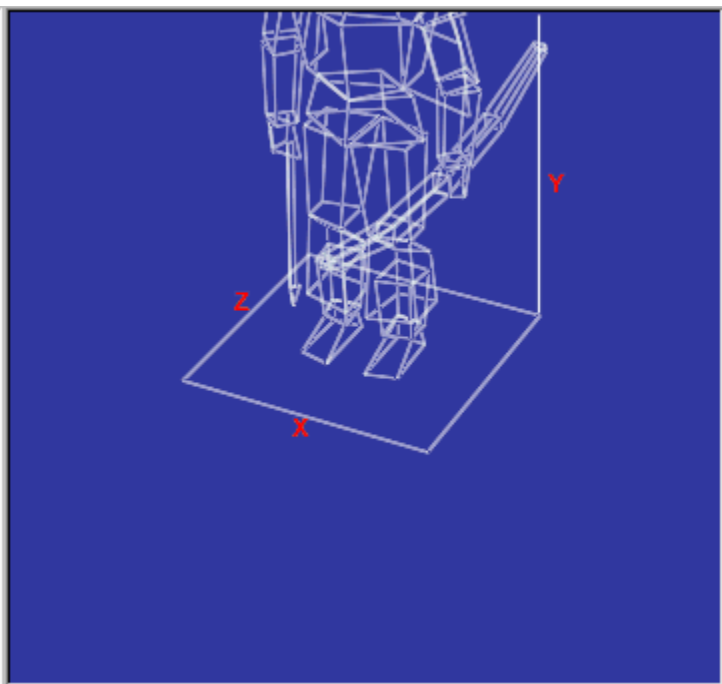
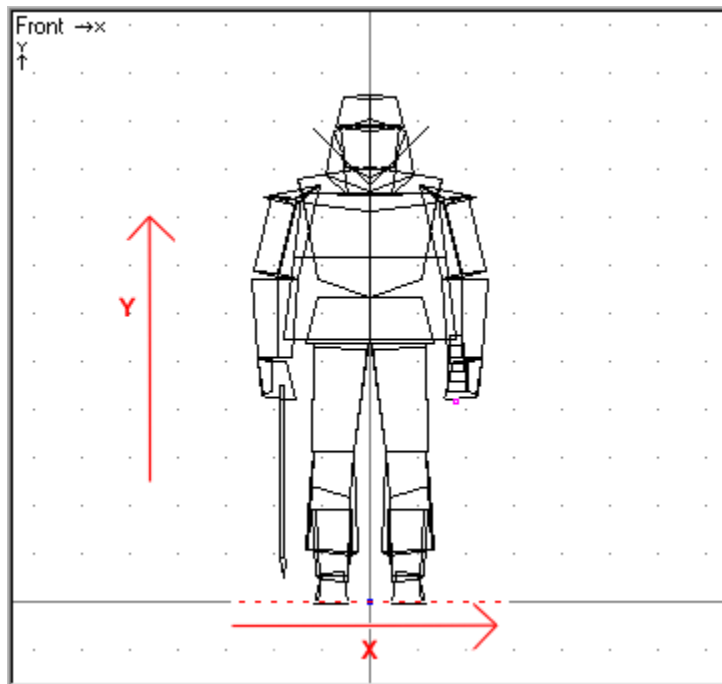


As an example, I will not use the traditional "arabowa", because there

are too many arts and faces in it, and the figures would have been messy.
We will use a missile launcher vehicle, a secret weapon from Elsin's arsenal. Errr, no, I guess its from TA.

Important note

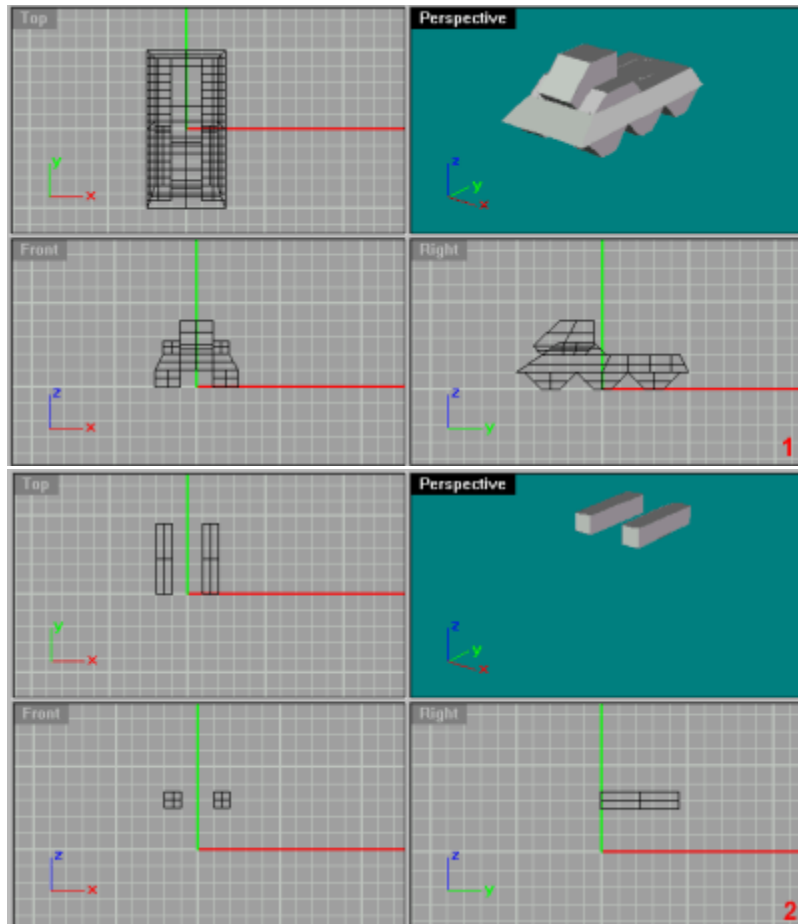
The X axis is the axis that goes from left to right, the Y axis is vertical, and the Z axis is from the front to the back.



The axes in TA:Kingdoms

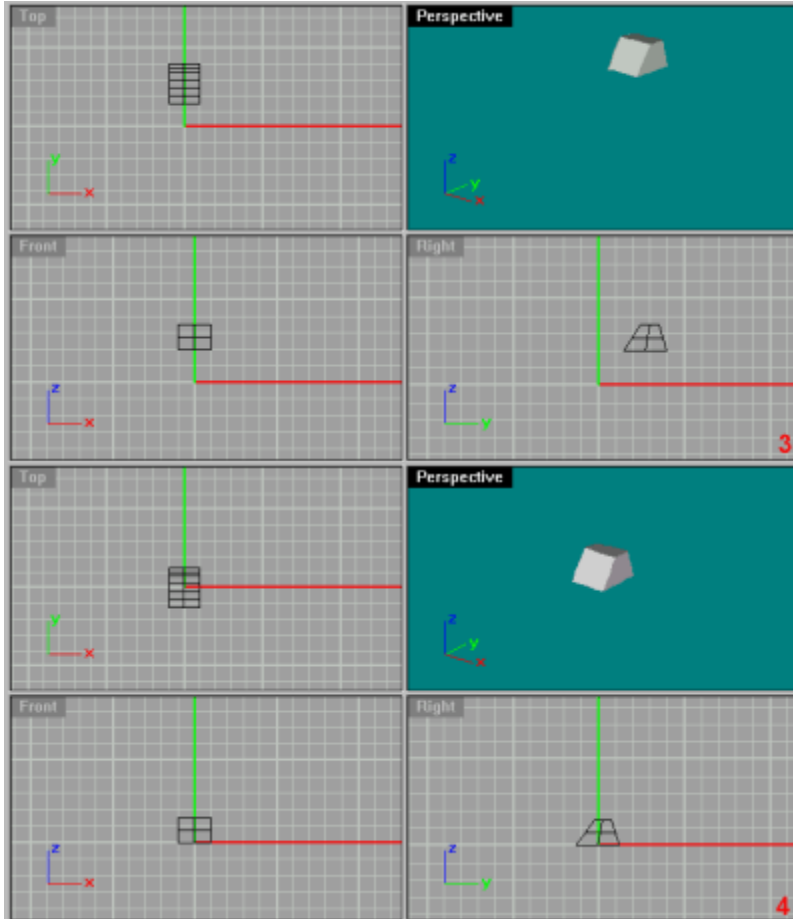
II: Exportation to 3Do builder + 2.0

1: Divide up your model into fixed parts that will be animated. For instance, separate the arm from the torso, from the leg, etc... Save each part in a separate .LWO file. Do this for the model of the unit, not for the dead and the stone model. They are not animated !



In the example of our missile launcher, three different parts : the chassis (1), (2) and the turret (3). In TA:K it is frequent to see 20 parts models of units

2: Then, you have to reopen each of these file, and center each part in it. This has to be done because the center of the axes is the center of the rotation of the part.



Here, we have centered the turret.

3 : All the .LWO files are now going to be imported in 3DO Builder +.

Note about *.DXF files

To make your unit, you can use *.DXF files too. But you must first "soap" this files before. To do this, lunch Visual Soap and configure it like this screenshot.

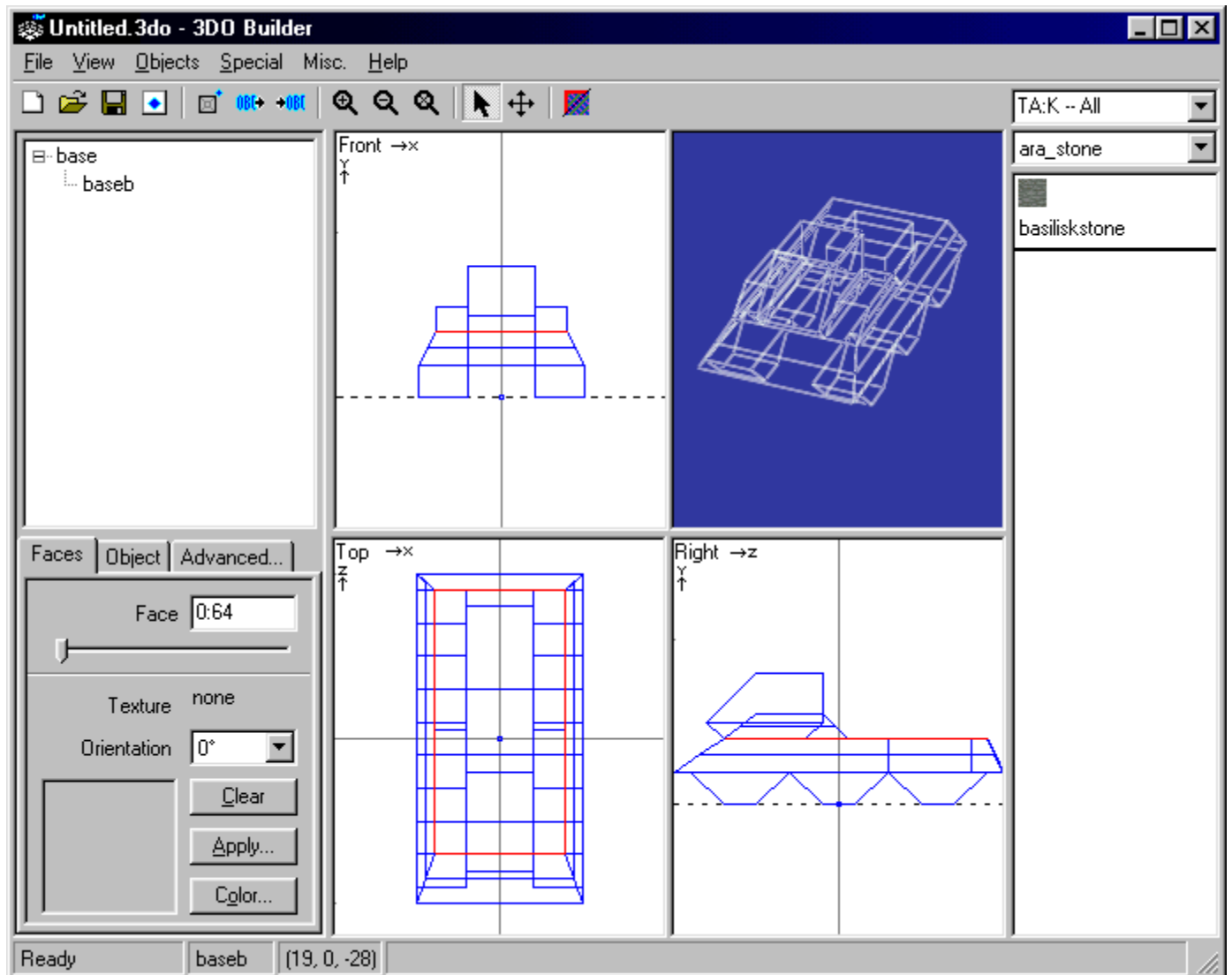


"File in" is your *.DXF file and "File out" is the modified *.DXF file. Click on "Soap it !" to modify your file and adapt it to 3Do builder.

III: Importation in 3Do Builder + 2.0

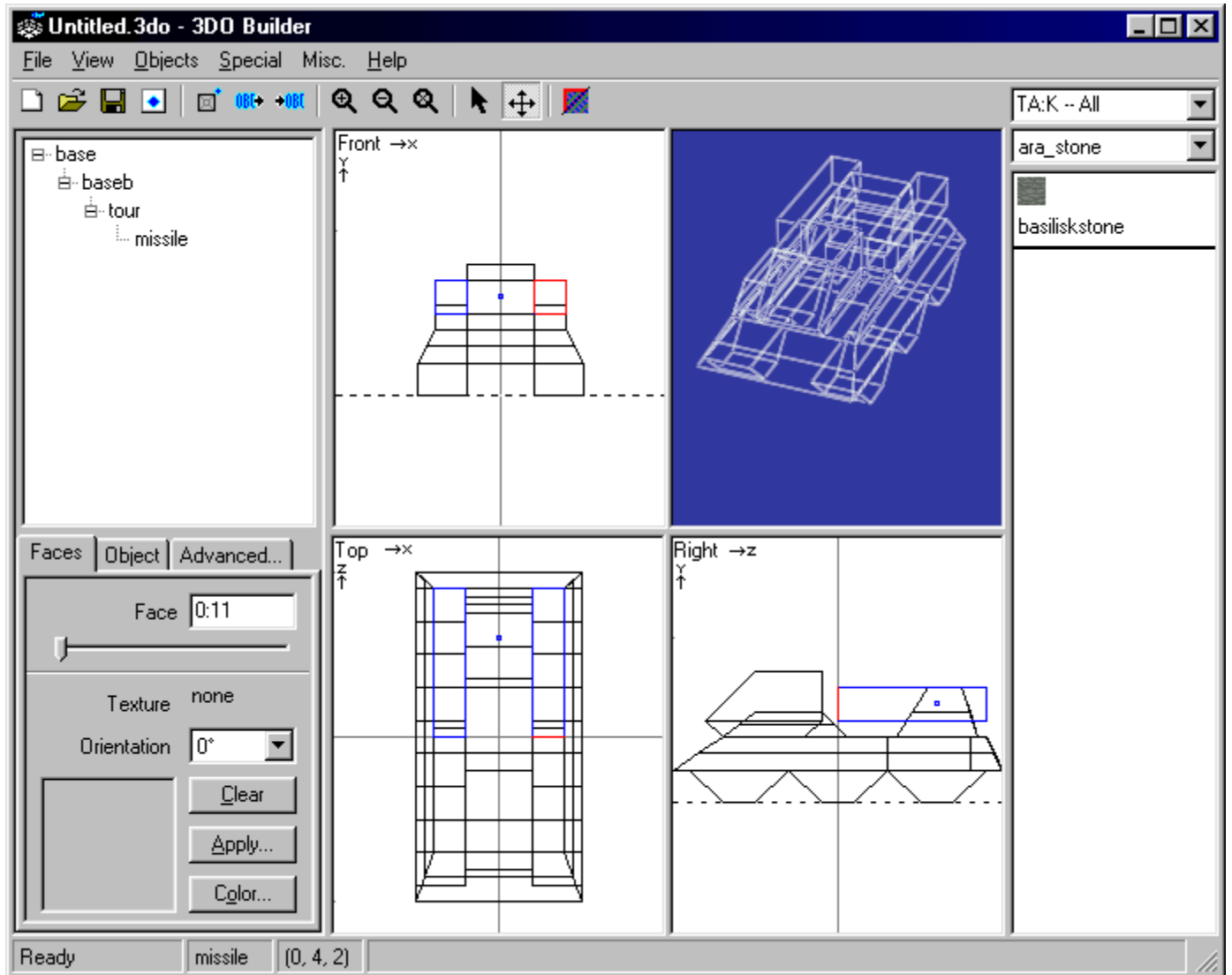
1: Launch 3DO Builder +. On the top left is found the tree of the pieces of your unit. For the moment, there is only one : the basis, which is normal. This tree represents how the pieces are linked together. When a piece at a given node of the tree moves, then all the pieces under this node move also. At the bottom of the window, you will find all you need to adapt your model to TA:K. On the right, are the textures you can map on your model. As 3DO Builder can manage TA and TA:K units, dont forget to select "TA:K-all" in the first pull down menu. At the center of the window, are the four views of your unit : top front and right, and a 3D rendered view. You can change this view between Wire frame, bare frame, shaded frame and textured frame.

2: Let us begin by rebuild our unit. To create a new piece, click on "Objetcs -> create object", and type in the name of your unit. It will appear in the tree on the left. Then, select this piece in the tree, and click "Objects -> import object" to import a .LWO file. Your part should now appear in all the views. You can "click and drag" in the 3D view. Adjust the position of that part, using the icon with the little arrows on it. You can then add all the other parts on that one. On the side views, the selected part is in red, the other one in blue, the blue square represent the rotation axis.



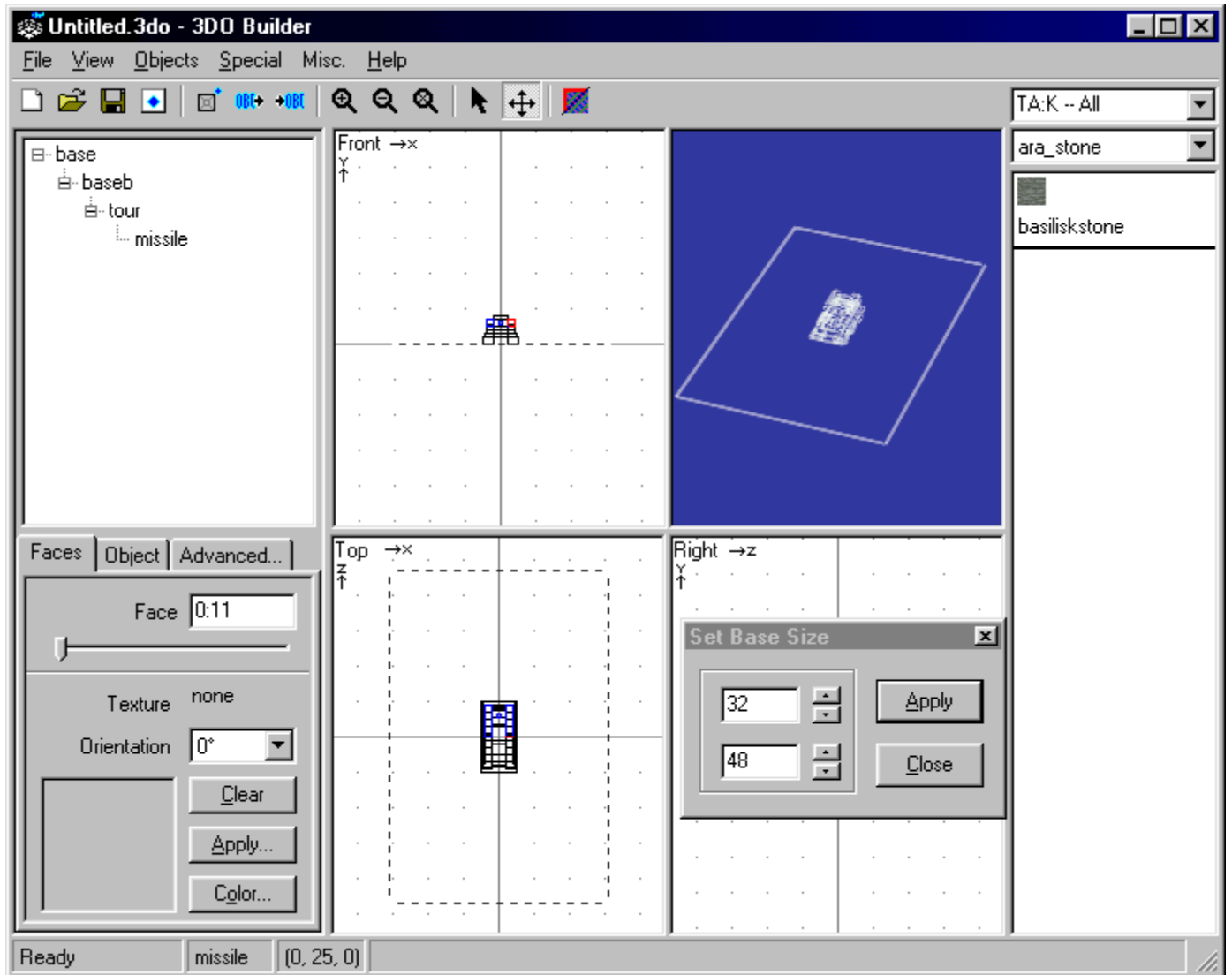
The first piece, the chassis has been imported, one can see it in the pieces tree, it is called "baseb". Don't worry if it is too big or too small, we will take care of that later. Use the zoom to adjust it.

3: Repeat the previous step until all the unit is made. Be careful: never import a .LWO file on the first piece "base", as it is the piece who will give the green circle (square in TA) when the unit will be selected. On the wire frame view, the unit should now appear.



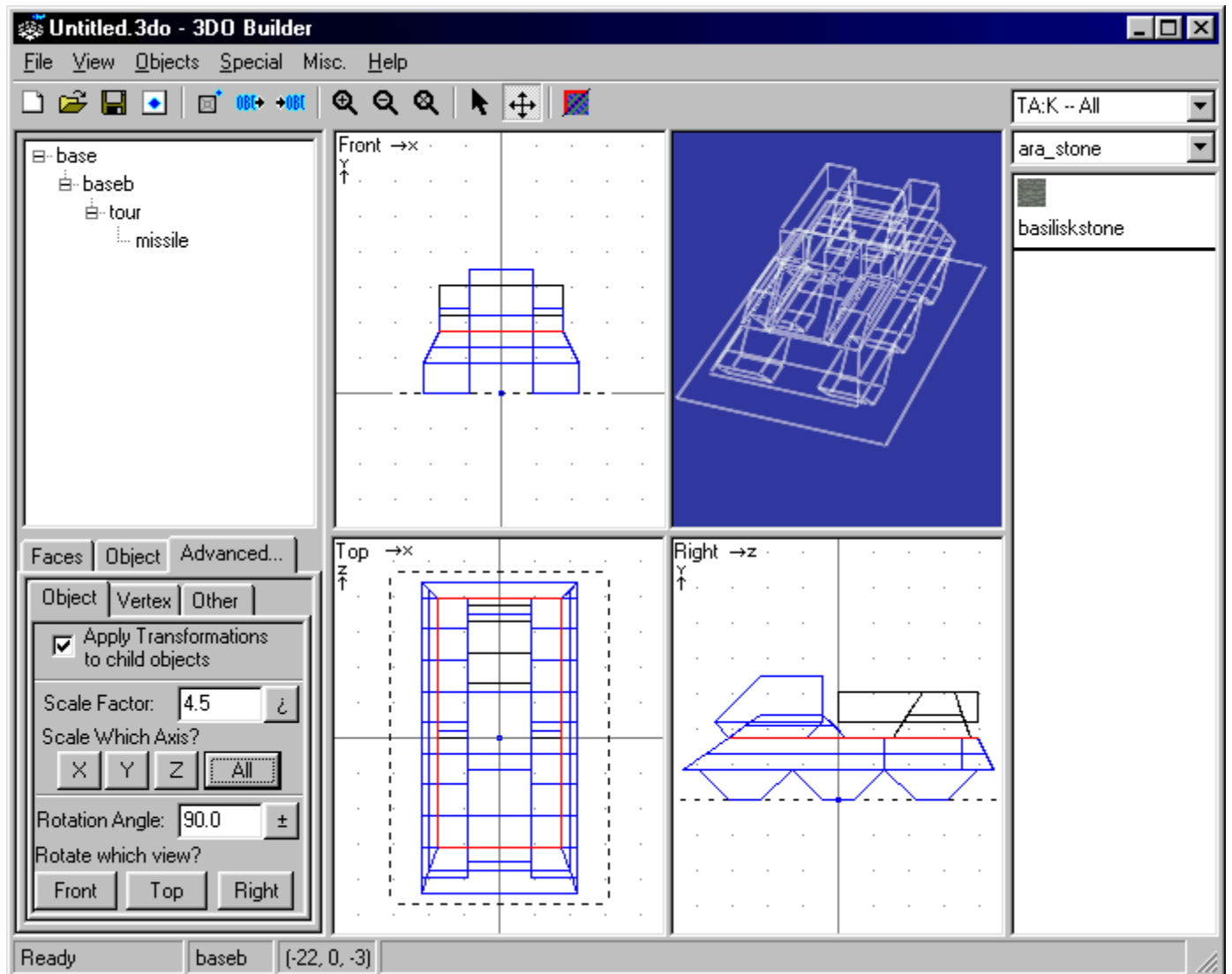
The unit is build. Here it is the piece of the missile tube that has been selected. One can see the center of rotation (blue square).

4: Now you need to adjust the scale of the unit. Select "base" in the pieces tree, then "Special ->Change Ground Plate Size...", and fill in the size you want to give to your unit. . The first figure corresponds to the X axis, and the second one to the Z axis. For instance, the Mage archer is 2x2, and the Gold Dragon is 4x4. Multiply this size by 12 (32x32 for the Mage Archer and 64x64 for the Golden Dragon). Click apply, the size of your unit shold have changed.



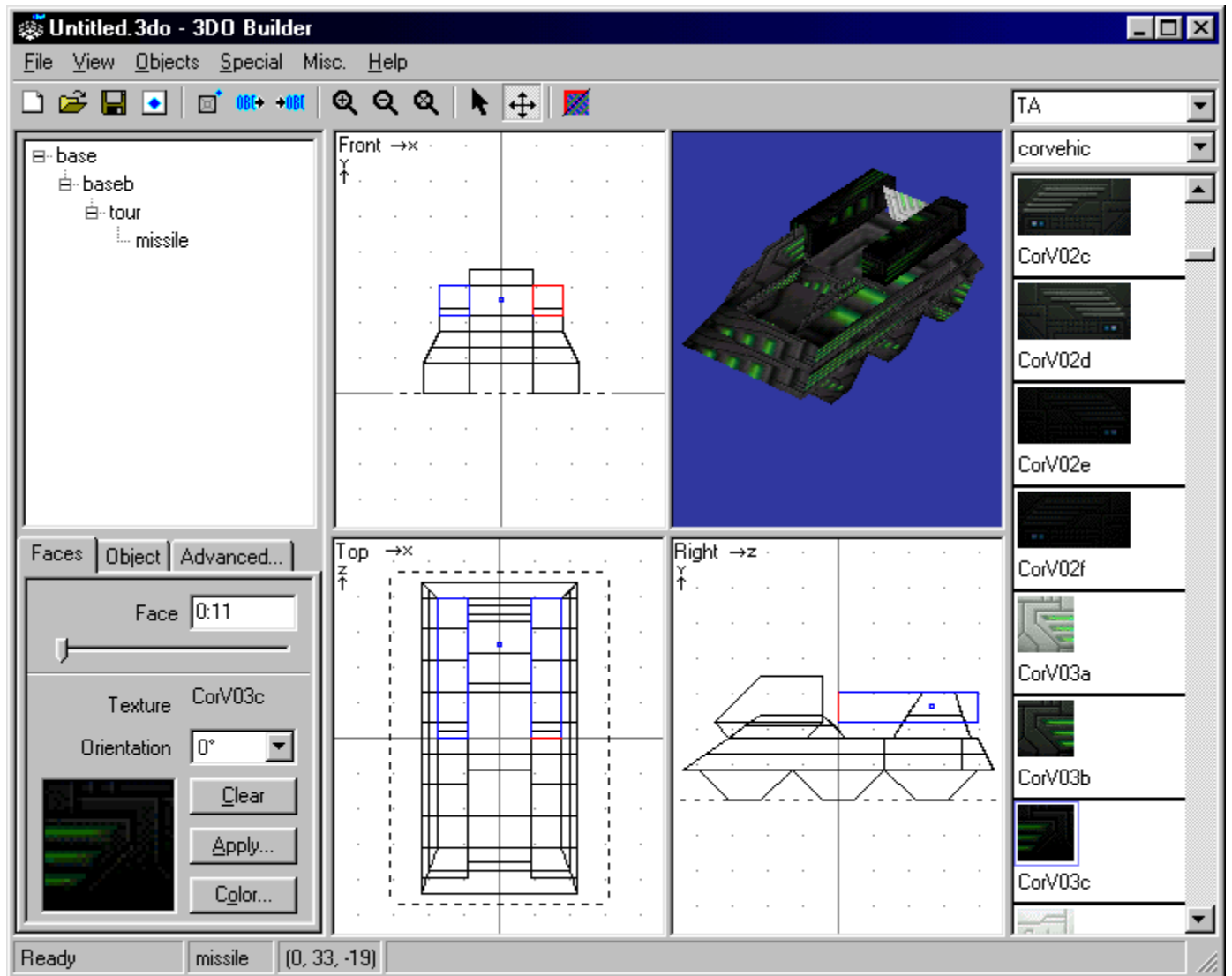
This rocket launcher is of size 2x3, thus I put 32x48. One can see that the unit is a bit small...

5: You then have to adapt the size of the window. Click on the first piece after the piece "base", then click Advanced. On the left, click "apply transformation to child object". Then, using the scale factor function, try to adapt the size of your unit to the one of the window. Then click on All to apply that on all the views.



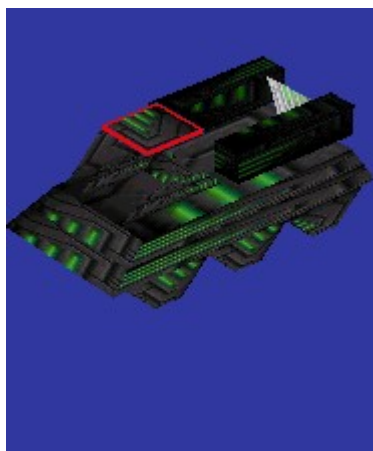
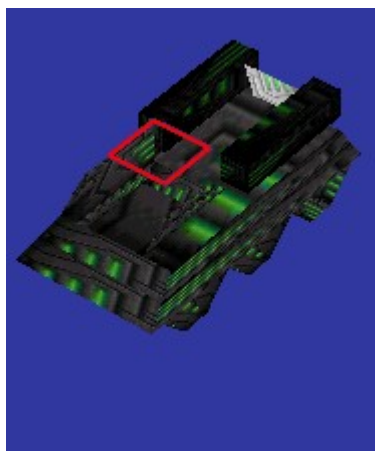
On example, zoomed in. The scale factor is 4,5.

6: Lets continue with the texture mapping. We just have one thing left : be sure that all the faces have the same orientation. If a face has the wrong orientation, we will not see any texture on it. Choose the 3D view, and apply a dummy texture on all the faces. Repeat the operation for each piece, apart from "base". Now, in the 3D view, the object should appear with the same texture ofr its pieces. If one face has no texture, that means it is oriented in the wrong way.



On this example, some textures (turret) are on the wrong side.

7: To put the faces on the correct sense, select them one by one, and hit F2. That's all.

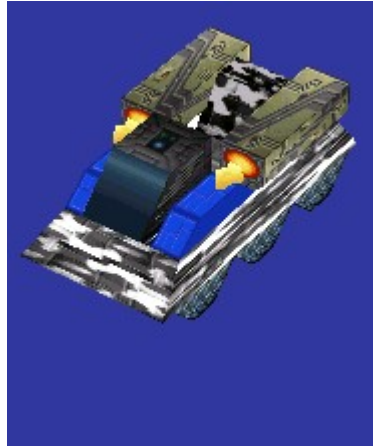


One face to be turned...

A F2 later, it is done...

Some more F2 later, our model is correct

8: You can now apply the texture you want. For this, select a face, double-click on the texture you want to apply. You can also apply a simple color, with the "Color" option.



The textured missile launcher. Ok, those are TA textures, but it is the same for TA:K.

9: Here it is ! Your 3D model should be finished. Now you can save it under the short name or your unit, and do the same for the stoned version and the corpse. The stone texture is called "basiliskstone". The corpse should be save in shortname_dead.3do, and the stone version in shortname_stone.3do.

The 3D models or our great arabowa



The body :
arabowa.3do



The corpse :
arabowa_dead.3do



The petrified body :
arabowa_stone.3do

IV: Import your three files .3do with TDF Edit

1: Click on the three "Open" button to select the three 3D models you just made.

2: You can edit your model by clicking on "Edit". It will automatically launch 3DO Builder +.

3: You can use an existing corpse or stone body by clicking on "I use an already existing..." and give a short name to the 3D object. Remind that you have to configure the [Features](#) for that.

THE NAME OF THE .3DO FILE IN THE BOX SHOULD ALWAYS MATCH THE ONE UNDER EDIT.

The screenshot shows a configuration window with three sections for setting 3D files. Each section has a text input field, 'Open' and 'Edit' buttons, and a checkbox for 'I use an already existing object (in data.hpi)'. The 'your unit' section has the path 'C:\Temp\arabowa\objects3D\ARABO' and the name 'ARABOWA'. The 'dead unit' section has the path 'C:\Temp\arabowa\objects3D\arabowa' and the name 'arabowa_dead'. The 'stone unit' section has the path 'C:\Temp\arabowa\objects3D\arabowa' and the name 'arabowa_stone'. A tip on the right explains that for Kingdoms packages, the filename should be just the name, not the path.

Unit Type	3D File Path	Object Name	Use Existing Object
your unit	C:\Temp\arabowa\objects3D\ARABO	ARABOWA	<input type="checkbox"/>
dead unit	C:\Temp\arabowa\objects3D\arabowa	arabowa_dead	<input type="checkbox"/>
stone unit	C:\Temp\arabowa\objects3D\arabowa	arabowa_stone	<input type="checkbox"/>

Info

I: Description of 3do Builder

3DO Builder + has many other features than the one described here. All the features are [here](#).

II: TA:Kingdoms textures creation for 3DO Builder +

1: Open the Data.hpi file, with HPI View

2: Create a temporary folder called "textures"

3: Save all the files of the textures folder of the data.hpi in the folder you just created.

4: Launch HPI Pack, select your temporary folder in "Directory to pack", and give a name "Textures.ufo" to the new file you are creating. Save this file in your TA:K directory. Select TA and TA:CC compression method.

5: Click pack, verify that the file has been created.

6: Launch 3DO Builder + and select TA:K all. If all went right, you should see a list of textures at the top right of the window. If not, or if the colors of the textures are not correct, that means that it did not work properly.

These steps are well-detailed in the file Compatibility.txt, from the .zip of 3DO Builder +.

SCRIPT

Description

The .COB files are found in this folder. It contains the animation scripts of the unit. This script is in fact a .BOS file, that has to be compiled into a .COB file. The TA:K Cobbler is not published yet.

Action

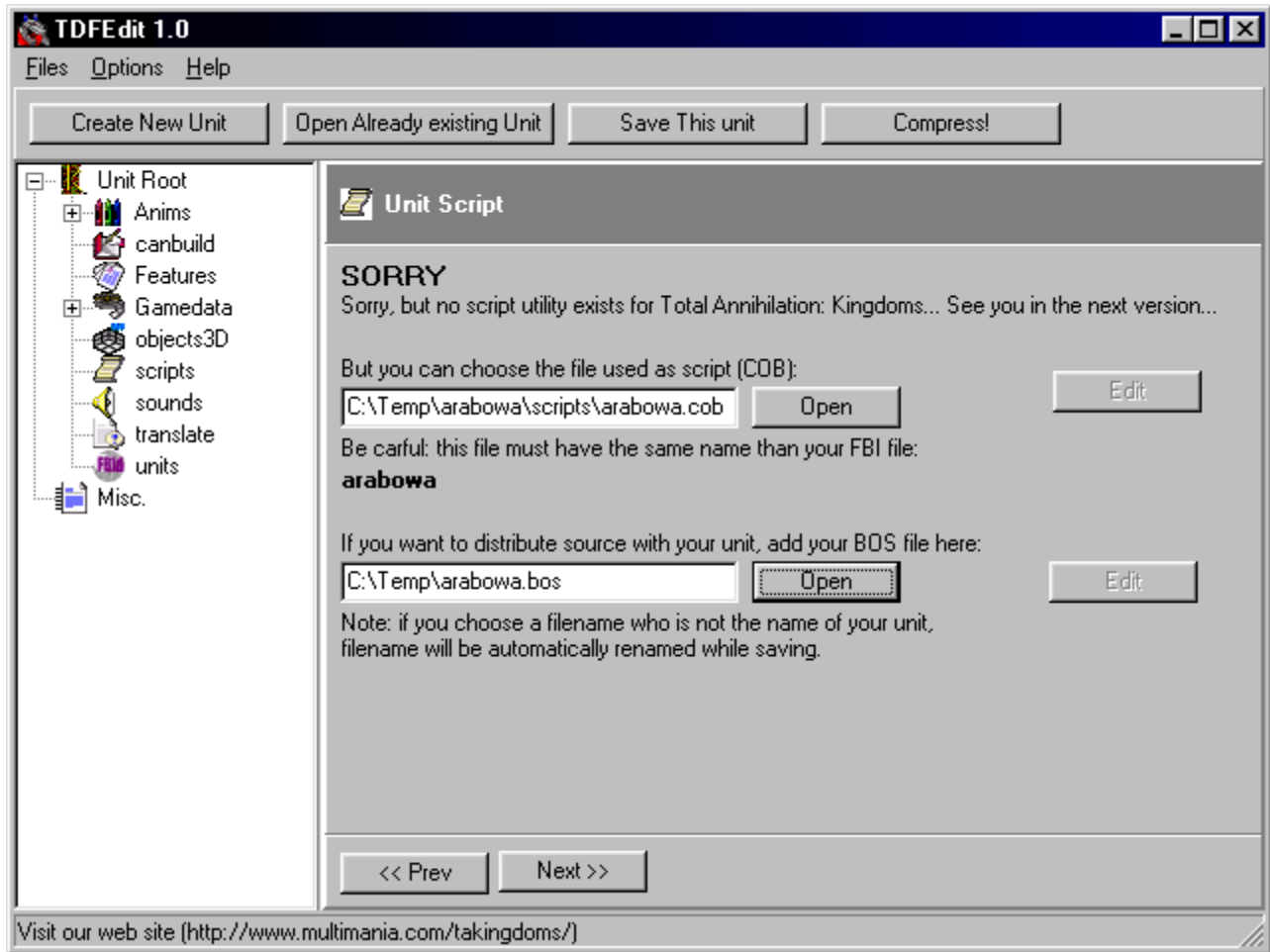
I: Creation of the script file (.BOS) and compilation into .COB

This step is not realizable for the moment, it will be described later.

II: Importation of the script in TDF Edit

1: Create your script .BOS and compile it a in .COB file, called shortname.cob.

2: In TDF Edit, click on the first Open button, to give your .COB file, and on the second one to give the .BOS file. The .BOS file is not needed, but if you put it here, you are sure you will not lose it !



The files "arabowa.cob" and "arabowa.bos" in TDF Edit.

Info

I: Manually editing this file

- 1: Create the .BOS and .COB files, as explained in the previous section.
- 2: Place these two files in the Scripts folder of your unit.

SOUNDS

Description

This folder holds the .WAV files of the new sounds you can add. You are advised to use file names in majuscules, without any spaces in them. These names will be used in the TDF file of the soundclasses folder. The

sampling rate of the files are : 11 025 Hz, 8 bits, Mono, Microsoft RIFF Wave.

Action

- 1: You need to have a .wav file ready to be put into TA:K. Save it using a 8 letters name, non special characters, no space.
- 2: Click Open, and select all the files you want to add.
- 3: Write down the names of the sounds, and go to the [Gamedata](#) section to add them.
- 4: To add new sounds, do as explained in the [Gamedata](#) section, but dont select anything in the list. Copy the filename in the right box, respect min / maj, and do NOT put the .wav extension.

Info

I: How to edit this folder manually

- 1: In the folder "sounds" of your unit, put the sounds that you want to add.
- 2: You can use these file name in the section "[gamedata -> soundclasses](#)"

TRANSLATE

Description

Cavedog has noticed that on the earth, not everybody was talking english. So they decided to add multilingual support to their games. So, TA:K exists in five languages : English, French, German, Italian and Spanish. How does it work ? In fact, when the game is launched, it is in english. Once everything is in memory, the engine loads all the files of the Translate folder, and read the translation of the texts and names. The language you have installed is recorded in the Windows Registry.

Action

- 1: Put in the first box the english name of your unit, that will be recorded in the .FBI file of the [Units folder](#).
- 2: Put in then the names in the other languages in the right boxes.
- 3: Do the same for the corpse and the stone body, apart from the first one, chosen in the [features](#) section.

Info

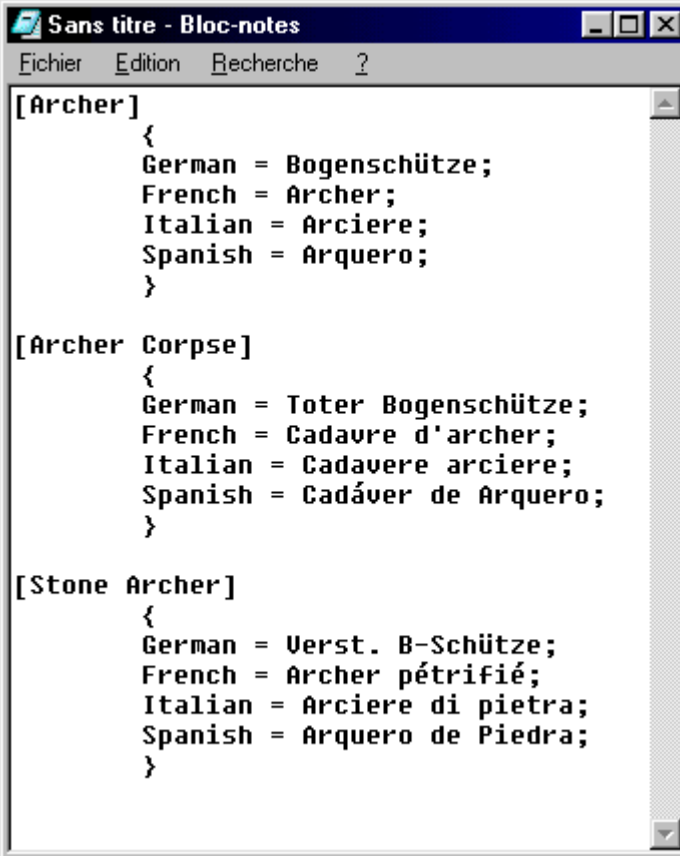
I: How to manually edit this file

You can translate things that you cannot reach which TDF edit.

1: With a notepad, write all this :

```
[ENGLISH_NAME]
{
german=TRADUCTION;
french=TRADUCTION;
italian=TRADUCTION;
spanish=TRADUCTION;
}
```

2: Replace `ENGLISH_NAME` by the word you want to translate, and `TRADUCTION` by its translation into the given language.



```
Sans titre - Bloc-notes
Fichier  Edition  Recherche  ?

[Archer]
{
German = Bogenschütze;
French = Archer;
Italian = Arciere;
Spanish = Arquero;
}

[Archer Corpse]
{
German = Toter Bogenschütze;
French = Cadavre d'archer;
Italian = Cadavere arciere;
Spanish = Cadáver de Arquero;
}

[Stone Archer]
{
German = Verst. B-Schütze;
French = Archer pétrifié;
Italian = Arciere di pietra;
Spanish = Arquero de Piedra;
}
```

Example of a file from the translate folder

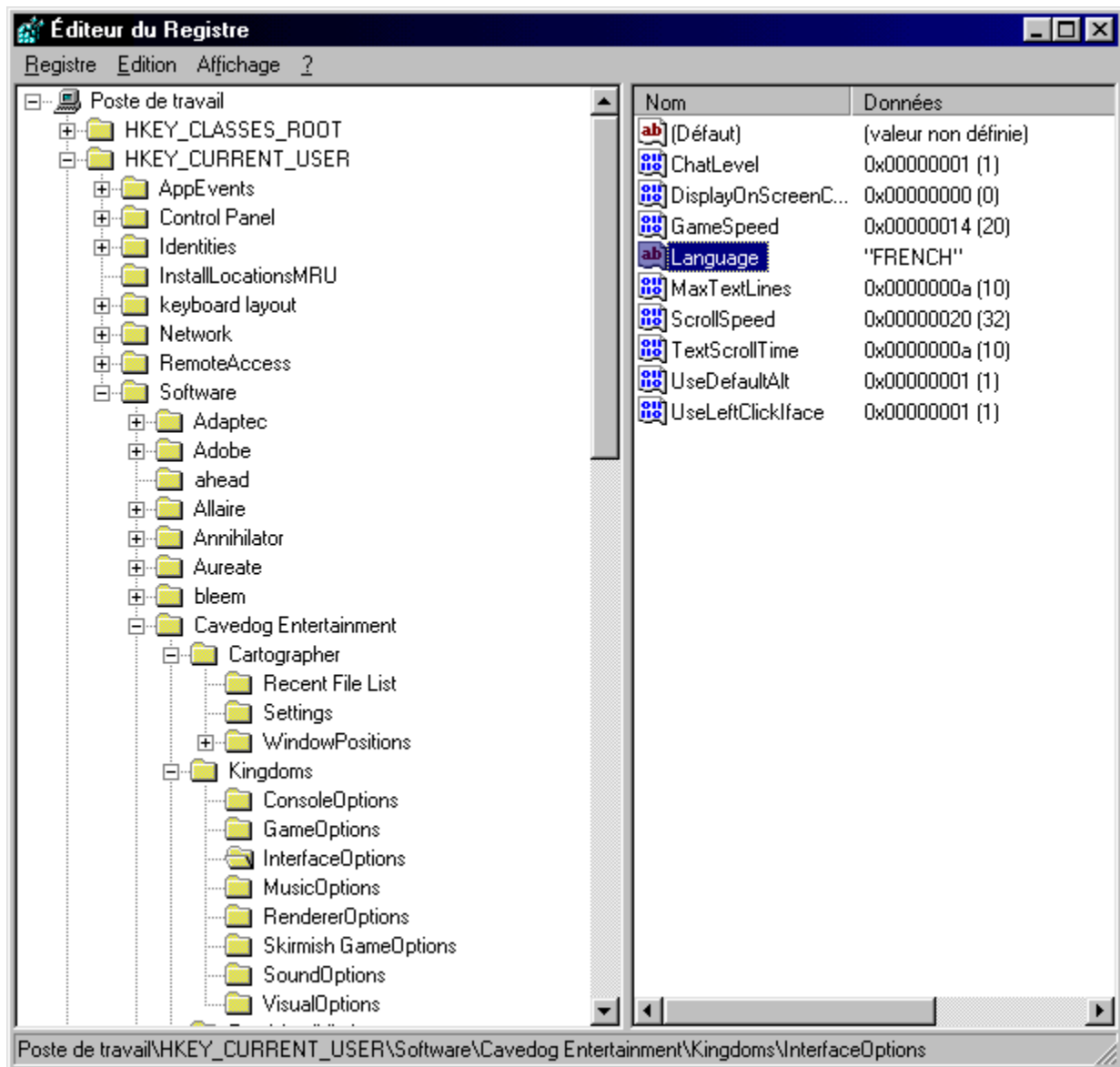
II: How to change the default language

Beware of the following ! Windows is an unstable OS, tweaking the registry without knowing what you are doing can lead to major crashes, definitive loss of data and programs, hours of reboot, reformats and numerous

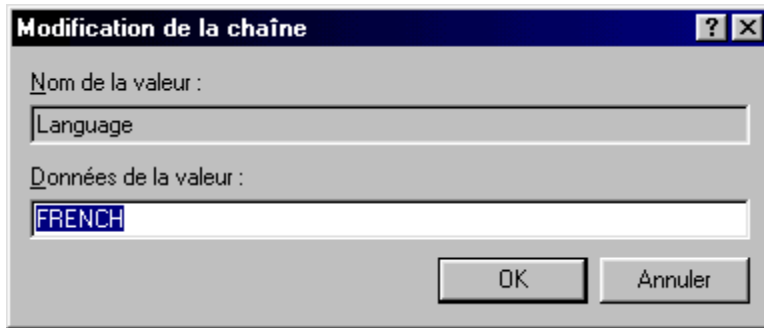
reinstallations of all your PC and hard drive. And terrible headaches. And, worst of all, big laughs from your linuxians friends. Furtive Fox, GFRUIT Rene and l'Arm à l'œil have warned you, and cannot be taken responsible for any damage to your computer or your health. If you want to have a complete, say, Italian game, you also need the file Italian.hpi in your TA:K installation directory.

1: Open the registry ***** BEWARE OF WHAT YOU DO NOW ***** No undo is available in this process. Startup -> execute, then type in regedit and return. You are in the registry editor.

2: In the left tree, chose HKEY_CURRENT_USER -> Software -> Cavedog Entertainment -> Kingdoms -> InterfaceOptions, double click on language.



3: Change the value ENGLISH into FRENCH, or GERMAN, or ITALIAN, or SPANISH. Click Ok. That's all.



Here it is, TA:K in french !

UNITS

Description

This folder contains a .fbi file that holds all the informations on the units parameters : name, resistance, speed, type, weapons, ...

Action

- 1: Click on Create in TDF Edit to create a new FBI file.
- 2: Click edit to edit this file.
- 3: Edit the .fbi file. Follows explanations on each variable. Click File -> Save and exit when it's done.
- 4: Edit the weapons part, explain below.

Info

I: The variables of the .fbi files

If you dont know which values to give to the variables, take a look at the existing .fbi to have an idea.

Valeur de FBI	Explication	Variable
Description		
Name-	Long name of your unit	name = Mage Archer ;
Description (side)-	Side or description of your unit	description = Aramon ;

Side-	Side of your unit	side = <code>ARA</code> ;
Unit Category-	Category of your unit: <code>ARA</code> = Aramon, <code>VER</code> = Veruna, <code>TAR</code> = Taros, <code>ZON</code> = Zhon, <code>MON</code> = Monster, <code>NPC</code> = Non-playable unit, <code>ATTACK</code> = can attack?, <code>Magic</code> = is unit with mana?, <code>FLY</code> = Flying unit, <code>FACTORY</code> = can your unit build?, <code>Monarch</code> = Monarch	category = <code>ARA ATTACK</code> ;
Unitname-	Short name of your unit	unitname = <code>ARABOWA</code> ;
Unit's 3do-	*3do file of your unit, without the extension *.3do (generally it's the short name of your unit)	objectname = <code>ARABOWA</code> ;
Unit Version-	Version of your unit	version = <code>1.2</code> ;
Unit ID	Number of your unit between 1 and 9999, this number must be unique	unitnumber = <code>1995</code> ;
Unit Capabilities #1		
Does the unit activate when build	Does the unit activate when build (the gate,...)?	activatewhenbuilt = <code>1</code> ;
Does the unit attracts gods	Does the unit attracts gods?	attractsgods = <code>1</code> ;
Is the unit a builder	Is the unit a builder?	builder = <code>1</code> ;
Can the unit NOT aid with the construction of other buildings/units	Can the unit NOT aid with the construction of other buildings/units?	builderlimited = <code>1</code> ;
Can the unit capture other units	Can the unit capture other units?	cancapture = <code>1</code> ;
Can the unit attack	Can the unit attack?	canattack = <code>1</code> ;
Can the unit stop	Can the unit stop?	canstop = <code>1</code> ;
Can the unit cloak	Can the unit cloak?	cancloak = <code>1</code> ;
Can the unit fly	Can the unit fly?	canfly = <code>1</code> ;
Can the unit guard	Can the unit guard?	canguard = <code>1</code> ;
Can the unit hover	Can the unit hover?	canhover = <code>1</code> ;
Can the unit transport other unit	Can the unit transport other unit?	cantransport = <code>1</code> ;
Can the unit load other unit	Can the unit load other unit? (for a transport unit)	canload = <code>1</code> ;
Can the unit move	Can the unit move?	canmove = <code>1</code> ;
Can the unit patrol	Can the unit patrol?	canpatrol = <code>1</code> ;
Can the unit clean (reclaim)	Can the unit clean (reclaim)	canreclaim = <code>1</code> ;
Can the unit raise dead units into ghol	Can the unit raise dead units into ghol?	cananimate = <code>1</code> ;
Can the unit bring units back to life	Can the unit bring units back to life?	canresurrect = <code>1</code> ;

Can the unit NOT be captured	Can the unit NOT be captured?	cantbecaptured = 1;
Can the unit NOT be turned to stone	Can the unit NOT be turned to stone?	cantbestoned = 1;
Unit Capabilities #2		
Is this unit a gate	Is this unit a gate?	gate = 1;
Is this unit a ghost	Is this unit a ghost?	ghost = 1;
Does this unit hover while attacking	Does this unit hover while attacking?	hoverattack = 1;
Does this unit become a feature when build (walls)	Does this unit become a feature when build (walls)?	isfeature = 1;
Does this unit NOT have a shadow	Does this unit NOT have a shadow?	noshadow = 1;
Does this unit NEVER become a veteran	Does this unit NEVER become a veteran?	noveteran = 1;
Can this unit be turned off or on	Can this unit be turned off or on?	onoffable = 1;
Should enemy units target this unit	Should enemy units target this unit?	shootme = 1;
Does this unit NOT tilt over uneven terrain	Does this unit NOT tilt over uneven terrain? (1 for walking units)	upright = 1;
Does this unit automatically switch weapon when out of mana	Does this unit automatically switch weapon when out of mana? (for unit which use mana for weapon)	weaponswitching = 1;
Can the unit float on water	Can the unit float on water?	floater = 1;
Can the unit NOT be transported	Can the unit NOT be transported?	cantbetransported = 1;
Health/Armor/Mana #1		
Body Type-	Type of body: armor = armor; flesh = without armor; stone = stone; scales = with scales; wood = in wood.	bodytype = flesh;
Damage Category-	!!!!	damagecategory = human;
Maximum Mana-	Max mana for the weapons.	maxmana = 120;
Mana recharge Rate-	Mana recharge Rate.	manarechargegerate = 1.4;
Maximum Hit Points-	Resistance of the unit.	maxdamage = 1421;
Heal Rate-	Heal Rate	healtime = 2;
Experience Points-	Experience points giving to the unit which kill your unit.	experiencepoints = 20;
Health/Armor/Mana #2		
Cloak Cost Standing-	Cloak Cost when your unit is standing.	cloakcost = 50;
Cloak Cost Moving-	Cloak Cost when your unit is moving.	cloakcostmoving = 500;

Minimum Cloak Distance-	Minimum Cloak Distance. (in pixel)	mincloakdistance = 50;
Mana Generated-	Mana created by the unit	mogriumincome = 8.2;
Mana Stored-	Mana Stored.	mogriumstorage = 5000;
Blood Color 1-	First color of blood.	bloodcolor1 = 160 30 0;
Blood Color 2-	Second color of blood.	bloodcolor2 = 160 30 0;
Blood Color 3-	Third color of blood.	bloodcolor3 = 160 30 0;
Stone-	Name of the *3do file of your stone corpse.	stone = arasword_stone;
Corpses-	Name of the *3do file of your dead corpse.	corpse = arasword_dead;
Misc Unit Variable #1	1	
Acceleration-	Accélération of the unit	acceleration = 10;
Maximum speed-	Max Speed	maxvelocity = 1.1;
Brake rate-	Brake rate (- than 1 = short turn, + than 10 = large turn)	brakerate = 0.15;
Maneuvering Distance-	Max distance for maneuvering (to avoid a stone for example, or to attack an other unit)	maneuverleashlength = 500;
Turn In Place Rate-	Turn Rate When standing	turninplacerate = 2500;
Turn Rate-	Turn Rate When moving	turnrate = 3000;
Cruise Altitude-	Cruise Altitude of a flying unit	cruisealt = 150;
Build Cost-	Build Cost	buildcost = 43944;
Build Time-	Build Time	buildtime = 8755;
Mission When Built-	standby for all the unit, except VTOL_standby for flying unit.	defaultmissiontype = Standby;
Misc Unit Variable #2		
Radar Distance-	Radar distance of the unit (in pixel): zone where your unit can see other unit in the minimap	radardistance = 500;
View Radius-	Line of sight (in pixel)	sightdistance = 135;
Sonar Radius-	Used in TA	sonardistance = 500;
Road Multiplier-	Rate which increase speed when the unit is on a road	roadmultiplier = 1.21;
Water Multiplier-	Rate which increase speed when the unit is in the water (you can use number between 0 and 1 to reduce the speed)	watermultiplier = 0.75;
Order When Built-	Not for building. Aggressivity of the unit: 0 = passive; 1 = défensive; 2 = offensive.	standingunitorder = 0;
Building Order When Built-	For building. Aggressivity of building units: 0 = passive; 1	unitstandorders = 2;

	= défensive; 2 = offensive.	
Building Slope Slack-	Max slope where a building can be build.	maxslope = 15;
Footprint X-	Size of your unit (width)	footprintx = 2;
Footprint Z-	Size of you unit (length)	footprintz = 2;
Misc Unit Variable #3		
Transport Unit Capacity-	Capacity to load other unit. (number of unit)	transportcapacity = 15;
Transport Distance-	Max range to unload unit (in pixel)	transportdistance = 300;
Transport Max Unit Size-	Max size (width) of a unit loaded by your unit	transportsize = 9;
Transport Capacity-	Capacity to load other unit. (in size of unit)	transportsizecapacity = 40;
Work Speed-	Work speed	workertime = 10;
Water Line-	Water line for a ship	waterline = 8;
Attack Run Lengh-	!!!!	
Hover Attack Altitude-	!!!!	hoverattackaltitude = 10;
Hover Attack Distance-	!!!!	hoverattackdistance = 15;
Movement Class-	Type of mouvement of the unit, look in other *.FBI for example.	movementclass = GROUND2;
Misc Unit Variable #4		
Total Unit Allowed-	Quantity of this unit allowed (for example, 1 for sacred dragon)	totalallowed = 1;
Yard Map-	<p>Inside of a building and how can other unit walk on it (for example, unit can walk on the front of a factory, where they are build or unit can walk trough a door when it is open)</p> <p>o: unit can't walk on it, never.</p> <p>c: unit can walk on it if the unit is actived (like the door: unit can walk on it (in fact trough it) when the doo is open)</p> <p>. (point): unit can always walk on it.</p> <p>The first character is the left bottom place and the lastest is the right top place of the batiment</p>	<pre>yardmap = 000000 000000 000000 000000 000000 000000 000000 ccccccc cccccccc ccccccc cccccccc;</pre>
Copyright-	allway Copyright 1999 Humongous Entertainment. All rights reserved	copyright = Copyright 1999 Humongous Entertainment. All

		rights reserved.;
Unknow Variable #1		
Soundcategory	Name of the *.TDF file where the sound informations are stored.	soundcategory = arafly ;
SoundClass-	Name of the *.TDF file where the sound informations are stored.	soundclass = arafly ;
Bmcode-	1 for units, 0 for buildings	bmcode = 1 ;
Maxwaterdepht-	Only fo TA	
Notargetcategory-	The (type of) target that your unitr can NOT attack	
Tedclass-	Type of unit (side)	tedclass = Aramon ;
Wpri_badtargetcategory-	The (type of) target that your unitr has difficult to attack (Wpri_ pour for the first weapon, Wsec_ for the second et Wter_ for the third)	wsec_badtargetcategory = VTOL ;
Unknow Variable #2		
Admultiplier-	!!!!	admultiplier = 2 ;
Animatetype-	!!!!	animatetype = 0 ;
Bankscale-	!!!!	bankscale = 1 ;
Corpseadjustx-	!!!!	corpseadjustx = 2 ;
Economybonus-	!!!!	economybonus = 5 ;
Fireatwillrandom	!!!!	fireatwillrandom = 1 ;
Moveratel-	!!!! Used by flying units	moveratel = 1 ;
Moverate2-	!!!! Used by flying units	moverate2 = 9 ;
Pitchscale-	!!!!	
Unknow Variable #3		
Transmaxunits-	Like transportcapacity	transmaxunits = 18 ;
Standingmoveorder-	Only for TA	standingmoveorder = 2 ;
Wind-	!!!!	wind = 100 ;
Canbuild	Like builder	canbuild = 1 ;
Commander	Only for TA	commander = 1 ;
Bonus		
Used for "magic" effects of your unit (healing, increasing power, increasing armor,...) (for example, the sacred fire Zhon)		
Adjust Armor	Influence resistance of the unit (protect your unit)	[AdjustArmor] { Variableci-dessous }
Adjust Joy (life)	Influence life of the unit (heal your unit)	[AdjustJoy] { Variableci-dessous }
Adjust Attack	Influence attack power of your unit	[AdjustAttack] {

		Variableci-dessous }
Intensity of adjustment	Intensity of adjustment: 1 = none, 2 = doubled, 0.5 = reduce by 2	adjustment=45;
Edge effectiveness	How decrease the effect of the unit with distance (1 = none)	edgeeffectiveness=1;
Radius	Influence zone (radius in pixel)	radius=90;
affects enemy	Does the unit affect enemi?	affectsenemy=1;
Weapons		
Describe below, in part IV	Describe below, in part IV	

II: Manually editing the .FBI

1: Open the .fbi file with notepad, and type in the following :

```
[UNITINFO]
{
VARIABLE1=XXX;
VARIABLE2=XXX;
...
}
```

2: Replace "VARIABLE1=XXX; VARIABLE2=XXX; ..." by the needed values, you can find them all in the above tables. Just take the needed variables, and leave the others. To ease your task, you can take an existing .FBI file and tweak it as needed.

3: Save the file under the name shortname.fbi in the units folder of your unit. Click [here](#) to show the [arabowa.fbi](#) file

4: Now you have to edit the weapons of this file, as explained in the next section.

III: The weapons

You can edit the weapons from TDF Edit, but I think it is easier to do it with notepad, as the window opened by TDF Edit is a bit small, and you can't see the text in it. I did not take time to list all the possible variables (it will be done in the next version of this tutorial), so you would better take an existing FBI file and edit it.

1: Open the FBI file, and note [WEAPON1] for the first weapon, [WEAPON2] for the second one, and [WEAPON3] for the third one.

```
[WEAPON1]
{
VARIABLEWEAPON1=YYY;
VARIABLEWEAPON1=YYY;
```

```

...
[DAMAGE]
{
DOMMAGEDEL'ARME=ZZZ;
}
}

```

2: Replace "`VARIABLEWEAPON1=YYY; VARIABLEWEAPON1=YYY; ...`" by the variables you need for that weapon. Copy it from an existing weapon, you will avoid typos and the like.

Variable de l'arme	Description
<code>aimtolerance = 1024;</code>	!!!!!!
<code>areaofeffect = 31;</code>	Area of effect (in pixels)
<code>builduptime = 0.5;</code>	!!!!!!
<code>buttonimagedisabled = ArrowParalyzePU;</code>	Picture of the desactied weapon (no enough mana,...) if the unit have more than one weapon  To know the aviabile pictures, open "data.hpi" with HPI View, the picures are in the folder "anim -> weaponpic" in *.JPG format
<code>buttonimagedown = ArrowParalyzePU;</code>	Picture of the weapon if the unit have more than one weapon: when the button is pushed 
<code>buttonimageselected = ArrowParalyzeSBh;</code>	Picture of the selected weapon if the unit have more than one weapon 
<code>buttonimageup = ArrowParalyzeSB;</code>	Picture of the weapon (by default) if the unit have more than one weapon 
<code>damagetype = paralyzer;</code>	Type of damage inflicted by the weapon (look in *.FBI)
<code>decaytime = 3;</code>	!!!!!!
<code>dontleadtargets = 1;</code>	!!!!!! the weapon do not follow the target (to avoid turnig arrow)
<code>duration = 9;</code>	Time of effect of the weapon
<code>edgeeffectiveness = 0.7;</code>	!!!!!!
<code>emittime = 45;</code>	Time of shoot of the weapon
<code>explosionclass = green_shockring;</code>	!!!!!!
<code>firestarter = 1;</code>	Can the weapon trigger of a fire(1 yes, 0 no)
<code>gravityadjustment = 4.25;</code>	Influence of gravity on the weapon (for catapult)
<code>hweffect = fire;</code>	!!!!!!
<code>innercolor = 255 255 255;</code>	Inner color of flash weapon
<code>middlecolor = 200 230 255;</code>	Middle color of flash weapon
<code>outercolor = 180 200 255;</code>	Outer color of flash weapon

lightmap = <code>small</code> ;	How the ground is lighting by a shoot of this weapon
lobpreferred = <code>1</code> ;	!!!!!!
manapershot = <code>500</code> ;	Mana used by one shoot
maxvariation = <code>5</code> ;	!!!!!!
minrange = <code>500</code> ;	Min range for a shoot
nimbus = <code>1</code> ;	!!!!!!
noairweapon = <code>1</code> ;	This weapon can NOT fire on air units (<code>1</code>)
model = <code>Araarrow</code> ;	3D Model of the projectile (3do file)
name = <code>Bow and Arrows</code> ;	Name of the weapon
particlespersecond = <code>5</code> ;	!!!!!!
radiusart0 = <code>ring_fx_redA</code> ;	!!!!!! Used by earthquake weapon
radiusart1 = <code>ring_fx_redB</code> ;	!!!!!! Used by earthquake weapon
radiusart2 = <code>ring_fx_redC</code> ;	!!!!!! Used by earthquake weapon
range = <code>450</code> ;	Max range of a shoot
reloadtime = <code>3</code> ;	Reload time
ringcount = <code>3</code> ;	!!!!!! Used by earthquake weapon
ringdelay = <code>0.45</code> ;	!!!!!! Used by earthquake weapon
ringduration = <code>1.2</code> ;	!!!!!! Used by earthquake weapon
shadowart = <code>weaponshad01</code> ;	Show below in "shadowgaf"
shadowgaf = <code>shadows</code> ;	*.GAF file which contains the animation of the shadow of the weapon. The name of this animation is in "weaponart"
shakeduration = <code>2</code> ;	Used by earthquake weapon, duration of the earthquake
shakemagnitude = <code>3</code> ;	Pwer of the earthquake
showeffect = <code>0</code> ;	!!!!!!
soundhitclass = <code>arrow</code> ;	Sound
soundhit = <code>arrow08</code> ;	Sound when the weapon hit the target
spinheading = <code>150</code> ;	!!!!!! Center of rotation (show in "spinpitch")
spinpitch = <code>-3000</code> ;	Speed of spinnig of the projectile of the weapon (for spinnig projectile, like the weapon of the aramon builder). Can be negative for a rotation in the other direction
spritecount = <code>35</code> ;	!!!!!!
subtype = <code>Earthquake</code> ;	Subtype of the weapon
turnrate = <code>180</code> ;	!!!!!!
type = <code>Ballistic</code> ;	Type of the weapon
unitonly = <code>1</code> ;	Weapon affects just unit and not building
variationtime = <code>2</code> ;	!!!!!!
veteranlevel = <code>10</code> ;	!!!!!!
veteranmodel = <code>araham10</code> ;	3D Model of the unit when the unit is a veteran
wanderendart = <code>flamevortexend</code> ;	!!!!!!
wanderloopart =	!!!!!!

<code>flamevortexloop;</code>	
<code>wanderstartart = flamevortexstart;</code>	!!!!!!
<code>waterexplosionclass = small water explosion;</code>	type of explosion when the weapon hit the water
<code>weaponart = cannbmed;</code>	Show below in "weaponart"
<code>weaponart = projectiles;</code>	*.GAF file which contains the animation when the weapon hit the ground. The name of this animation is in "weaponart"
<code>weaponvelocity = 750;</code>	Velocity of the projectile

3: You must now fix the amount of damage this weapon inflicts. Replace "`DOMMAGEDEL'ARME=ZZZ;`" by "`default = ZZZ;`" to give the base power of the weapon. Then by using "`nomcourt=ZZZ`", you can adjust this damage as a function of the attacked unit.

4: Save your new .FBI file, that's all ! To show ARABOWA.fbi, click [here](#) ([arabowa.fbi](#))

MISC

Description

Here you can add a file like a readme but this file will not be readed by normal TA:K players because this file will be compressed with the unit.

Action

1: Click on "Open" to add a file and on "Remove" to remove it.

COMPILATION OF THE UNIT

Now, you must compil your unit into a *.ufo file. Do not forget to change the TA:Kingdoms shortcut and add "`-disablecavedogverification`".

Compilation with TDF Edit

Just click on "Compress" in TDF Edit. Your unit will be compressed in the folder where you saved it. Now put this file "shortname.ufo" in your TA:Kingdoms folder.

Do not forget to check error by clicking on "Options -> Check Errors".

Compilation with HPI Pack

1: Open HPI Pack

2: Select the folder where you can find the "anims", "canbuild",... folders in "Directory to pack" box and give a name (the short name of your unit for example) with the extension *.ufo. in "Destination HPI File" box

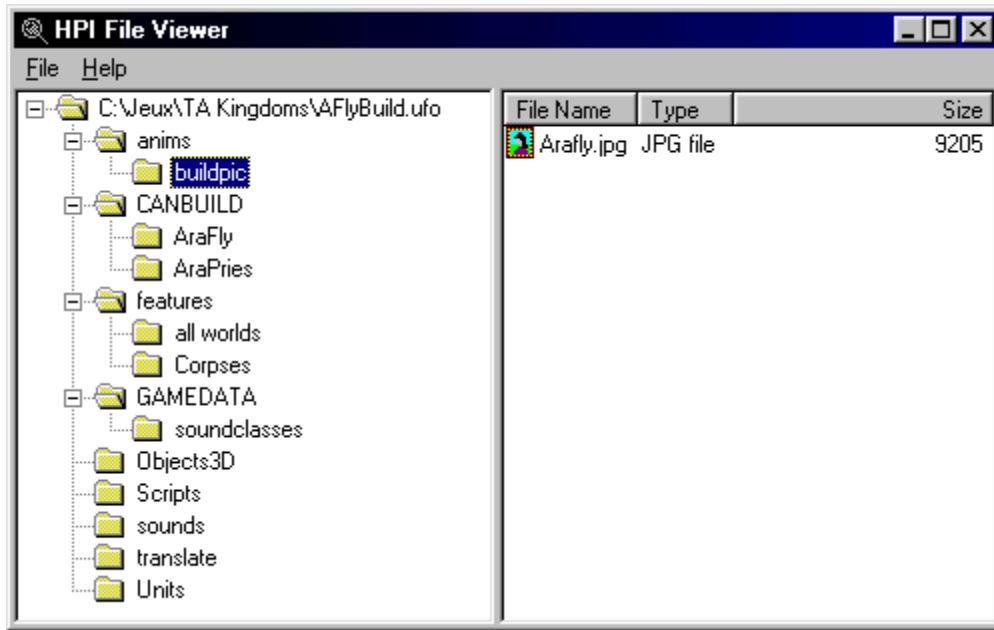
Selectionnez le dossier ou se trouve tous vos dossier "anims", "canbuild",... dans "Directory to Pack" et le fichier *.ufo (n'oubliez pas de noter l'extention .ufo) où vous voulez compresser l'unité. Il n'est pas obligatoir de donner le nom court de votre unité au fichier *.ufo.

3: Select "TA:K" dans "Program" and click on "Pack"

INSTALLATION AND CONFIGURATION OF THE PROGRAMS

I: HPI View by [JoeD](#)

This excellent piece of software can uncompress .HPI, .UFO, .CCX and .KMP files. These files contain all the infos TA:K needs to run. You need 1.9 version to be compatible with TA:Kingdoms.

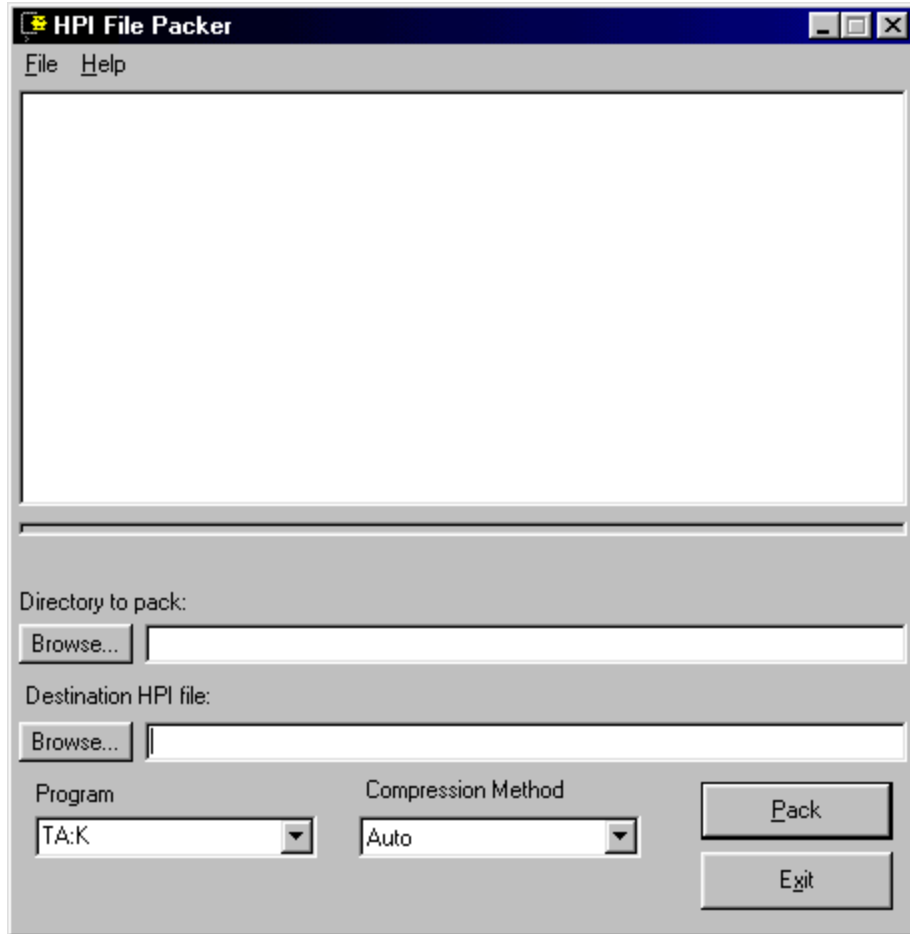


Installation:

Uncompress the .zip file in a folder, and create the shortcuts you need.

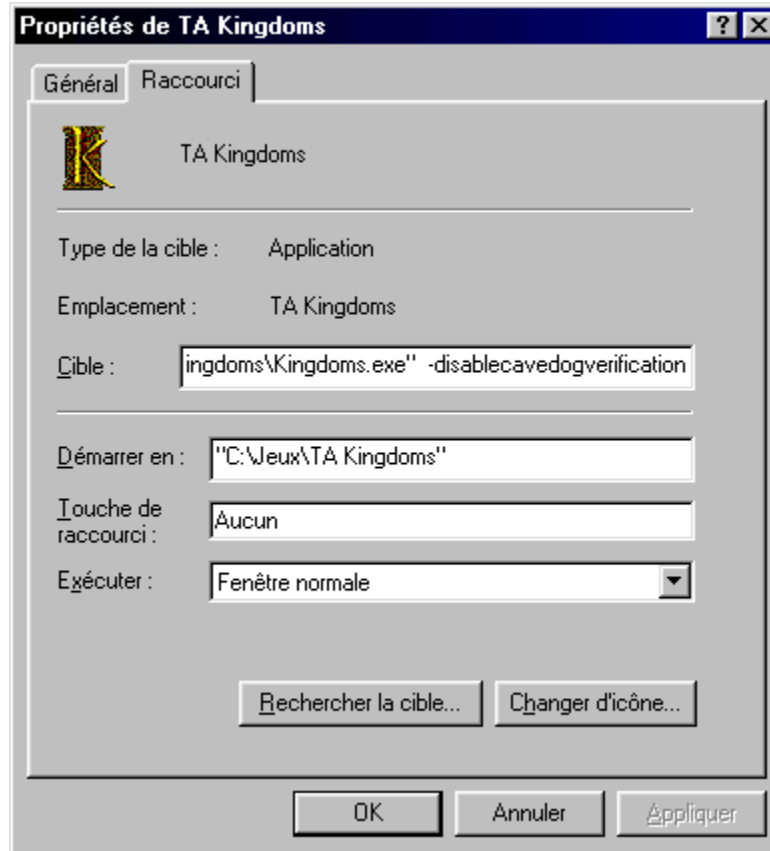
II: HPI Pack by [JoeD](#)

This program can build .HPI, .UFO, .CCX and .KMP files. It is the little brother of HPI View, that does the opposite. You need 1.7 version for TA:K.



Installation:

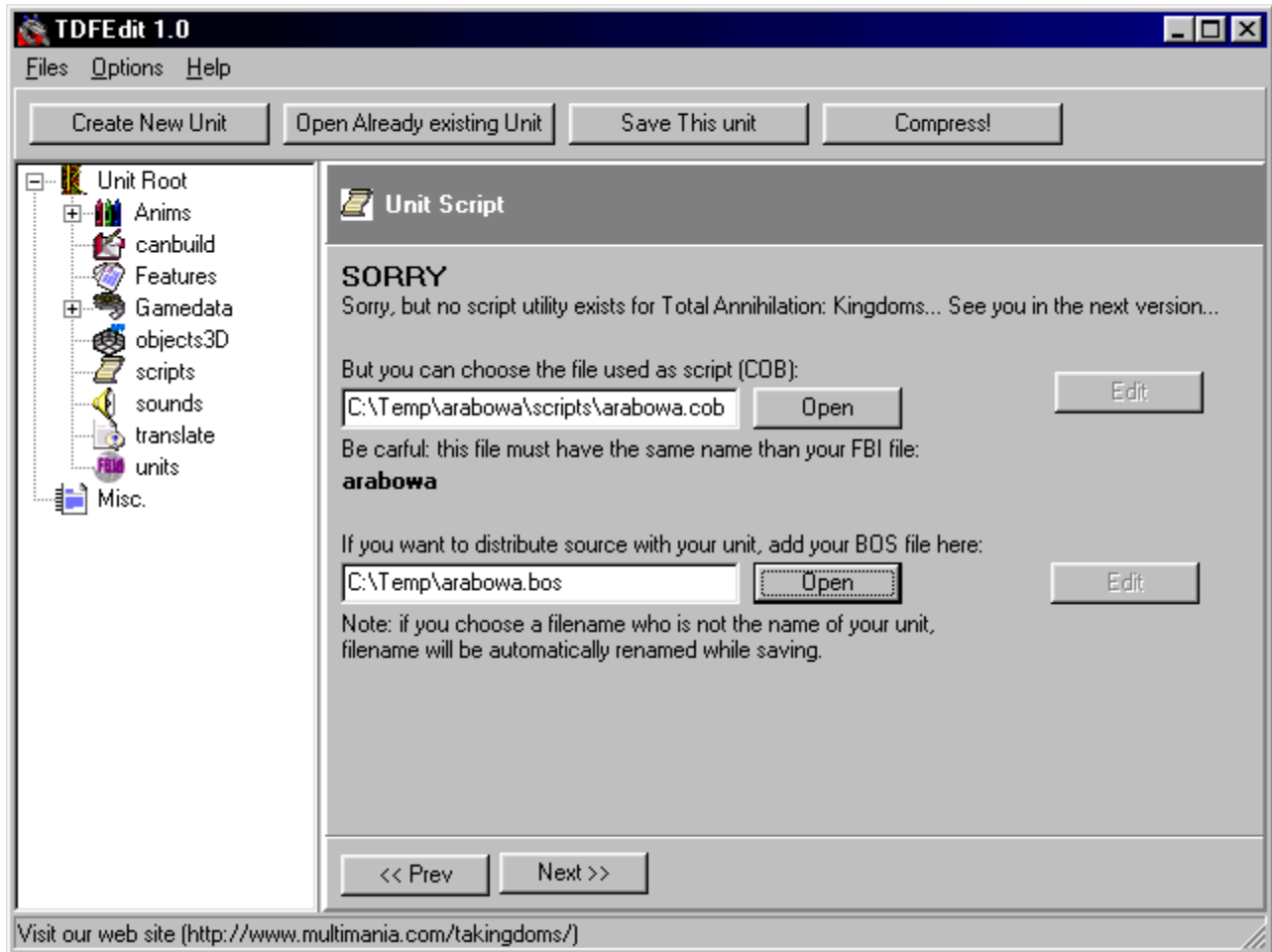
As simple as HPI View. You need to uncompress the .zip in a folder. To make units compatible with TA:K, you just have to add "[-disablecavedogverification](#)" after kingdoms.exe on the command line of your shortcut.



The modified shorcut

III: TDF Edit by [TA:Kingdoms_Fr](#)

This software will help you in all the creation process of your unit.



Installation

Uncompress the .zip in a folder, respecting the hierarchy of the folders / files. You need then to create two folders : one for the sounds, and one for the images.

Help:

To have this tutorial when you click on "Help -> Tutorial" ou CTRL F1,copy all this tutorial (don't change the files and folders names) in the subfolder "Tutorial" of the folder where you have installed TDF Edit.

Sounds :

- create a folder on the desktop named "taksounds"
- extract all the files from english.hpi (english version) or french.hpi (french version) or spanish.hpi, german.hpi, italian.hpi (guess which one is for the italian version ?) in the taksounds folder you just created. You can use HPI View to do this.
- In this folder, delete all the subdirectories, except the "sounds" one.
- Launch HPI Pack, select your "taksounds" folder in "Directory to pack", and save that in your TDF Edit folder, under the name "sounds.hpi" (Destination HPI file). Select TA in "Programs", and click "pack".

Images :

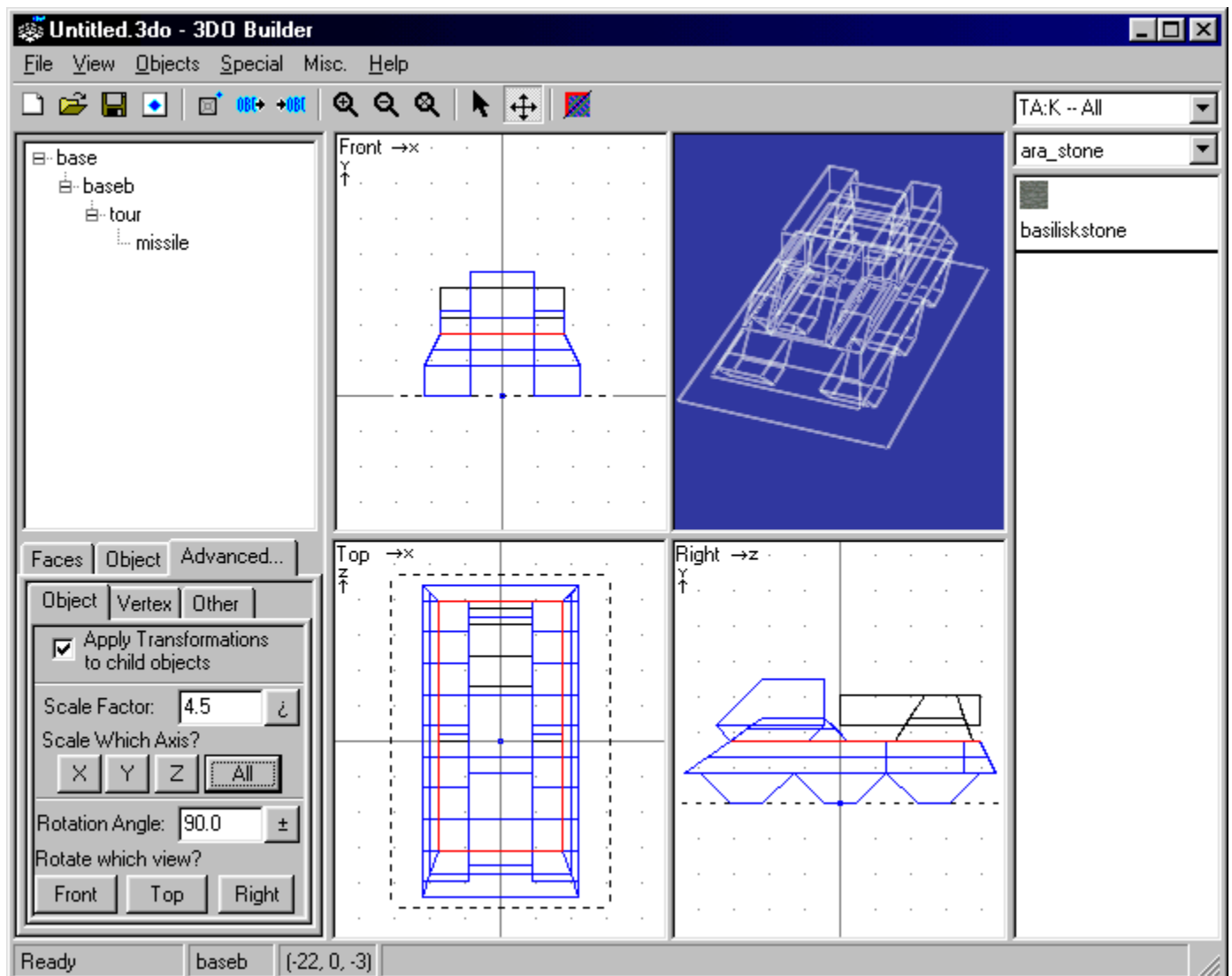
- create a folder on the desktop, called "takpics"
- extract all the files from "data.hpi" (on your TA:K CD if you chose

minimal installation), with HPI View

- delete all the directories from the "talpics" folder, apart from "anim"
- in this subfolder, delete all the subfolders, apart from "buildpic"
- launch HPI Pack, and pack the takpics folder in a file called "buildpic.hpi" in your TDF Edit folder.

IV: 3Do Builder + by [Quantum](#)

This program is used to create units from .DXF or .LWO files. You need version 2.0 to work with TA:K. You also need a 3D modeler software that can save in .DXF ou .LWO formats.



Installation:

- Extract all the files from the .zip, and launch 3dobuilder+.exe to configure the program
- you should have created TA:Kingdoms textures, as explained in the

[objects3d](#) section of this tutorial

- If you have the TA UnitViewer, you can use it here, but it will not work with TA:K units. Click on File->Export yo Unit Viewer to choose the right Unit Viewer, compatible with TA:K units.
- Click on "View -> Set custom TA Location" and select the folder where TA resides (if you have it), and click on "View -> Set custom TA:K Location" to select the folder where TA:K is.
- if ou click on "View -> Display options", you can choose the colors of the layout, and also diable the 3D View. You should be aware that this view is repsonsible for 80% of this soft crashes.

To enable the 3D view, you need to have OpenGL installed on your PC. It is available from [Quantum](#)'s site.

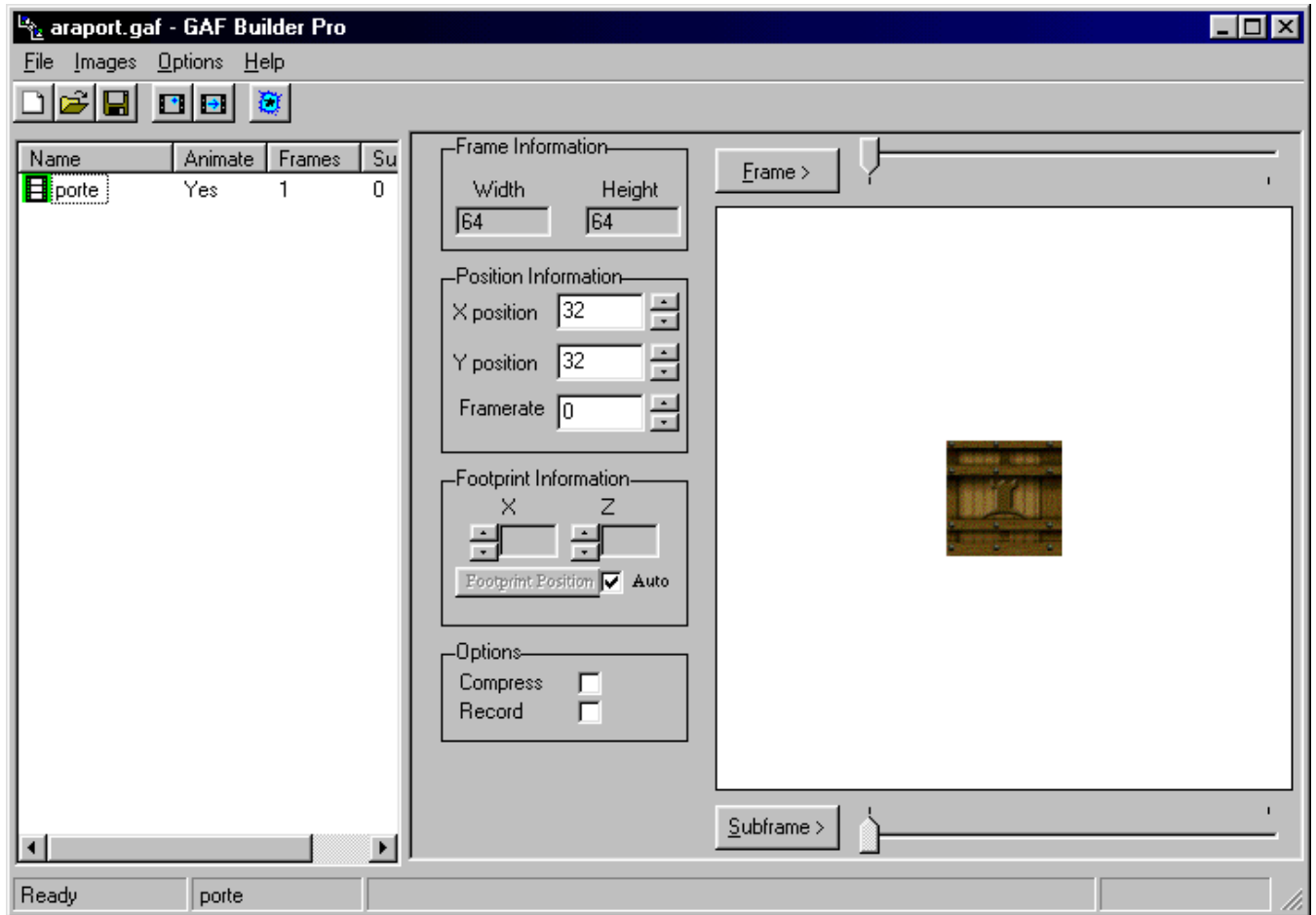
V: Visual Soap

You need it if you want to use *.dxf file in 3Do Builder +



VI: Gaf Builder Pro par C_A_P

Use it to create custom textures for your unit and custom features for your maps.



Installation:

Uncompress the .zip file in a folder, and create the shortcuts you need.

3DO BUILDER +

Some functions are described in the [objects3d](#) section of this tutorial.

I: L'INTERFACE

1: Icon menu

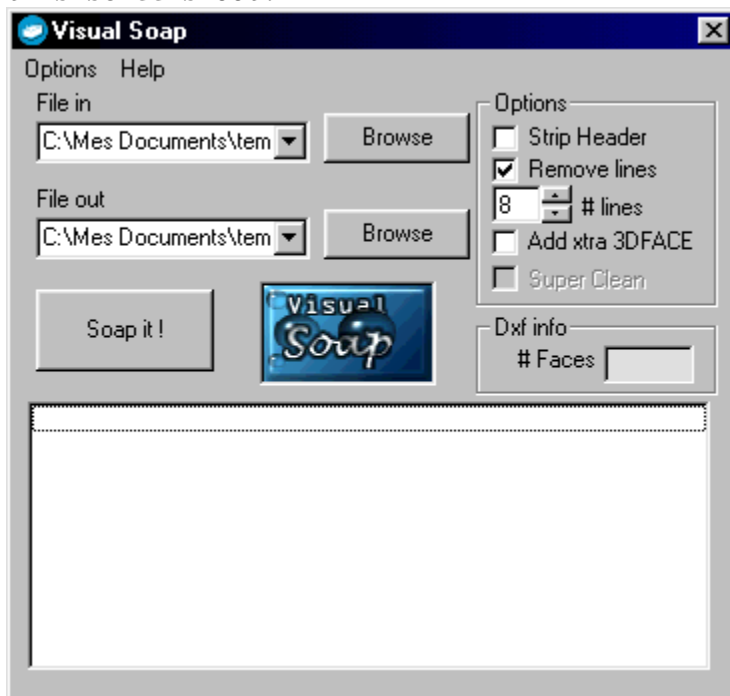


- 1: Create a new 3D model
- 2: Open an existing 3D model
- 3: Save a 3D model

- 4: Create an image of a 3D model from the Unit Viewer. Use it for TA units only, as this Unit Viewer is not compatible with TA:K Units.
- 5: Create a new piece of the object.
- 6: Import a piece. Supported formats of the import module :
 - *.DXF: files created by most of the 3D modeler
 - *.LWO: "lightwave" files, generated by 3D modeler
 - *.OBJ: "object" files, generated by 3D modeler
 - *.3de: file created by 3DO Builder. It is a textured piece.
 - *.3do: file created by 3DO Builder. It is a textured piece, taken out of its context.
 - *.3dt: another 3DO builder file. Holds a piece, with all its subpieces.
- 7: Export a piece. Supported formats are : *.DXF, *.3de, *.3dt.
- 8: Zoom in (the four views are affected)
- 9: Zoom out (the four views are affected)
- 10: Zoom 100%
- 11: Default cursor
- 12: Cursor that enables you to move a piece in the four views.
- 13: Draw a red border around the selected face in the 3D textured view.

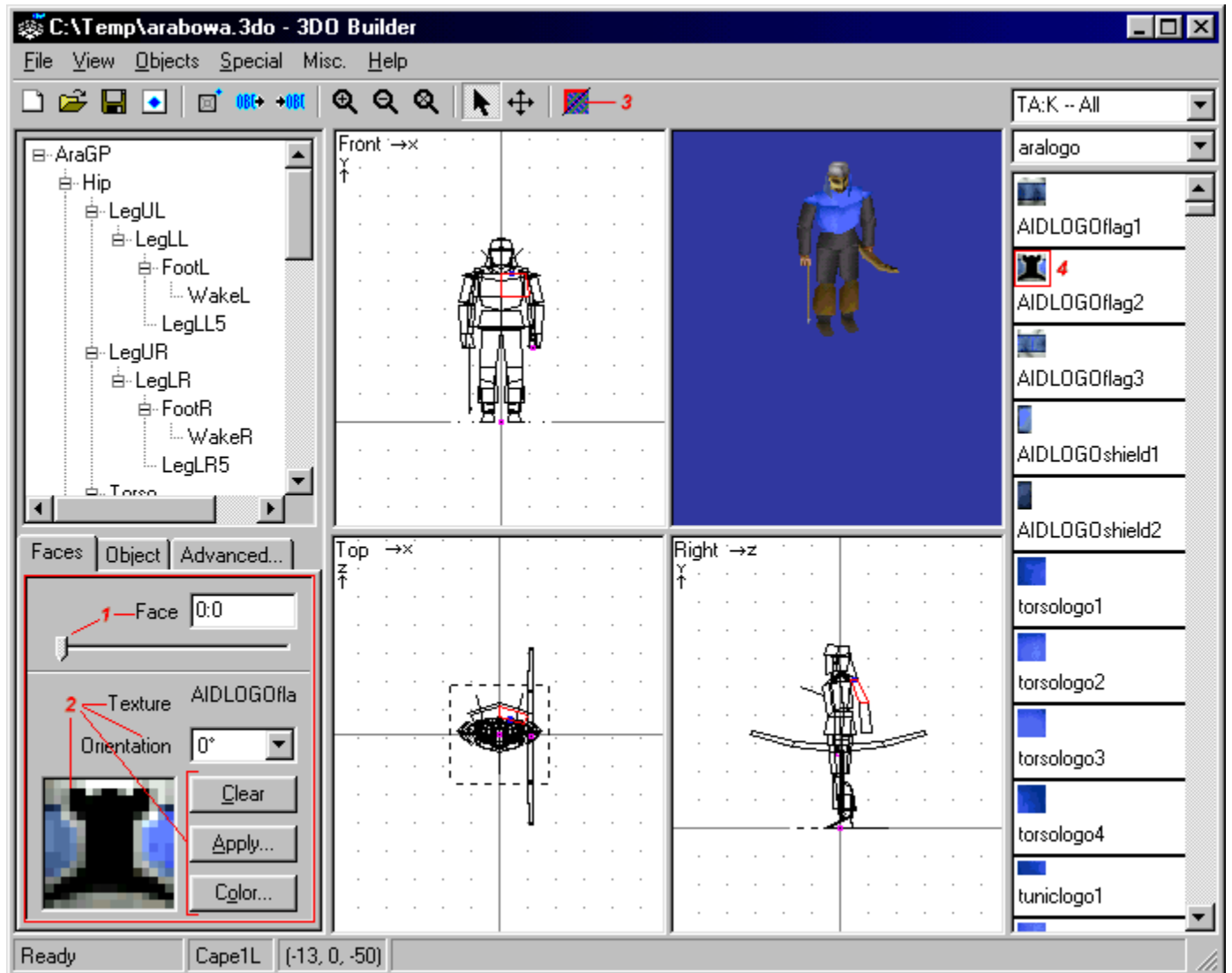
Note about *.DXF files

To make your unit, you can use *.DXF files too. But you must first "soap" this files before. To do this, lunch Visual Soap and configure it like this screenshots.



"File in" is your *.DXF file and "File out" is the modified *.DXF file. Click on "Soap it !" to modify your file and adapt it to 3Do builder.

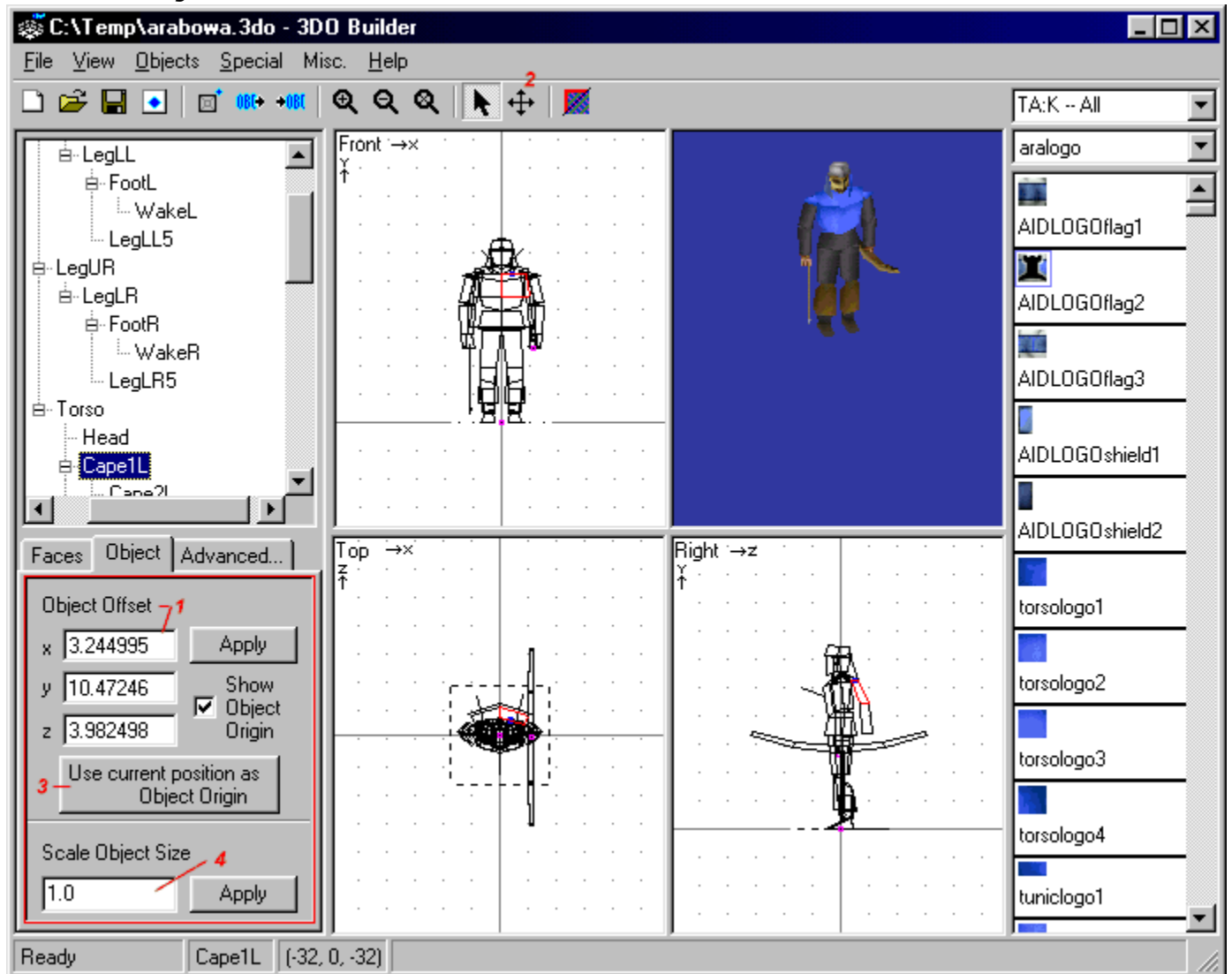
2: Table "Face"



1 : Face selector. A face is selected on the selected part in the pieces tree just above. The face is in red on the three plane views, and also on the textured view if the option is selected (3).

2 : Possible modifications for this face : - Mapping of a texture by double click on the right menu (texture chooser) (4). The selected texture is shown under "Orientation", and its name is shown near "Texture" (2) - Change the orientation of the texture with "Orientation". You can turn in of 90°, 180°, 270°. - By clicking on "Clear", one can delete the texture from the face. - By clicking on Apply, one can map the texture on more than one face, or on the complete piece. ("Apply to the entire object") - By clicking on Color, you can apply a simple color on a face.

3: Table "Object"

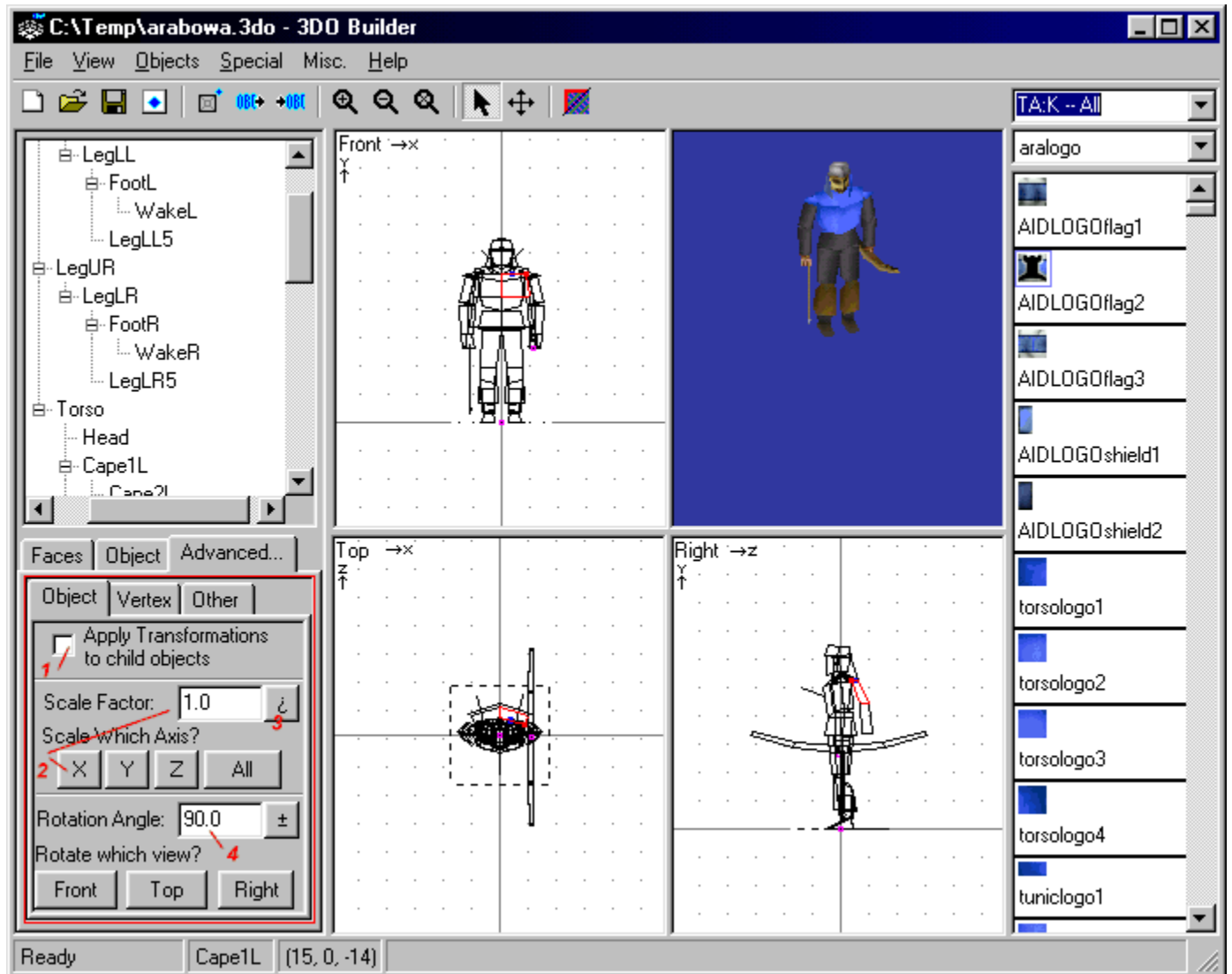


1: Enables you to move the selected part more precisely than using the cursor (2). Click on Apply to validate your modifications.

3: Enables you to move the rotation axis on the origin of the axes. You can use that if you misplaced while creating your .LWO file. The rotation axis is shown with a little blue square on the three plane views.

4: Is used to zoom your part. A more powerful function will be explained in the Table "Advanced". It acts as a multiplier on the size of the part. Thus a figure smaller than one reduces it, and bigger than one enlarges it. Click Apply to record the modification.

4: Table Advanced



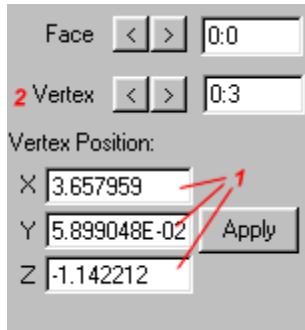
A: Object



1: Check this box if you want to apply the transformations you made on this piece to all its subpieces. Useful if you want to move an arm composed of three subpieces.

2: Is used to change the size of the piece. It is more powerful than the object zoom in / out, as you can apply it axis by axis, by selecting "X", "Y", "Z" or "All". You can also apply this modification to all the subpieces of that piece. The button marked with an inversed ? (3) undoes the modifications.

4: Is used to rotate a piece as well as all its subpieces. To select which rotation you want to apply, you have to click on the "Front", "Top" or "Right" view.

B: Vertex



On this table you can move vertices of the faces, to go from  to  for instance. When you click on this table, a small red square appears on the views, marking the vertex that is to be moved. Set the new position of the vertex in the box (1) and "apply" it. You can change of vertex by using the arrows (2) next "Vertex".

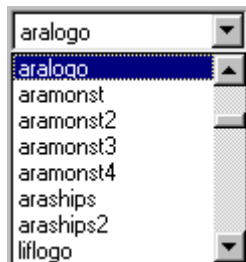
C: Other

This table is empty, I dont know what it is doing here...

5: Menus



This menu enables you to choose which type of texture you want to use. If TA is not in the list, either you dont have TA (you should !), or you dont have [configured](#) 3DO Builder properly. Check it [here](#). If TA:K is not in the list, either you dont have TA:K (...) or 3DO Builder is not configured properly, or that your texture pack is not correct. To build it, go here : "[Objects3d](#)". To make TA:K units, I advise you to choose TA:K -- All.



This menu is a chooser between different types of textures.

II: THE MENUS

1: File



As you can see it, all the classical operations are available from this menu.

"Export to Unit Viewer..." makes possible to preview a unit in Cavedog Unit Viewer. Not compatible with TA:K...

"Export to Unit viewer (custom)..." allows you to test your TA unit scripts. Not TA:K compatible.

"Import Textures..." to import your own textures. Does not work with TA:K.

2: View



"Grid" : to enable the grid in the three plan views

"Grid Size..." : to change the grid size

"View" : to choose the X, Y or Z view

"OpenGL view" : to choose what kind of 3D view you want to see, textured or not

"Startup Option..." : to tell the program where are your TA / TA:K folders.

"Display Option..." : to customize your interface

"Keep textures ratio in Face box" : uncheck this to make the program faster.

3: Objects

Objects	Special	Misc.	Help
I	m	p	o
Import Object...		Ctrl+I	
E	x	p	o
Export Object...		Ctrl+E	
E	x	a	l
Export All Objects...			
C	r	e	a
Create Object...		Ctrl+C	
R	e	m	o
Remove Object		Maj+Del	
I	m	p	o
Import Scale Value...			

Is used to create a new piece in the pieces tree ("Create Object") or to delete one ("Remove Object"). You can import a piece ("Import object..."), export one or all of them ("Export Object..." and "Export All Objects..."). All these functions are available from the icon menu. "Import Scale Value..." : to fix a zoom value when you import a piece.

4: Special

Special	Misc.	Help
I	n	v
Inverse Face		F2
I	n	v
Inverse Object Faces		F3
I	n	v
Inverse All Faces		F4
M	e	r
Merge Duplicate Vertices...		
C	l	e
Clean Up Model...		
A	d	d
Add Unit Ground Plate		
S	e	t
Set Face as Unit Ground Plate	Ctrl+G	
C	h	a
Change Ground Plate Size...		
S	c	a
Scale All Objects...		
C	r	e
Create Unit Picture...		

"Inverse Face" : to inverse the sense of a face, if the texture appears on the wrong side.

"Inverse Object Faces" : inverses all the faces of a part.

"Inverse All Faces" : inverses all the faces of a 3D object.

"Merge Duplicate Vertices..." and "Clean Up Model" : are used to delete vertices to reduce the number if the faces. Be careful when using this, you are likely to get a strange result !

"Add Unit Ground Plate" : adds the green square around the foot of a unit (TA), or the green glowing circle (TA:K). The value is given in pixels, you need a 16x16 square for a 1x1 unit.

"Set face as Unit Ground Plate" : to use the selected face as the unit ground plate

"Change Ground Plate Size..." : to change its size...

"Scale all Objects..." : change the size of the whole object

"Create Unit Picture..." : use this in TA only. The picture is used in the build menus to represent the unit.

5: Misc.

Misc.	Help
Select Next Face	F12
Select Previous Face	F11
Copy Piece	Ctrl+Ins
Copy Branch	
Paste Piece/Branch	Maj+Ins

"Select next Face", "Select Previous Face" : to select the next / previous face of the part.

"Copy Piece" : more simple than to export / import a part. Textures are also copied.

"Copy Branch" : same as above. Alle the subpieces are also copied.

"Paste Piece/Branch" : to paste what you have copied.

SHORT NAME LIST

Each unit has a long name (Flying Builder) and a short name, composed of 8 letters, understood by the TA:K engine. One of the first things you want to do is to choose a short name for your unit. The first three letters are related to its camp : ARA, TAR, VER or ZON.

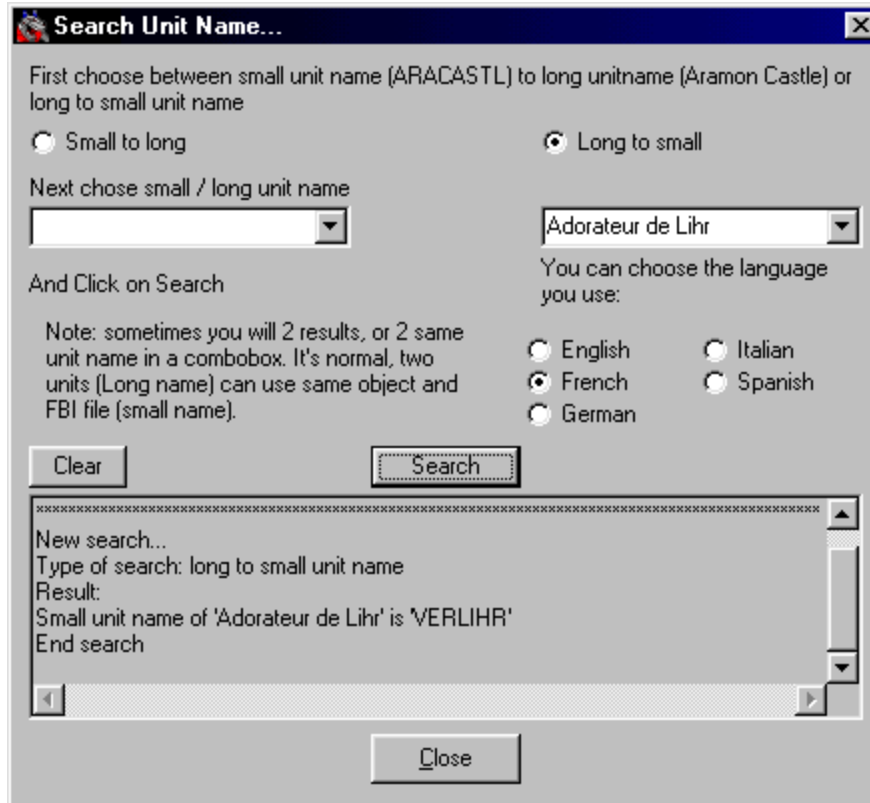
TDF Edit has a function that tells you the short name of any unit, giving its long name, and its long name giving its short name.

Looking for a name with TDF Edit

1 : Click "Options -> Search Unit name..."

2 : Type in a long or a short name then "Small to long", or a long name then "Long to small". Choose your language.

3 : Click search, the result is written in the text window.



As an example, I looked for 1'"Adorateur de Lihhr" in the french version, and I learnt that it was VERLIHR.

Creating Build Pictures

From Acolytes Shrine(<http://www.takmcc.com/acolytes/>)



- Open your **units.3do**.
- Find the angle you want. Make sure it's no bigger than 64*48. I know, it's small but you have to live with it :)
- Press **ALT + PRINTSCREEN** (on your keyboard)
- Open MS Paint
- Paste the screenshot. (CTRL-V) (you can directly pasted into Photoshop but it's more comfortable for me :)

- Using the outline tool, select the area around your unit image, and copy it (CTRL-C)
- Open Photoshop
- Create a new file; goto **File + New.**
- Make sure it's 64*48 pixels and in RBG color.



- Paste your from *Step 2* (it should in new layer).
- Select the "Magic Wand" tool



- Using your "wand", select all the "blue" (or the background color and press **Delete.** Increasing your magnification will help

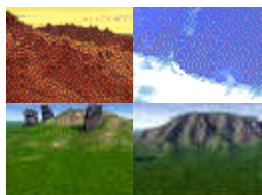
- From the **Layer** box, right-click on layer your units pic is on and select **Blending Options.**



- A new window should a appear
- Click (and select) **Outer Glow**
- Select the settings and experiment to get the effect you want.



You should have should like the bp to your left.



Now, to added a background. First find one (Or select one here).. Do so now, I can wait



- Got your pic? Good.
- **Right Click** on your picture and select **Copy**
- Goto to back to Photoshop.
- Paste the picture (**CTRL + V**) (make sure it's a layer behind your units picture)
- And now you got a background. And your finished at this point



- Now saving. The easiest part.
- **File** and **Save As...**
- Make sure you save in JPG. And since TA:K will take any (file) size, and color, make sure you set quality to **Maximum (12)** to get the best quality :)

Customizing TA:K to Allow Third Party Races without Iron Plague

By Delozier

All Race Custom Installation

The predominant problem I have been receiving feedback about lately is installation of third party races when the Iron Plague is not available. If you really love playing Kingdoms, I recommend that you purchase the Iron Plague..... even if you have to shop the www to find it. I know for a fact that the expansion can be purchased online through vendors such as Amazon.com, and Chips and Bits.....

So what I did was re-install Kingdoms without the Iron Plague and started trying to get the races to work there myself. I ended up creating some kinda large files to download (25mb total) and I am hopeful that this custom installation will alleviate enough race installation problems to be worth the file sizes. I have NOT included the Creon race, or any of the

Iron Plague map files. I have left the sidedata files separate for those of you who wish to modify it to include only certain races. The NoIP version included with the NoIPfiles does NOT support the Creon race. The files contain:

1. Races contains the versions of the 3rd party races currently installed in my original Kingdoms folder. These have been edited for sidedata and ai conflicts. An unfinished AI (not all races are included) that can be updated if all else proves to work as planned. Also, a copy of the Creon supporting allsidedata is included.
2. NoIPfiles contains the Ip files needed to load the 3rd party races. It also contains the No IP sidedata file. It does NOT contain the Creon race.
3. Supporting files contains the extra files needed to perform the installation below. (version upgrade files, and Cavedog unit files)

Download those files to your hard drive and unzip them. DO NOT unzip them into your Kingdoms directory! Instead, unzip them into another directory that you have created. This is what I have done in order to get to the third party races to work without the Iron Plague installed:

1. Performed a FULL INSTALLATION of Kingdoms from the CD.
2. Added the -disablecavedogverification to my shortcut.
3. Installed 1x-2bd.EXE, then 2x-2bd.EXE. These are the Cavedog version 1 to 2 upgrades contained in the supporting files download. I did NOT install version 3!
4. Installed the Cavedog units included in the supporting files download.
5. Added the races and ai folder included in the races download.
6. Added the new version of the AllRaceNoIP file included in the No IP files download.
7. Added the NoIPsidedata file included in the No IP files download.