



BUSH'S GEOTYPICAL MICROTERRAINS - A BEGINNERS GUIDE



TABLE OF CONTENTS

<i>Bush's Geotypical Microterrains - A Beginners Guide</i>	1
Introduction	3
Category 1 - Old Pros	4
Category 2 - Experienced Intermediates	4
Category 3 - Absolute Beginners	4
Section 1 - Initial Setup	4
1.1 - Required Downloads	5
1.2 - Optional Downloads	5
1.3 - Install the BIS Tools	5
1.4 - Unpack Game Assets with Arma2P	6
1.5 - Do Sgt Ace's Tutorial !	8
1.6 - "Tag" or "Namespace" folders & The little "Configs-Only" CA folder	8
1.7 - *Optional Advanced Step * - Buldozer & Visitor Advanced Configuring	9
Section 2 - The Source Files	12
2.1 - Names and Naming Conventions	12
2.2 - Installing The Source Files Package	12
2.3 - The "Data" Folder	13

The Menu Image	14
The "Middle MCO"	15
A Typical "Ground Texture Set"	15
The "Satout" Texture	17
2.4 - The "Source" Folder	17
The "Terrain" Subfolder	18
mapLegend.png	19
The Sat_lco.png	19
The Mask_lco.png	20
The "layers.cfg"	20
2.5 - * Optional Folder *- The "UI" Folder	22
2.6 - The "Main Project Directory Files"	23
KhargharValley.hpp - the "Names.hpp"	23
KhargharValley.pew - The "Main Project File"	23
cfgClutter.hpp	24
cfgSurfaces.hpp	26
The "Config.cpp"	28
Section 3 - Loading the Terrain Into Visitor	35
3.1 - Set the "Layers Folder" Path	36
3.2 - Import Satellite and Mask	36
3.3 - Run Bulldozer	37
3.4 - Export the 'Worldfile'	37
Section 4 - Binarization	38
4.1 - Setting up BinPbo	38
BinPBO - Main Window	38
BinPBO - 'Options' Window	39
4.2 Binarize!	40
'Advanced' Appendices	41
Credits & Thanks	41

INTRODUCTION

This guide is primarily designed to accompany the *Bush's Geotypical Microterrains* series of source files packages. As such, it's *not* really a complete, from-scratch, teach-you-everything-you'll-ever-need-to-know-about-terrainmaking tutorial. There *is* a "beginners" section which, after some initial install and setup instructions, directs you to *Sgt Ace's Tutorial* - thereafter, the rest of this guide is more like a *mechanics manual* - the simple mechanics of getting basic terrain structures working and in-game are discussed in some detail... the files you need, what they are, what they do and where to put them.

After an introductory setup section for beginners, the main part of this guide takes the form of a walkthrough of the *Afghan Valley - Ghor Province* example terrain source files. As we consider each folder and file, there's instructions on repathing and renaming each of the example files into your own "Namespace" or "Tag Folder", including renaming the terrain itself. By the end of the guide, the reader should have a complete and working renamed version of *Afghan Valley*... *your* terrain now, and *your* working structure! You can use that as a basis for your own project - adding your own enhancements to that basic framework, or use the whole folder structure as a project template.

Some of those enhancements will be illustrated by subsequent terrains in the series - with "advanced-level" appendices added to the end of the guide. Etah Plateau, for example, will include custom "terrainHit" sounds (footsteps to you and me), and appropriate discussion of the extra files and steps involved will be added to the guide. The *Faffindale* source files pack will contain its associated *WorldTools* project file, so there'll be some brief matching discussion of basic forest masks and importing vegetation, etc.

What's *not* discussed much, if at all, is any of the truly *creative* stuff, like sculpting *heightmaps*, or painting *Satellite* or *Mask* files. Nor will I be covering roads, or placing buildings and objects, etc. When you get to that creative stage there's a wide variety of options - real or artificial heightmaps? satellite or hand-painted satmaps? Regardless of how you choose to actually *make* these parts of your terrain project, the final result will depend on one common factor - getting them into Visitor in a coherent structure it expects, then getting that terrain through binarization and into the game. This guide will focus on that basic, practical procedure at a very simple "entry" level.

Since you're actually bothering to read any further I'm guessing you fall into one of these three main categories...

1. You've made terrains before and you just want a quick peek at the terrains in Visitor for interests sake. You know what all the files are and what they do and you're not intending to repath or adapt and, even if you are - you know what you're doing anyway.
2. You've got a proper & working P:\ drive Dev Setup, you've done a proper tutorial like Sgt Ace's - *successfully* - and now you want to mess around with something slightly more complex. Some discussion of the files might be useful, as well as instructions on how to repath and adapt stuff, since you might actually *use* some of it sometime.

3. You're an absolute beginner and you've never done a proper tutorial or managed to get a P:\ drive to work properly so far. Reasonably detailed basic setup instructions would be *very* welcome, plus a brief look at the relevant sections of [Sgt Ace's Tutorial](#), then maybe you'd feel a bit more confident about moving on to a look at the Source Terrains with the category 2 guys.

Decide now which best applies to you and then follow the instructions below for your chosen category...

CATEGORY 1 - OLD PROS

You guys don't really need instructions at all, do you?... well, since you're here anyway...

All the files are pathed to my usual "Bush" namespace, so - for a fast look in Visitor - make a "Bush" folder on P:\ - copy the source files in there. Load the .pew and re-establish the Layers files by re-importing the Sat & Mask files...

Obviously, you guys know all about the pitfalls of actually working on stuff which is carrying tags from an already-released project, so any actual serious editing of these terrains should be preceded by a full re-path & rename exercise of all the usual files.

The rest of this guide is pretty much all aimed at the category 2 & 3 guys, so... You're done!

CATEGORY 2 - EXPERIENCED INTERMEDIATES

No need for setup instructions? P:\ drive fully configured? Have a "namespace" or "tag" folder in place? Managed Sgt Ace's Tutorial without problems and got the terrain binarized and in-game? - Well done!!!

I'm assuming that you guys will be most likely to want to adapt these source files - maybe as basic structures to which you'll add your own landscape files, or maybe you're thinking of remodelling one of these existing heightmaps... whatever... The most important primary task is to repath and rename all the files - *All* the files...

Serious conflicts can occur if terrains use the same classnames, or even the same filenames sometimes... it's Bad Practice all round, so - since repathing and renaming has to be done anyway - we'll try to learn something from a good look at all the main files while we're at it... Since you don't need to slog through the initial setup procedures, **you can skip forward to "Section 2 - The Source Files" now...** The beginners have some preliminary work to do.

CATEGORY 3 - ABSOLUTE BEGINNERS

Okay... You guys have opted for the Full Deal... That starts directly below... Follow the steps carefully - one at a time... I'll send you off to do the relevant part of Sgt Ace's Tutorial at an appropriate point... However - the initial setup part of that otherwise excellent tutorial is now a little out of date... So - start below and, when the time comes - kick into Sgt Ace's Tut at the indicated chapter - then you can come back to this guide and move on to the more complicated stuff...

SECTION 1 - INITIAL SETUP

Before we start, you'll have to acquire some of the basic tools... The list of "required" downloads below are... well... *required*... so download them now... The "optional" downloads list is mainly for future reference, and is by no means a complete list... worry about them as and when you find you need them...

1.1 - REQUIRED DOWNLOADS

[BIS Editing Tools 2.5.1](#)

A required download - the latest version of the BIS Tools package - all the basic tools you need are in this.

[Mikero's Dos Tools](#)

Mikero's no-frills Dos Tools are a mainstay of the editing community... if there's a tricky, or repetitive or downright impossible job - there's probably a Mikero Tool which does it automatically while you go make coffee... The only package we want just now is "**Arma2P 2.5.1a.rar**" - download that file from the list - it contains everything you'll need.

** Mikero is constantly updating his tools... so, if you follow the link above and you see an Arma2P package with a **higher** version number - like "2.5.1b" or something - grab that instead! Always use the highest version number you see for the tool you're interested in. **

1.2 - OPTIONAL DOWNLOADS

[File Renamer Basic](#)

I first came across this deeply handy little freeware program ages ago... I've used it a lot for things like renaming MP3's and media files, etc. Since there's some bulk texture file-renaming to do later you *might* want to take a look at this excellent free tool... it's handy for all sorts of stuff.

[L3DT Standard Edition](#)

Later on, once you've successfully reconfigured some of these terrains, you might want to fool around with remodelling areas of the heightmap, or maybe even start making your own!... L3DT Standard Edition is by far the best free heightmap editor around... There's also a demo of the Pro version available... save your 90 days usage of that for when you're ready to start playing with Sat & Mask generation though... For basic heightmap editing the free Standard version will be fine...

[Shezan74's World Tools 1.8](#)

Shezan's *WorldTools* contains clever utilities for autoplacing objects - both Artificial and Natural. Since the *Forest Generator* section of WorldTools was used in a couple of these example terrains, there will be a short "advanced" section later in this guide which briefly discusses how Forest Generator was used to create the vegetation cover on those terrains. Those terrain Source File packs will also include the appropriate WorldTools project files and masks as appropriate).

1.3 - INSTALL THE BIS TOOLS

The first thing we need is somewhere for the BIS Tools to live... **You'll need either a separate partition of about 50Gb or a spare second drive.** I use a spare 500Gb drive installed as D:\... You can use any letter you like

- except "P" - that's reserved by default for the BIS "virtual development drive" which will be created as part of the Tools install process.

Once you have this set up - **create a folder called "BIS TOOLS" on the root directory of your new partition or drive.** Now you're ready to install the BIS Tools package...

The BIS Tools are self-installing and pretty much idiot-proof. Install them all! The only major point to remember is - as you install each individual tool - alter the path so it installs into your new BIS TOOLS folder...

Sgt Ace provides an [excellent video](#) covering basic tools installation... watch that, then go ahead and install them all... **Reboot when you're done**, and you should notice that now you seem to have a P:\ drive appearing in Windows Explorer. It's not a real drive of course - it's a "virtual drive" created during the Tools install. If you take a look inside your newly-filled BIS TOOLS folder, you'll see a folder named "ArmAWork" - this is where the actual contents of that virtual P:\ drive actually live - but you'll never mess around in here directly. To all intents and purposes you can treat P:\ as a real drive, and indeed - that's where all the action will be happening from now on...

Like all terrainmakers, I suffer from a Powerful Compulsion to Make Backups! Sometimes I even make backups of backups! You'll have your own backup fetish soon enough - meantime you can humor mine by making a new folder somewhere and calling it something like "Default P Contents"... Then copy the "BIS TOOLS\ArmAWork" folder I mentioned before into there... This isn't a "in case the drive crashes"-type backup... more of a "I'm gonna mess around in here now so best have a copy of all the original files - just in case"-type backup...

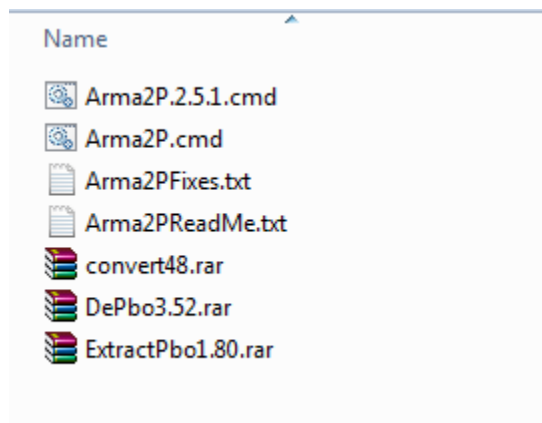
Now, take a look inside P:\ and you'll see that the install process has placed a whole load of files in here already. Notice that "CA" folder? - it's *very* important! If you look inside there you'll see some "Roads" folders... These contain *specialty-formatted road parts* for the Visitor road tool. These are different from the road-parts contained within the Game, which is why they're included in the Tools install (and why we made a backup!)

The rest of the stuff you'll use in Visitor - buildings, fences, rocks & trees, etc - are included with the Game itself, of course, and your next job is to unpack the rest of your Game Assets into this CA folder alongside the roads... This has to be done in a carefully structured arrangement of folders - precisely mimicking the Game itself - in unpacked form...

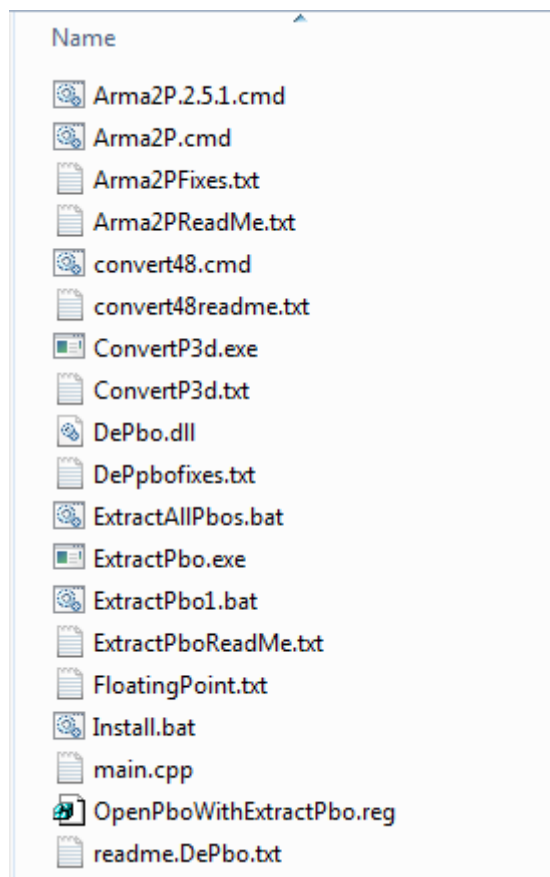
Sound complicated, laborious and seriously off-putting?... It *was* - until Mikero's Arma2P Dos Tool! This turns a full evenings picky and laborious work into a coffee break! So let's play with that next...

1.4 - UNPACK GAME ASSETS WITH ARMA2P

You downloaded Arma2P 2.5.1a earlier, didn't you? - now's the time to unpack that... Make yourself a "Mikeros Tools" folder somewhere - beside your "BIS TOOLS" folder might be a good place - you'll collect more Mikero Tools as you go along, but for now - copy your Arma2P .rar file over there and unpack it... you'll see something like this...



Arma2P requires the contents of those other three .rar files - convert48, DePbo & ExtractPbo - to be here in the same directory as Arma2P itself, so right click those in turn and "Extract here"... watch out for "convert48.rar - one of the files it produces when you unpack it is another .rar file! - "ConvertP3d" - unpack that too!... When you're done you can delete all the original .rar files and you should have a folder like this...



Now you're ready to go... But, before you do - **reboot!**

Then come back to this folder and doubleclick "Arma2P.2.5.1.cmd"... A DOS Window will appear and you'll see reports of files being processed... This could take 15 minutes or so to complete, and Arma2P is shifting some serious numbers of files around, unpacking and arranging them as it goes... best not disturb it by doing other stuff while it's busy... go fetch a coffee or something - don't forget a biscuit!... it'll be just about done by the time you get back...

Rarely, Arma2P has been known to throw a memory error, find a dodgy file and hang, or otherwise fail to complete... Give it plenty of time, but if it doesn't end with a "*Press any key to continue...*" message at the bottom of the DOS window, it might have encountered a problem... If this happens - try rebooting and running it again - it'll intelligently clean up its last attempt before trying unpacking a second time...

OK... With a bit of luck you now have the latest tools installed - plus an almost fully configured P:\ development drive established! *Well done!*...

1.5 - DO SGT ACE'S TUTORIAL !

You have a good head start! The whole of the first section of the tutorial is concerned with the process we've just completed - setting things up! We've done that already - the fast way!, so you can skip directly to the tutorial itself...

Start Sgt Ace's Tutorial at [THIS STAGE](#) but *-see below!*

- **Ignore the "Update Bulldozer" section which follows immediately afterwards!** Since the new tools version 2.5.1 was released, there's *no* need to mess about "using Arma2OA.exe as bulldozer" or anything like that at all!
- Follow the rest of the tutorial steps carefully - all the way. You should end up standing on a terrain which you basically constructed from scratch - fully binarized and in-game! That *is* a Major Accomplishment! Don't underestimate it! Many people don't make it this far... they don't get to check out the grass blowing in the wind, run down to the shore and hear the whooshy sea, check out the hills in the distance and think - "I actually *made* this!". Better still is knowing that you can change those hills, make the sea a different colour, change the grass, even change the wind if you want!

A good way to see how to do all those "next steps" is to take apart and rebuild a couple of little ready-made terrains that have all that stuff included... that's what these terrains and this guide is for! So wave goodbye to *Tut SampleMap* and head on to the next chapter and your P:\ drive - there's some more fiddly stuff to take care of...

1.6 - "TAG" OR "NAMESPACE" FOLDERS & THE LITTLE "CONFIGS-ONLY" CA FOLDER

My standard development structure for these terrains uses a "*tag folder*" or "*namespace*" structure... Strictly speaking, this *isn't* necessary - you *could* have your project folders directly on the P:\ root directory... But since Sgt Ace's tutorial uses a "*TUT*" tag folder, and all these sample island files are pathed to my "*Bush*" tag folder structure, I'll be recommending that you do also... for the purposes of this exercise at least...

So... now it's time to make your own, and therefore time to decide on a "*Tag*" for yourself. If you haven't got one already, decide on something and then head over to [OFPEC](#) and register your tag.

Once you've done that - **make a "*namespace*" or "*tag*" folder on your P:\ drive.** For the purposes of this walkthrough I'll use "*YourTag*" as a "dummy" tag in all examples...

Okay - **you now have a "*P:\YourTag*" folder!**

For complicated reasons which we don't need to understand, when we binarize an island containing buildings with ladders or doors, BinPbo needs to be able to access the config.cpp files for these objects... Arma2P

unpacked your objects into that important P:\CA folder, so you're cool, right?... not quite... You need a special copy of that entire CA folder, containing *only* the config.cpp files - *not* all the objects & textures - we'll call this the little "**configs-only CA folder**"...

Arma2P made you one of these as well! - isn't it clever?... Look on your P:\ drive and you'll see a folder which Arma2P created called "**WRP_PROJECTS**" - inside there is a "**ca**" folder... this is your ready-made "little configs-only" ca folder!

Copy it into your "tag" folder - so you have "P:\YourTag\ca"

Now - all doors and ladders, gates, etc - every object with an animation - will work properly on your terrains!

1.7 - *OPTIONAL ADVANCED STEP * - BULDOZER & VISITOR ADVANCED CONFIGURING

At this point your Terrain Development Setup *is* complete and ready to go, but there is a small "advanced" configuration procedure which you might usefully consider... Absolute beginners could happily proceed to the next chapter and come back here later. Otherwise - read on...

When you're working with Visitor you'll often have Bulldozer open as well... "buldozer.exe" is essentially a chopped-down version of "Arma2OA.exe" designed as a "viewer" program, so you can simultaneously work on your terrain in 2D and 3D (sort of... most of the time)...

Since Bulldozer.exe is basically the same as Arma2OA.exe, it can have its own configuration and profile files - just like Arma2OA does... it's handy to actually create and define these files specifically for bulldozer, since then you can dictate parameters specifically for your Visitor / Bulldozer combo... Parameters like - window size & resolution, viewdistance, graphics quality, position & size of the bulldozer "infoline", specific key & mouse assignments, etc, etc...

Let's get these basic files created and into place. First we'll go and take a look at the two config files your Arma2OA uses... In Windows 7 these are stored in "Libraries\Documents\Arma2\" - your setup may differ.

The files we're looking for are "**ArmA2OA.cfg**" and "**yourgamename.Arma2OAProfile**"... Find these and take a look! You'll see that ArmA2OA.cfg is mostly concerned about graphics settings, sizes of windows, resolution, etc... handy! YourGameName.Arma2OAProfile is longer and has a lot of stuff in it - including things like key assignments, etc... also handy!

What we'll do for Bulldozer is to use an adapted version of your ArmA2OA.cfg for one of its files, and force it to generate its own .profile file for the other... So - while you're here looking at your main game files - **right click your ArmA2OA.cfg file and choose "copy"** - then head back to your P:\ drive...

Back viewing your P:\ drive now, so right click in there and choose "paste" and you have a copy of ArmA2OA.cfg on your main root P:\ directory! - **Rename it now!** - I call mine "bdozer.cfg"... Once you've done that, open it up in Notepad and take a look... Here's what mine looks like - yours will be a little different...

```
language="English";
adapter=-1;
3D_Performance=93750;
Resolution_Bpp=32;
Resolution_W=1440;
Resolution_H=900;
refresh=60;
```

```
winX=14;
winY=34;
winW=1440;
winH=900;
winDefW=1440;
winDefH=900;
Render_W=1440;
Render_H=900;
FSAA=0;
postFX=0;
GPU_MaxFramesAhead=1000;
GPU_DetectedFramesAhead=1;
HDRPrecision=16;
lastDeviceId="";
localVRAM=1041694720;
nonlocalVRAM=2147483647;
Windowed=1;
vsync=1;
AToC=7;
```

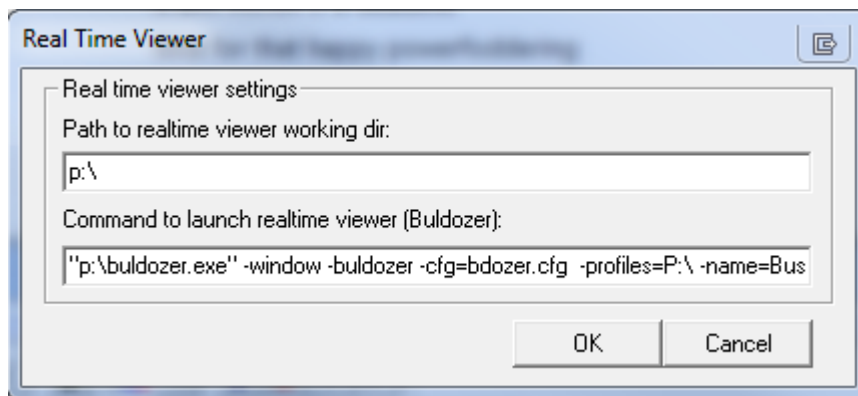
The main things to spot here are...

Windowed=1; - yours will be "0" - change it to "1"

... plus notice how all my "resolution" parameters are set to 1440x900 - since my monitor is 1920x1200 that gives me a decent sized Buldozer window, plus some desktop room (I keep Visitor itself on a second monitor)... Decide for yourself what sort of resolution suits you best and enter the figures here... or just try 1440x900 for now and later on when you have Buldozer open - after we've told it to use this config - you can just drag the buldozer window to the size you want, and it'll write those values directly into this file!

Next step is to tell Visitor to actually use your new "bdozer.cfg", plus we'll ask it to create a fresh "basic".ArmA2OAProfile file... So - open Visitor!

Once your empty Visitor window is open, choose "Tools / System Preferences" and you'll see a window something like this...



You can't see the whole launch command line in the picture but this is what it says in full...

"p:\buldozer.exe" -window -buldozer -cfg=bdozer.cfg -profiles=P:\ -name=Bushlurker -exThreads=0

- **Delete whatever is in your command line, and paste in my line above - the whole thing!**
- **Change "bdozer.cfg" if you used a different name for your file.**
- **Change "Bushlurker" to "YourGameName", or any other name you want.**

You're done! - Press "OK" to exit...

Now... Run Bulldozer! - (Click the "!" icon top/left).

A Bulldozer window should open - probably a 1440x900 one if you just copied my figures earlier... this shows that Bulldozer is now using your config! - Success there!... You'll also see the usual Bulldozer square white cursor and probably a "UI infoline" - this may be an odd size, and/or in the middle of the view... we'll fix that next... If you'd like a different sized bulldozer window, notice you can drag it to whatever size you like... You're using a bulldozer-specific config now, so whatever size you decide on will be what's used from now on.

When you're happy - close Bulldozer, close Visitor and head back once more to the P:\ drive...

If you look closely you should see a P:\Users folder that wasn't there before! - look inside! - a "YourGameName" folder!... and, inside that, a freshly generated .ArmA2OAProfile file!

Open this up and take a look - bearing in mind the "full-game" version you had a look at earlier... You'll see that this file is a lot shorter... One obvious thing missing is a list of keyboard key assignments... we'll have to add them ourselves... more on that later, but first - a quick glance shows some interesting stuff here... there's "viewDistance", "shadingQuality", "shadowQuality" - obviously you can now tweak these values to suit your preferred Bulldozer experience - without affecting your Main Game! Notice also those "uiTopLeft", "uiBottomRight", "IGUIScale" parameters? - You guessed it - they control the scale and position of your "UI infobar" in Bulldozer...

I won't just paste my personal .ArmA2OAProfile file here - because it's too big, because of the "keyboard controls assignments part I copied and pasted from my main game file...

[Here's a LINK](#) to the file so you can see exactly what I did, plus you can see what parameters I use for all the other values...

Beware of just copying my key layout, however... I use a slightly tweaked "mouse 'n arrow keys" setup you may not like...

If you actually run your Full Game and go to the keyboard controls assigning section, you'll see that you can actually define all the Bulldozer keyboard and mouse controls right there! When you quit back out of the game, those values you set will now be in your usual "YourGameName.ArmA2OAProfile" file we looked at earlier...

If you want to reassign keys - do it in-game, then copy the key-assignment section out of your game profile and paste it into your Bulldozer one... just like I did in mine...

Okay... At long last, you're finished! Completed! Done with this whole beginners initial setup section... You've started out for the right reasons - the Creative Impulse to actually make your own Landscape - an Admirable ambition... You've felt the thrill of running around on Sgt Ace's TUT SampleMap which, even if you didn't actually *make* it yourself, you *did* successfully assemble it from the same basic parts all terrains are made of... You've slogged through endless setup instructions - basic background development structure and mechanics... By now you're thinking, "*When does the Creative stuff start, dammit?!'*"... Well... not quite yet... you've used ready-made terrain "parts" before, when you assembled *TUT SampleMap* - now it's time to look at a new set of *Source Files* - one at a time - so we can see exactly what they do and what happens if we change things...

You're still learning the language... soon you'll be able to write poetry with it - or graffiti, if you prefer...

SECTION 2 - THE SOURCE FILES

Now we come to the main purpose of this guide... A walkthrough of all the files and folders contained in the accompanying *Bush's Geotypical Microterrains* packages. All of these quite different terrains were built around the same basic "template structure" of files and folders, so in this section I'll explain the basic process of "grabbing" this structure for yourself... Since that involves "handling" all the basic files, we'll go a step further than Sgt Ace's tutorial and actually look a bit more closely at all of the files in turn. We'll see which parts you need to rename and repath in order to make the "template" your own, and also hopefully become a bit more familiar with the files themselves, what they do and what other bits we can change...

There will be five "microterrain" source file packs available (*eventually*) - all essentially the same as far as the main files and structure are concerned, though textures and details like clutter mixes and types will differ, of course. Additionally, some terrains include more "advanced" features such as custom footstep sounds and recoloured sea parameters (kindly donated by prowler.wolf), and/or basic vegetation placed via Shezan74's *WorldTools* (appropriate files included). All terrains also include a simple intro powered by Tupolov's *Map Intro Script*.

I'll explain each of these additional features in later *Appendix* sections, and refer you to those other source files at that time, but for the main files walkthrough I'll use "*Afghan Valley*" as my primary example.

2.1 - NAMES AND NAMING CONVENTIONS

Before we begin it's best if we're all clear and agreed on a few things which will save confusion later.

For the purposes of this exercise I'll be assuming you have decided on a "**Tag**" and created a Tag or Namespace folder... All of these source files currently expect to live inside a "**Bush**" tag folder structure - **we'll be repathing them all to a "YourTag" structure so - whenever you see "Bush" in the forthcoming examples, I'll be changing that to "YourTag" - you, of course will use your actual Tagname when you do it for real!**

Secondly, of course, the name "*AfghanValley*" or sometimes just "AV" for short, is going to crop up a lot! - in almost every file we look at...

If you were repathing one of these packages as a basic template for your own project, you would probably have decided on a name already. Or you may be planning a realworld terrain, in which case the name may already be decided for you... For this exercise I *could* use "*YourTerrain*" in all examples, as I'll be doing with the Tag folder, but that tends to make the examples clumsy and confusing, so I'll decide on a new name specially for this walkthrough...

Hmmm... Afghan valley high in the mountains, accessible only by donkeys... quick Google... Pashto for Donkey is *Khar*, for Mountain it's *Ghar*... **My new name for repathing this package will be "*KhargharValley*" or just "**KV**" where appropriate** - that should make the examples a bit more readable, and it sortof sounds vaguely authentic too - which is always a bonus.

Let's get started!

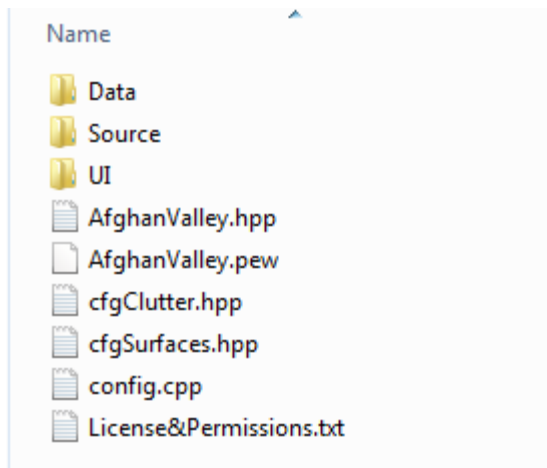
2.2 - INSTALLING THE SOURCE FILES PACKAGE

Download and unpack the "Bush_AfghanValley_v02_SourceFiles" package. All the files are contained within the "AfghanValley" folder...

- **Copy the folder to your P:\YourTag\ folder so you have P:\YourTag\AfghanValley**
- **Rename the folder so you have P:\YourTag\KhargharValley**

That's the first renaming step done! Now we'll look inside! You should see this...

Contents of P:\YourTag\KhargharValley\ - The "Main Project Directory"

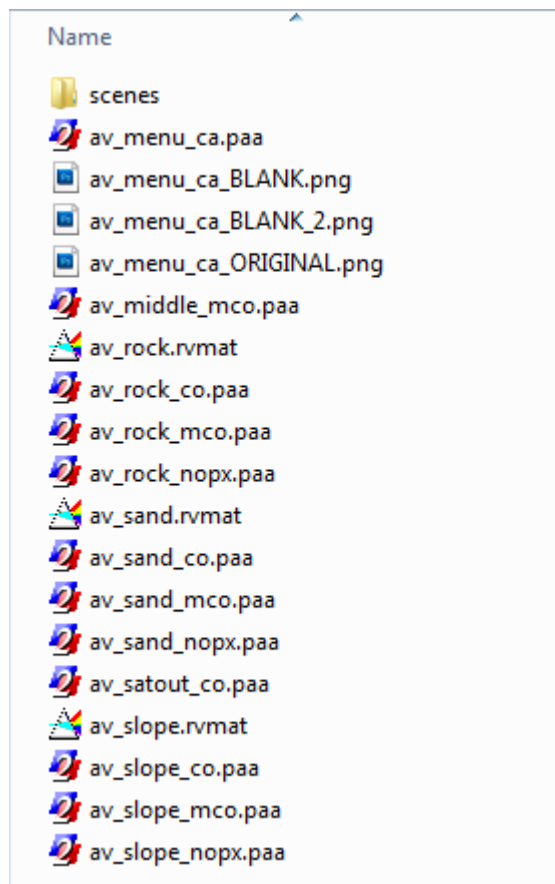


Those "AfghanValley" files need renamed, of course, but we'll worry about these important **Main Project Directory Files** later. First we'll look at each of these folders in turn - see what's in them and what we need to do.

2.3 - THE "DATA" FOLDER

Open the "**Data**" folder and you should see this...

Contents of P:\YourTag\KhargharValley\Data\ - The "Data folder"



The "**scenes**" folder contains the "*Intro Cutscene*" - we'll look at that separately and later on. The main purpose of the *Data* folder is to contain all the "*Ground Texture Sets*", plus a few other associated texture and image files. We're here to deal with them first!

First thing to notice is that all the files seem to be "tagged", but not with "*Bush*", as you might expect. Instead, they're tagged with "**av**". I generally follow the BIS example and tag these files with something short and relevant to the terrain itself... in this case "**av**".

Unlike later, when we'll be walking through config files - renaming things manually to get a feel for what they do, there's nothing really to learn from manually renaming texture files, so it's for thankless jobs like these I mention "File Renamer Basic" in the Optional Downloads section...

Choose your renaming method now, and work your way through the process of retagging all these files - from "av**" to "**kv**"... I'll assume you've done this for all subsequent discussion of these files!**

We'll have a little more work to do when we come to the ".rvmat" files, but for now - let's start at the top of the list and quickly step through some of these files to see what they are and what they do...

THE MENU IMAGE

- **kv_menu_ca.paa**

As the name implies, this is the 512x128 pixel menu image which is displayed in the in-game editor menu. This current version belongs to the "old" *Afghan Valley* - complete with my logo and the terrain name. There's also a .png version of this file - labelled "*Original*", so you can take a look. There are also two readymade "blank"

terrain shots which you could use to make your own menu picture - with "*Kharghar Valley*", or with whatever name you decided on...

THE "MIDDLE MCO"

- **ka_middle_mco.paa**

This texture file is our first encounter with those *Mysterious MCO's*... We'll run into them again in just a minute as part of *Ground Texture Sets*... A full explanation belongs in a texture-making tutorial, but, basically... An "MCO" is a "*Detail Texture*"... In-game they appear enlarged and mostly transparent, and are used as "*Detail Shaders*" to overlay some extra detail in the "middle distance view" of textures... In this case, the overall *Satellite Layer*...

You've seen these "*middle mco's*" in action - even if you haven't realised it... This file in use here was poached from *Takistan*, and is a *seriously* good example! If you've ever played on *Takistan* you'll have seen hills from a decent distance... that's the "*Sat_Lco*" layer - pure and simple... Really close up - that's the ground textures... but in that "middle distance" - you see it best taking off in a chopper if you look down as you lift off - just as the actual close-up ground textures seem to lose detail, there seems to be some sort of rocky, stony "overlay detail" enhancing the basic Sat Layer when you're this close, and you might start noticing the Sat Layer is actually pretty low res... that's the "*Middle MCO*" - this file, working very well!

For now - it's enough that you know you need this file, and you have a basic idea of why. If you're making a predominately "grassy" terrain, maybe borrow the middle_mco from Chernarus... if it's a more "rocky" terrain, use the *Takistan* one - as I have in this case...

A TYPICAL "GROUND TEXTURE SET"

- **kv_rock.rvmat**
- **kv_rock_co.paa**
- **kv_rock_mco.paa**
- **kv_rock_nopx.paa**

These four files comprise a typical "*Ground Texture Set*". For now you'll probably just copy these ready-made from one of the BIS islands, or maybe scrounge an interesting or handy custom set from another terrainmaker... This particular "rock" one is a custom one of mine, which you can use of course... in fact - you'll see this texture set in use on most, if not all of my little demo terrains - usually with the palette shifted a little to suit the overall theme. The three ".paa" files are texture images of one sort or another - the ".rvmat" is best thought of as a little "config" file for each texture set... Let's quickly look at the .paa textures and then move on to the .rvmat - which we need to fool around with a little...

The "**_co**" texture file is basically the ground texture you see - directly under your feet in-game - if you need to tweak the colours of a ground texture slightly to match your overall theme - like I mentioned above with the rock texture - then this is the only file you need to mess with...

The "**_mco**" texture file does exactly the same job as the "overall" middle_mco discussed above - except on a smaller scale - for the ground textures only.

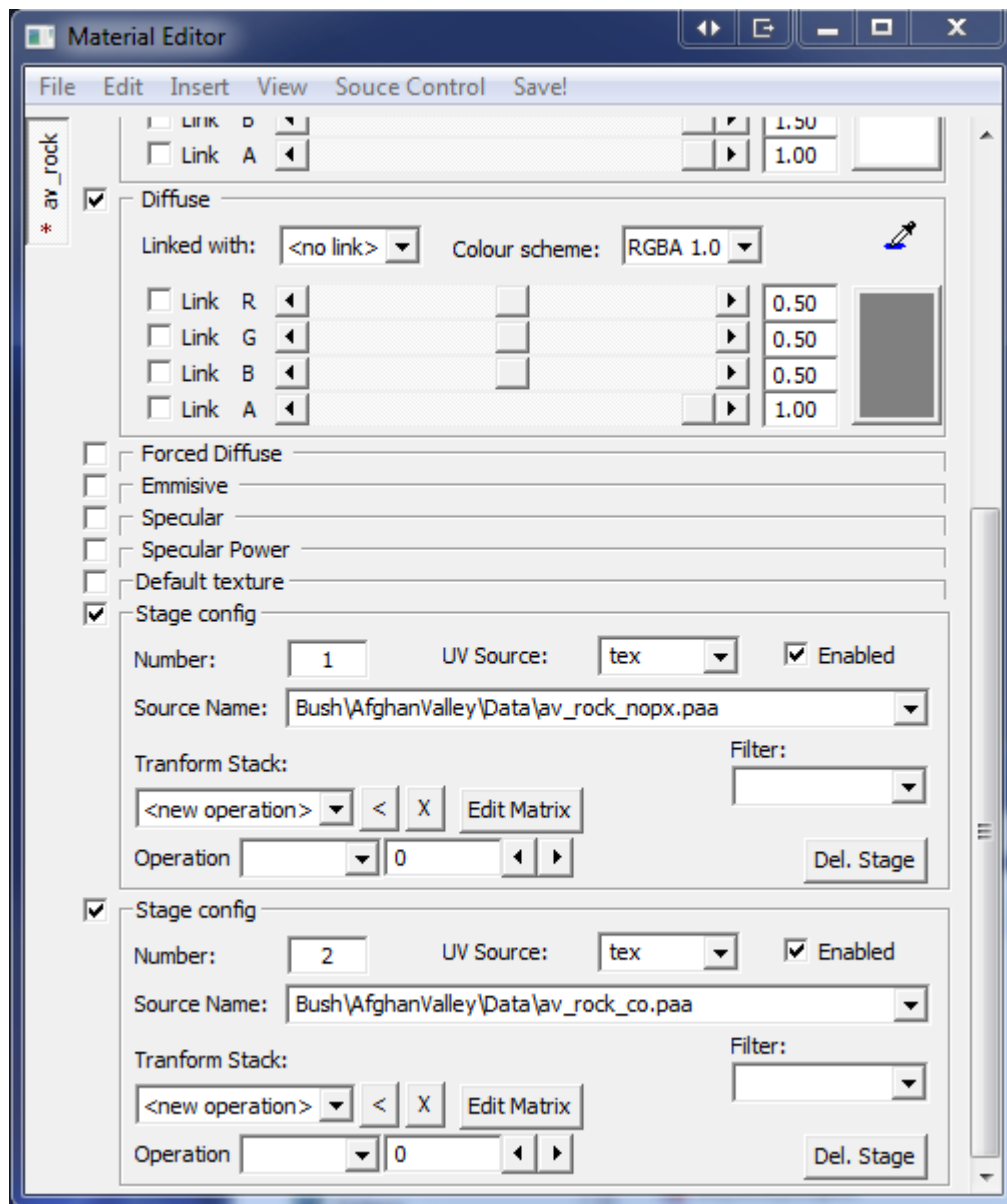
The "**_nopx**" is a "Parallax Displacement Map" which is basically a type of *bumpmap*... it makes the close up "_co" texture look a little more 3D and bumpy...

...And that really *is* all you need to know about these files for now... Making your own is an advanced thing - for now there's plenty of ready-made BIS sets to choose from to cover most situations...

The final file in the set is the **".rvmat"** file... this is basically a text file, just like any other config file, so you can open it in notepad if you like and edit it there... However - BIS provide a little **"rvmat GUI"** - the *Material Editor* - which should automatically be associated with unbinarised textfile-based .rvmats. and they should open in that when you doubleclick. Like most config files, these .rvmats - notice there's one for each Texture Set - contain paths and filenames...

We need to repath and rename the info inside each of these .rvmats! Let's do that now with our example "kv_rock.rvmat" one - but **remember to follow the same procedure for ALL the .rvmat files in this folder!**

Doubleclick "kv_rock.rvmat" and it should automatically open in a window like this...



Well... actually you'll need to scroll down a little so it looks like this...

It all looks awfully intimidating and complicated in here... that's mainly because .rvmats *can* be pretty complicated - for models mostly... ours are really easy - all we'll ever do is alter those two "Source Name" paths you can see in the picture above... they're even in little "popdown" windows for easy repathing!

The first Source Name path is currently pointing to the old "Bush\AfghanValley\Data\av_rock_nopx.paa" structure... **Click the little popdown arrow on the right and choose the "Browse file" option... browse to your newly-renamed "kv_rock_nopx.paa" file and choose it. Do the same for the "_co" texture in the second Source Name window...**

Your Source Name paths should now read...

YourTag\KhargharValley\Data\kv_rock_nopx.paa

YourTag\KhargharValley\Data\kv_rock_co.paa

Now click the "Save!" option at the top of the window...

You're done!

There's three Texture Sets in this folder - Rock, Sand and Slope... so now **do the same thing for all the rest of the .rvmat files in this folder** - repath to the appropriate renamed texture files in each case.

THE "SATOUT" TEXTURE

- **ka_satout_co.paa**

The "*Satout*" texture is fairly simple and self-explanatory - it's the "**Satellite**" texture for the "**Outside Terrain**"!

Unlike the main area of your terrain, which has a *huge* Satellite Image draped over it (the "*Sat_Ico*" image which we'll discuss later)... *outside* the borders, the Game Engine will simply tile this basic *Satout* texture endlessly. If you're designing a "landlocked" terrain, with "infinite Land" all around, it's particularly important that this texture blends in reasonably well with the overall "average tone" of your Main Satellite Layer. For this reason you may find yourself tweaking the colour of this texture file slightly - or indeed making one yourself. For this particular "Afghan-style" terrain however, this file - borrowed from *Takistan* - blends in nicely - so we'll just use it as it is. There'll be a little more explanation of "*Outside Terrain*" later, when we enable it and specify it's parameters in the "*config.cpp*"...

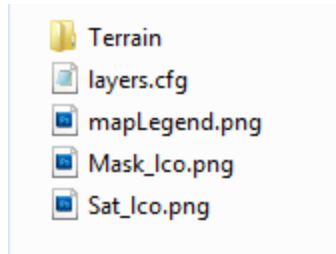
And that brings us to the end of the **Data** Folder discussion - for now... We'll come back here for a closer look at that "*scenes*" folder and cutscene intros in the first of the "Advanced" *Appendix* sections, but right now - we've established the new names of all these important texture files (plus learned a little about them), so now it's time to move on to some assorted *config* files - most of which will be calling on these texture files in one way or another - which is why we got the renaming part done first! - Well Done so far!

Now - step up a level to your *Main Project Directory*... See that "**Source**" folder? - We'll open that next and take a look!

2.4 - THE "SOURCE" FOLDER

Open the "**source**" folder and you should see this...

Contents of P:\YourTag\KhargharValley\Source - The "**Source folder**"



As the name implies, the "Source" folder is basically just a place to store some of the main data sources our terrain will use - the "**Heightmap**" (in the "Terrain" subfolder) - the "**Sat & Mask**" layers, plus there's a little "required infofile" - **mapLegend.png** - and a very important config file - **layers.cfg**.

Actually making some of these files - the Heightmap, the Sat layer and Mask layer - is well beyond the scope of this basic guide. Right now our primary goal is to make all the changes necessary to make this whole terrain package "your own". Once you've achieved that - you could, for example, simply swap out these Height, Sat and Mask files and substitute new ones, and you'd immediately have a viable and working totally different terrain!

You've encountered and used all these files before when you followed Sgt Ace's Tutorial, of course, but we'll take a quick tour of them all anyway - just so we know what they are and basically what they do, then we'll move on to that important layers file.

THE "TERRAIN" SUBFOLDER

The Terrain subfolder is basically just a handy container for your Heightmap data file(s) - take a look inside! Two files! - **Rename these to "KhargharValley.png" and "KhargharValley.pbl"!**

- **KhargharValley.png**
- **KhargharValley.pbl**

Sgt Ace's Tutorial gave a basic explanation of this pair of files... the .png file is your "*Heightmap*" file - **a 16bit Greyscale PNG** - the .pbl is a little text "config" file which dictates the "*square size*" or "*cellsize*" of your heightmap, plus the "*maximum and minimum height values*".

Take a look at it in your favourite art program... Notice it's 1024x1024 pixels in size. The .pbl file specifies a "square size" of 5 (meters), so - $1024 \times 5 = 5120$ (meters) - this terrain is 5120x5120 meters - or 5.12km x 5.12km

What are we actually *seeing* though? Well... essentially - these Heightmap files are maybe best thought of as a grid... Better still... imagine a fishing net stretched out flat - it's a *mesh*, with knots at the corners - we'll call those knots "*vertices*"... Now - still imagining that flat horizontal mesh plane... if we were to grab a vertice and pull it upwards it would make a jagged point - doesn't look much like a hill! - if we were to pull up all the other vertices around that one though - not quite as much, and the ones further out a little less, we could at least approximate a more rounded and hill-like shape. You can see immediately that - if we had a very fine mesh, with lots of vertices, quite close together - we could make very detailed ups and downs in the mesh... With fewer vertices, further apart, things might look a little more "squared off" and jagged...

When you use the ".png & .pbl" format for your heightmap, the image basically represents your 1024x1024 vertice mesh - with the colour of each pixel representing the height of that particular vertice... The .pbl file specifies the in-game spacing of those vertices - in our case - 5 meters, and it also specifies the "height range" to be used... A totally "black" pixel will mean that vertice is set to the *minimum* height value specified, while a totally "white" pixel will cause it's corresponding vertice to be set to the *maximum* height value specified...

- **KhargharValley.xyz**

Some heightmap editing programs export in ".xyz" format. Since Visitor 3 knows how to Import (*but not Export!*) this format, it's often used as a quick substitute for the ".png and .pbl" method. The "cellsize" and "max / min height values" are embedded in this format, so there's no need for that .pbl file at all. You can think of an .xyz file as basically a spreadsheet... in this case, a 1024x1024 cell spreadsheet - each cell containing the height value for the corresponding vertice...

Although I mostly use .xyz format myself, I've provided a .png and .pbl set here - they allow more scope for beginner to fiddle with water levels, etc plus they're familiar already from Sgt Ace's Tutorial.

You won't actually *use* these files at all in our little walkthrough - one of the handy things about having a ".pew" or "Terrain Project File" is that the heightmap is embedded inside it! When you come to load this project into Visitor - later on in the guide - you'll see that the heightmap is already in there!... No need to re-import! So, basically - these files are only here to show you where Heightmap files go - and what they look like... When you've finished adopting this template structure to your own needs - this is where you'd put your heightmap files - in one of these two formats...

Okay - step back up a level to the main "Source" folder again and we'll take a look at the rest of the files.

MAPLEGEND.PNG

- **mapLegend.png**

The official explanation of the "mapLegend" file is even more cryptic than usual! While I was working on CWR2 Everon recently, I devoted several days to figuring out precisely what the hell this file is for and why it's necessary... I figured it out - eventually... It's cunning... it's devious, and... it's of absolutely *no* interest or real, practical use to us Community mapmakers at all! It's essentially a "lookup table" for colours which may be used in your "Mask_lco" file... As such - it merely needs to be present... **And that's basically all you need to know about this file... It has to be present - here - in this folder...** You can use the one I've provided here - *no need to rename it!* - I made this one during my experiments, but you could use the one that was supplied with the Sgt Ace Tutorial if you prefer - it doesn't matter...

THE SAT_LCO.PNG

- **Sat_lco.png**

The "Sat_lco" or "Satellite Layer" is a "24 bit" or "8 bit RGB" .png file. It's basically the overall satellite image which will be draped over your heightmap to provide the actual landscape you see *in the distance* in-game. Up close, you'll see the close-up ground textures we looked at earlier of course, and - in that "middle distance" you see a combination of the "Sat Layer" and that clever "Middle MCO" texture we looked at earlier...

Making your own Sat Layer is an Art in itself! A good one can really enhance your terrain... a bad one can look just awful! This is one of the major "creative" areas in terrainmaking, where you can really let rip with those painting skills, if you have them. If you don't, or if you're planning a realworld terrain, you may prefer to acquire a real satellite image of your target area... Generally, the best Sat Layers involve some clever combinations of both...

Once again, actually *making* one of these files is *well* beyond the scope of this guide... there are some tutorials around and, once you've worked your way through this guide and you're planning a full scale project of your

own, you'll probably want to go looking for those... For now, there's the original Afghan Valley one here to use...

Take a look at it in your favourite art program... Notice it's 10240x10240 pixels in size... We did this with our "Heightmap" file - remember? - it was 1024x1024 pixels, so this file is *10 times bigger*! That's good! We talked about the underlying Heightmap "*mesh*" earlier, remember - with its vertices 5 meters apart? That's actually pretty high-res for a BIS terrain! I think Takistan is around 8 meters, Chernarus around 10 meter spacing... Still, if you were to imagine looking around or walking across a terrain of these resolutions you'd think it looked pretty crude! - yet Takistan and Chernarus look great! That's because of the much higher-res Satellite Layer! - You don't actually see the underlying mesh at all, because this *much* higher Satellite Layer is "draped" over it...

Okay... when you come to making your own heightmaps, you'll also need to consider Sat_Lco's in more detail... Right now, we know what it's for and a little more about it... It has a partner - the "Mask_Lco" file... let's look at that next...

THE MASK_LCO.PNG

- **Mask_Lco.png**

Once again - no need to rename this file - it's always called "Mask_Lco"... Once again, open it up in your art program of choice and take a look! It's 10240x10240 pixels again - just like the Sat_Lco, and it looks suspiciously like a "basic colours" version of the Sat_Lco... That's exactly what it is!

The Mask Layer is essentially a "lookup table" which controls close-up ground textures... The Sat Layer provides the overall "photograph" which you see - every pixel on it is "pretending" to be Rock, or Sand, or Grass - the Mask layer ensures that - if you were to run over there to that "grassy looking bit" on the Sat Layer - you'd see close-up grass texture - and it's associated clutter - when you get there!

Ideally, the Mask_Lco should correspond almost exactly to the Sat_Lco. Wherever "grass" occurs on your Sat Layer, the corresponding "control colour" for "grass" should occupy the same area on your Mask_Lco. This "control colour" will in turn be associated with a "Ground Texture Set" - you'll define these "control colours" and their associations just shortly in the "layers.cfg" file.

Since there is a engine limitation, as well as practical ones, on the number of Ground Textures you can effectively use, there may only be three or four "control colours" in use - so the Mask_Lco will be composed of very few colours - one for each texture set in use...

As you noticed earlier when we were looking around the "Data" folder, there's only three Ground Texture Sets in use on Afghan... sorry - *Kharghar* Valley, you can see how the Mask Layer consists of only three colours - one each for the "Rock", "Slope" and "Sand" surfaces...

As you've guessed - making Mask_Lco's is almost, but not quite as tricky and arty as making Sat_Lco's! In fact, they're generally made in tandem - changes in one requiring a change in the other, as you'd expect.

Anyhow - by now you should at least have a basic grasp of what these files *are*... let's move on to the final file to be considered - the "*layers.cfg*", and see how it pulls all these files together.

THE "LAYERS.CFG"

- **layers.cfg**

The last file we need to look at in this "Source" folder is - "**layers.cfg**". This is an important file, but very simple and straightforward. It's *only* used when you import your Sat and Mask layers into Visitor, and it essentially just defines which "*control colours*" exist on your Mask, and which *Ground Texture Sets* are associated with each.

Let's look at the basic Afghan Valley one - you can see immediately the bits to change (in red) - pretty obvious, really - plus I'll explain things a little as we look at each section (the green text).

P:\YourTag\KhargharValley\Source\layers.cfg

```
class Layers
{

class av_rock
{
    texture = "Bush\AfghanValley\Data\av_rock_mco.paa";
    material= "Bush\AfghanValley\Data\av_rock.rvmat";
};
```

These few lines above provide the information Visitor needs to know about the "Rock" areas on your *Mask Layer*. Notice how the classname has been "tagged" with "av" - just like the texture files we renamed earlier. Renaming is required here too, of course - so **change that "av_rock" to "kv_rock" to start with.**

The following two lines define the "texture" - that's that MCO "detail texture" we discussed earlier, plus the "material" it's made of.... Notice how the "material" we're declaring is actually an *.rvmat* - not a *texture*? Remember we looked at the *.rvmat* files back when we were renaming textures in the "Data" folder? We repathed the *.rvmats* - which were little config files, pointing to two textures - the "_co" or "visible" texture, plus the "_nopx" or "bumpmap" texture.

Obviously, we need to repath these two lines so they point to the correct newly-renamed files, so do that too! Just in case it's not all pretty much blindingly obvious by now, that means you'll change them to...

```
texture = "YourTag\KhargharValley\Data\kv_rock_mco.paa";
material= "YourTag\KhargharValley\Data\kv_rock.rvmat";
```

```
class av_sand
{
    texture = "Bush\AfghanValley\Data\av_sand_mco.paa";
    material= "Bush\AfghanValley\Data\av_sand.rvmat";
};
```

Exactly as before, these lines define the Textures Set to be used for the "Sand" areas on your *Mask_lco*. **Retag the classname and repath the "texture" and "material" lines** to point to your renamed textures.

```
texture = "YourTag\KhargharValley\Data\kv_sand_mco.paa";
material= "YourTag\KhargharValley\Data\kv_sand.rvmat";
```

```
class av_slope
{
    texture = "Bush\AfghanValley\Data\av_slope_mco.paa";
    material= "Bush\AfghanValley\Data\av_slope.rvmat";
};
```

Only three ground textures on this terrain, remember? This is the last one - the "eroded slopes" texture set. **Retag that classname and repath the following lines as before...**

```
texture = "YourTag\KhargharValley\Data\kv_slope_mco.paa";
material= "YourTag\KhargharValley\Data\kv_slope.rvmat";
```

```
};
```

```
class Legend
{
    picture="Bush\AfghanValley\source\mapLegend.png";
```

Here's the reason why that "mapLegend.png" file has to be present in your "Source" folder... because with this line you tell Visitor it's available and where it is! As I mentioned before, there's no real advantage to be gained by messing with this file - just consider it part of the basic "mechanics" of the structure. Make sure the file itself is present in your "Source" folder, then use the path above to point to it. **Repath yours appropriately.**

```
picture="YourTag\KhargharValley\source\mapLegend.png";
```

```
class Colors
{
    av_rock[]={196, 177, 123}};
    av_sand[]={220, 210, 102}};
    av_slope[]={228, 215, 137}};
};
```

Having defined the classnames of the *Ground Texture Sets* to be associated with each of our *Mask Layer* colours - it's time to define the colours themselves!

The three "classes" we defined above are listed in this section. We retagged those names, remember? - better retag their names here too! **Change those three "av"'s above to "kv"!**

The three numbers which follow each classname are the RGB Values of the Colours used on your Mask Layer to represent those areas.

We're just using the original Mask_lco from Afghan Valley for this exercise, so we haven't changed any of the colours, so *we don't need to change them here*. When you come to the stage where you've created your own Mask_lco - this is the section where you'll define the colours you used, and the texture class names you want associated with them.

```
kv_rock[]={196, 177, 123}};
kv_sand[]={220, 210, 102}};
kv_slope[]={228, 215, 137}};
```

```
};
```

... and that brings us to the end of *layers.cfg* and also completes our tour of the "Source" folder... pretty straightforward stuff there, really!

Time to step back up a level once more to the *Main Project Directory* and take a look at the last folder - "UI".

2.5 - * OPTIONAL FOLDER *- THE "UI" FOLDER

- av_logo512_ca.paa
- bushlogo2_ca.paa

Since this folder is present in the source files package - I'd better explain it I suppose. However - it's *not* part of the "essential structure" necessary for a terrain, it's basically just a convenient location to store two logo graphic files which are used in the *Intro Cutscene*. I could have called this folder any name I wanted really... Come to think of it, I could even have stored these files in with the cutscene itself, which might have made more sense... Oh well, the folder is here now and it'll be automatically included in the final binarized island so the graphics files are available in-game when the intro requires them.

Just to be thorough, I'll suggest you **rename "av_logo512_ca.paa" to "kv_logo512_ca.paa"**, though you can leave **"bushlogo2_ca.paa"** as it is... that'll remind you it's *my* logo! - make your own if you decide you want one later!

I'll mention these two files again when we come to taking a look at Cutscenes, but for now it's time to step back up a level for the last time to that *Main Project Directory*... Some of the most important files of all live at that level, so - for want of a better description - we'll take a look at the *"Main Project Directory Files"* next...

2.6 - THE "MAIN PROJECT DIRECTORY FILES"

We saw these files earlier when we installed the Source Files package - time to rename those two "AfghanValley" ones now... **Change "AfghanValley.hpp" to "KhargharValley.hpp" and the all-important "AfghanValley.pew" to "KhargharValley.pew"** - you should end up with the following files...

- **KhargharValley.hpp**
- **KhargharValley.pew**
- **cfgClutter.hpp**
- **cfgSurfaces.hpp**
- **config.cpp**

These files are particularly important, and involve some key concepts in action... You've *used* the same types of file before of course - in *Sgt Ace's Tutorial* - but let's look at them more closely - one at a time - see what we need to change and why.

KHARGHARVALLEY.HPP - THE "NAMES.HPP"

As you'll see if you open this file, it's currently empty! Later on - when you've progressed beyond this simple guide, you'll come to the stage where you'll want to define "Keypoints" in Visitor... These are used to define the *Name* of an area or settlement, plus there are other, more complex uses for Keypoints... Once you've created them in Visitor, they'll be written as config entries to this file - Handy! Since it's a text file, you may also wish to manually type in additional keypoint definitions yourself. Ultimately, this whole file will be "included" in your Main Config.cpp - *that happens automatically at the binarization stage!* - as we'll see later.

So - since settlements and settlement names lie beyond and outwith the scope of this walkthrough, we'll leave this file for now - safe in the knowledge that if/when you do get to that stage - it's here and ready and appropriately named.

KHARGHARVALLEY.PEW - THE "MAIN PROJECT FILE"

This file in particular is likely to become the main focus of your rapidly-developing Backup Fetish - with good reason! This is your "core" project file! If it really, really, *really* came to it - you could *almost* recover from losing your entire project - as long as you had this file... You may need to recreate Sats and Masks - rewrite entire configs - but this file contains those hundreds upon hundreds of hours of work you poured into carefully

arranging those big cities you're *definitely* gonna make for your Big Project! - the trees, the roads - everything! It's all in this file!

Treat it like gold! Back it up frequently! - Back it up **NOW**!

However, there's not really much else to actually *say* about this file - **it's name should always be the same as the project folder - and the terrain itself** - other than that - Keep it safe!

CFGCLUTTER.HPP

Another ".hpp" file! The last one we looked at - KhargharValley.hpp - the "*Names.hpp*" above - was mentioned as being *automatically included* in the Main Config.cpp at the binarization stage, remember? *This file works in exactly the same way*. Think of it as another "external chunk" of the Main Config - we can edit it separately - safe in the knowledge it'll be "included" when required.

This file essentially defines the individual "*Clutter Classes*" which - later on - we'll want to associate with different "*Ground Surfaces*". Once again, it's pretty logical and straightforward - essentially we just create a "*classname*" for each type of clutter, plus specify which clutter model to use, and a few additional handy parameters for each...

Here's the complete file, with my usual "*bits you need to change*" in **red**, plus suggested changes and comments in **green**...

P:\YourTag\KhargharValley\cfgClutter.hpp

```
class AV_BrushHard: DefaultClutter <----- classname - change the "tag"
{
    model = "ca\plants_E\Clutter\c_Brush_Hard_EP1.p3d"; <----- the clutter model and path
    affectedByWind = 0.4; <----- how much it "blows in the wind"
    swLighting = 1; <----- is this clutter allowed to be colour shaded by the Sat Layer? 0=No, 1=Yes
    scaleMin = 0.9; <----- smallest randomised size allowed
    scaleMax = 1.3; <----- largest randomised size allowed
};
```

Almost no need for further explanation here at all! First thing to do of course, is to **replace all those "AV" tags with our "KV" tag**... might as well just do them all!

Having done that - look at this "*class definition*" above as a "whole" for a moment... see how it forms a coherent "chunk"? Now look below - all the remaining definitions are just the same "chunk" over and over! The only things that vary are the name, the clutter model being used, plus the parameters have slightly different values... but, essentially - they're all the same!

You've renamed the classname "tags" already, haven't you? Let's look at the rest of the parameters then...

model=

There's lots of clutter models to choose from - this parameter is simply a path to the chosen model.

affectedByWind=

Does exactly what the name implies. I'm not sure what the valid range for this parameter is - probably 0 > 1.0... The higher the number, the more the clutter type will "sway" in the wind.

swLighting=

Clutters have their own texture and colour, obviously - they're models, albeit simple ones. This parameter seems to be a simple on/off switch which controls whether the colour tone of the overall *Satellite Layer* is allowed to influence the shading of the clutter - or not.

scaleMin= & scaleMax=

These two parameters control the scaling of the clutter type. Once again, I'm not entirely sure of the allowed range, but common sense plus a little experimentation shows that anything smaller than around "0.4" or larger than about "1.4" starts to look a bit silly. More on clutter scaling below...

```
class AV_BrushSoft: DefaultClutter <----- replace "AV" with "KV".
{
    model = "ca\plants_E\Clutter\c_Brush_Soft_EP1.p3d";
    affectedByWind = 0.8;
    swLighting = 1;
    scaleMin = 0.75;
    scaleMax = 1.4;
};

class AV_PlantsViolet: DefaultClutter <----- replace "AV" with "KV".
{
    model = "ca\plants_E\Clutter\c_Plants_Violet_EP1.p3d";
    affectedByWind = 0.7;
    swLighting = 1;
    scaleMin = 1.0;
    scaleMax = 1.25;
};

class AV_PlantsWhite: DefaultClutter <----- replace "AV" with "KV".
{
    model = "ca\plants_E\Clutter\c_Plants_White_EP1.p3d";
    affectedByWind = 0.7;
    swLighting = 1;
    scaleMin = 0.85;
    scaleMax = 1.1;
};

class AV_WeedThistle: DefaultClutter <----- replace "AV" with "KV".
{
    model = "ca\plants_E\Clutter\c_Weed_Thistle_EP1.p3d";
    affectedByWind = 0.7;
    swLighting = 1;
    scaleMin = 1.0;
    scaleMax = 1.25;
};
```

And that's pretty much all there is to know about the *cfgClutter.hpp*! It really *is* just a basic list of simple Class Definitions! While we're discussing Clutter, however...

Since most terrains - including the BIS ones - feature clutter, it's often interesting to take a look at the clutter definitions used on terrains you like, which are similar to the one you're currently making. All the clutter class definitions above, for example, were lifted directly from Takistan. However - remember I mentioned previously - several times - that all these ".hpp" files get "included" in the *Main Config* during binarization? That means that when you unpack Takistan, or any other island - don't expect to find a "cfgClutter.hpp" file - they're just for our convenience at the Visitor stage! Look for the Main "config.bin" file - which you'll then need to "unbinarize" with a suitable utility (such as [Mikero's "Derapify"](#)). You'll find all the clutter definitions in there, you can see what parameters were used and even just lift whole "chunks" and re-use them (after some suitable renaming/retagging, like we did above).

Finally, a few words about Clutter Scaling! I'm a Big Fan of clutter, you can really add atmosphere and believability to a terrain with a well-chosen clutter mix or two, but when it starts to get in the way of actual

play - people will often turn it off if they have the option. That's bad news for your carefully crafted atmospherics! If you can manage to keep most clutters relatively low and unobtrusive, they don't get in the way too much - people leave clutter switched on, and they benefit from the extra ambience. "Keep It Low" is a good rule to bear in mind. The two Clutter Scaling Parameters discussed above are worth playing around with - rather than just blindly copying.

Only two more files to consider, and they're the Big Ones! So far we've defined our *Ground Texture Sets*, then we defined our *Clutter Types* - now it's time to pull both of these together and define final *Ground Surfaces*! We'll do that in yet another ".hpp" file, the *cfgSurfaces.hpp*.

CFGSURFACES.HPP

The *cfgSurfaces.hpp* is yet another one of those external "chunks" of config which it's handy to keep separate at this Visitor stage - like the others, it'll be "included" in the final Main Config at binarization. Essentially, this file consists of two sections.

In the first half we define the final *Ground Surface Classes*. These consist of a *Ground Texture Set*, a "*Clutter Mix*" (or "*Surface Character*"), and a few other useful parameters. Much like the *Clutter Type* classes we defined previously - these are obvious repetitive "chunks" so it's easy to copy/paste, change a few parameters and add additional classes.

In the second half - slightly confusingly - we'll define the "*Clutter Mixes*" which we previously assigned to some of the Ground Surfaces. These "*Clutter Mix*" definitions are precisely that - named sets of one or more of our previously-defined Clutter Types in varying proportions.

Sound a little confusing? It is - a little, at first... It'll all fall into place and make sense eventually! For now, let's just keep stepping through the basic mechanics of repathing and renaming... just by doing that a few times you'll find you begin to grasp the overall idea. Time for some more red and green text! You know the routine by now!

P:\YourTag\KhargharValley\cfgSurfaces.hpp

```
class CfgSurfaces
{
    class Default {};

    -----
    class AfghanValleysandSurface : Default <---- the Surface Classname - rename!
    {
        files = "av_sand_*"; <---- the Ground Texture Set to use for this Surface - retag!
        rough = 0.1; <---- how "bumpy" it is when you drive over this surface.
        dust = 0.1; <---- how much "dust" vehicles and choppers raise.
        soundEnviron = "dirt"; <---- the "Environmental Sounds Set" appropriate to this Surface.
        soundHit = "hard_ground"; <---- the "footstep" sound type.
        character = "AV_sandClutter"; <---- the "Clutter Mix" to use for this Surface - retag!
    };

    -----
}
```

Once again, this "chunk" above represents one whole *Surface Class*... the ones below are just more of the same thing really... We'll look briefly at each of the lines in a little more detail, then move on to the second half of this file. **Make sure you've done all the renaming you need to do for this file - the red bits, as usual.**

files=

This line controls which Ground Texture Set is used for this surface. "kv_sand_*" (you *did* rename, didn't you?) just means "all the texture files named kv_sand_ 'something'".

rough=

I'm not sure of the valid range for this parameter, though you usually see low values of around 0.1 to 0.3 or so. It basically controls how "rough" the surface behaves when you drive over it in vehicles... You could use this, for example, to make "ploughed fields" a bad option for a cross country short cut!

dust=

Again - valid range unknown but low values like 0.1 to 0.3 seem to be the norm. This parameter, as you'd expect, controls the amount of dust raised by driving vehicles and/or low hovering choppers. Try to go easy on this value, and suit the amount to the type of surface you're defining! (something I sometimes forget myself!)

soundEnviron=

This parameter specifies the "Ambient Environment Sounds" to be used with this Surface. I'm not sure how many default ones there are - "dirt", "rock", "grass", "gravel", "drygrass", "sand", "forest", "road" and "concrete_ext" are all valid. You could also define your own Ambient Environment Sounds! -that's outwith the current scope of this guide however... maybe in a future edition.

soundHit=

This is another sound parameter specific to each Surface. This one defines the "footstep" sound. Again I'm not sure of the entire list of default sounds available... "hard_ground", "soft_ground" and "concrete" are a few you'll see in use in these terrain configs.

You can also define custom "footstep sounds" if you like - I *will* be giving an example of this in the future - to accompany the "Etah Plateau" terrain, which really benefits from those distinctive "scrunchy snow step" sounds!

character=

This last parameter refers to the chosen "Surface Character" or "Clutter Mix" for this Surface. If you're defining a Surface like "beach sand" or "rock" you may choose to have no clutter at all - in that case you can simply use "Empty" as the value for this parameter. Otherwise, *it should be the classname of an appropriate Clutter Mix*. We'll be defining these clutter mixes below just shortly...

```
class AfghanValleyslopeSurface : Default <----- replace "AfghanValley" with "KhargharValley"
{
    files = "av_slope_*"; <----- replace "av" with "kv"
    rough = 0.1;
    dust = 0.1;
    soundEnviron = "dirt";
    soundHit = "hard_ground";
    character = "AV_slopeClutter"; <----- replace "AV" with "KV"
};
class AfghanValleyrockSurface : Default <----- replace "AfghanValley" with "KhargharValley"
{
    files = "av_rock_*"; <----- replace "av" with "kv"
    rough = 0.3;
    dust = 0.05;
    soundEnviron = "rock";
    soundHit = "hard_ground";
    character = "Empty"; <----- notice there's no "Clutter Mix" assigned to the "Rock" surface!
};
```

Moving on now to the *Surface Character or Clutter Mix* definitions - **replace all the following "AV" tags with the usual "KV" tag!**

```
class CfgSurfaceCharacters
{
    -----
    class AV_sandClutter <----- the "Clutter Mix" Classname - retag!
    {
        probability[] = {0.06,0.05,0.005,0.005}; <----- FOUR values here - totalling Less Than 1.0!
```

```
names[] = {"AV_BrushSoft", "AV_BrushHard", "AV_PlantsWhite", "AV_WeedThistle"}; <----- FOUR Clutter  
Types - proportions of each dictated by the values above.  
};
```

Pretty straightforward there, huh? Firstly we define a Name for this "Mix" - "KV_SandClutter" (you *did* rename, didn't you?), then we dictate some relative proportions - *adding up to 1.0 - or less!*, then in the final line we list our choice of the previously-defined Clutter Types for this Clutter Mix - four of them... one for each of the random "probability" values in the line above. This is a one-to-one correspondence, so the First "probability" value relates to the First clutter type "name", the Second to the Second, and so on...

probability[]=

Each "Surface Square" in-game (usually about 2x2 meters or so) which has a Surface with a Clutter Mix defined will feature a random mix of clutter models. They're already randomly *sized*, thanks to the parameters we set back in *cfgClutter.hpp*, and at this stage we can dictate relative probabilities for the *proportions* of each clutter type on a per square basis. This helps to avoid a regular or patterned effect.

names[]=

This line lists the individual *Clutter Types* used in this "Mix" - one for each of the probability values defined in the line above.

```
class AV_slopeClutter <----- replace "AV" with "KV"  
{  
    probability[] = {0.1,0.05,0.01,0.01,0.01};  
    names[] = {"AV_BrushHard", "AV_BrushSoft", "AV_PlantsWhite", "AV_WeedThistle", "AV_PlantsViolet"};  
};  
};
```

... and that brings us to the end of *cfgSurfaces.hpp*! It was all about surfaces - predictably...

We've left the most important file 'till last... this is the Master File - the one that brings all aspects of the terrain together and makes it work in-game. The "*config.cpp*"!

THE "CONFIG.CPP"

When you're working on your terrain in Visitor, and viewing it in Buldozer, the ***config.cpp*** isn't actually in use! At that development stage it's enough to have a limited "view" of your terrain-in-progress. You won't see Clutter, for example, or hear Ambient Sounds, or see Ambient Wildlife, like birds, rabbits, etc. When you come to preparing your terrain for use in-game however, it has to "stand alone" and know everything about itself that the game might need... From in-game map grids, to latitude and longitude, airport locations, which ambient sounds to use, which ambient wildlife, which lighting... all of these parameters need to be specified for your terrain - in its *config.cpp*.

For a sophisticated project you might want to do exactly that - specify *everything* - tailor every ambient parameter to your liking. You end up with a very big *config.cpp*! - take a look at Takistan's sometime! However - you can take advantage of "*inheritance*" and keep your configs fairly short and tidy - like the one below. Inheritance can seem confusing at first, and this guide *isn't* intended to be a programming lesson, but the way it works for terrains is particularly simple. Here's a very brief and simplistic explanation, then we'll take a look at inheritance actually "in action" in the *config.cpp* itself - that'll probably be much clearer and more obvious than the explanation!... still... here goes...

Basically, Arma2 (CO) comes with some inbuilt terrains - Utes, Chernarus, Takistan, etc, etc... Arma2 "*understands*" these terrains already, and it keeps all the information required for these terrains on a couple of different "*levels*".

At the "**CA World**" level, it holds information common to all terrains - a "*superconfig*" if you will... this allows the game engine to understand basic terrain things... all terrains will have "land" of some kind, they may potentially have "sea"... very basic low-level stuff we needn't concern ourselves with...

At the **next sublevel** are the individual Arma2 terrains themselves... they're all "CA Worlds" - ie: they all use the same basic fundamental info required for any terrain to work at all, but they also have individual *config.cpp*'s. These individual configs provide additional information to the game engine about that *specific* terrain.

For example, when you load Takistan in the editor, the game engine reads the Takistan config.cpp... (in plain english, that would go something like this)...

"I'm a CA World" (so all the basic defined stuff for them applies)

"however, I have individual info specific to me" (oh, ok... tell me)

"My name is Takistan" (roger that)

"I'm located at Lat/Long coordinates 'etc, etc'"

"My Ambient wildlife is like this..."

"My Sky and Lighting looks like this....."

And on, and on... until every specific aspect of Takistan has been detailed in its config.cpp...

Now here's the sneaky part... We're currently working with this "Afghan Valley" terrain... it's an *Afghan-style* terrain... Takistan is an *Afghan-style* terrain - *the game engine already has the full definition for Takistan!*

So for our Afghan Valley config.cpp we could just say (again, in plain english)...

*"This is Afghan Valley - Son of Takistan! - assume it to be identical to Takistan in **all** ways, **except** those I specify below"...*

Effectively, we'll *inherit* all parameters from Takistan - so we don't need to bother specifying them! For the parameters that obviously *do* need to be different - the Name, the Lat/Long coords, the menu picture, etc, etc... we'll specify them ourselves, and - by doing so - they'll be different from Takistan, and appropriate for our specific terrain. Anything we *don't* specify - we get Takistan's ready-defined parameters - whatever they may be.

You can "*inherit*" from any of the "built-in" terrains in this way, and save yourself a lot of unnecessary typing. As you'll see in some of the other microterrains in this series, they *inherit* from the most appropriate BIS terrain (Takistan or Utes mostly), so their config.cpp's are all fairly short.

Let's take a look at the config.cpp from *Afghan Valley*. As before I'll highlight the bits you need to change for our example exercise in **red**, and I'll provide additional relevant comments and explanations in **green** below each parameter. The config naturally falls into separate sections so I'll highlight these and we'll look at them in turn. Between the comments and the sectioning, this means that what is actually a pretty short config will be strung out over the next few pages! so - since you're editing it anyway - have your *Afghan Valley* config.cpp open in notepad side-by-side with this guide... you'll be able to do your renaming and repathing to *Kharghar Valley* as we go along, whilst also seeing the config "as a whole"...

Don't panic! This is the most important config of all, but it's no more complex than any of the others we've looked at already... Remember! As you go along, in **all** cases - **replace "Bush" with "YourTag", "AfghanValley" with "KhargharValley", "AV" or "av" with "KV" or "kv"** - just the same as before...

P:\YourTag\KhargharValley\config.cpp

```
#define _ARMA_  
  
//Class config.bin{  
class CfgPatches  
{  
class AfghanValley <----- the "classname" for this terrain  
{  
units[] = {"AfghanValley"}; <----- the "unitname" - must be the same as the "classname"  
weapons[] = {};  
requiredVersion = 1.0;  
requiredAddons[] = {"Takistan"}; <----- we'll be "inheriting" from Takistan, so it must be available  
version = "30/07/2011";  
fileName = "AfghanValley.pbo"; <----- the "filename" - must be the same as "classname" too!  
author = "Bushlurker"; <----- it's your terrain now! - put your name here!  
mail = "terrain@bushlurker.com"; <----- and a contact email - or just leave blank  
};  
};
```

A fairly understandable intro section there... Just declaring the *classname*, *unitname* and *filename* - **all of which must be the same to avoid the "Save Game Bug"!**

We declared "Takistan" to be a "*required addon*" because we want to "inherit" from it later. Since Takistan is part of Operation Arrowhead, this obviously means we just made our addon dependant on Arrowhead. If we were intending to inherit the majority of our parameters from Utes or Chernarus, we'd make them a *required addon* here instead of Takistan...

Make sure you've done your renaming to "KhargharValley" and move on...

```
class CfgWorlds <----- announces that the whole following section is part of a "cfgWorlds" definition  
{  
class CAWorld;  
class Takistan: CAWorld  
{  
class Grid;  
class DefaultClutter;  
};
```

Nothing to change in this section, and the only thing you really need to understand about this section at this stage is that it's part of the preparation for "inheriting" from *Takistan*... if you were planning to inherit from *Utes*, you'd replace "*Takistan*" with "*Utes*" here, or "*Chernarus*" - no other changes. Later terrains in this series *do* inherit from Utes, so you can easily compare the configs and see what's different...

```
class AfghanValley: Takistan <----- this line finishes the "inheritance" - it says "Afghan Valley - Son of Takistan"  
{  
cutscenes[] = {"AV_Intro1"}; <----- the "classname" of the cutscene - we'll define this later, but - see below!  
description = "Afghan Valley, Ghor Province."; <----- The Terrain Name, as it appears in the in-game menus  
worldName = "Bush\AfghanValley\AfghanValley.wrp"; <----- must be the same as "classname", etc  
pictureShot = "Bush\AfghanValley\data\av_menu_ca.paa"; <----- the in-game menu picture
```

Repath and rename the lines above - you know the correct paths by now, surely?... oh.. ok then... Change...

"Bush\AfghanValley\AfghanValley.wrp" to "YourTag\KhargharValley\KhargharValley.wrp".

"Bush\AfghanValley\data\av_menu_ca.paa" to "YourTag\KhargharValley\data\kv_menu_ca.paa"

Notice that "*worldName*" parameter... as before, **it must be the same as classname, unitname and filename.** We don't actually have an "*AfghanValley.wrp*" file... as yet - **repath and rename anyway!** This is the "*worldfile*" and we'll make and export it from Visitor soon!

Now... read this next paragraph carefully!

The "**cutscenes[]**=" parameter defines which *cutscene(s)* are associated with this terrain. We'll actually define the cutscene's *classname* later in the config, but another important point to realise is that this parameter also acts as an "On/Off" switch! If we define this parameter as "empty" (**cutscenes[]={};**), then instead of an actual cutscene, we'll get a static view of the terrain from the "*seagull position*" (I think!) - we'll define that later too.

Cutscenes get a little *Appendix* section of their own later in this guide, where, as a very simple example, I'll discuss the use of *Tupolov's Map Intro Script*. Once you've completed the main part of this guide you can move on to the "Advanced" Appendix section(s), the first of which will be - cutscenes! When you follow that section you'll be given instructions to come back to this place in the config.cpp and "re-activate" the Cutscene, but for now, for this first run through with the terrain, we'll *deactivate* it...

Change - cutscenes[] = {"AV_Intro1"}; to cutscenes[] = {};

```
startTime = "07:00";
startDate = "05/03/2001";
startWeather = 0.2;
startFog = 0.0;
forecastWeather = 0.6;
forecastFog = 0.0;
```

These parameters need no explanation other than to say that they're the "defaults" for your terrain... when people load your terrain into the editor for a quick look, these are the parameters which are used... You might want to fool around with these - pick some nice time of day or year, to - literally - show your terrain in its best light...

```
centerPosition[] = {2560,2560,500};
seagullPos[] = {2560,2560,500};
```

For now - **make sure these two parameters are set to the center of your terrain - in meters!** This terrain is 1024 pixels wide, and each pixel represents 5 meters, so it's 5120 x 5120 meters. The "*centrePosition*" will therefore be at the 2560x2560 mark... The third parameter - "500" - indicates "height". *If you adapt this config for a different sized terrain, remember to change these.* Right now, however - they're OK as they are. Notice in this section the "seagullPos" I mentioned earlier. This position *doesn't* have to be in the exact centre of your map. I *believe* it controls, possibly amongst other things, the *default viewpoint* when there is no cutscene enabled, as discussed above.

```
longitude = 65; <---- values in Degrees, positive values mean "East"
latitude = -34; <---- values in Degrees, positive values mean "South"
elevationOffset = 2000; <---- values in meters
```

Whether you're making a *Geospecific*, or a *Geotypical* terrain, you should decide roughly where in the actual Real World it is, and find the approximate Lat/Long coordinates - to the nearest Degree will do... Then enter them here. This can be important since - impressively - the Game Engine accurately varies lighting, lengths of day & night, positions of stars, etc to reflect these realworld coordinates.

A prime example of this is the Antarctic terrain, *Etah Plateau*... set to accurate Antarctic coordinates, it really does become a Land of the Midnight Sun at certain times of year.

No need to change any parameters here for the purposes of this walkthrough though...

That final parameter - "**elevationOffset**=", does exactly as the name suggests - it will add (or subtract) an additional offset - in meters - to your entire landscape.

```
midDetailTexture = "Bush\\AfghanValley\\data\\av_middle_mco.paa"; <---- the "Middle MCO" we met earlier!
minTreesInForestSquare = 2; <---- controls green "forested areas" on the in-game map
minRocksInRockSquare = 2; <---- controls the "rocks" symbol on the in-game map
```

More self-explanatory stuff there!

The "**midDetailTexture**=" line points to that "*overall Middle MCO*" texture we discussed earlier - it lives in the "*YourTag\KhargharValley\Data*" folder - that's where we saw it, when we renamed it to "*kv_middle_mco.paa*", remember? - **better repath that line appropriately then!**

The remaining two lines - "*minTrees*" and "*minRocks*" control the in-game map symbology, as I already mentioned. Values are "*per terrain cell*" - I *think!* Areas with trees which score under this value will have individual little green "tree" circles.

```
ilsPosition[] = {1024,1024};
ilsDirection[] = {0.5075,0.08,-0.8616};
ilsTaxiIn[] = {};
ilsTaxiOff[] = {};
drawTaxiway = 0;
class SecondaryAirports {};
```

Since we don't have an Airport on this terrain - why do we need parameters for one?. Well, it's down to that "*inheritance*" thing again... Remember how it works? - we inherit *everything* Takistan has - ***everything!*** - *unless we specify differently!* Takistan has Airports, unless we specify different ones - *or none at all* - we'll get Takistan's ones! - totally out of place!

Note that sneaky other option... *specify "none at all"* - that's exactly what all these "*ils*" parameters do!

So - once again, **nothing to change here!** These values will effectively prevent us inheriting any actual values from Takistan, so we'll have no visible runways or taxiways on our map which don't belong there.

When you get to the stage where you're planning an actual airport of your own, there's an excellent tutorial from *Marksman*.

```
class OutsideTerrain
{
    satellite = "Bush\AfghanValley\Data\av_satout_co.paa"; <----- the "satout" satellite texture
    enableTerrainSynth = 1; <----- on/off switch - 0 is "off", 1 is "on"
    class Layers
    {
        class Layer0
        {
            nopx = "Bush\AfghanValley\Data\av_sand_nopx.paa";
            texture = "Bush\AfghanValley\Data\av_sand_co.paa";
        };
    };
};
```

This section controls the *BIS Procedural Terrain* generation for areas beyond the borders of your actual terrain. For actual island terrains, and often for terrains with at least *some* coastline, you may prefer to keep *Outside Terrain* switched off... results are unpredictable and you may end up with unwanted procedurally-generated "islands". For fully landlocked terrains, such as our *Afghan Valley* example, Outside Terrain is usually used, and often works quite well!

The first parameter - "**satellite**=" - is a simple path - pointing to the "satout" texture we learned a little about when we were renaming textures in the "Data" folder, remember? This is that little "mini" satellite texture used by the Outside Terrain, being called on, at that location, by this config parameter. We know that file is there, we renamed it! So now - **repath this parameter to "YourTag\KhargharValley\Data\kv_satout_co.paa"**.

The next parameter - "**enableTerrainSynth**=" is self-explanatory... we want it on so we have "1".

As I mentioned briefly before, the Outside Terrain basically works in a similar way to your Main Terrain, but at a much simpler level... *Afghan Valley* has a fairly large and detailed *Satellite Layer* - the "*Sat_lco*" file we looked at earlier... this is "draped" over the heightmap to provide the overall "*distance texture*" you see, when flying over the valley, for example. As I explained before - the Outside Terrain uses the "*satout*" texture - infinitely tiled - to provide a basic and simplified "Satellite Layer" for the Outside Terrain - so when you're flying over that, it looks at least OK, and matches in reasonably well with the "real" terrain - hopefully.

But what happens if you're flying over Outside Terrain and you decide to *land*? What do you see *close-up* on the ground? Once again, Outside Terrain uses a single "simplified" version of the more complex "Ground Surfaces" we've encountered before. Look again at the definition above, you can see two other parameters pointing to texture files. The class "Layer0" section defines the Outside Terrain's single basic Ground Surface. Let's think about this for a moment... What we're defining here is a single, overall, close-up "look" for the Outside Terrain... If your actual landscape is predominately "grassy" you'll already have a set of "grass" textures in use across the main part of your terrain... we want this Outside Terrain to blend reasonably well with the Main Terrain - which is why it's important that the "satout" texture blends well with your overall Satellite Layer "look" - so, we can cunningly re-use our "dominant" Main Terrain ground texture files for the Outside Terrain! Of course, the Outside Terrain ground texture is rendered in a different, less sophisticated way than within the boundaries of our main terrain, but it still looks OK, serves the purpose and, most importantly, helps to keep your Outside Terrain colour tones and overall "look" consistent with your Main Terrain.

Since Afghan Valley - apart from some limited "slope" or "rock" areas, is largely textured with the main "sand" texture set, I've used its "nopx" and the "co" textures for the Outside Terrain - as you can see in the definition above. Of course, to continue with our walkthrough, you should **repath the "nopx=" line above to "YourTag\KhargharValley\Data\kv_sand_nopx.paa"**, and also, **don't forget the "texture=" line, change it to "YourTag\KhargharValley\Data\kv_sand_co.paa"**.

Finally, I should mention that it's perfectly possible to have a "nopx" and a "co" texture set *specifically* for the Outside Terrain if you wish. You'd store them in the "Data" folder along with all the other textures and simply call on them using the parameters above.

```
-----
class Grid: Grid
{
  offsetX = 0;
  offsetY = 5120; <----- size of the terrain - in meters
  class Zoom1
  {
    zoomMax = 0.15;
    format = "XY";
    formatX = "000";
    formatY = "000";
    stepX = 100;
    stepY = -100;
  };
  class Zoom2
  {
    zoomMax = 0.85;
    format = "XY";
    formatX = "00";
    formatY = "00";
    stepX = 1000;
    stepY = -1000;
  };
  class Zoom3
  {
    zoomMax = 1e+030.0;
    format = "XY";
    formatX = "0";
    formatY = "0";
    stepX = 10000;
    stepY = -10000;
  };
};
-----
```

This section, as the class name suggests, controls the in-game "Map Grid".

Map Grids can be a complex, not to mention contentious, matter sometimes. At this early stage, the only thing we need to worry about is conforming to the basic BIS standard, as used on *Takistan*, for example. This complete Grid definition above will do that - as it stands. The only parameter you need to consider - but not *change*, in this case, is at the top - the "**offsetY**=" parameter - this Value should be equal to your overall terrain *width* - in meters. In our case 5120 meters is correct.

```
-----
class Clutter
{
    #include "cfgClutter.hpp"
};
class Names
{
    #include "AfghanValley.hpp"
};

};
};
-----
```

Here at last are two of those "*included .hpp*" files we looked at before, finally being called upon here in the Main Config. As you can see above, we declare that we're talking about the "*Clutter*" class - then don't actually say anything! We said it all back when we prepared our "*cfgClutter.hpp*"! When *BinPbo* binarizes our config later on - along with the rest of our terrain and files, the "*#include*" command here will literally *include* our whole *cfgClutter* file into the Main Config, at this point.

The exact same thing happens in the section below for the "*Names.hpp*". We discussed this, and renamed, this - currently blank - file earlier. Though it's not actually in use for the purposes of this project, it's best to maintain consistency and **rename it to "KhargharValley"** here as well.

```
-----
class CfgWorldList <---- announces that the whole following section is part of a "cfgWorldsList" definition
{
    class AfghanValley{};
};
class CfgMissions
{
    class Cutscenes
    {
        class AV_Intro1
        {
            directory = "Bush\AfghanValley\Data\scenes\intro1.AfghanValley";
        };
    };
};
-----
```

This short "*cfgWorldList*" section serves two purposes...

Firstly, it declares "*AfghanValley*" as a "*world*" to be added to the "*list*"... **We're renaming to "KhargharValley" of course, so rename that line now.**

Secondly there's a section which declares the "**classname**" for our "*cutscene*", then provides a path to its location. Remember the "*cutscenes[] =*" parameter we encountered at the beginning of this config? We renamed from "*AV*" to "*KV*" and declared that "*cutscenes[] = {\"KV_Intro1\"};\"...? Well, this is where we actually define what "*KV_Intro1*" actually is - by providing the path to the mission folder.*

We'll look at cutscenes in a *little* more depth in one of the *Appendix* chapters, meantime - for *almost* the last time, **rename the "class AV_Intro1" above to "class KV_Intro1"**. Notice how we'll rename this section as usual, rather than remove it. We disabled the cutscenes earlier, so this whole definition is redundant - for now, but we'll leave it in here anyway - it'll do no harm and later on, when you come to follow the Cutscenes "Advanced" section, you won't have to replace it... In fact, we'll rename it in advance!

Change that "directory=" parameter - from "Bush\AfghanValley\Data\scenes\intro1.AfghanValley";

to "YourTag\KhargharValley\Data\scenes\intro1.KhargharValley";

```
#include "cfgSurfaces.hpp"
```

Yes - it actually *is* literally the last line of the config - the *"include"* command for that clever *"cfgSurfaces.hpp"* file we worked on earlier... There's not much I really need to say here, and **nothing to be changed**... Our carefully repathed *cfgSurfaces.hpp* will be included here by *BinPbo*, and the *"cfgSurfaces"* and *"cfgSurfaceCharacters"* classes we defined in there will be available in the final *"config.bin"* for our terrain to use in-game.

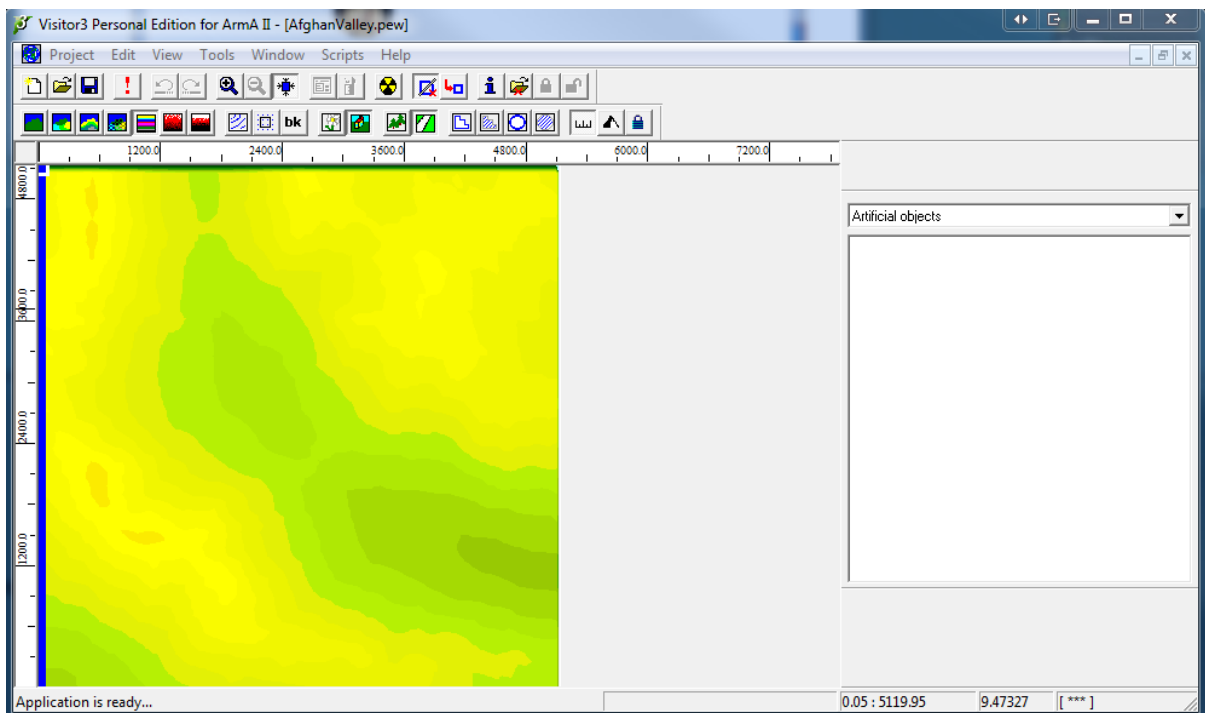
Congratulations if you've made it this far! We've now looked at ***all*** the files and folders required to create a basic terrain. You should by now have at least a vague idea of what all of the files are, what they do and where they usually live! If you've been faithfully repathing and renaming the relevant source files all along as suggested then you've also just made a basic *Terrain Template Structure* for yourself - already pathed within your own *"MyTag"* Namespace... That's going to be handy in the future!

You could make a copy of this whole *"KhargharValley"* folder - rename it to *"MyNewProject"*, or whatever - then just follow the exact same repathing and renaming procedure we've just completed - changing *everything* to your new chosen project name... Once you know what you're doing, and you're confident enough to glance through the configs, spotting errors easily - you can use *"find and replace"* on almost all the files, and create a new Terrain Project Folder pretty quickly!

We haven't quite finished repathing! there's one final path yet to be done, but that one is contained in the Main Project File - the *KhargharValley.pew*. To edit that, we'll need to load it into Visitor...

SECTION 3 - LOADING THE TERRAIN INTO VISITOR

The hard work and endless manual typing stuff is done! From here on in its "sit back with a coffee and click stuff" - *mostly*... Let's get started! Launch Visitor, then **choose "Project\Open"** - navigate to your *"P:\MyTag\KhargharValley\"* main project directory and load *"KhargharValley.pew"*... you should see something like this...

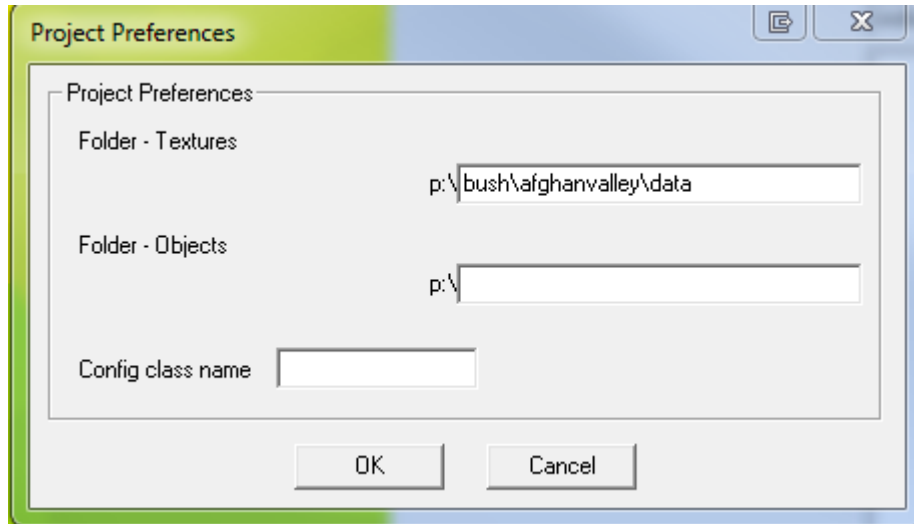


Yours will say (KhargharValley.pew) at the top, of course...

In the main window section you can see the *heightmap* displayed - it's embedded in the Project File ".pew", so no need to re-import it! Before we do anything else, however, there's one last parameter we **must** repath...

3.1 - SET THE "LAYERS FOLDER" PATH

Click on "**Tools\Project Preferences**" and the following window should pop up...



This window should be familiar from *Sgt Ace's Tutorial*, and it's immediately obvious what we have to change here. **Change that "*bush\afghanvalley\data*" path to "*yourtag\khargharvalley\data*" and click "OK".**

This path controls the location where Visitor will create it's "*Layers*" folder when you "*Import Satellite & Mask*". **It's important you always make sure that this path points to the appropriate "*p:\yourtag\yourprojectname\data*" folder.** On import, your Satellite & Mask files will be chopped into sections and converted, and the resulting files stored in the "*Layers*" folder. When you subsequently run Buldozer for the first time after the import, more conversion and file creation will take place... this makes the "*Layers*" folder *Very Big*, which is why it *wasn't* included in the Source Files pack... you can create it yourself! We'll be doing that next...

3.2 - IMPORT SATELLITE AND MASK

Click "**Tools\Import Satellite + Mask...**"

- A selection window will appear, headed "*Select layer configuration file*"... **navigate to inside your "Source" folder, select "*layers.cfg*" and click "Open".**
- Another small popup window will appear, headed "*Rvmat selection*"... **choose to "*save .rvmat files as 'Text'*" and press "OK".**
- Another selection window will appear, headed "*Select Satellite Map*"... **select your "*Sat_lco.png*" file and click "Open".**
- A final selection window will appear, headed "*Select layer mask*"... **select your "*Mask_lco.png*" file and click "Open".**

The "Importing Satellite Data" progress bar will appear and start... progressing... If you try to do something else in another window while this is happening, Visitor sometimes has an unfortunate habit of saying its "Not Responding"... it's actually fine, and it *will* complete the import. Visitor quite often says it's "Not Responding"... usually, it actually means "I'm just not saying anything else until I'm finished"... so *don't panic* if you see this now and then.

3.3 - RUN BULDOZER

Once the Importing Sat & Mask Progress Bar has completed its run and disappeared, **click the "Save" icon (Floppy disk symbol)**! It *wasn't* actually necessary to save at that point, it's just good to start getting into the habit of saving regularly! Now we're ready to launch *Buldozer* for the first time with this project...

Click the "!" icon in Visitor - **your *Buldozer* window should open**... Since this is the first time viewing the terrain after "Importing Satellite & Mask", the files that were created in that *Layers* folder will be further processed. You'll see this happening in a DOS window which will appear and scroll for a while - reporting the files being processed. This procedure *only* happens the first time you view a terrain after (re)importing Sat & Mask files and it might take a while. Don't worry! This processing of your Sat & Mask layers into "tiles" is necessary to make it work in *Buldozer* and in-game.

If all has gone well, the DOS window should eventually disappear, *Buldozer* will ponder things for a moment or two and... you should be done! By default, the "viewing cursor" will be at the extreme Top/Left of the terrain... things don't look too good up here, so **use "Alt+Tab" to switch back to the main Visitor window, point your mouse somewhere in the middle of the terrain and press your Centre Mouse Button, or Mousewheel**. This should move the square cursor to that location... **now you can "Alt+Tab" back to the *Buldozer* window** and you should see your new *Kharghar Valley* terrain!

Congratulations again! Another little milestone reached! That *Layers* folder was far, far too big to include in the Source Files pack, but you've now successfully re-created it for yourself! In the correct location! There's one more absolutely essential file that we need to recreate... we told the "*config.cpp*" where to find it already, so we need to make sure it exists - at that location! It's the "**Worldfile**" (.wrp).

3.4 - EXPORT THE 'WORLDFILE'

You **can only "Export World" when *Buldozer* is running**! So, if you closed that *Buldozer* window, open it again... **Now, "Alt+Tab" back to the Visitor window, click the "Project\Export World" option** - a File Selector window should open. **Navigate to your Main Project Directory** - if the file selector window isn't pointing there already. *Now you need to manually type the filename!* **In the "File Name" slot - type "*KhargharValley.wrp*" and press "Save"**.

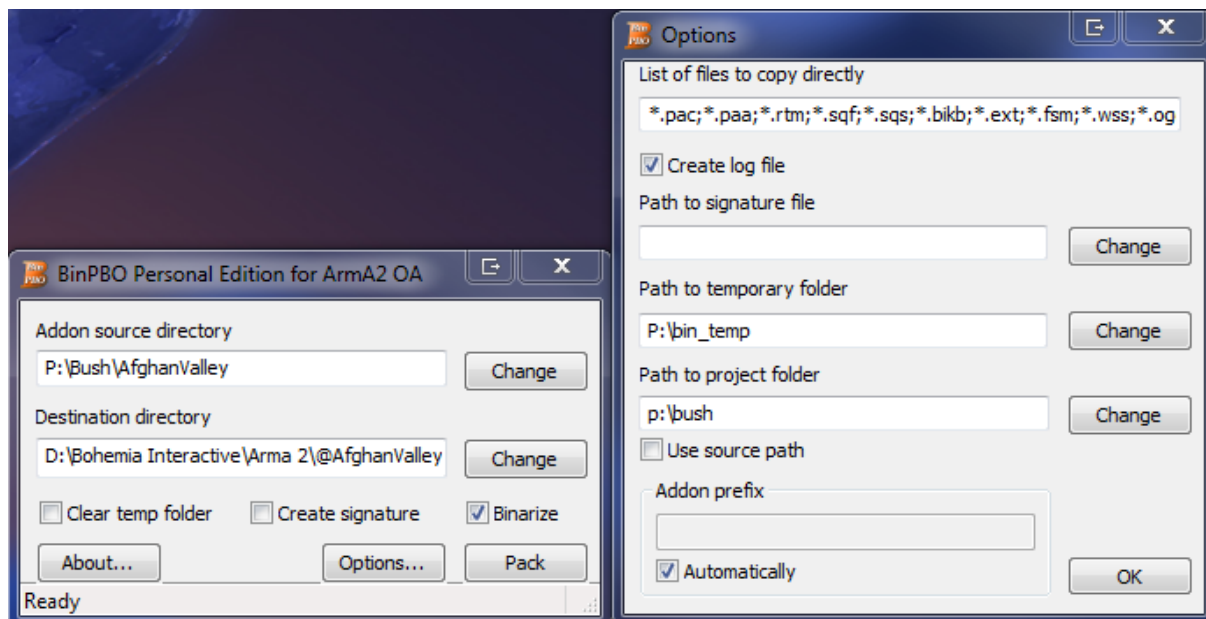
Now you should have that all-important *worldfile* in your *Main Project Directory*, named "*KhargharValley.wrp*" - precisely as we stated earlier in the *config.cpp*. That's the final piece of the puzzle in place! We're ready to "*Binarize*" and put the terrain in-game! We've finished with Visitor for now, so - just to reinforce your *Backup Fetish* - **click that Floppy Disk "Save" icon one more time** - just to be on the safe side, then you can **close both *Buldozer* and Visitor**... Time to navigate to your "*BIS TOOLS\BinPBO Personal Edition*" directory!

SECTION 4 - BINARIZATION

Binarization is the final step in preparing your terrain for use in-game, and it's an *essential* step for several reasons. Firstly, and most importantly, it improves in-game performance. Secondly, it "protects" your work - the "Source" file or .pew *isn't* included in the final binarized output, the *config.cpp* is assembled and turned into a "locked" *config.bin*, and that, technically at least, says "*hands off without permission*". Thirdly, binarizing is a handy way of "error checking" your work... problems with *configs*, etc will show up as errors at the binarizing stage, plus, a "log" file will be created during the process which is another handy source of information about potential flaws in your creation.

4.1 - SETTING UP BINPBO

Since you've done *Sgt Ace's Tutorial* (haven't you?), you've used *BinPBO* before, however, let's set it up to binarize our new *Kharghar Valley* terrain, and look at the options and settings in a bit more detail... Launch BinPBO and you should see something like this...



... well, the smaller left-hand panel for now, anyway...

BINPBO - MAIN WINDOW

- The first parameter is labelled "**Addon Source Directory**"... think of this as "**Main Project Directory**". As you can see from the left-hand pic above - that's exactly where this path points to. Click the "**Change**" button - it'll give you a directory navigating window - use this to point to our *Kharghar Valley* main project directory - it should now read "**P:\YourTag\KhargharValley**".
- The second parameter - "**Destination directory**" - is exactly as the name suggests... the final destination for our completed and binarized "*KhargharValley.pbo*". Technically, this path can point anywhere you like - to the *Desktop*, for example - or any other convenient location. However, since in order to actually try out our final terrain in-game, we'll need to launch the game with a "*modfolder*" containing our terrain addon, we might as well make one now! If you've been fooling around with *Arma2* for any length of time you should know how "*modfolders*" work, and how to make them - if by

any chance you *don't*, you should read the [Armaholic Guide to Modfolders](#) immediately! Once you've done that - **navigate to your "Main Arma2 Directory", create a folder called "@KhargharValley", then create another folder inside that called "Addons"**. Now we have somewhere to send our addon to when it's binarized! **Back in BinPBO - click the "Change" button and navigate to your new "@KhargharValley\Addons" modfolder...** Your path may vary, depending on your game installation, but, for example, in my case the new path would read... "D:\Bohemia Interactive\Arma 2\@KhargharValley\Addons".

- The "**Clear temp folder**" checkbox is also pretty self-explanatory... When you binarize for the first time, it may take quite a while for all the files in that *Layers* folder, plus everything else to be processed. Later, you might have added a village or two and want to binarize that version to check it out in-game... All that's really changed is that few buildings being added - you can leave "**Clear temp folder**" unticked - *binarizing will be pretty fast*, since it will only process the files that have actually changed! If you imported a completely new *Sat & Mask* or something, however - you might want to "flush" your "**Temp folder**" of the old stuff, so you could tick this box and do a completely clean, fresh binarize... you get the idea... usually you can **leave this unticked**.
- The "**Create signatures**" checkbox is - obviously - part of the "*bisign*" signature-creation process... This will be covered in one of the "Advanced" *Appendix* sections... eventually... for now, just **leave this unticked** as well...
- The "**Binarize**" checkbox should be **ticked** - of course... that's why we're here! Enough said!

Now there's a few *Additional Options* which must be set, so - **click the "Options" button** - a new "*options*" window should pop up, just like the right-hand side of the pic above. Let's look at these parameters now!

BINPBO - 'OPTIONS' WINDOW

- The "**List of files to copy directly**" parameter specifies which "*filetypes*" we **don't** want BinPBO to attempt to binarize. A detailed discussion of the whys and wherefores of each filetype is beyond the scope of this simple Guide. However, we're up against some slightly *dodgy BIS default parameters* here! In *Sgt Aces Tutorial* you were instructed to remove "**.wrp*" from this list, remember? When you install BinPBO at first, "**.wrp*" is on the list of files to copy directly! *That's Bad!* We *want* it to be binarized! **So if "**.wrp*" is still on your list - remove it!** Conversely, there are a couple of filetypes we need to add to this list! It's not important right now, since we disabled cutscenes for this practice walkthrough, but later on, when you add a cutscene, it'll contain a "*mission.sqm*" file! There's also often a "*description.ext*" file! - **Neither "**.sqm*" nor "**.ext*" are on this list by default! Add them now!** Otherwise they'll "disappear" during the binarization process and the *Intro Cutscene* will fail. When you're done you can **check your complete list against mine below**. If they match then you're OK and you shouldn't have to worry about this parameter again, unless you reinstall the tools for some reason...

.pac;.paa;*.rtm;*.sqf;*.sqx;*.bikb;*.ext;*.fsm;*.wss;*.ogg;*.wav;*.fxy;*.csv;*.html;*.lip;*.txt;*.sqm;*.bisurf

- The "**Create logfile**" checkbox... creates a logfile! That can be handy for chasing errors so just **leave this option ticked at all times**.
- The "**Path to signature file**" parameter can be left blank for now. We'll look at Signature Creation in a little more depth in one of the *Appendix* sections sometime later.
- Once again, the "**Path to temporary folder**" parameter was covered in *Sgt Aces Tutorial*. You should already have a "**P:\bin_temp**" folder... if you haven't - make one! Then **use the "Change" button to point to it** - just as in the illustration above. The "temp folder" is just a folder that BinPBO uses during the binarization process...
- The "**Path to project folder**" parameter means **exactly** what it says - it's the path **to** your project folder - **not** the project folder itself, but **where it's located**! Our *KhargharValley* project folder is located in the "**P:\YourTag**" folder - so that's where this path should point! **Use the "Change" button to navigate to your actual "Tag Folder"** so this path is correct.
- We've just specified the *path to our project folder*, so we must **leave the "Use source path" checkbox unticked**.
- "**Addon prefix**" is another parameter we'll never use in a terrain-building context, so just **leave that blank too**.
- And the final checkbox - "**Automatically**" should **always be left ticked**.

That completes our settings for all these additional Options, so you can now **click "OK"** and that window will disappear... At last... We're *ready*!...

4.2 - BINARIZE!

We're back in the main BinPBO window now so... **click the "Pack" button!**

This is the first time you've binarized this terrain so it may take some time... go grab a coffee and a biscuit - you deserve it! You'll see BinPBO makes at least *some* attempt to keep you informed of what it's up to on its little bottom line info section... Assuming you've made no ridiculous mistakes anywhere along the line, eventually, this will read "**Ready**" - and it is!

Launch your game with the "**@KhargharValley**" modfolder enabled and head for the Editor... You should see "*Kharghar Valley*" in the list of terrains!

Congratulations! You've made it to the end of the Guide, and you should have a completely working copy of the new terrain - "*Kharghar Valley*" - structured and pathed to your own personal *Tag Folder*!

For the moment... this Guide ends here! The preceding *Main Sections* have covered its principal purpose - a detailed walkthrough of the basic files, structure and procedure for making a simple terrain. The accompanying *Afghan Valley* source files served as a practical example for us to work through. Job done! Except, it isn't of course... you've only just begun! All the real work of building your own World is yet to come... An original Heightmap, a matching Sat & Mask, Roads, Forests, Settlements... you have a lot of Forum Reading to do! I *won't* be discussing that stuff - except forests... and heightmaps... a bit... maybe. However, there *are* other "*Microterrains*" in this series, and each of these has some little "advanced" technical aspect over and above the "basics" we've already learned. As I release each of the subsequent terrain source files in the series, I'll update this guide with an appropriate "*Appendix*" chapter, which will discuss some little advanced feature specific to that particular terrain. I'll also cover some other non-specific technical stuff like Signatures, etc...

'ADVANCED' APPENDICES

Subsequent versions of this Guide will feature additional content here... such as...

- "cfgMaterials.hpp" files and "sea colours"
- "Custom "terrainHit" footstep sounds
- Signatures and Signing Addons
- Breaking the 4 colours Mask limit
- Basic WorldTools Forest Generator usage
- ... and probably other little practical or technical stuff as well

... all that to come in future Editions, so keep an eye on the FORUM THREAD for updates!

CREDITS & THANKS

Many people have contributed directly and indirectly to this guide - from the Old Pros who answered my own dumb beginners questions back when I was starting out, to more recent and direct contributions, in the form of custom parameters, files and scripts contributed specifically for this series... Without your early encouragement and ongoing support I'd have missed out on all the truly unique and creative *fun* making your own World from scratch can be!

Gentlemen... I thank you all...

- Opteryx
- OldBear
- IceBreakr
- Gnat
- Planck
- Beton
- Atsche
- Alliex
- Shezan74
- Tupolov
- Prowler.Wolf
- Mondkalb
- Q / Kju / PvPscene

