

# **Blitzkrieg**

# **“HOW TO”**

# **Tutorial**

**CREATED BY:**

**MAJOR PAIN**

**IN ASSOCIATION WITH:**

**PAINFULLY SOFT USA**

**COPYRIGHT APRIL 2009**

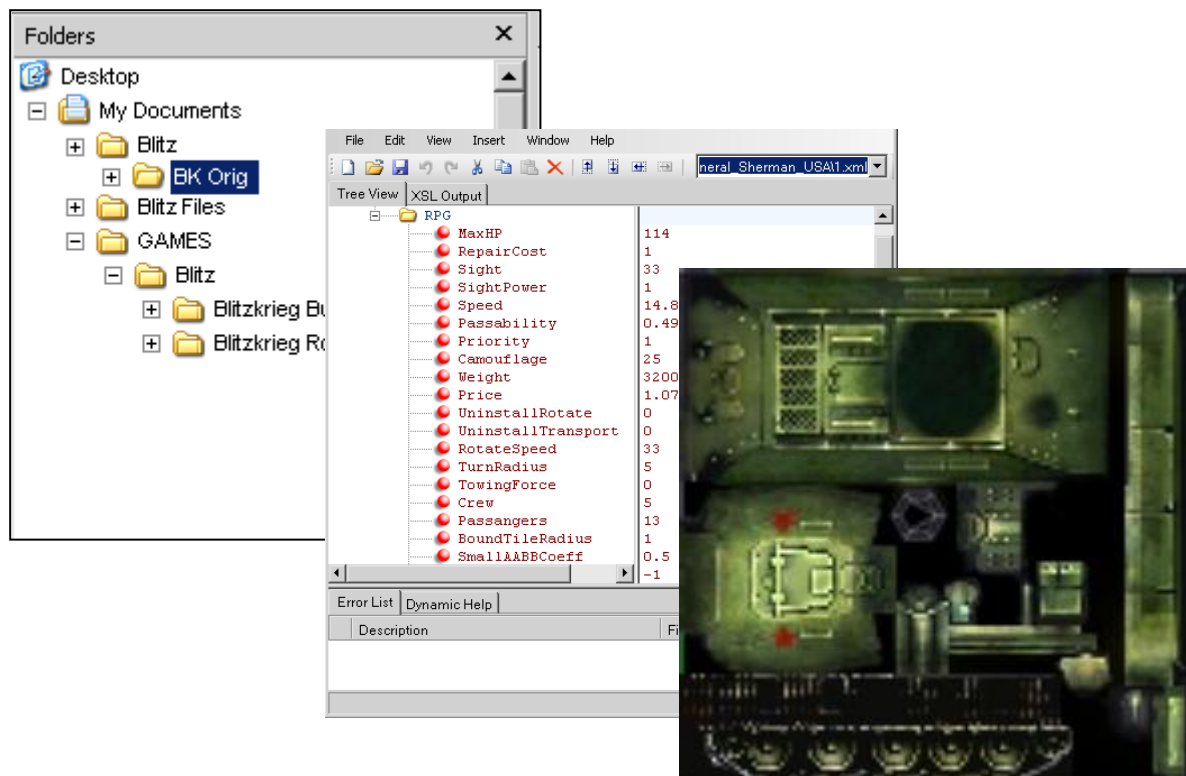
# Blitzkrieg "HOW TO" Tutorial

Much has been written and contributed over the years concerning the modification of Blitzkrieg units or models. The goal of this tutorial is to serve as a guide to access the various files that are used by the game. I will attempt to show you ways that will allow you to manipulate the game folders and files and simplify the methods used to modify the skin files, or command functions that define each object.



While this will not be the final authority, it is intended to help you gain the very basic knowledge to help you become familiar with the files and overcome the fear of the unknown.

If you learn anything from this Tutorial, then the mission has been successful.



MAJOR PAIN  
COPYRIGHT APRIL 2009

**THIS DOCUMENT MAY NOT BE DISTRIBUTED WITHOUT PERMISSION FROM AUTHOR  
AND MAY BE DISTRIBUTED ONLY BY THE AUTHOR OR BLITZKRIEG HEADQUARTERS**

# BLITZKRIEG - HOW TO

## DESCRIPTION AND FUNCTION OF GAME FILES

### BEFORE YOU START

#### PART 1

.PAK FILES - The Primary Game Files

How to Access

How to Modify



#### PART 2

.DDS Files - SKIN FILES

How to Access

How to Modify

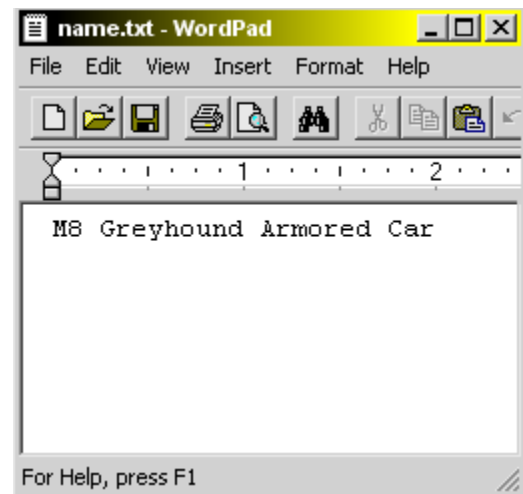
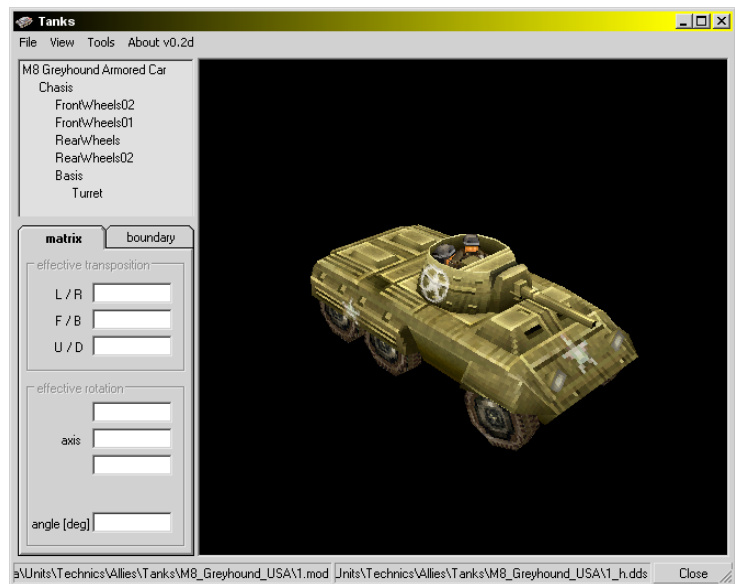
#### PART 3

.XML Files - UNIT DATA FILES

How to Access

How to Modify

## OBJECT COMMAND FUNCTIONS



## DESCRIPTION AND FUNCTION OF GAME FILES

Blitzkrieg uses various files which define every aspect of the game. This includes the unit models, flora/trees, landscape, buildings and more. The function of each file type is pretty straight forward.

**.pak files** / These files contain the various object/unit/model files. The .pak extension is in reality a zip file. This type file can be easily extracted, unpacked, and repacked. The use of this file allows the Game to take advantage of the compressed file structure, which saves disk space.

**.xml files** / Each OBJECT has a file named 1.xml. This file can be edited easily in any text editor, but the file structure must be obeyed for it to perform correctly. This file stores the OBJECT parameters which define the Hit points, sight, speed, and other attributes, as well as the type weapons/guns used by the OBJECT (if any). It also defines armor characteristics and other vital data that are unique to a specific OBJECT. Every OBJECT (model, unit, etc) requires a 1.xml file.

**.mod files** / This file is the particular OBJECT such as a tank or truck. Most OBJECTS are comprised of many pieces which are nothing more than a 3d polygon model. But it is not that simple. This tutorial will not explore this file type in depth, due to the complexity of the subject. It is enough to state that the software and hardware requirements in terms of cost will prevent most gamers from acquiring the capability to modify this file type. You will see these files listed as 1.mod, 2.mod, and 3.mod. Each of these serves a different function pertaining to the OBJECT. There will be continued discussion later on these functions.

**.dds files** / This file is the actual skin or map that is specific to each OBJECT 1.mod file. This type file can be modified by Photoshop or any graphic software program that has the required plug-in file.

**.txt files** / This type file is used to by the game to describe the OBJECT within the game to the player. The name.txt is a simple file that provides the OBJECTS name. The description.txt is a file that describes the OBJECTS specifications, history, or particulars about the OBJECT.

**.tga files** / A picture of the OBJECT that is used by the game for various purposes, including the map editor and resource editor.

The combinations of these files define each and every part of the game. This Tutorial will attempt to teach you how to modify and manipulate these files, with the exception of the 1.mod files which is beyond the scope of this Tutorial.



## Before you Start...

Before we get started, there are some important programs that we will need. The use of these programs will help you with the very tedious tasks of file handling, copy, moving, deleting, and so on. It is assumed that you will be using some version of Microsoft Windows.

**Quick Note:** I use Windows XP Pro. You may use a different operating system. It is vital that you can run the game in your operating system before you attempt to modify the files. Blitzkrieg will run in variety of operating systems.

**FILE MANAGEMENT:** One of the most important Windows files that I find handy is a program called **Explorer**. This is a file program and should not be confused with Internet Explorer. Some of you may use My Computer, but for what **Explorer** can do, I find it more useful for locating and manipulating files. It is named **explorer.exe** within the Windows Directory. After you locate the file, <right click> on it with your mouse, and then select **Send to > Desktop** (create shortcut). Go to your desktop and confirm that you do have this shortcut. <Right click> on the icon and **Rename** to **Windows Explorer** or to a name of your choosing as long as you can identify it when you need it.

**TEXT/XML EDITORS:** You will need a text editor so you can modify the *1.xml* files and *text* files. I use **WORDPAD**. You can usually locate **WORDPAD** in **START > ALL Programs > Accessories** on your desktop menu. I recommend making a copy on your desktop. After you locate the file, <right click> on it with your mouse, and then select **Send to > Desktop** (create shortcut). You can rename if you wish.

Microsoft also has a very good *xml* editor available, **XML NOTEPAD**, available at:

<http://www.microsoft.com/downloads/details.aspx?familyid=72d6aa49-787d-4118-ba5f-4f30fe913628&displaylang=en>

**ZIP/RAR UTILITIES:** In order to modify the *.pak* files you must have a fully functional version of a *ZIP* program. I use **WINRAR**, **WINZIP** and **ZIPMAGIC**, but just about any will work. There are many shareware versions of *ZIP* available on the internet. Some have time limits, so do not. If you are using a late version of XP or XP Pro, you likely already have a Zip Utility Program.

From time to time you may encounter *.RAR* files. You will need a *.RAR* extraction program. **WINRAR** works fine for this: <http://www.rarlab.com/>

The following site has ZIP/RAR Utility Programs available for review and download.

<http://www.bestfreewaredownload.com/freeware/t-free-7-zip-freeware-wrlnehvb.html>

**IMAGE SOFTWARE:** If you plan to modify OBJECT skins, you will need a graphics program like **Photoshop**. I personally prefer **GIMP** (version 2.2.13). It is freeware and available on-line. You will also need the **GTK** runtime environment and the *.dds* plug-in in order to access and modify *.dds* files. GIMP is available for download at: <http://gimp-win.sourceforge.net/stable.html>.

**IMAGE VIEWERS:** Another program that you will find useful is the TANKS VIEWER v0.2d by Pesmontis. This program allows you to View the game *1.mod* (3d objects) with their *.dds* (skin file). It can be downloaded at: <http://tattooinebase.star-fleet.org/>

A very good IMAGE file that will allow you to view the *.dds* files in EXPLORER Thumbnail View is DDS VIEWER. It is available at: [http://download.nvidia.com/developer/NVTextureSuite/DDS\\_viewer.exe](http://download.nvidia.com/developer/NVTextureSuite/DDS_viewer.exe)

Another Viewer that allows Explorer to display *.tga* files in Thumbnail View is THUMBPLUG TGA, available at: [http://www.greggman.com/downloads/thumbplug\\_tga1.10.exe](http://www.greggman.com/downloads/thumbplug_tga1.10.exe)

I truly recommend that if you plan on doing a lot of BK modifications; keep all of the programs that you will use together on your desktop. This will make them much quicker to locate. And remember, you may be limited by your machine speed, RAM memory, Hard Drive capacity, and operating system, as to how many files you can keep open and work on at one time. You must get in the habit of saving very often since there is nothing so frustrating than losing an hour or two of work if a crash should occur.

As time goes on and you become comfortable with the BK files, other tools and program can be used to further your skills and capabilities.

**NOTE:** To simplify the terms that may be used within this document, the use of Object, unit, and model all refer to the same thing. A tank is a unit, and a unit is an object. Other objects can be trees, buildings, soldiers, rocks, terrain landscape, roads, fences; the list applies to about everything in the game. I will mostly use the term “unit” when discussing things like tanks and trucks and “objects” when discussing terrain and buildings. But you should not become confused if you see a reference within a discussion when any of the terms could apply.

# PART 1

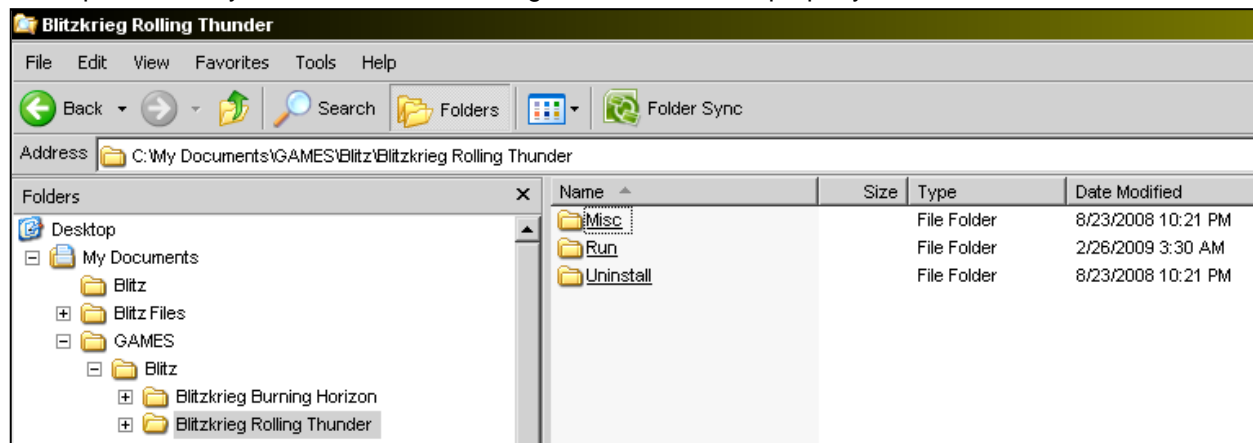
## .PAK FILES - The Primary Game Files

The game depends on the .pak file for everything from instructions to missions, and maps to objects.

**VERY IMPORTANT:** Before beginning to modify any game file, you must make a backup copy of the file that is to be modified. In the event that you make a mistake, you will have the original file intact that can be replaced. Never use the original file for modification. Only use a copy. Sometimes things can happen during the file copy process. Better be safe than sorry.

### Where to locate the .paks

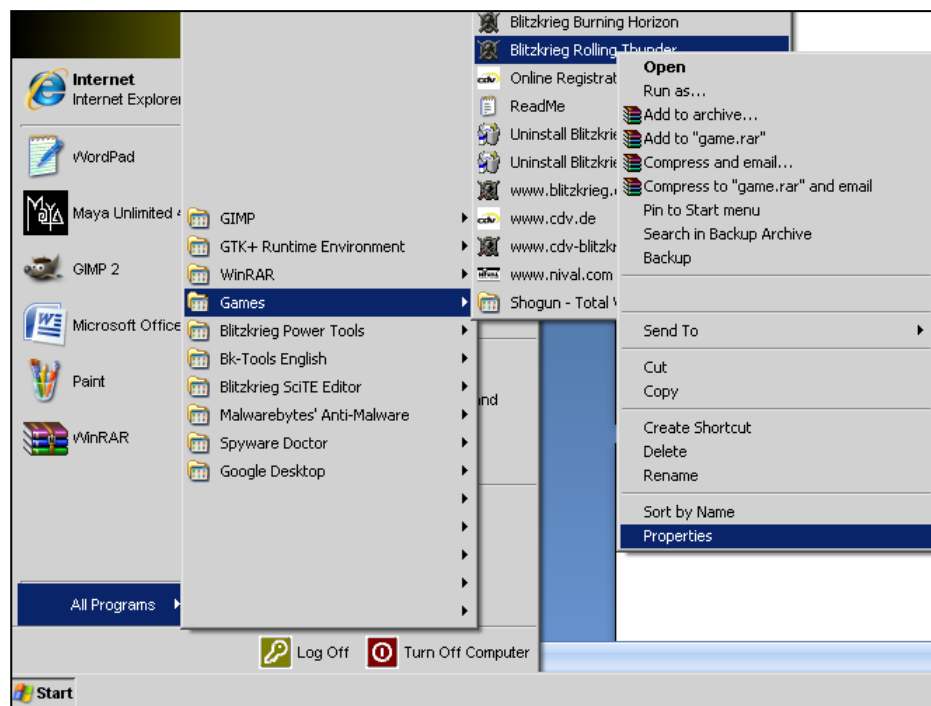
Everyone has a file directory, and everyone's directory is different. The directory pictures that are displayed in this tutorial will be very different than yours. But the game structure will be nearly identical in all respects. If they were not the same, the game would not run properly.



Look at the **Explorer** File Picture above. On the left, the folders show the Folders tree. You should become very familiar with where your Blitzkrieg files are located. Using **Explorer**, you should be able to find the Blitzkrieg Folder.

If you have trouble finding the Blitzkrieg folder, you may be able to locate the folder by going to your **START MENU**.

Click on **START** > Navigate to **Blitzkrieg** and <right click> then Select **PROPERTIES**.

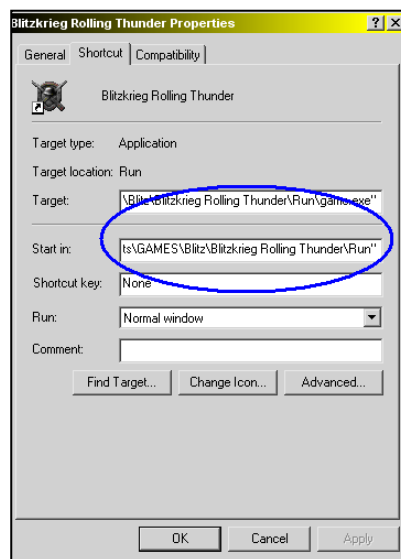


When the **Properties** Menu comes up, look at the Target: or the Start In: box.

You may have to click in the box and scroll to see the entire address, but this is the part of your directory that the Blitzkrieg Folder is located. Write this information down. Be careful not to erase the contents of the box. If you do erase the contents, click on **Cancel** and exit.

Now go back to either **Explorer** or **My Computer** and you should be able to Navigate to the Blitzkrieg Folder. Once you have located the Game folder, you may choose to make a shortcut on the **Desktop**, but this is not necessary if you can find it again easily.

**Important:** Do not move any original game file from the original locations. Depending on the file, you could cause the game to crash if expected or required files cannot be located by the game engine.



Now we are ready to find the **.paks**. They are located in the Blitzkrieg \ **Run \ Data** Folder.

Folders	Name	Size	Type	Date Modified
Desktop	additional_update_2_0.pak	960 KB	PAK File	10/14/2004 8:50 AM
My Documents	AddressBook.xml	1 KB	XML Document	9/21/2004 7:18 PM
Blitz	Anthology BH-RT Texture Patch.pak	71,083 KB	PAK File	8/17/2007 5:43 AM
Blitz Files	AP_AT Mines.pak	14 KB	PAK File	1/5/2008 3:05 AM
GAMES	BH_text_english.pak	1,060 KB	PAK File	10/14/2004 12:05 AM
Blitz	BH_unit_icons.pak	5,612 KB	PAK File	9/27/2004 7:16 PM
Blitzkrieg Burning Horizon	BH_units_names_english.pak	93 KB	PAK File	10/13/2004 11:58 PM
Blitzkrieg Rolling Thunder	credits.pak	4 KB	PAK File	10/14/2004 1:03 PM
Misc	data.pak	173,383 KB	PAK File	10/6/2004 8:50 AM
Run	objects.gdb	553 KB	GDB File	1/12/2009 3:20 AM
data	objects.xml	1,047 KB	XML Document	1/12/2009 3:16 AM
	Passable Water.zip	4 KB	WinRAR ZIP archive	12/7/2008 7:41 AM
	RT_040926_english_loca_1_02.pak	303 KB	PAK File	10/13/2004 8:32 PM
	RT_balanced_silhouettes.pak	99 KB	PAK File	10/2/2004 6:02 PM
	RT_data.pak	68,958 KB	PAK File	10/5/2004 9:40 PM
	RT_maps.pak	7,062 KB	PAK File	10/11/2004 9:17 PM
	RT_szenarios.pak	0 KB	PAK File	10/11/2004 9:42 PM
	RT_units.pak	73,393 KB	PAK File	9/30/2004 9:36 PM
	RT_update_1_3.pak	8,727 KB	PAK File	10/11/2004 11:59 AM
	SP_menu.pak	4 KB	PAK File	10/5/2004 12:58 PM
	tex_high.pak	364,568 KB	PAK File	8/29/2004 7:45 PM
	tex_low.pak	144,369 KB	PAK File	8/29/2004 7:41 PM
	texture.pak	96,334 KB	PAK File	8/29/2004 7:57 PM
	UI_patch.pak	753 KB	PAK File	1/22/2006 8:55 PM
	update_1_3.pak	8,727 KB	PAK File	10/11/2004 11:59 AM

Here is an example of my Blitzkrieg Directory Files, which displays the **.pak** files.

Every **.pak** file has a specific purpose. The primary **.pak** file is the one named [ **data.pak** ]. Look at the **SIZE** of the file, 173,383 KB; (yours may be slightly different). As mentioned previously, a **.pak** file is nothing more than a file that has been compressed, a **ZIP** file. The **data.pak** file contains the files that pertain to the game **OBJECTS** (**1.mod**), the **OBJECT** Parameters data (**1.xml**) and the **icon.tga** which is the picture file used by the map editor that displays the **OBJECT**. But it also contains other files for things like buildings, missions and maps.

At this point I recommend creating a safe work area where we can work on these files. In this Safe Area, we can copy the original files so we can have files that we can modify and work with.

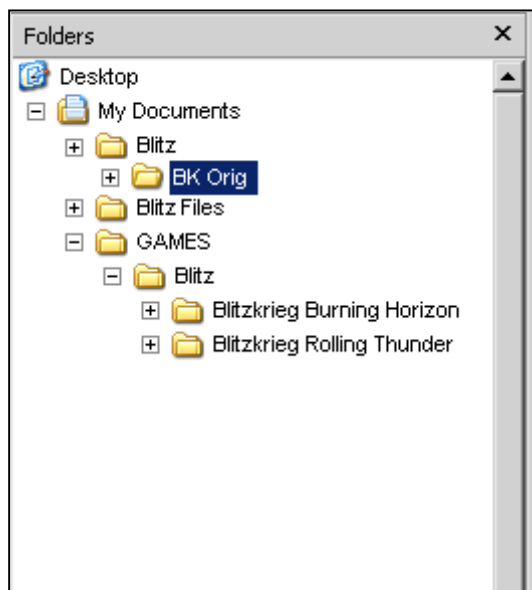
It does not matter where you put this safe work area, except it cannot exist in the **run** or **data** folder of your **Blitzkrieg Folder**.



I prefer to create my folder in an area of it's own. I created a folder called **Blitz**, and then created a sub folder called **Blitz Orig**. Then to make certain that I keep the proper directory structure, I created a **Run** subfolder under **Blitz Orig**, then a **Data** folder under **Run**. Refer to the picture below. The data folder is where we will build our new files.

By locating the safe area close to the Blitzkrieg Folder, you can copy and paste files without having to change the view of the Explorer. This will be important as we begin to build our work files.

I might mention that if you have a slower or older machine, some of the copy/paste functions may take a while to perform.



Another thing that you should make sure of is that you have enough hard drive space to create the work files. In order to truly work on these files, you will need a very large area of the hard drive that you can commit to this project. Once these files are un-zipped, they will occupy more disk space than when they are compressed or zipped. Keep in mind, you are working on copies. When you make a copy, that file alone, still zipped, is taking twice as much space.

Think about this for a moment; the **data.pak** file is 173.3 mb and we are going to copy it to make a work copy. Now we are dedicating almost 350 mb to these files alone. Then we will un-zip or extract one of those copies. So if you have very limited drive space, you should consider freeing some space up, or add another hard drive that you can use for this project. If you have been around computers for very long, you will remember when 350 mb was much larger than most hard drives.

*If you are comfortable with your drive space requirements, we can proceed.*

Go to the original **data.pak**, <right click> on the file and select **Copy**. Then switch to the work area data folder, and **Paste** the **data.pak** file. It should appear under the data folder, and remember, on slower machines it can take a minute.

Now <right click> on **data.pak** and select **Rename**. **Carefully Replace [ pak ] with [ zip ]**. Then press **Enter**. Now your **data.pak** file should be named **data.zip**. Sometimes, you will receive an **Alert Box** asking if you really want to perform this function. Click **Yes**. Your directory should look similar to the picture below.



When you are satisfied that you have the **data.zip** file and the safe work area the way you want it, we can proceed to extracting or unpacking the **data.zip** file.

## UNPACKING/UNZIPPING/EXTRACTING data.zip

The Title of this section says it all. We are going to extract, explode, unzip, unpack, expand the **data.zip**. These terms basically all mean the same thing.

If this is the first time you have worked with a *zip* file, don't worry... remember, we are working with a copy of **data.pak**. There is absolutely no way you can render your Blitzkrieg game to certain death, if you have done everything correctly up to this point.

Take a deep breath, let it out slow and cross your fingers. This will not have any effect on the following operation, but it may help you relax. If this does not work, oh well, back to square one, right?

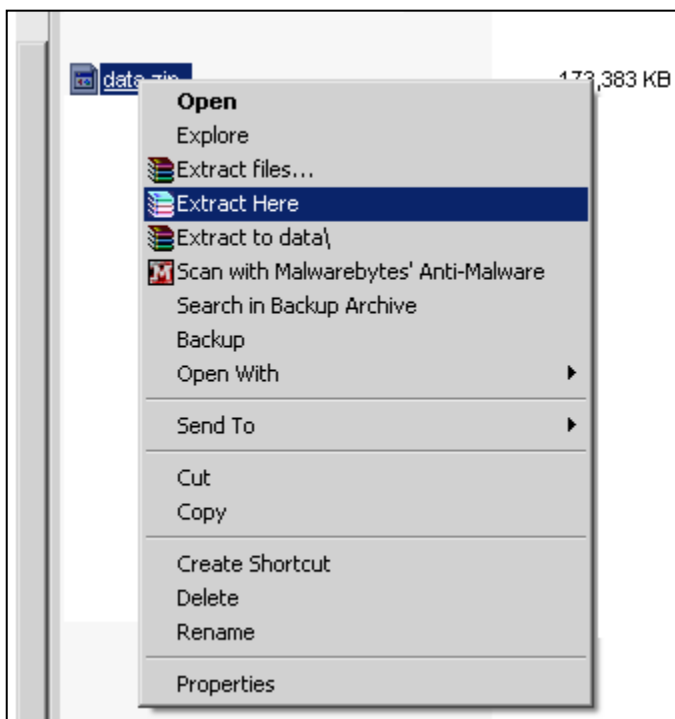
So here we go. <Right Click> on **data.zip**.

If your ZIP program has been installed properly, you should be able to select **EXTRACT HERE**, as shown in the picture to the right. If you have this choice, click on **EXTRACT HERE** now. If this worked, skip down to **Result**.

If you do not have this choice, I recommend that you go back and reload your **ZIP Program**. You will need this to work right in the future.

But if you are in a hurry, and exceptionally brave, you can 'set' the *Zip* program by selecting **OPEN WITH >**. You will given a menu where you can select the program you wish to use to open **data.zip**. In this case you must select your *Zip* Program.

Now you will have to Navigate in the *ZIP* program to the correct folder, then extract or unzip **data.zip**.



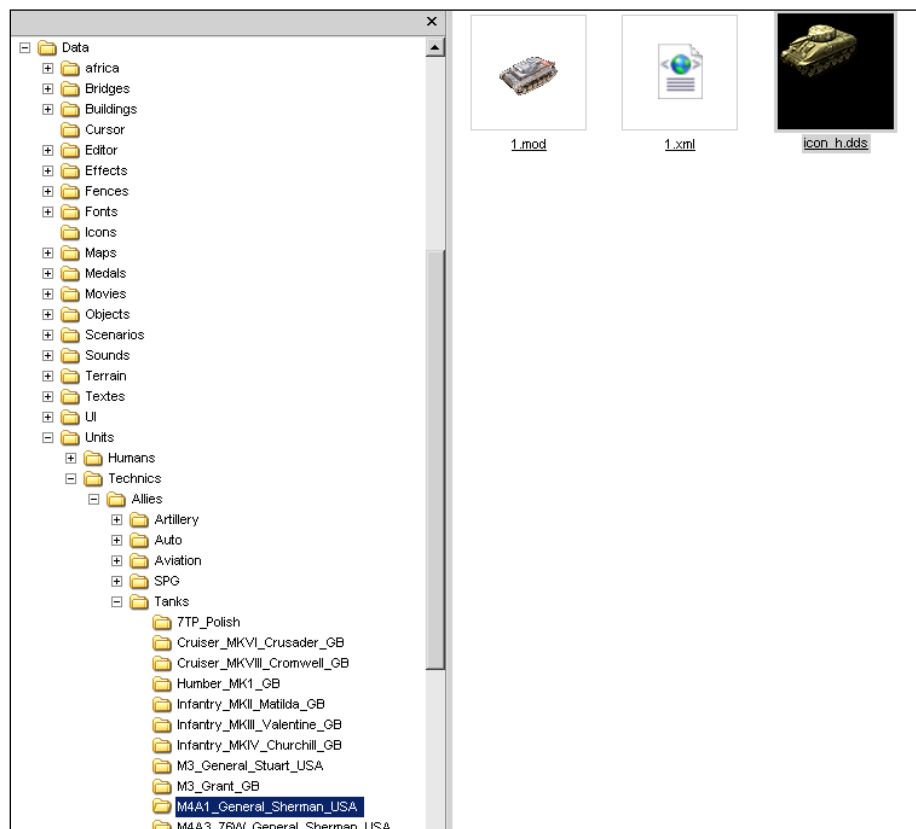
And for the **RESULT**, you will find this, or something very close depending on the **data.pak** version.

Folders	Name	Size	Type	Date Modified
Desktop	africa		File Folder	3/9/2004 1:46 PM
My Documents	Bridges		File Folder	3/9/2004 1:46 PM
Blitz	Buildings		File Folder	9/27/2004 7:54 PM
BK Orig	Cursor		File Folder	3/9/2004 1:47 PM
RUN	Editor		File Folder	3/9/2004 1:47 PM
Data	Effects		File Folder	3/9/2004 1:47 PM
	Fences		File Folder	3/9/2004 1:47 PM
	Fonts		File Folder	3/9/2004 1:47 PM
	Icons		File Folder	3/9/2004 1:47 PM
	Maps		File Folder	10/11/2004 8:41 PM
	Medals		File Folder	9/27/2004 7:59 PM
	Movies		File Folder	8/29/2008 11:00 PM
	Objects		File Folder	9/27/2004 7:55 PM
	Scenarios		File Folder	9/27/2004 7:56 PM
	Sounds		File Folder	3/9/2004 1:46 PM
	Squads		File Folder	9/27/2004 8:00 PM
	Terrain		File Folder	3/9/2004 1:48 PM
	Textures		File Folder	9/27/2004 8:14 PM
	UI		File Folder	9/27/2004 8:02 PM
	Units		File Folder	9/27/2004 7:58 PM
	Water		File Folder	3/9/2004 1:50 PM
	Weapons		File Folder	9/27/2004 8:13 PM
	data.zip	173,383 KB	WinRAR ZIP archive	10/6/2004 8:50 AM

I might add that procedure will take a little time and will depend on the speed of your machine.

You now have the game file structure where you can begin to make modifications. If you look at the folders, you will begin to see how the game uses the files. Open an OBJECT folder, you will see this:

The only file that we can really modify right now is the 1.xml, which defines the OBJECTS specific parameters.



Most of the OBJECT files contain three files:

- 1.mod, which is the 3d OBJECT
- 1.xml, which is the file that defines the OBJECT parameters
- icon.tga, which is the picture that the Game Editor uses to display the OBJECT in selection mode

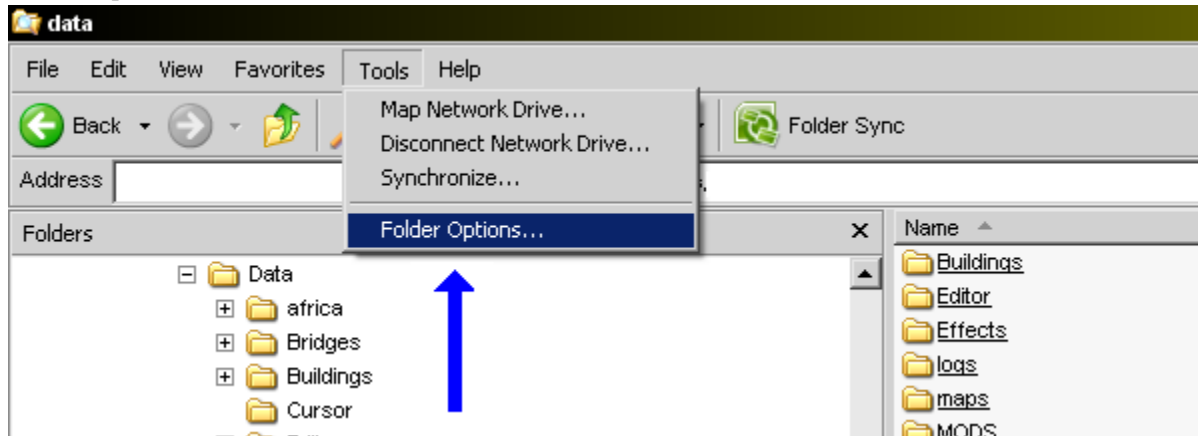
**.xml files** / Each unit/model has a file named 1.xml. This file can be edited easily in any text editor, but structure must be obeyed for it to perform correctly. This file stores the unit/model parameters which define the Hit points, sight, speed, and other attributes, as well as the type weapons/guns used by the model (if any). It also defines armor characteristics and other vital data that are unique to a specific model. Every object (model, unit, etc) requires a 1.xml file.

## FOR XP / XP-PRO USERS

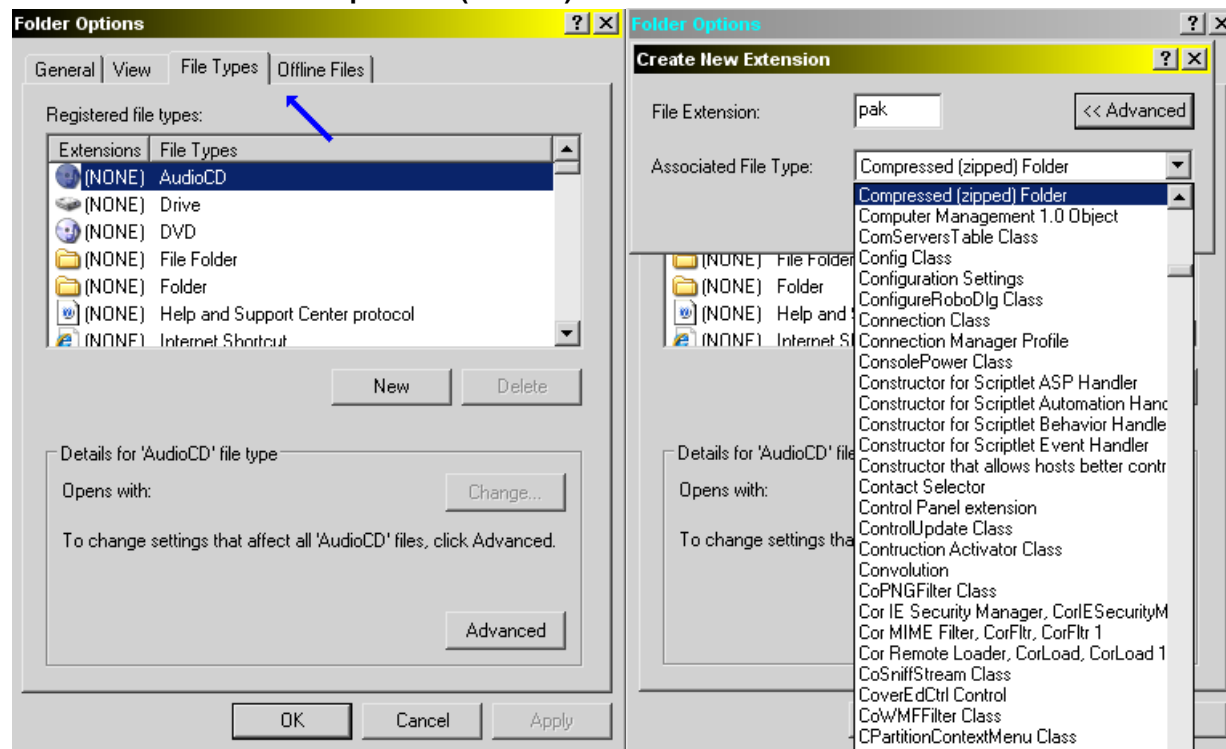
If you do not have XP or XP pro, this information will not be of any value to you.

The XP/XP Pro Operating System has an integrated ZIP utility program called **Compressed (ZIPPED) Folder**. It has the capability to extract and compress .PAK Files. You must 'associate' .pak files to **Compressed (ZIPPED) Folder** in **EXPLORER**.

GO TO **EXPLORER**, CLICK ON [ **TOOLS** ] ON THE **MENU** AT THE TOP / SELECT [ **FOLDERS OPTIONS** ]



Click on [ **File Types** ] then Click on **New**. Type "pak" in the File Extension box. Click [ **Advanced** ]. Scroll down and Select **Compressed (ZIPPED) Folders**.



Now you should be able to **extract/unzip/unpack .pak** files by Selecting the **.pak** file, <right click> and Select **EXTRACT HERE**.

Note: **Compressed (ZIPPED) Folders** is not a full function Zip program, but it will work fine for your purposes in BK.

## UNPACKING/UNZIPPING/EXTRACTING tex\_high.zip

### Our work is just beginning.

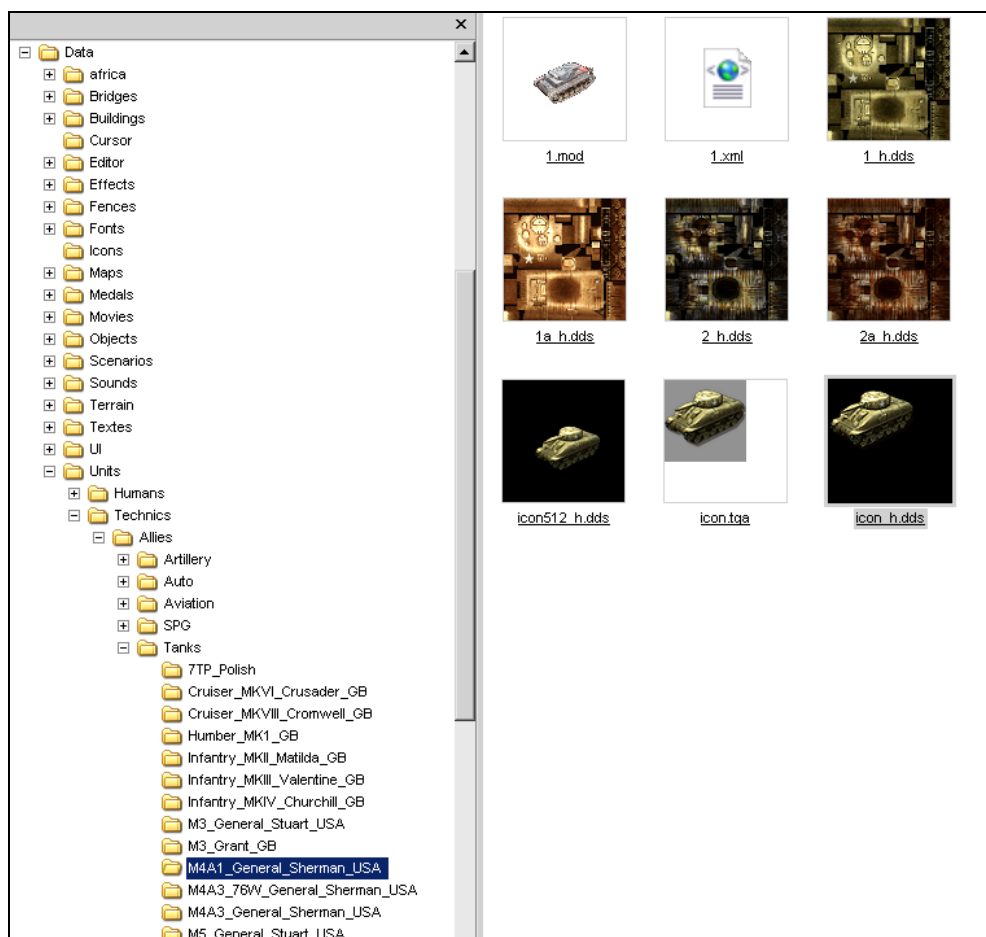
Go to the original **tex\_high.pak**, <right click> on the file and select **Copy**. Then switch to the work area data folder, and **Paste** the **tex\_high.pak** file. It should appear under the data folder, and remember, on slower machines it can take a minute.

Now <right click> on **tex\_high.pak** and select **Rename**. **Carefully Replace [ pak ] with [ zip ]**. Then press **Enter**. Now your **tex\_high.pak** file should be named **tex\_high.zip**. Remember, you may receive an **Alert Box** asking if you really want to perform this function. Click **Yes**.

If you have not yet properly installed your **ZIP** program, you should again think about doing it now.

You will perform the same procedure as you did previously for the **data.zip** file. If your **ZIP** program has been installed properly, you should be able to <right click> on **tex\_high.zip** then select **EXTRACT HERE**.

As you watch the progress of the unpacking, you may not notice any change to the file directory. But when it has finished, go back to the same objects folder that you previously open after we unpacked **data.zip**, and look at what changed. It should look similar to this:



See the difference? You have just merged the available skins for each OBJECT file to the folders. Remember, these are the **.dds** files.

**.dds files** / This file is the actual skin or map that is specific to each mod file. This type file can be modified by Photoshop or any graphic software program that has the required plug-in file.

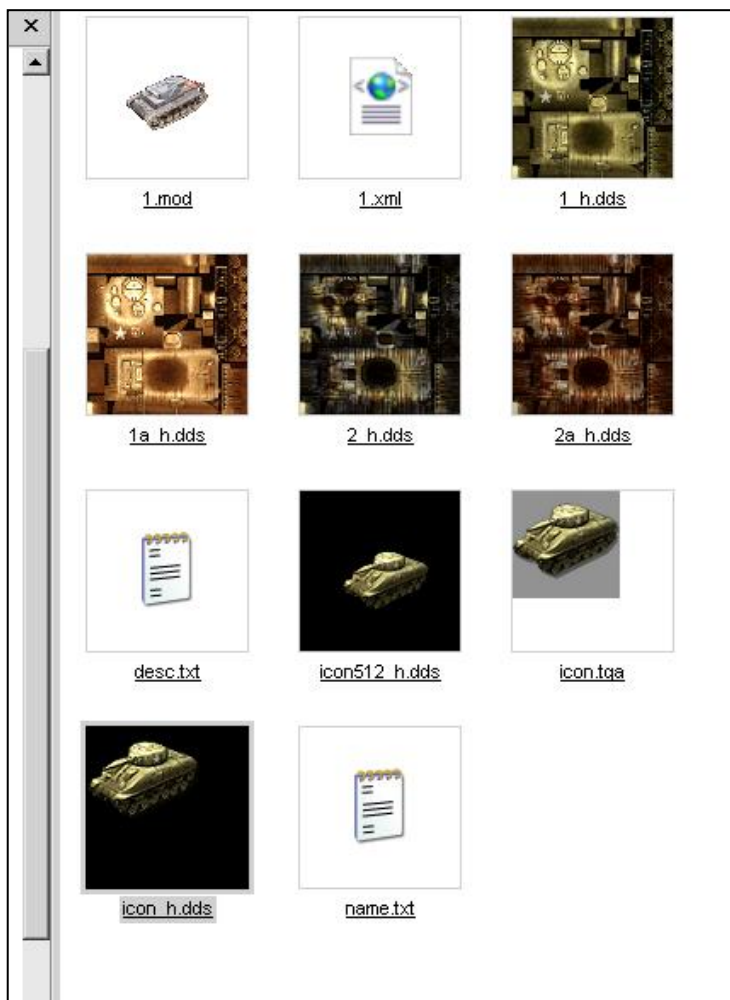
We can still add some other files. I recommend that you locate the **name.txt** and the **description.txt** files. The name of the .pak file that contains the text files will depend on which version of BK you have. In this case, Rolling Thunder, I will use **BH\_text\_english.pak**.

I will follow the same procedure as I did for **data.pak** and **tex\_high.pak**. Copy the original **BH\_text\_english.pak**, paste it to the new folder.

Rename **BH\_text\_english.pak** to **BH\_text\_english.zip**.

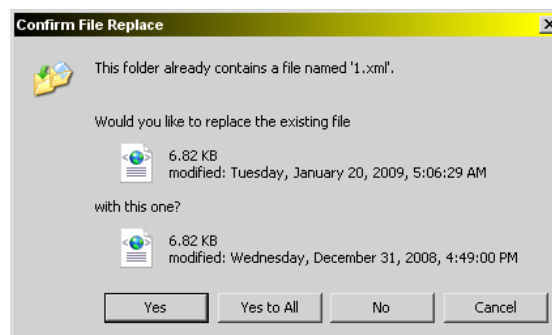
Then unpack by <right click> then Select **EXTRACT HERE**.

Again, you may not see any changes to the directory as the file unpacks. But now the text files have been added to each object file. You can verify this to be sure that you accomplished the desire effect. You should now see: **1.mod**, **1.xml**, **icon.tga**, the **.dds** skin files, **name.txt** and **description.txt**.



Now that you have a firm grasp on how to convert .pak files to .zip files, unpack the zip files and merge the contents of several files together, you can begin to fill in some of the other files that you may need. There may be **updatexxx.pak** files that include skins (.dds) or updated 1.xml files that are newer than what you have already unpacked and merged. The game engine will always use the file or folder with the most recent date, so it is important that you find these.

Whenever you begin to merge the files, if a newer file is encountered by **EXPLORER**, it will stop and prompt you for a decision. Since you know the latest version is the updated or newer file, you will also know that this file is the one that was used by the game engine in your game. If you are satisfied, with that selection, click **Yes**. There are times that you may feel that this will need further investigation. In that case, select **No**. Be sure to write down the folder and file so you can go back and check it in the .pak file. You can always pull it out later if you need it.

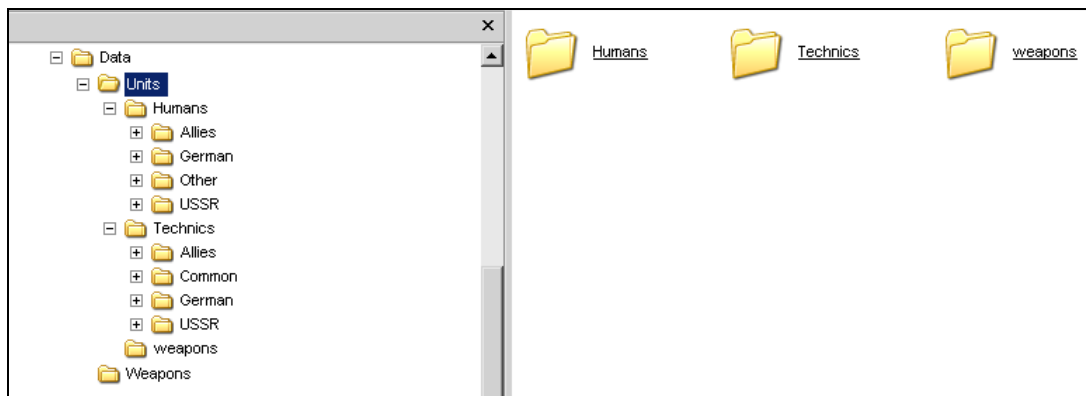


You should spend some time reviewing all of the original .pak files in the game \ run \ data folder so you can make sure you have everything you need.

## PACKING/ZIPPING FOLDERS

Quickly I will discuss re-packing folders. Much like the *.pak* files that you extracted, you can reverse the steps to zip or pack files and then convert it into a *.pak* file.. I have found it useful to have all of the specific files for certain OBJECTS contained in one single *.pak* file.

The first thing you may think of is how large that file would be, and you would be right to have a concern. I try to rebuild my *.pak* files with similar types of objects, such as **buildings.pak**, **allies\_units.pak**, **german\_units.pak**, etc. This gives you the benefit of having all of the similar type objects in one *.pak*, that includes all of the specific files that each object will need. This becomes very important if you ever need the source files to make a change. When you get comfortable manipulating the *.pak* files and folders, you can arrange them any way you wish. You must make certain that they adhere to the file folder format that you have already seen, but you can now see the possibilities.



Above is an example of creating a new *.pak* file. If you wish to build a **UNITS.pak** file, you should create a new folder and place a copy of all of the OBJECTS (units) in the folder. In the picture above, it shows the **UNITS** folder under the data folder. Create a new folder named **UNITS** then copy the Units folder to it.

Then all you have to do is use your *zip* program to zip or pack it. Once that has been completed, you can rename it to **UNITS.pak**. Then copy the **UNITS.pak** to your game Data Folder and the game should read the file.

If the **UNITS.pak** contains new objects, you will need to make certain that these items are indexed properly in the **objects.xml** file.

### About **objects.xml** and **modobjects.xml**

These files are very important to the game. Everything that the game uses is indexed in the **objects.xml**. Each item/object is listed by its name and path. The path is the physical location of the object within the file structure. This is how the game engine knows how to find the object. If an item has been indexed incorrectly, the item cannot be used, and may cause a crash, depending on the situation. Usually, new items or mods use a **modobjects.xml** to index the new items, so that the original **objects.xml** does not need modification. But you may structure your **objects.xml** any way you wish, as long as you observe proper file structure.

This will be covered in more detail in a later section if you think you missed something.

## THE GAME FILE STRUCTURE

As mentioned previously, everything that is used by the BK Game Engine, the map Editor, or the Resource Editor is an object of some type; the units, terrain, flora, buildings, etc. It takes many types of files to define each object; what it is, what it does, how it behaves, its appearance, etc.

One thing that you should remember is when you modify an object, you should never modify the original. Only modify a copy. If you modify an object, it is quite easy to test it before you pack or *.pak* it.

One of the misconceptions about the File Structure is that everything must be contained within a *.pak* file before the game can use it. That is FALSE. What is TRUE is that the object must adhere to the proper file structure rules. If you modify a Tank, if the modified tank is not put into the correct location of the file structure, the game will not use the modified file (tank). So placement of the file is very important.

When I test a new or modify an object, I don't pack it inside of a *.pak* file. If it doesn't work the way I want it to, I will spend more time packing and unpacking the *.pak* file than I will performing modifications. So you should adopt the same method.

So how do I get my OBJECT into the Game?

Go to your **BK Game Folder**. You will see the **Data** folder within the **Run** folder. In some versions of BK, you might see **Scenarios**, **Weapons** or other folders under the **Data** folder. Any folder under the **Data** folder usually contains files or data that the game uses.

You can place any Game File or Object File within the **Data** Folder and use it within the game, but it must adhere to the proper Game File Structure. So if you create a new Folder named **Units** in the **Data** folder, you can place modified Units into that folder and the Game will use your modified files rather than the original files.

**Example:** I want to modify the existing **M4A3\_General\_Sherman\_USA** by changing the summer skin. Once I have modified the skin the way I want it, then I can easily add it to the game.

Go to the Game Folder. <Left Click> on the Data Folder.

If you don't see a *Unit* Folder within the *Data* Folder create one. <Left Click> on **File** on the Menu Bar, <Left Click> on **NEW**, <Left Click> on **Folder**. A new folder is created; name it **UNITS**.

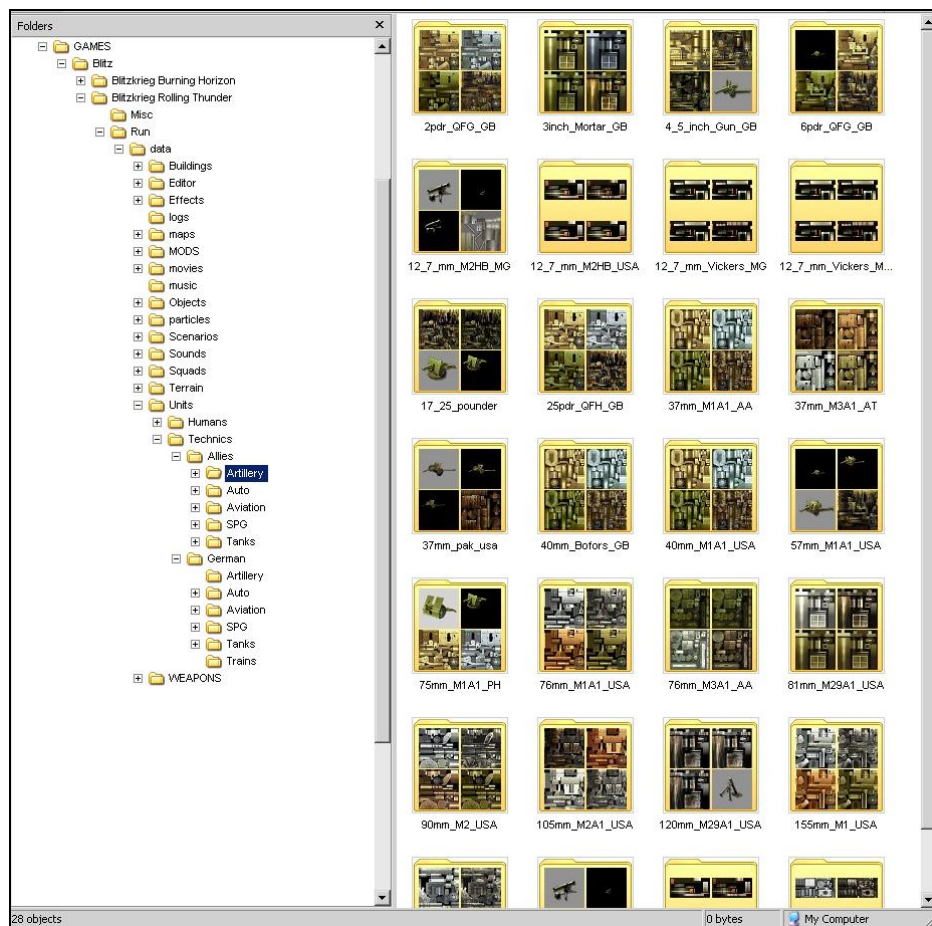
<Left Click> on the **UNITS** Folder, <Left Click> on **File** on the Menu Bar, <Left Click> on **NEW**, <Left Click> on **Folder**. A new folder is created; name it **TECHNICS**.

<Left Click> on the **TECHNICS** Folder, <Left Click> on **File** on the Menu Bar, <Left Click> on **NEW**, <Left Click> on **Folder**. A new folder is created; name it **ALLIES**.

<Left Click> on the **ALLIES** Folder, <Left Click> on **File** on the Menu Bar, <Left Click> on **NEW**, <Left Click> on **Folder**. A new folder is created; name it **TANKS**.



Now you can go to your modified **M4A3\_General\_Sherman\_USA** folder and copy the entire folder to the Game\Data\Units\Technics\Allies\Tanks folder. Since this is just a modified file, you do not need to change the *objects.xml*. The game will now use the new skin file.



You can add any folder to the **Data** folder, as long as you observe the proper file structure, and the game will use that data that you have modified. Once you are satisfied that your objects are the way you want them and they are working correctly, then you can move them into a *.pak* file; such as **my\_modified\_units.pak**.

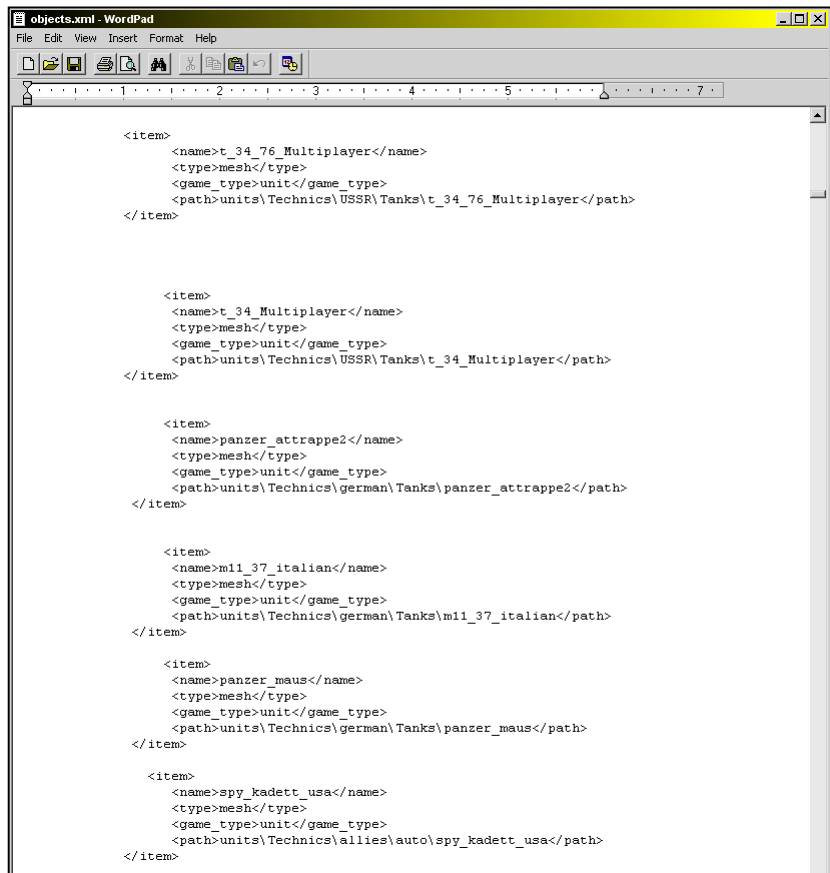
What you have already probably noticed is the data folder structure works exactly the same way as the *.pak* file structure. The game does not care if the data comes from a *.pak* file or within your **data/unit** folder. It reads all of the data in both areas. The Game Engine and Map Editor will always use the newest version of the data you supply.

What you may notice about my **Data** file is that I have what appears to be the entire directory of the data.pak file. No, that isn't the case. But since I am constantly working on everything from units to buildings and other objects, I simply created all of the necessary folders that I may need to test new objects. Objects that are still under development or that may still need some work remain in the data folder. Once I have completed a project, I build a *.pak* file. New objects can either be included within the **objects.xml** or a **modobjects.xml** depending on the situation. Items that will be released for others to use as in the case of a mod.pak will have an attached **modobjects.xml**.

## THE OBJECTS.XML FILE

So you have seen the Objects.xml file mentioned several times now. So what does this file do?

The objects.xml is the index or directory file that defines the data for each object used in the game. If an item is not listed or indexed, the game will not use it. Each object is defined by name.



A screenshot of a WordPad window titled 'objects.xml - WordPad'. The window shows the XML content of the objects.xml file. The XML is structured with multiple <item> tags, each containing <name>, <type>, <game\_type>, and <path> elements. The objects listed include t\_34\_76\_Multiplayer, t\_34\_Multiplayer, panzer\_attrappe2, m11\_37\_italian, panzer\_maus, spy\_kadett\_usa, sherman\_can, sherman76\_can, m3\_grant\_usa, churchill144, cromwell144, bt\_7\_span, bt\_5\_span, ba20\_span, and gmc\_can\_ari.

```
<item>
  <name>t_34_76_Multiplayer</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Tanks\t_34_76_Multiplayer</path>
</item>

<item>
  <name>t_34_Multiplayer</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Tanks\t_34_Multiplayer</path>
</item>

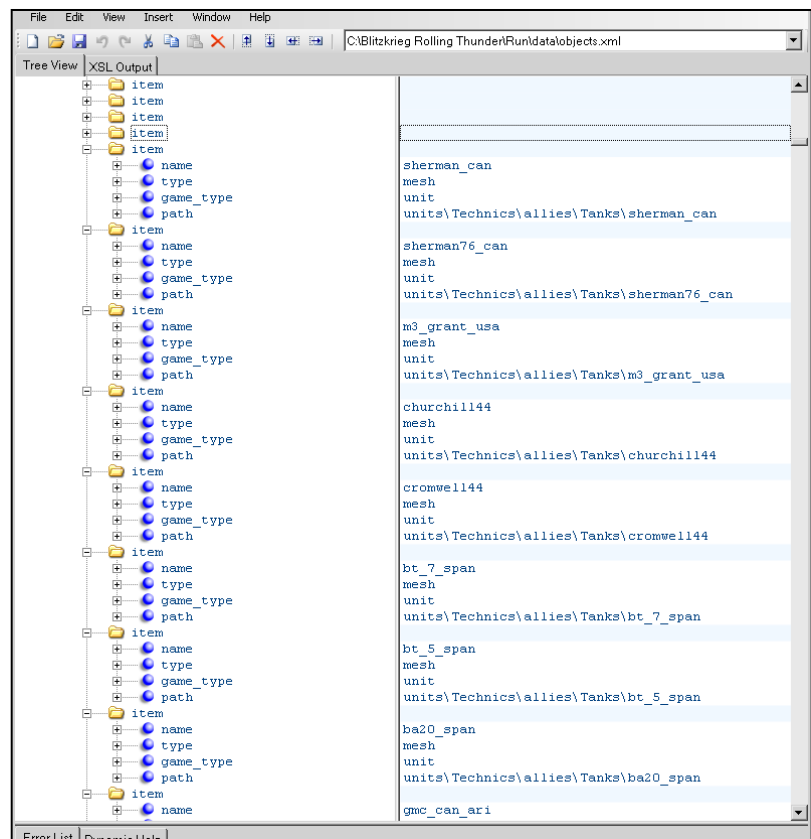
<item>
  <name>panzer_attrappe2</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\german\Tanks\panzer_attrappe2</path>
</item>

<item>
  <name>m11_37_italian</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\german\Tanks\m11_37_italian</path>
</item>

<item>
  <name>panzer_maus</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\german\Tanks\panzer_maus</path>
</item>

<item>
  <name>spy_kadett_usa</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\allies\auto\spy_kadett_usa</path>
</item>
```

Viewing the *objects.xml* in WORDPAD



A screenshot of XML Notepad showing the contents of objects.xml. The left pane displays a tree view of the XML structure, with 'item' elements expanded to show 'name', 'type', 'game\_type', and 'path' attributes. The right pane shows the corresponding XML data for each item, such as 'sherman\_can', 'sherman76\_can', 'm3\_grant\_usa', 'churchill144', 'cromwell144', 'bt\_7\_span', 'bt\_5\_span', 'ba20\_span', and 'gmc\_can\_ari'.

Viewing the *objects.xml* in XML NOTEPAD

Let's examine the file.

Each *OBJECT* is separated by `<item>` and `</item>`. When the game engine reads this file, it places the data into the **Game Database**.

Each time it reads `<item>`, the game engine realizes that this is the marker or index for the next object. When it reads `</item>`, it knows this is the end of the object data.

Each object is referenced by a name; `<name>122-mm_A-19</name>`. ►►

This `<name>` is not the NAME used by the Game Engine while playing the game. The Game Engine uses the name data contained in the **name.txt** file within the *objects* folder.

The **Map Editor** and the **Resource Editor** will use the `<name>` in the **objects.xml** for reference.

Then each object is identified as a `<type>` and `<game_type>`. ►►

This is very important. If you try to list something as an incorrect type or game type, the game will crash.

And finally, each object is defined by its location, or the `<path>`. ►►

This tells the game where it can expect to find each object within the file structure.

If the object does not really exist at this `<path>` (location), then the game will not use the object or it may cause a game or Map Editor crash.

```
<item>
  <name>10_cm_K_18</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\German\Artillery\10_cm_K_18</path>
</item>

<item>
  <name>120_mm_38</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Artillery\120_mm_38</path>
</item>

<item>
  <name>122-mm_A-19</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Artillery\122_mm_A_19</path>
</item>
```

```
<item>
  <name>10_cm_K_18</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\German\Artillery\10_cm_K_18</path>
</item>

<item>
  <name>120_mm_38</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Artillery\120_mm_38</path>
</item>

<item>
  <name>122-mm_A-19</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Artillery\122_mm_A_19</path>
</item>
```

```
<item>
  <name>10_cm_K_18</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\German\Artillery\10_cm_K_18</path>
</item>

<item>
  <name>120_mm_38</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Artillery\120_mm_38</path>
</item>

<item>
  <name>122-mm_A-19</name>
  <type>mesh</type>
  <game_type>unit</game_type>
  <path>units\Technics\USSR\Artillery\122_mm_A_19</path>
</item>
```

Now that you know what the file does, you can see the importance of the file structure.

I strongly recommend that you make a copy of the **OBJECTS.XML** and place it in your game folders **data** folder. This way you can easily modify the file to add new units for testing purposes. Study the structure carefully and be certain that you always create a backup. Always place new items at the beginning of the file so they are easily located.

When you add new items, you might adopt a method of creating *Comments* which reference what you are adding.

You can add a comment by surrounding your text with:

**<!-- (text here) -->**.

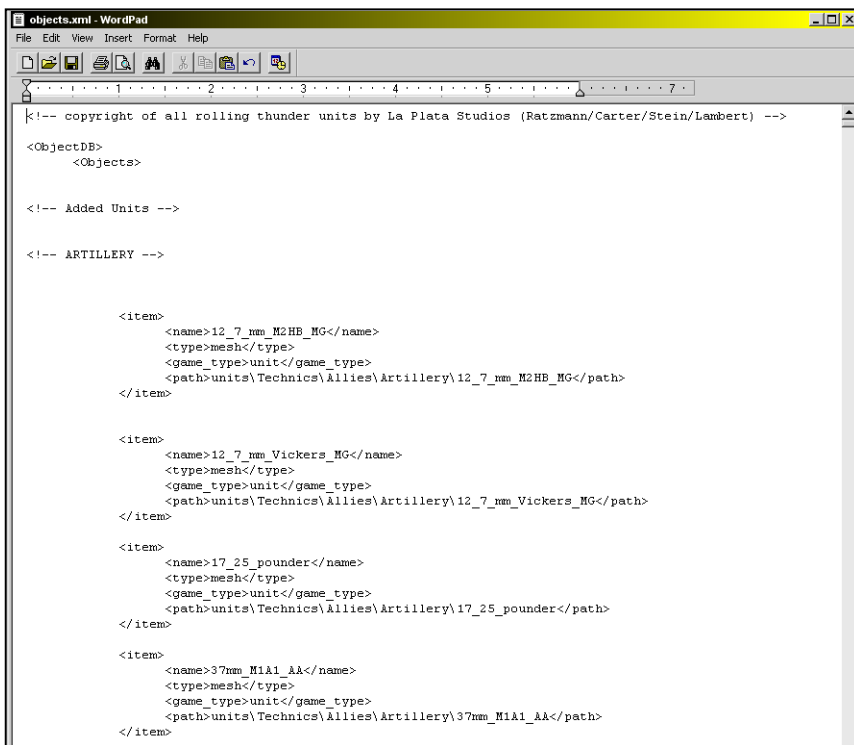
I usually use **WORDPAD** to modify the objects.xml, because of the *copy/paste* functions. **XML NOTEPAD** or any other XML Editor will work just as well, and will probably reduce the chance of mistakes in the file structure.

Once you open any .xml file in WORDPAD, NOTEPAD or

any other Text Editor, you must not change the file extension. This will cause the file to become unusable. Likewise, you cannot create a new .xml file within a Text Editor and save it as an .xml. If you try to name it as an .xml file, it will actually save the files as: [filename].xml.txt.

Always start with an existing .xml; you can always borrow one from something else that is a close resemblance when you need to create a new one. After it has been modified, you can properly place it where it needs to go. Alternately, and what I do is create the copy where I want to put it then modify the file.

XML Files will be covered later in more depth in this Tutorial.



## PART 2

### .DDS FILES - The OBJECT SKIN Files (TEXTURE MAP)

Every object in Blitzkrieg has a Skin File or Texture Map. In most cases, objects like tanks or trucks have multiple skins which are used for various maps; Summer/Spring, Desert and Winter. But they will also have a Death skin that shows each unit after it has been destroyed.

**List of SKIN FILES**

<u>High Resolution Skin Files</u>	<u>Low Resolution Skin Files</u>	<u>Compressed Skin Files</u>
1_h.dds Summer/Spring	1_l.dds Summer/Spring	1_c.dds Summer/Spring
2_h.dds Summer/Spring Dead	2_l.dds Summer/Spring Dead	2_c.dds Summer/Spring Dead
1a_h.dds Desert	1a_l.dds Desert	1a_c.dds Desert
2a_h.dds Desert Dead	2a_l.dds Desert Dead	2a_c.dds Desert Dead
1w_h.dds Winter	1w_l.dds Winter	1w_c.dds Winter
2w_h.dds Winter Dead	2w_l.dds Winter Dead	2w_c.dds Winter Dead
Use with +32mb Video Cards	Use with older Video Cards	GE Force256/ later Video Cards

These skin files, (.dds), are specific to each **1.mod** file that defines the Object. You could not use a .dds file from a tank and use it for a truck for example. But there are times that one skin file may be used for more than one Object, like in the case of a certain model of tanks with different specifications.

Blitzkrieg can use three different resolution settings, high, low and compressed. See the Table above for reference. Most BK Gamers and Mod/Unit Developers use High Resolution for their projects. This tutorial will be specific to the High Res files, but the same principals can be applied to the other two resolution types. You should not try to use High Res files if you have an older computer or limited drive space.

One of the issues that have been discussed over the last several years that Blitzkrieg has existed, is the missing skin files for certain maps. Some objects did not have a Winter Skin when the game was released. With the various releases of .pak files and MOD packs, this discussion has almost all but ended. As BK gamers began to explore the Open-Source possibilities that the game presented, they found ways to fill in the gaps, and go light years past in terms of what could be done to improve the game.

If you are interested in modifying the object skin files, you are at the door and only have to enter and read through this Tutorial, to learn "HOW TO" SKIN.

In Part 1, you learned how to access the various game files. If you have extracted the .pak files and have created a work folder for your modified files, then you already have access to the .dds skin files. If you have not extracted the files, then I suggest that you go back to Part 1.

Below are the links for two specific viewers which will allow the .dds skin files and the .tga object picture to be viewed in the Thumbnail View of **Explorer**. You may find these helpful for locating specific files later.

[http://download.nvidia.com/developer/NVTextureSuite/DDS\\_viewer.exe](http://download.nvidia.com/developer/NVTextureSuite/DDS_viewer.exe)

[http://www.greggman.com/downloads/thumbplug\\_tga1.10.exe](http://www.greggman.com/downloads/thumbplug_tga1.10.exe)

A program that you will find useful is the TANKS VIEWER v0.2d by Pesmontis. This program allows you to View the game 1.mod (3d objects) with their .dds (skin file), and allows you to see the changes to your skin as you make them. It can be downloaded at: <http://tatooinebase.star-fleet.org/>

The most important tool you will need for modifying an Object skin file, is a *Graphic or Image Manipulation Program*. Two very good programs that can handle the .dds files are **Adobe Photoshop** and **GIMP**. Photoshop has a hefty price tag, around \$200 upto \$1000 depending on the version. GIMP is a shareware program and is free, and available from GIMP at <http://gimp-win.sourceforge.net/stable.html>. I use GIMP version 2.2.13. Gimp has now released version 2.6.5. While I am sure it will work fine for BK skins files, I do not have any experience with it. Version 2.2.13 is still available as well.

FROM GIMP: GIMP (GNU Image Manipulation Program) is a freely distributed piece of software suitable for such tasks as photo retouching, image composition, and image authoring. It is a powerful piece of software with capabilities not found in any other free software product. It can be used as a simple paint program, an expert-quality photo-retouching program, an online batch-processing system, a mass production image renderer, or an image-format converter. GIMP is modular, expandable, and extensible. It is designed to be augmented with plug-ins and extensions to do just about anything. The advanced scripting interface allows everything from the simplest task to the most complex image-manipulation procedures to be easily scripted.

Because Photoshop may be difficult for many modders to obtain, I will use **GIMP** for examples within this tutorial. Most of the information will be very similar for Photoshop.

Once you have the Image Software, you must also have the .dds plug-in which allows you to load and save .dds files. The plug-ins are available from both Adobe and GIMP.

When you install the Graphic Software, if you have loaded the program and plug-in correctly, .dds files should be automatically associated with the Graphics program. You should be able to click (select) a .dds file and automatically load the Graphic Program.

If selecting a .dds does not cause the software to load, you can Associate the .dds file with the graphic software in **EXPLORER**.

**To Associate .dds Files in Explorer**, <select> [ **Tools** ] (on the menu at the top) <click on> [ **Folder Options** ]. When the Option Box appears, <select> [ **FILE TYPES** ]. Search the list of extensions and find .dds.

**IF .dds appears in the list**, <click on> .dds and check the program that is associated with it. If it does not show your Graphics program, <click on> **Change** then select the Graphics program from the list.

**IF .dds does not appear in the**, <click on> [ **New** ]. <Type> .dds in the **Create New Extension** box. Then <click on> [ **Advanced>>>** ]. <Select> the Graphics program from the list.

Once you have checked to see if your Graphic program opens .dds files, you are ready to take a look at how a skin works.

Since it is likely that you checked the file association by clicking on a .dds file, you probably now have the Graphics program open with a display of a .dds file.

At this point, I recommend that you CLOSE or EXIT the program, EXIT all programs that are open, including **EXPLORER**, and perform a RESTART. Modifying .dds files can and will use a lot of RAM. You should start with a fresh STACK on your system before you begin to manipulate your files.

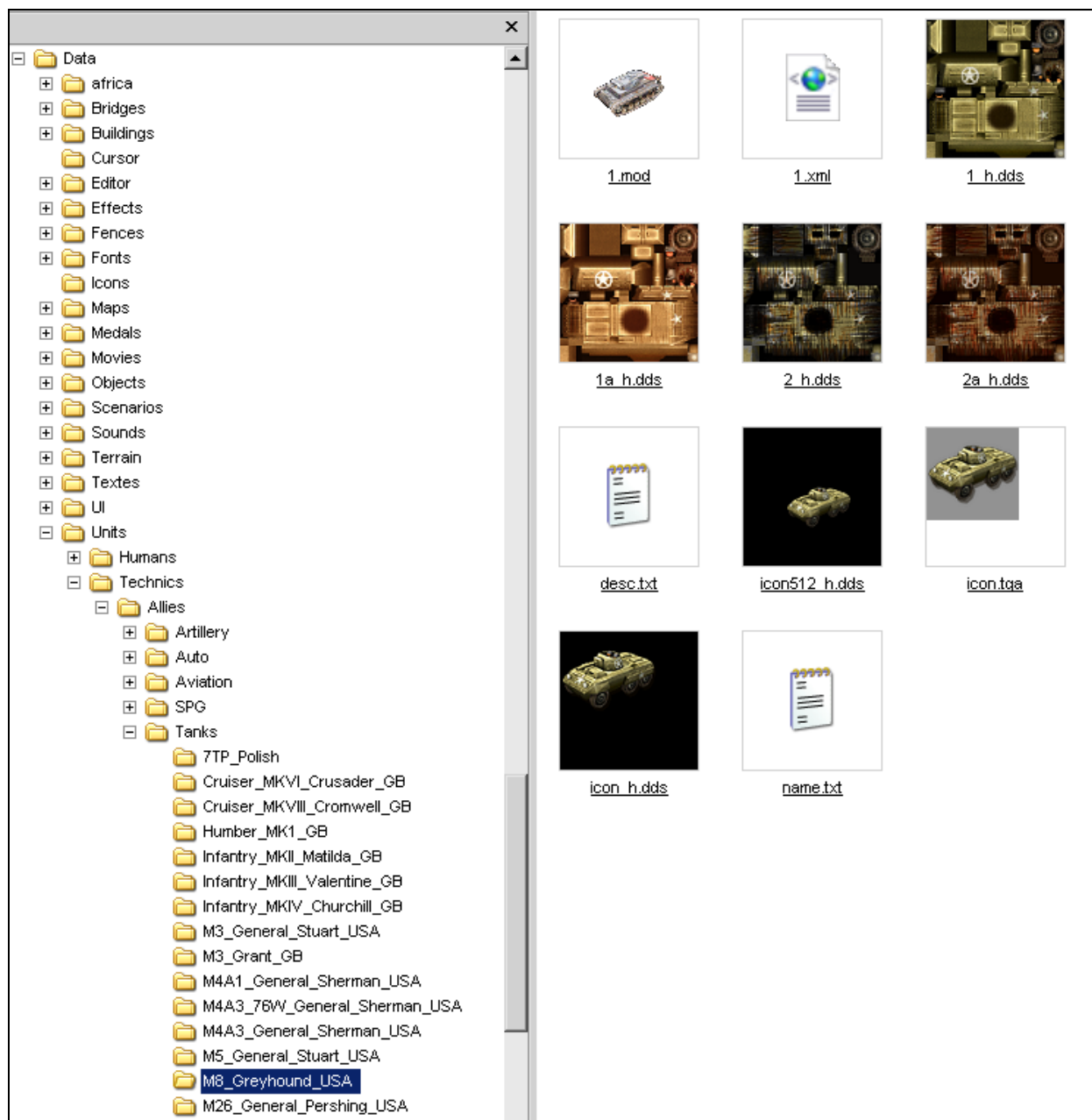
Note: Graphic Files use large amounts of memory. If you ever detect that the program is slowing down or seems to hang on COPY/PASTE functions, you should SAVE your work and do a Restart.

Get in the habit of Saving your work often, especially when modifying skin files. Also check your changes after each change. If you don't like the change, you can UNDO the change, then Re SAVE.



## .DDS FILES – HOW TO ACCESS

Accessing the .dds Files is easy once you have set up a work area. Go to your BK Work Area. Navigate to the **M8 Greyhound**: DATA \ UNITS \ TECHNICS \ ALLIES \ TANKS \ M8\_GREYHOUND\_USA.



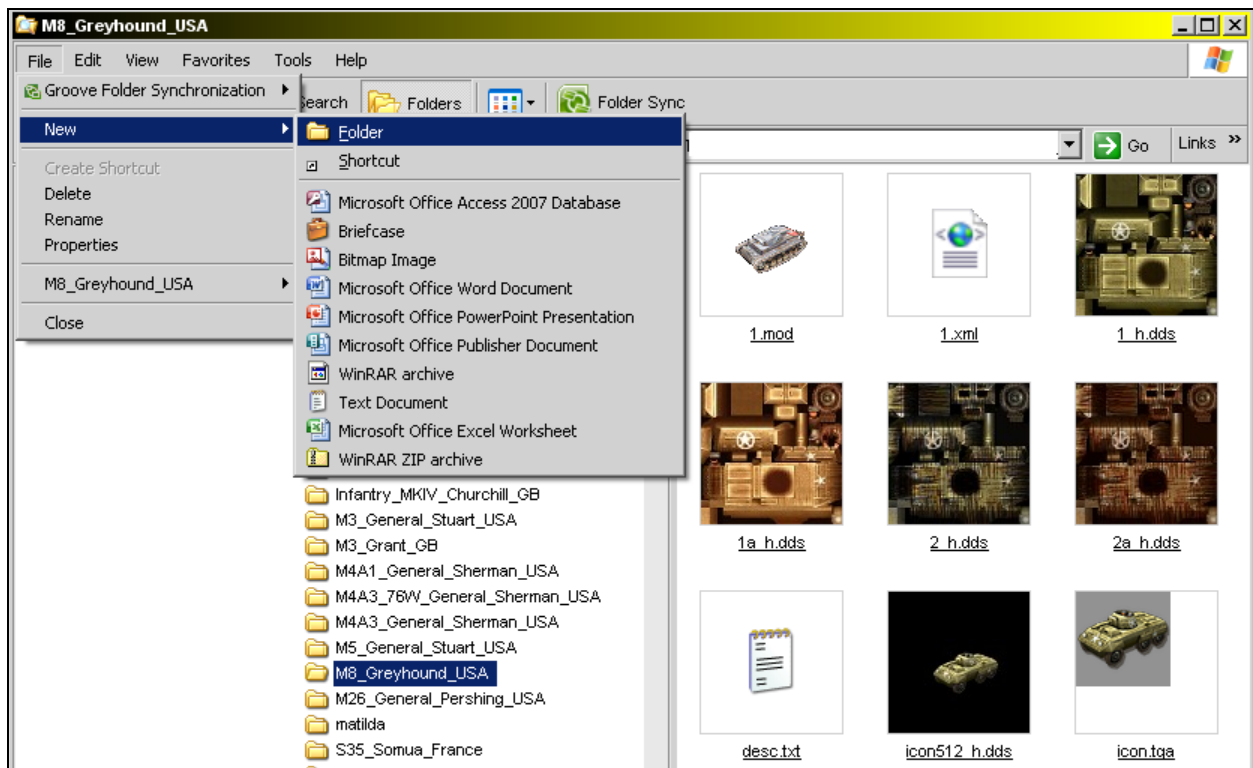
If you have extracted the .pak files properly, you will see something similar to above. You should load the DDS and TGA Viewers and set EXPLORER to Thumbnails in **VIEW**, so you can display the graphic files.

Before we load our Image Program, it is important to have a plan at this point. If you just need a .dds (skin) file for a winter map, you will notice that the M8 does not have one. While that file has been created by many people, you may not have found the one you need. It is important to know that if the BK Game Engine can't find a skin for a season/map, it will display the model with a checkered skin. Because the checkered skin doesn't look very realistic or even appealing, we will quickly create a winter skin for the M8.

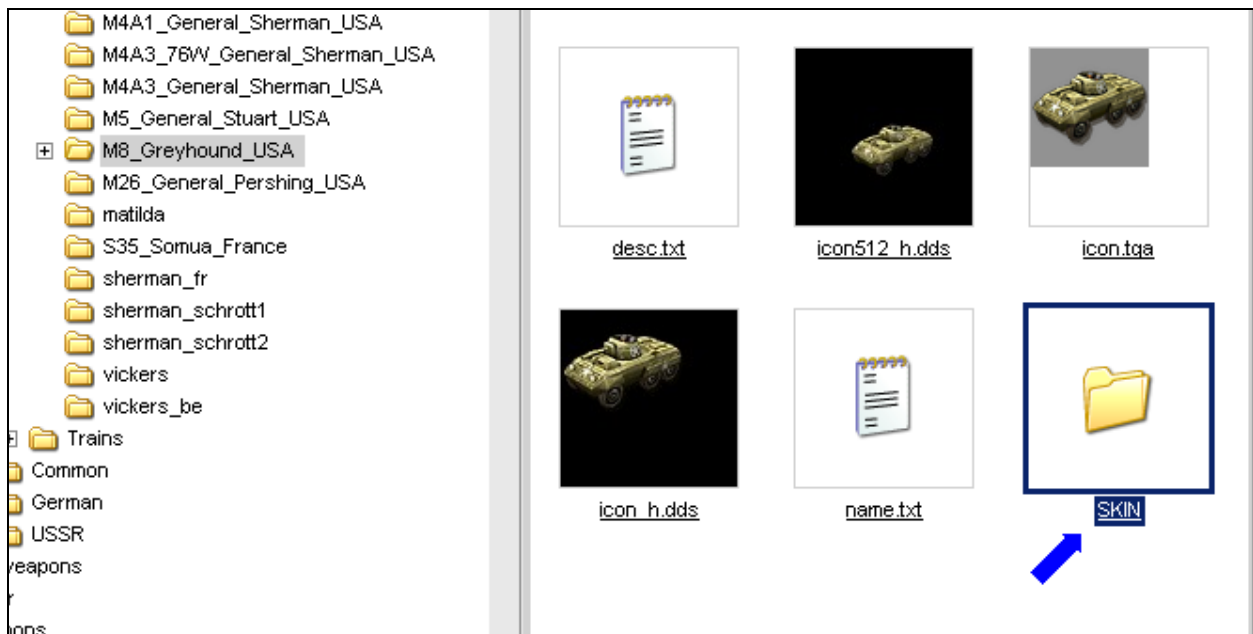
## Creating a New Skin - The Quick Way

In EXPLORER, <click on> the **M8\_Greyhound\_USA** Folder.

Go to the Menu at the top and <click on> **File**, <select> **New** and <click on> **Folder**.



A new folder has been created in the M8 Folder. Name it **Skin**.



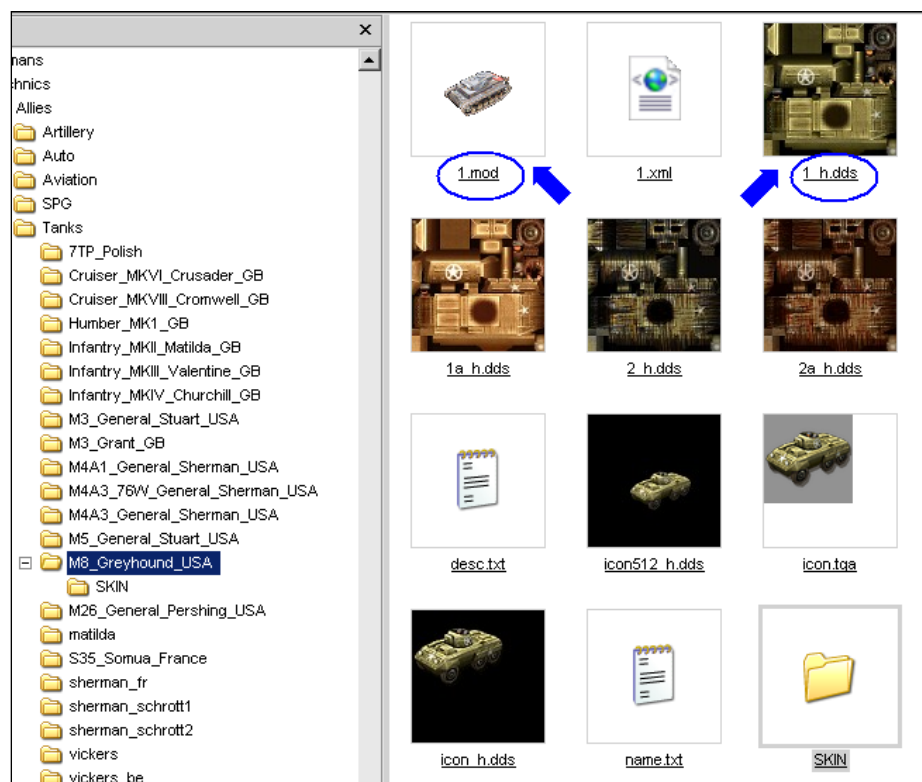
Now you have created a 'Skin' folder to use for new skin files for the M8. We will use the Skin folder as a holding area while we create the Winter Skin for the M8.



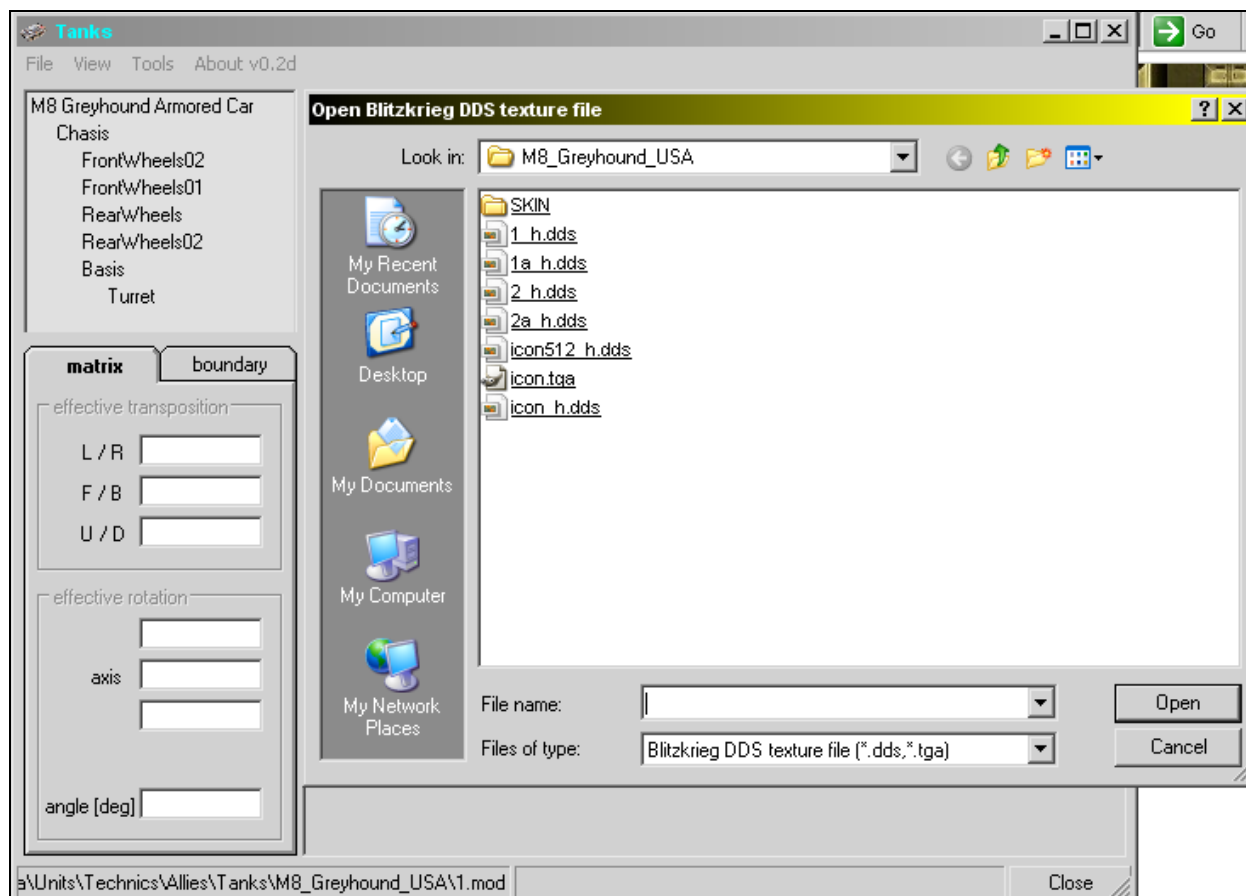
Now we will focus our attention on primarily two files in the M8 Folder, *1.mod* (the object model) and *1\_h.dds* (the object skin).

With your mouse, <click on> *1.mod*. If a box opens that ask which program you wish to use to open the file, select **TANKS v0.2d**.

**Note:** The .mod extension is also used for Music files. If your music browser opens, you will have to Open the Tanks v0.2d program from the Start menu or Explorer. I suggest that if you plan to use the Tank viewer and 1.mod files frequently, you should change the Association of the .mod file to the Tank Viewer.



When **Tanks** opens, it will prompt you to select a *DDS* texture file. <Click on> *1\_h.dds*.

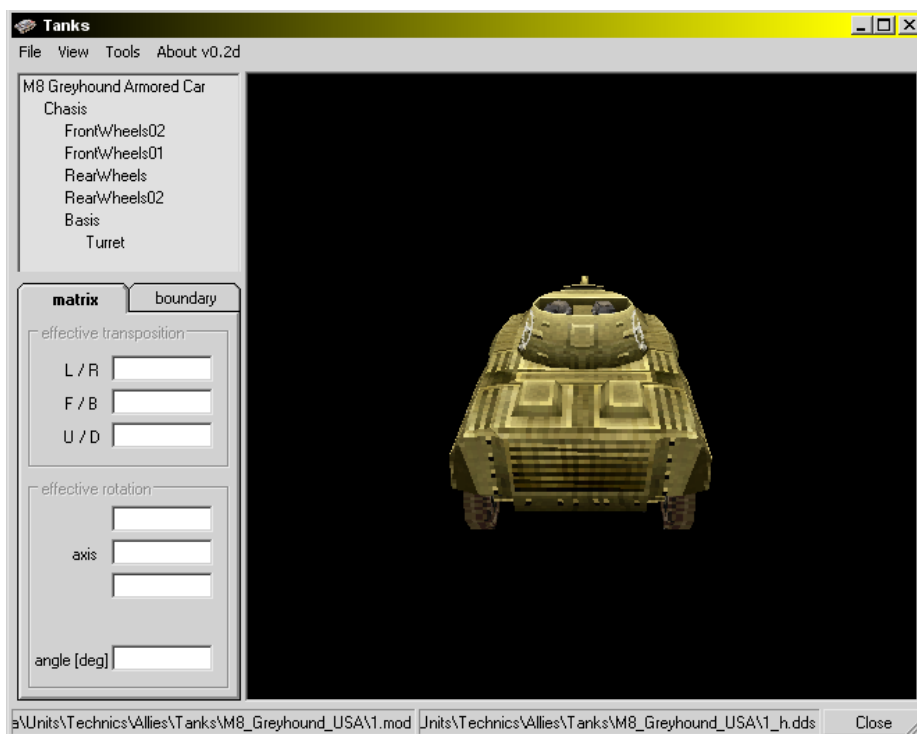


You should now see this...

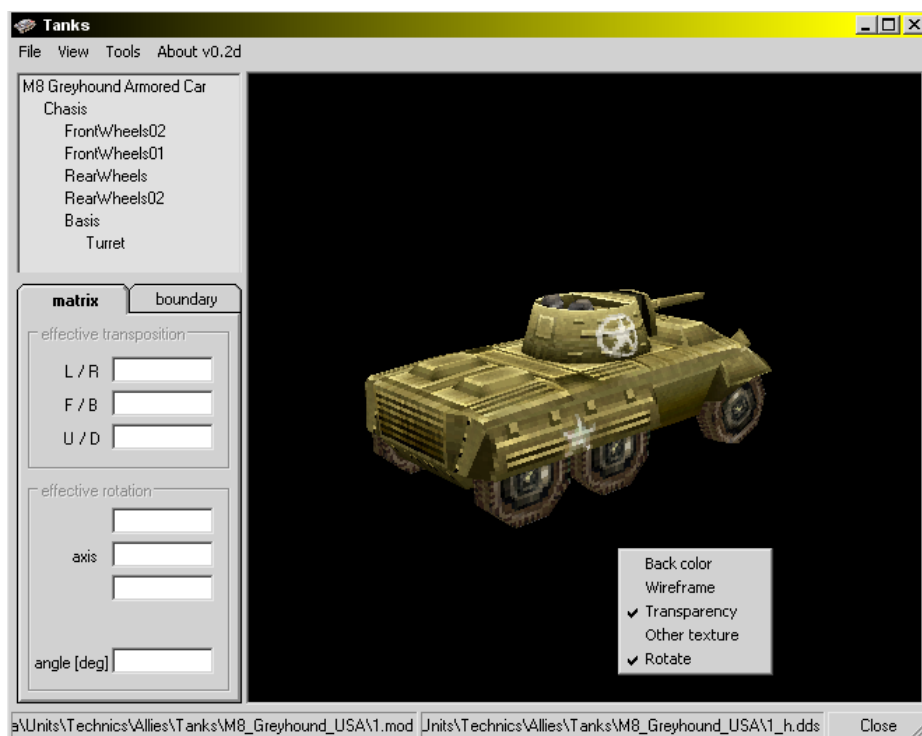
As soon as the program opens and displays the Object Model, it will begin to rotate. You will notice that the Tanks Viewer also displays the Object Model with the *1\_h.dds* skin, which is the summer texture.

<Right Click> on the **Object** in the Viewer to activate a Menu.

**Play with the Menu Options a bit to become familiar with what they do.**



- Back Color – Allows you to change the background color
- Wireframe – Display the 3d Wireframe of the Model without the skin.
- Transparency – Will Display or Hide any Part of the Model that is transparent when using the skin.
- Other Texture – Prompts you to load additional .dds skin texture skin files.
- Rotate – Turns the Rotate function on or off.

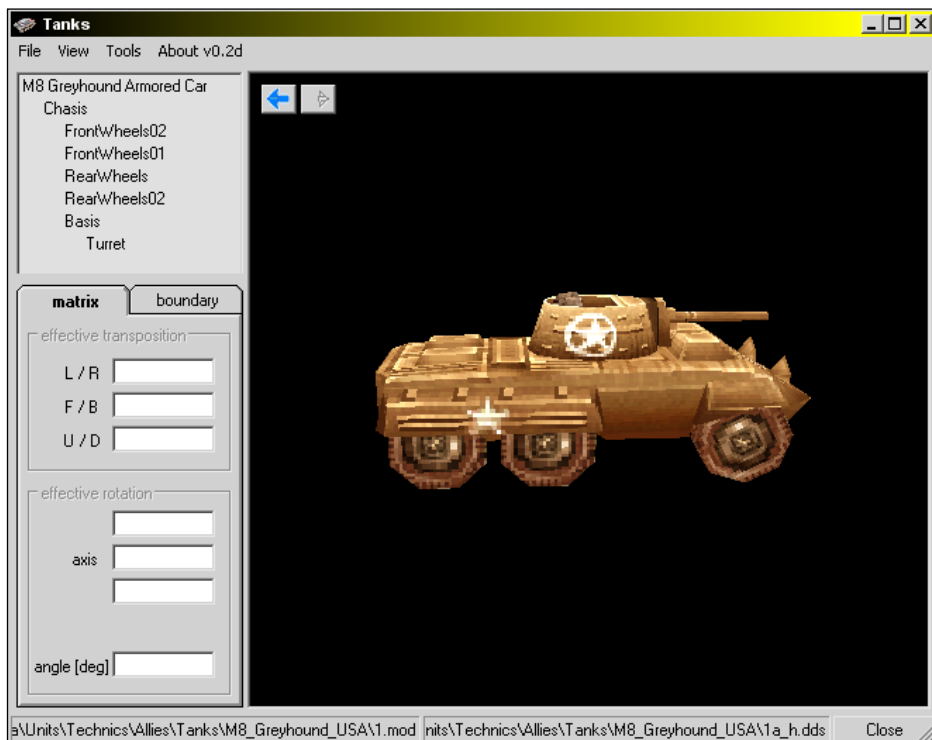


You might notice additional Menu items at the top left. Take a look at them, but you will not use them very often. Just below the top menus is a box that displays the name of the Object, the name of the parts that make up the 3d model and the matrix and boundary data. While this information is interesting, you will not need this data to create skins.

<Right click> on the **Model**, <click on> **[Rotate]**. <Right click> again on the **Model**, <click on> **[Other texture]**. When the prompt box opens <select> **1a\_h.dds**.

Tanks will now display the Model with the Desert Skin. Since we do not have a Winter Skin, let's create one.

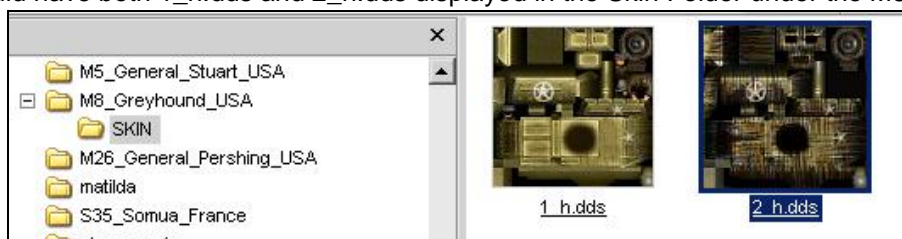
Go to **EXPLORER**. In the M8 folder, <Right Click on> **1\_h.dds** and select **[Copy]**. Open the Skin folder in the M8 Folder, <right click> in the folder (when it opens) and <select> **Paste**.



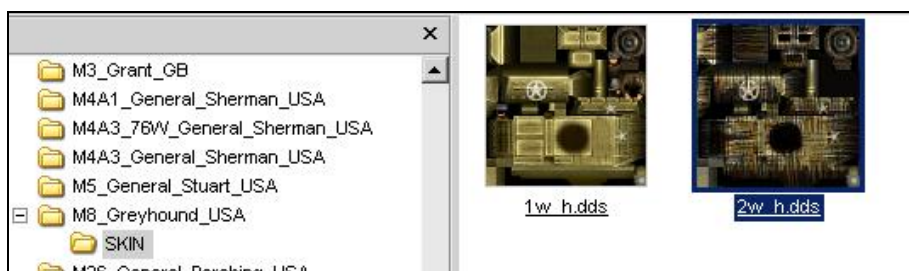
You see **1\_h.dds** displayed in the Skin folder.

Repeat this but this time make a copy of **2\_h.dds** and paste it into the Skin folder.

Now you should have both **1\_h.dds** and **2\_h.dds** displayed in the Skin Folder under the M8 folder.



<Right click on> **1\_h.dds** <click on> **[Rename]**. Carefully rename the file to **1w\_h.dds**. Do the same to **2\_h.dds**, but Rename it **2w\_h.dds**



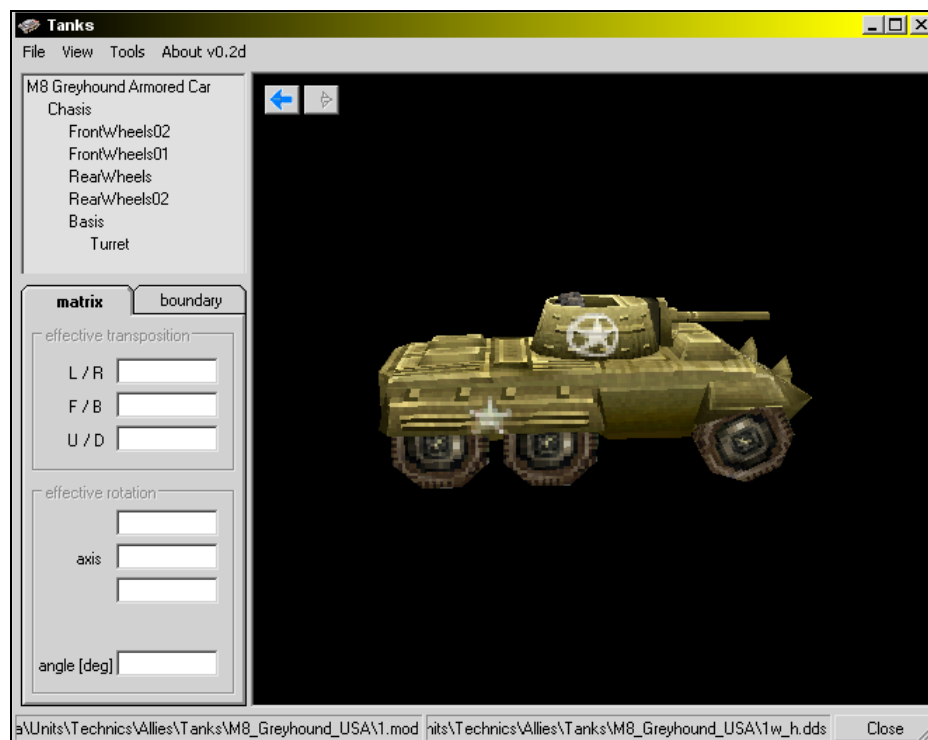
**Helpful Tip:** When Renaming a file, click in the name box. This will leave the current name displayed. Add the letter 'w' after the number (1 or 2) and press ENTER. Be careful not to erase the name or extension. If you should erase the extension, then you must be very careful to complete the correct name and extension or the file may become corrupt.

Now let's move the newly created Winter Skins to the M8 Folder. The easiest way to do this is by placing your mouse over one of the files, press the [ **SHIFT** ] key, then move your mouse over the next file. Once they have both been **Highlighted**, <press and hold> the Right Mouse Button in and move the files to the M8 folder. You must make certain that M8\_Greyhound\_USA folder name is also **Highlighted**. When you have the files over the M8 folder, release the mouse button.



Now click on the M8 folder. You should see the Winter Skins now displayed.

Go to Tanks. <Right click> on the **Model**, <click on [ **Other texture** ]>. When the prompt box opens <select> **1w\_h.dds**. Tanks will now display the Model with the Winter Skin.



But of course the new winter skin is nothing more than a copy of the summer skin. But the main difference now is if you need a winter skin, you now have one. It is better than the model showing the checkered skin we discussed earlier. When we created the LIVE M8 winter skin, we also created the Death winter skin. So now you can use this model on a Winter Map, and it will work just fine.

This method will work as a quick fix when you need a specific skin and you don't have time to find one or paint one.

In the next section, we will modify the winter-summer M8 skin, and really winterize it.

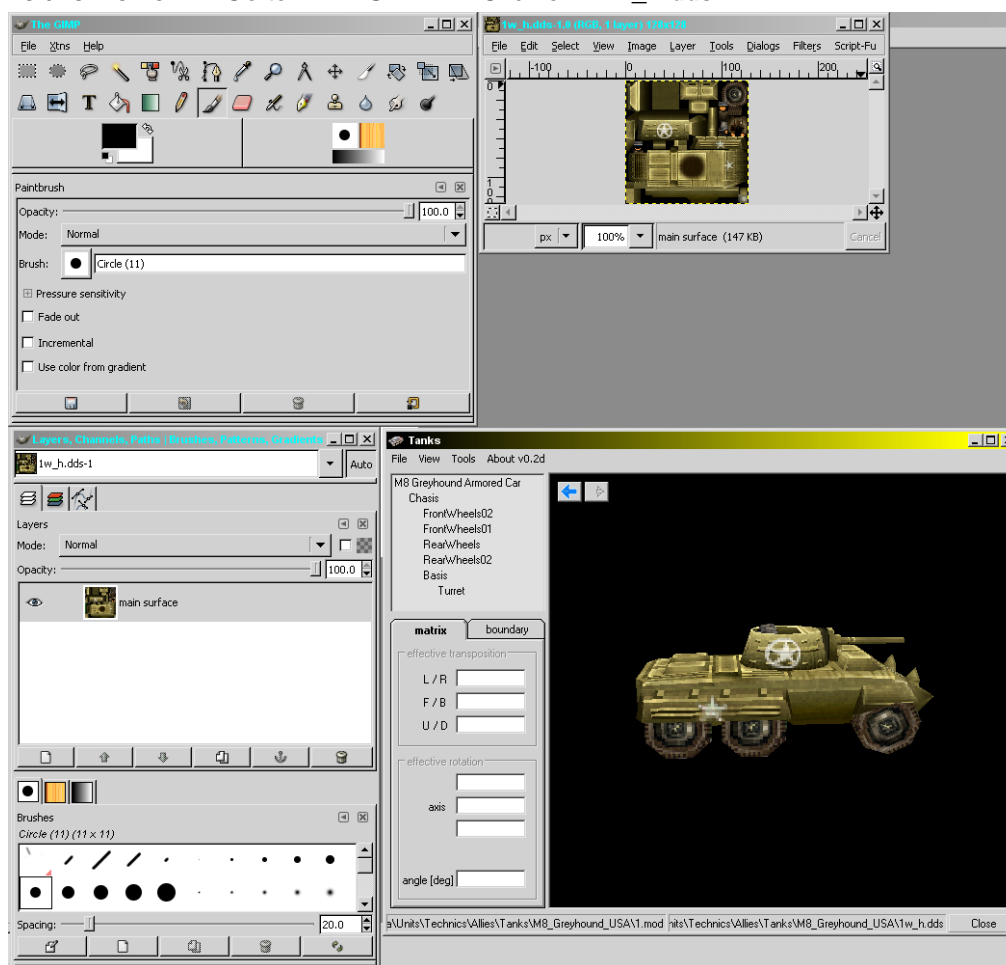
## .DDS FILES – HOW TO MODIFY

In the previous discussion, we created a temporary winter skin for the M8 Greyhound. Since we used summer skin, it might look a bit out of place on a Winter map. During Winter Operations, troops tried to camouflage themselves and their vehicles so they didn't stand out like a sore thumb. A Drab Green Armored car would make a great target since it would be very visible against the white landscape.

But troops did not always have everything they needed to really paint their vehicles, so they had to improvise. Most US units used a watered-down whitewash to help break up the green color scheme. Notice I said to break up the color scheme. That is what camouflage is about, to break up the normal appearance so it blends in. So what you did not usually see were White Tanks. What you did see were semi-white and green tanks. Troops entering a battle zone that were fresh, usually had not taken time to "camo" their vehicles. They did that as they begin to get set up. They used whatever they had on hand, whitewash, foliage, roots, hay, tree limbs, etc.; anything they could find to break up the usual pattern of their vehicle.

Keep in mind, you can do anything you want to make your M8 skin blend in. Even if you create the best "camo" winter skin, where you can't even see where it is, the game will still see it and the AI will attack it if it is in range. So 'camo' has absolutely no bearing on the game AI, unless you change the M8's parameters in the 1.xml file. But that is a different part of this Tutorial.

Let's winterize this M8 now. Go to **EXPLORER**. <Click on> 1w\_h.dds.



This is a screenshot of my monitor displaying the main GIMP Tool Menu, Layers/Channels Box, the 1w\_h.dds skin texture file and TANKS, which is displaying the M8 1w\_h.dds skin file.

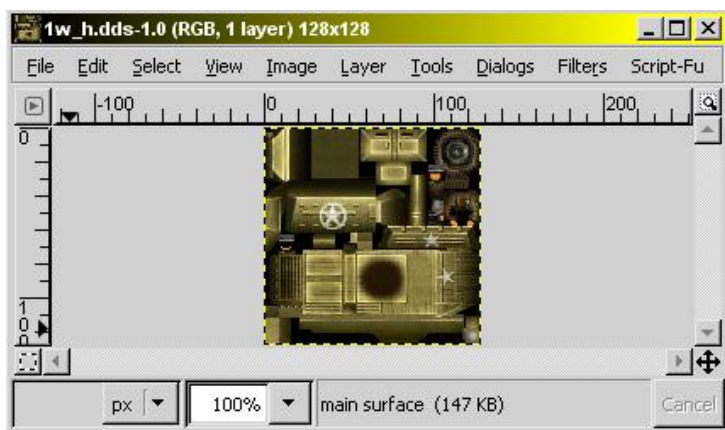
Since I am using **GIMP**, it will vary somewhat from **Photoshop**. Some of the basic things will be very similar, but it is important that you read any help files, start up files, or read me files that pertain to your Image program. The scope of this Tutorial is not to teach you everything about your Image program, but to help you on the track to a quick start. If something does not make sense to you, I urge you to not get frustrated. The tools and methods used in this tutorial will become very familiar to you as you develop your skills. Just as you now can extract and access the game files, you are just now embarking on another learning mission. It will take some time to learn everything about your Image program, but the basics hopefully will become clear very soon.

One of the great things about **GIMP** is the flexibility of its various menus and tools. Each part of **GIMP** exist in its own window. You can set or arrange each of these windows anyway you want by dragging on the header. I use a 22" Wide Monitor and have enough space to keep **GIMP** and **Tanks** handy all of the time. If you are using a smaller monitor, you will have to use the program tabs at the bottom of your monitor screen to task between various tools or programs. Your programs and monitor may not display exactly what I will show you within this tutorial, but hopefully you can create an environment that works for you.

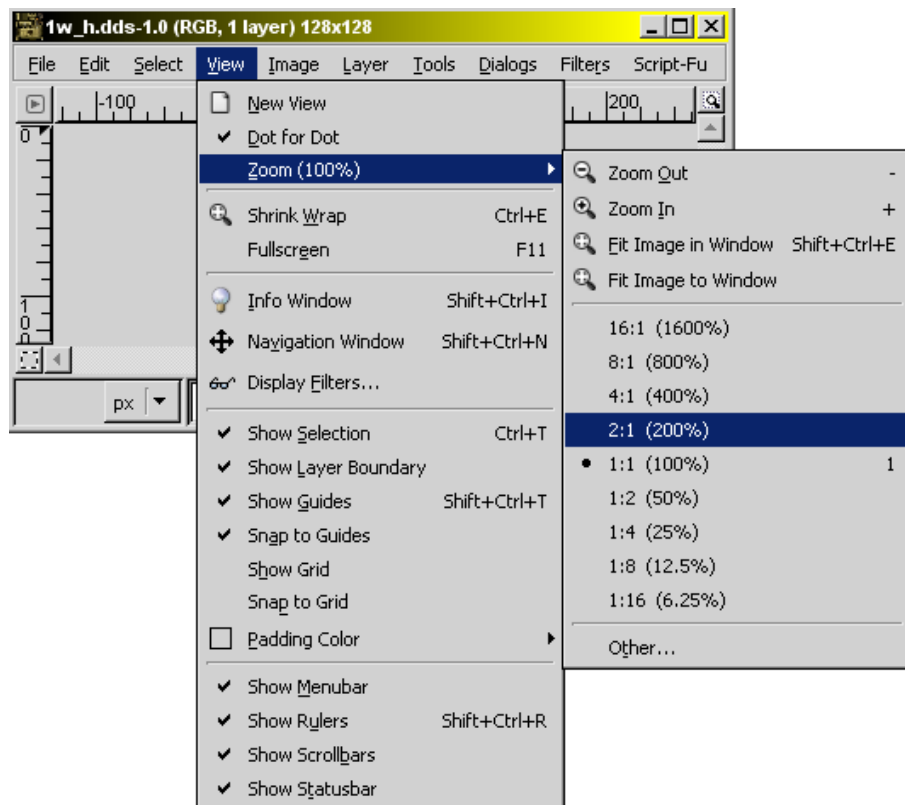
Lets look at the *1w\_h.dds* file.

You will notice that the size of this skin file is 128x128 pixels in size. This important due to how the BK game engine depends on the size of the .dds file.

I need to resized the window a bit, and change the View to increase the display level to make it easier to modify.



<Select> [ **View** ] on the Menu Bar, Use your mouse to Highlight [ **Zoom (100%)** ] and <click on> [ **2:1 (200%)** ].



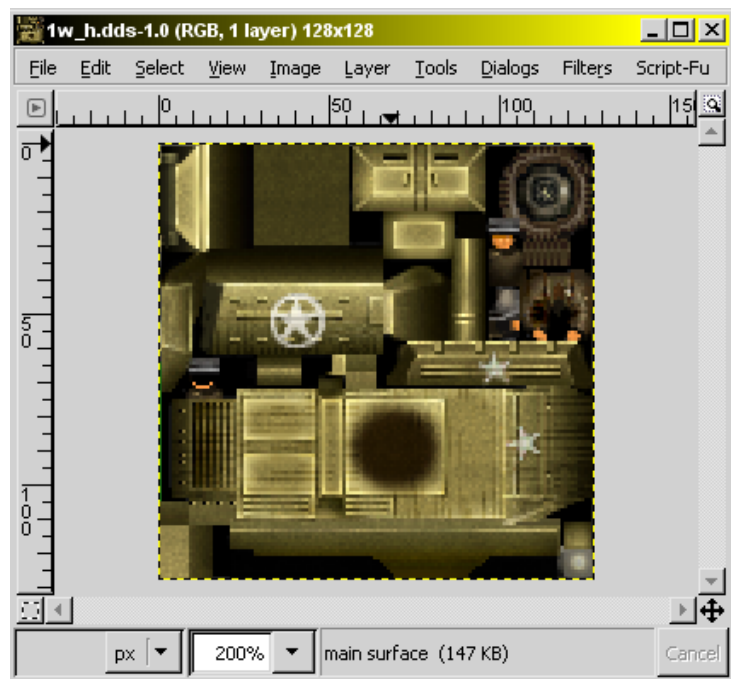


Now I must resize the frame around the window. I will use the mouse and the left button to drag from the bottom left to resize the window.

I can resize the window and zoom level to any size I need but it is a good idea to only size the image to the zoom level needed to get the job done. Obviously, if you are going to do very detailed work, you want it larger. Since the work we are going to do on this skin does not require a lot of detail, this should work just fine.

Examine the skin. Can you see how the pieces fit together? Can you see the soldier?

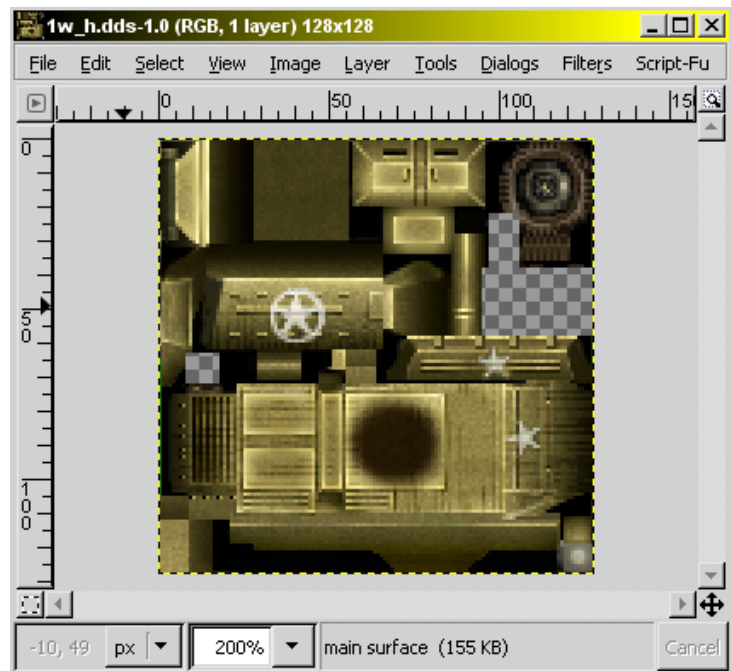
What you may notice is that not all of the parts are situated in a way that are adjacent to each other.



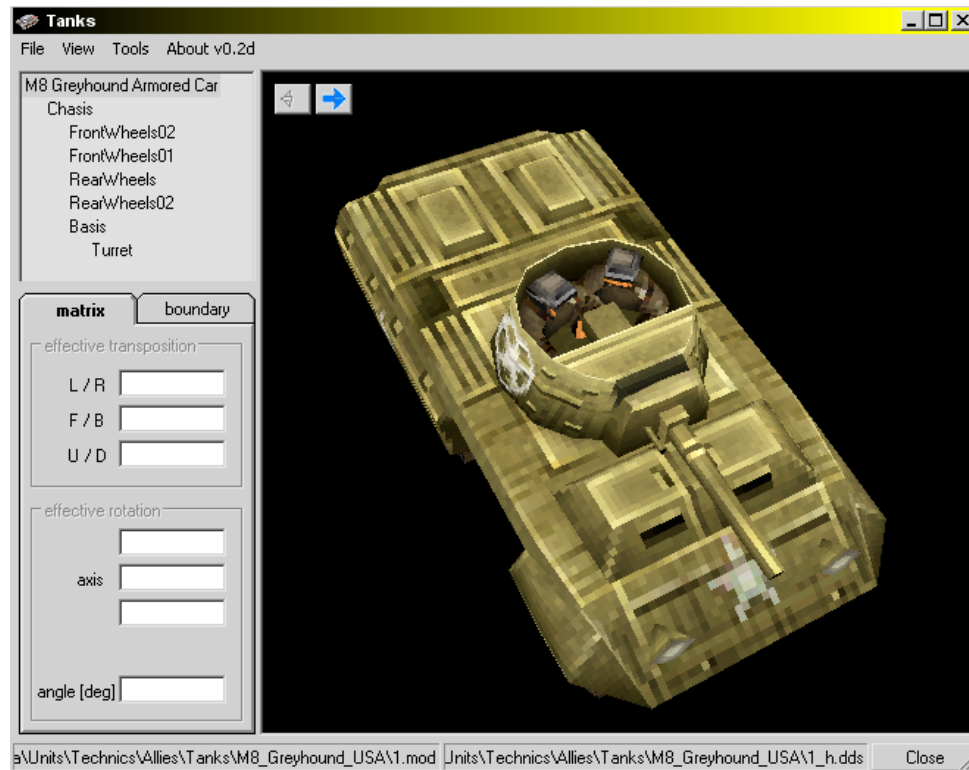
This because when the model is created in a 3d Editor like MAYA, the modeler must un-wrap the model and build what we refer to as the UV. In order to get all of the faces of the model to fit the UV, they are moved into places that provide the best fit for the texture or .dds file. You may have noticed that the soldier's face is not anywhere near his body, and his helmet is scattered around the skin. We also refer to this skin as the texture or the map. You must remember that if you move anything around on this map, the game engine will not know that and will continue to put the various parts of this map where they were before you made any changes. This means if you move the soldier parts, he will not look like a soldier when you view it again.

Example: I'm going to delete the soldier's part of the skin to make my point. In this case, I will use the Rectangle Select or Free Select Function by pressing the "R" key on the keyboard. I can navigate at the **Menu Bar** under [ **Tools** ] to get the same tool. With the Rect Select tool, I can drag a box around anything on the map, then delete that area by Selecting [ **EDIT** ] on the Menu Bar and <select> **Cut**. Then I will **SAVE** the changes: **Menu Bar** [ **FILE** ] [ **SAVE** ].

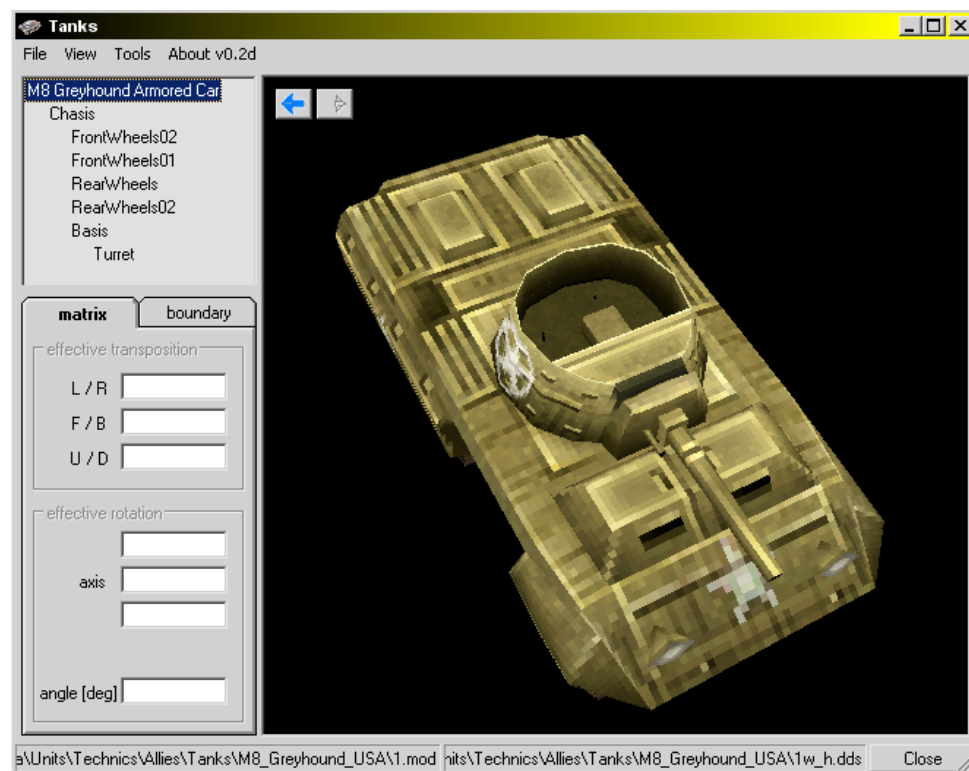
Now compare the Image to the right to the image above. See the difference? Where the little checkered areas are will become transparent on the model. This becomes important later. Let's go to Tanks and see what changed.



Since the Winter Skin is a copy of the Summer Skin, I will go back and look at the Summer Skin first. See the little Arrows on the Model Display? This allows you to flip between texture files.



And now the Winter Skin...



Well, it looks like the Soldiers left the vehicle. If I <right click> on the model and <click on> **Transparency**, they will be displayed again. This is because they are not really gone, they are just transparent or invisible to the game editor.



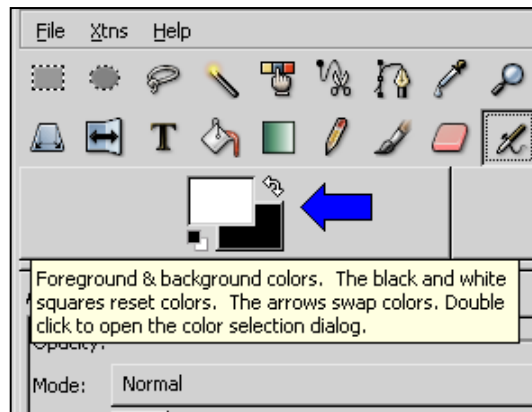
Now since I don't want this to be one of my final edits or modifications, I will UNDO the changes until the soldiers are back on the map. Go to the **Menu Bar**, select [ **Edit** ] then [ **Undo** (action) ]. Then I will SAVE the file to restore it in the folder: **Menu Bar** [ **FILE** ] [ **SAVE** ].

So now you are asking "What does that have to do with winterizing the M8?" Nothing. But it is a demonstration of the potential of this powerful Image Program and the flexibility of the BK game files. Game Modders are discovering something new all of the time. And just remember, they all started new, just like you.

Let's winterize the Winter Skin.

<Select> the Airbrush in the GIMP window, or you can press the "A" on the keyboard. ►►

<Set> Foreground color to **White** by clicking on the Arrows to reverse the colors. ▼▼

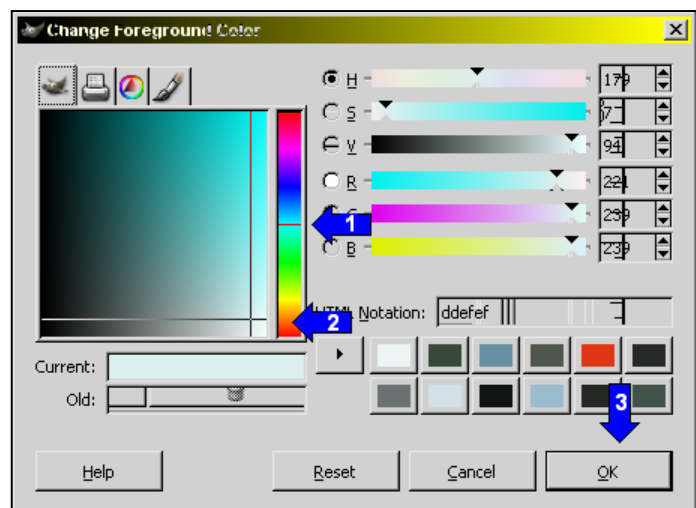
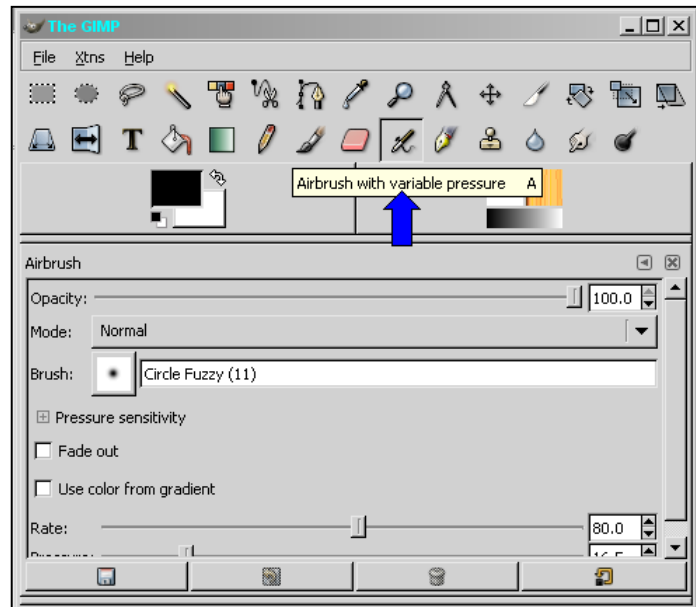


<Double Click> on the **White Box** ▲▲

The **Change Foreground Color** Menu will open. ►►

Let's adjust the white a little.

- 1) <Click> in the middle of the color bar.
- 2) <Click> in the lower right area of the color box.
- 3) <Click> on OK



Note:

The Foreground color has been changed. ►►

We can change the color as needed to darken and lighten the "Snow" tone. Because Snow shows dirt easily, it will not always be pure white. And as we already discussed, US troops usually created a White Wash to brush their vehicles just to break up the green paint scheme. Keeping this in mind, we do not want the M8 to be all white, but it will have shades of white. We will also try to give a hint of new snow which has fallen on the vehicle.



Go to the **Layers** Window.

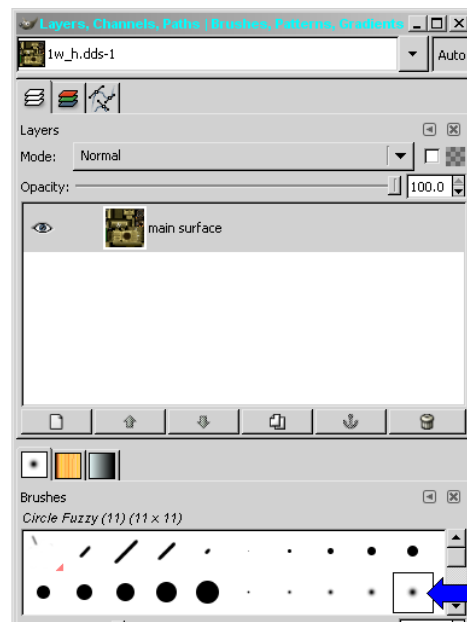
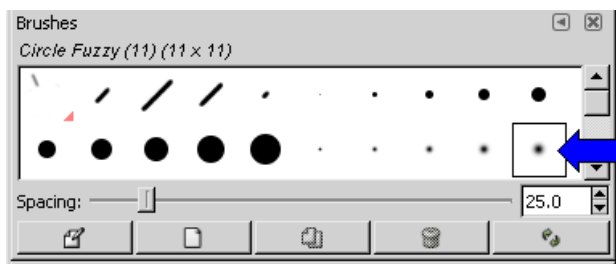
The Layers window gives us a choice of different Brushes to apply our color. ►►

We will leave the Mode: as Normal. ►►

We want the Opacity: set at 100%. ►►

Since we are using the **Airbrush**, we do not want a solid brush.

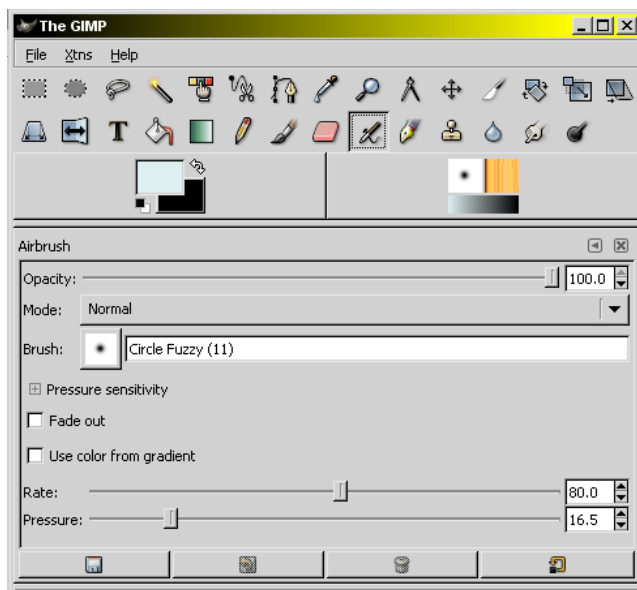
On our skin, I will choose the Circle Fuzzy (11). ▼▼



On the Gimp menu, we can adjust the Rate and Pressure. ►►

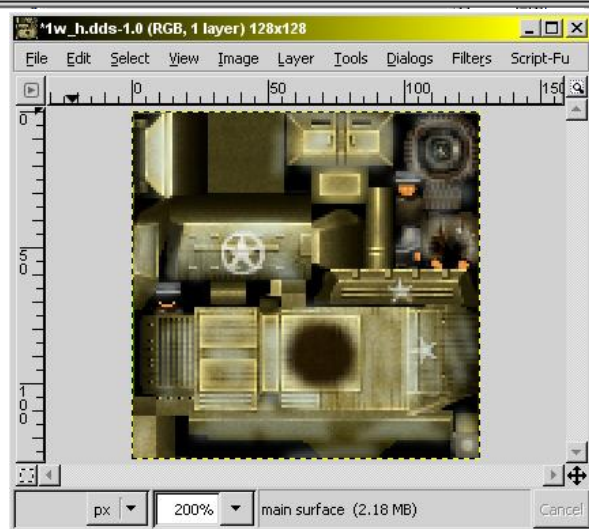
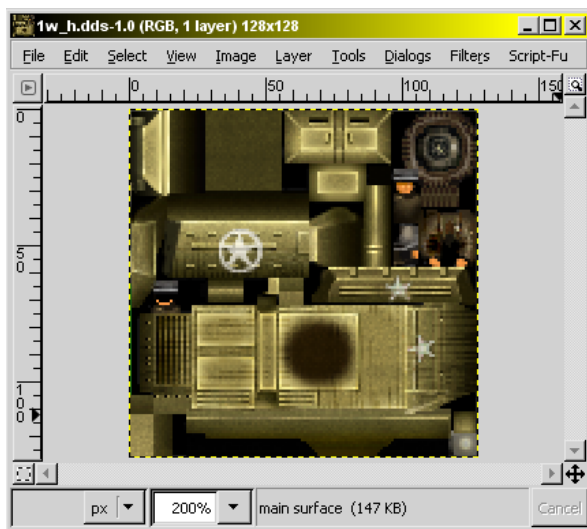
Set the **Rate** at 80 and **Pressure** at 16.5. ►►

Again we can see that the Brush has been set to the Circle Fuzzy (11). ►►



<Select> the *1w\_h.dds* Window, which is the M8 skin map.

We can now apply the airbrush to the skin map to give it the white was appearance. ▼▼



Use the left mouse button to spray/paint using short burst and strokes. We just want to apply a patchy look. As you apply the color, you can <select> **FILE** <Click> **SAVE**, and then view the result in **TANKS**.

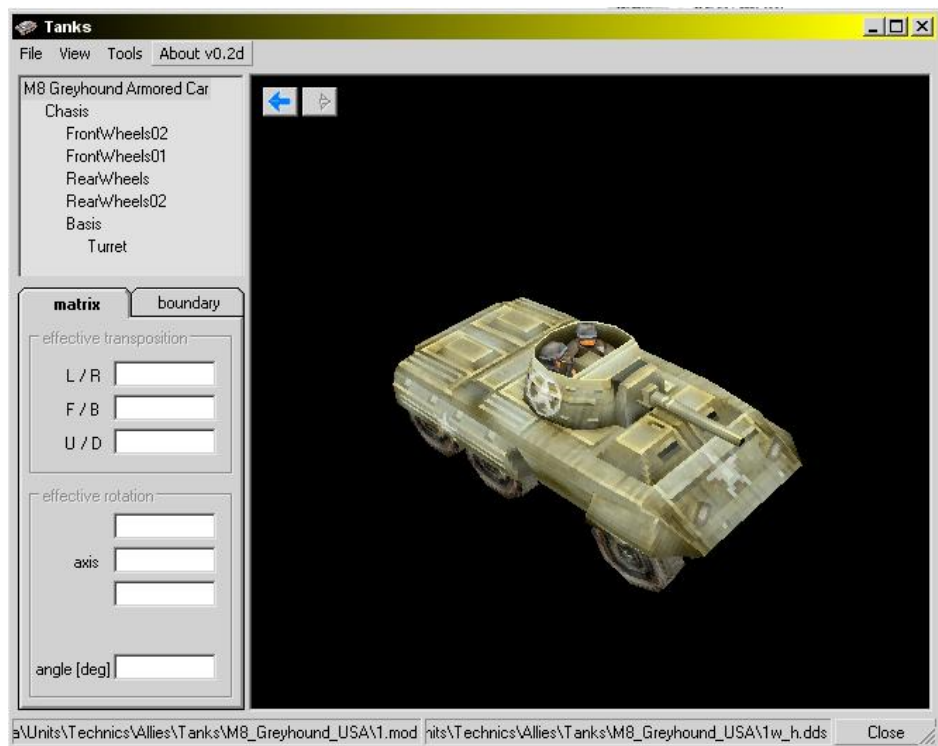
We can compare the modified skin with the desert and summer skin by clicking on the arrows in **TANKS**.

We can always UNDO the last changes: On the skin **Menu** bar, <select> **EDIT** <click> **Undo Airbrush**.

We can always change the brush size and type.

We can change the color that we need to apply.

We can change the Zoom level.



Be sure to rotate and zoom the model in Tanks to look at it from different angles. One important aspect of creating a skin is the highlights or detail of the model. Think about snow covered vehicles that you have seen. The snow varies in tone across the body. Areas closer to the ground are darker, due to the mud/dirt picked up as the vehicle travels. The top surfaces are lighter, whiter and more reflective. Areas around engine parts, vents and fuel caps are also darker, but not as dark as the lower levels. Tires and wheels may also have a whiter tone, but still darker than body parts. Headlights and front areas will usually accumulate snow/ice. Areas between vents, in front of windshields and turrets will also accumulate snow/ice.

The idea is not to make the whole vehicle white, but to blend it with the conditions and break up a pattern. The model should show some original color, or a white wash appearance over the original green.

You can use smaller brushes to detail or highlight for effect.

Once you are happy with the skin, make certain you have saved it and exit TANKS and GIMP.

Copy the 1w\_h.dds and the 2w\_h.dds into the game data folder, and test it in a map to see how it looks.



Learning to use your image software and manipulating the .dds files is really one of the easiest ways you can make worthwhile changes to the game. If you use a base or original unit for a foundation, you can develop additional new units that may be similar or different.

In the base Units, the Allies have the use of the M4A1\_Sherman, M4A3\_Sherman and the M4A3(76)W Sherman. While they were all very similar, and all were Sherman Tanks, they are all quite different. But there were many more Sherman Tanks used by the Allies. If you want to use an M4A2 with a 75mm gun, you can make one by simply copying a M4A3 and modify the skin to suit your needs. If you need an M4A2 with a 76mm Gun, you can copy the USSR version of the M4A2 Lend Lease, but unless you want to use the USSR skin, you will have to again modify the skin. Here are some examples.



Above you see two examples where you can define many versions of a tank from a single model. The skin can define the differences as in the Engine Grilles on the rear of the Tank Hulls. You might also notice the differences in the Bogies Track Wheels. The M4 Sherman Tank is an excellent example of using a single model to create many variations. This can be applied to many objects in BK.

When you skin or create a new texture for a model, it is usually easier to use an existing skin file, the **1\_h.dds**, to base your creation. By looking at the skin, you should be able to identify the various parts of the object. In some cases, some parts or areas will not be identified until you begin to make modifications. This is why the Tank Viewer is so important. Each time you make a change, save the change, then look at the model in the viewer to see how it looks or what has changed. Sometimes a skin or texture may look very good in the Tank Viewer, but not so good in the game. You must keep in mind that very precise details will be lost in the game due to the game resolution. It is important to note that any details that you might wish to stand out will probably appear over-exaggerated in the Tank Viewer.

Keep in mind that the skin file is your palette. You simply have to use your imagination and talent to express yourself. Coloring and toning has a huge effect on the models appearance. Surfaces facing upward should be lighter than side surfaces. Think about how things appear in real life. Lighter colors versus darker colors can show shadows, indentions, and various other effects.

Study other skimmers work or even the original skin textures. You can get a feel for where you want to go when you see where someone else has been. It is also very helpful to find pictures of the real unit when you want to study the colors and details.

It was not so much my goal to teach you how to make a good skin as it was to teach you how to skin and actually be able to manipulate the game files and tools. At this point you must venture forward, learn how to best use your image software, then paint the game the way you think it should be.

Remember: Beauty is in the eye of the beholder.

## PART 3

### .XML Files - UNIT DATA FILES

In this part of the Tutorial we will investigate the mysterious *1.xml* files. Each object folder contains a *1.xml* file that defines the objects parameters.

Here is what the *1.xml* file will look like. Go to any unit or object folder and <Left Click> on the *1.xml*. More than likely your HTML or Internet Browser will open the file and display something that looks like the example.

```
<?xml version="1.0" ?>
- <base>
- <History>

    <PreviousPath>s:\complete\units\technics\german\tanks\pz_v_panther_ausf_d\current.msh</PreviousPath>
    <PreviousDate>29.01.2003</PreviousDate>
    <PreviousTime>16:22:11</PreviousTime>
    <PreviousOwner>Alexander.Valencia</PreviousOwner>
    <Action>Exported as</Action>

    <CurrentPath>s:\versions\current\data\units\technics\german\tanks\pz_v_panther_ausf_d\1.xml</CurrentPath>
    <CurrentDate>29.01.2003</CurrentDate>
    <CurrentTime>16:22:13</CurrentTime>
    <CurrentOwner>Alexander.Valencia</CurrentOwner>
  </History>
- <RPG MaxHP="143" RepairCost="1" Sight="30" SightPower="1" Speed="15.59" Passability="0.42" Priority="2"
  Camouflage="0" Weight="43000" Price="1.03" UninstallRotate="0" UninstallTransport="0" RotateSpeed="44.2"
  TurnRadius="5" TowingForce="0" Crew="5" Passangers="0" BoundTileRadius="2" SmallAABBCoeff="0.5" TowPoint="-1"
  EntrancePoint="-1" FatalitySmokePoint="10" ShootDustPoint="-1" LeavesTracks="1" TrackOffset="0.05"
  TrackWidth="0.2" TrackStart="0.2" TrackEnd="0.2" TrackIntensity="0.7" TrackLifetime="10000" MaxHeight="500"
  DivingAngle="20" ClimbAngle="20" TiltAngle="30" TiltRatio="2">
    <KeyName>Pz V Panther Ausf D</KeyName>
    <StatsType>Mech</StatsType>
    <DamagedHPs />
    <Defence0 MinArmor="40" MaxArmor="90" Silhouette="1" />
    <Defence1 MinArmor="40" MaxArmor="90" Silhouette="1" />
    <Defence2 MinArmor="40" MaxArmor="90" Silhouette="1" />
    <Defence3 MinArmor="40" MaxArmor="90" Silhouette="1" />
    <Defence4 MinArmor="40" MaxArmor="90" Silhouette="1" />
    <Defence5 MinArmor="40" MaxArmor="90" Silhouette="1" />
    <Type>arm_medium</Type>
    <AIClass>track</AIClass>
  - <Commands Size="51">
    - <BitArray>
      <item data="15" />
      <item data="67" />
      <item data="0" />
      <item data="0" />
      <item data="128" />
      <item data="32" />
      <item data="4" />
    </BitArray>
    </Commands>
  - <Exposures Size="0">
    <BitArray />
  </Exposures>
- <AnimDescs>
- <item>
```

It is important to note that you should never attempt to modify the original *1.xml* file. Only modify a copy and be very careful when changing any part of the file. The smallest mistake will cause a game crash, You also must acquire a keen eye to be able to spot issues when a mistake happens. Notice, I said, "When a mistake happens". You will make a mistake, just about everyone does and has, So to protect yourself, always keep the original or a backup *1.xml* available when you are making changes.



## 1.XML FILES - UNDERSTANDING THE PARAMETERS

Lets study some of the basic components of the *1.xml* file. You must understand that *.xml* files maintain a structure much like the tree directory used by your computer in your file system. Different parts of the *1.xml* exist within a type of folder or subfolder.

I prefer to use **WORDPAD** to make changes to the *1.xml* simply because I can use copy/paste when I'm modifying many files at the same time. And even when I am only making changes to one file, I still prefer **WORDPAD**.

Each *1.xml* must include the **Header**; which defines the file as an **xml file**. Any file that does not include this Header will not function properly and will crash the game.

```
<?xml version="1.0"?>
```

The file must include the **<base>** keyword which lets the game engine know that the following data is to be used by the database.

```
<base>
```

The file History usually follows at this point, but is not required by the game engine. It is more useful to establish the History of the unit, the path and name, who made it, what day and what time.

```
<History>
<PreviousPath>c:\units\technics\allies\tanks\m4\current.msh</PreviousPath>
<PreviousDate>12.12.2008</PreviousDate>
<PreviousTime>00:00:00</PreviousTime>
<PreviousOwner>DUNKEL</PreviousOwner>
```

```
<Action>Exported as</Action>
```

```
<CurrentPath>c:\units\technics\allies\tanks\M4A3E2(75)\1.xml</CurrentPath>
<CurrentDate>12.12.2008</CurrentDate>
<CurrentTime>00:00:00</CurrentTime>
<CurrentOwner>MAJORPAIN</CurrentOwner>
</History>
```

Pretty much everything that follows within the file pertains to defining the actual **Parameters** of the object, which in turn tells the game what this thing is, what it does, how it works, the purpose of the unit and various other attributes.

But before we can define anything, we must tell the game engine that we are now defining the parameters with the following keyword:

```
<RPG
```

That now brings us to the actual values and definitions of the object. These are pretty straight forward and do not need much discussion.

PARAMETER VALUE	DESCRIPTION
MaxHP="175"	HIT POINTS Strength of the Object
RepairCost="1"	Cost to Repair or Heal the Object
Sight="31"	Sight Distance that the Object can effectively see
SightPower="1"	Sight Power Factor – used in Sight Calculation as a multiplier for bonus or handicap
Speed="13"	Speed or Velocity of the Object – Higher is Faster / Zero = does not move
Passability="1"	Objects Passability in relation to the Terrain - Each Terrain Type has a movement value
Priority="1"	Priority of the Object when several objects are moving at the same time. Who yields to whom.
Camouflage="30"	How visible the Object is to other objects – higher number is more camo
Weight="40000"	Objects Weight – Important on many objects such as artillery, can it be pushed or must be towed
Price="1"	The Price of the Object, also used in the repair or healing formula
UninstallRotate="0"	Time to uninstall and rotate
UninstallTransport="0"	Time to uninstall from Transport
RotateSpeed="40"	Speed that Object Rotates – higher is faster
TurnRadius="5"	How large is the radius or footprint when static or turning
TowingForce="0"	How much weight can the Object tow – usually only trucks
Crew="5"	The number of crew members on the object
Passangers="13"	The number of passengers that can be loaded in or on the object.
BoundTileRadius="1"	The actual radius of the object in terms of tile space
SmallAABBCoeff="1"	Again refers to the size of the object
TowPoint="-1"	Where the towing point is or if it can tow. Usually a "1" or other number referred to as the towpoint
EntrancePoint="-1"	Location where passengers load
FatalitySmokePoint="-1"	Location that smoke is displayed when damaged
ShootDustPoint="-1"	Whether it leaves dust when moving or shooting
LeavesTracks="1"	Whether it leaves track on terrain / Yes = "1" – No = "0"
TrackOffset="0.15"	The space between the tracks or tires
TrackWidth="0.15"	The Width of the tracks or tires
TrackStart="0.2"	Where the track begin in relation to the model on the footprint
TrackEnd="0.2"	The back end of the tracks / TrackStart-TrackEnd = length of the footprint
TrackIntensity="0.7"	The intensity/depth of the tracks in relation to the object weight
TrackLifetime="10000"	How long the tracks last on the map
MaxHeight="500"	Aviation Max Height
DivingAngle="20"	Aviation Diving Angle
ClimbAngle="20"	Aviation Angle of climb
TiltAngle="30"	Aviation Side to Side Tilt Angle when turning, climbing or diving
TiltRatio="2">	Aviation ratio of tilt in relation to speed of objects

Define the <KeyName> which does not serve any purpose in the game; it is more of a comment here. We also establish the <StatsType>.

<KeyName>M4A3E2(75)Jumbo</KeyName>  
<StatsType>Mech</StatsType>

The following data is the relational data between Damage, HipPoints and Armor. This is defined in the model and limited on effect.

```
<DamagedHPs/><Defence0 MinArmor="40" MaxArmor="90" Silhouette="1"/><Defence1 MinArmor="40" MaxArmor="90" Silhouette="1"/><Defence2 MinArmor="40" MaxArmor="90" Silhouette="1"/><Defence3 MinArmor="40" MaxArmor="90" Silhouette="1"/><Defence4 MinArmor="40" MaxArmor="90" Silhouette="1"/><Defence5 MinArmor="40" MaxArmor="90" Silhouette="1"/>
```

Define what <TYPE> of Unit this is and the <AIClass> (track, wheeled, etc.)

```
<Type>arm_heavy</Type>
<AIClass>track</AIClass>
```

The Command Size and its BitArray define actions that the object will be capable of.  
EXAMPLE: ATTACK, RESUPPLY, AMBUSH, DEPLOY ARTILLERY, ENTRENCH, ETC.

```
<Commands Size="51">
<BitArray>
<item data="47"/>
<item data="67"/>
<item data="0"/>
<item data="0"/>
<item data="128"/>
<item data="32"/>
<item data="4"/>
</BitArray>
</Commands>
```

The Exposure Size and its BitArray define the actions that can affect the object.

EXAMPLE: LOAD

```
<Exposures Size="5">
<BitArray>
<item data="16"/>
</BitArray>
</Exposures>
```

The numbers or values for each <item data> is a Binary number that defines several specific functions that the object may perform or affected by.

Remember the Binary System? 1 and 0 arranged in a Bit Array where 1 is on and 0 is off. The BK Bit Array is an 8-Bit Binary Code.

Location	8	7	6	5	4	3	2	1
VALUE	128	64	32	16	8	4	2	1

Each Location defines a specific function. The Value for that Location is added to the Values of the other Locations that are used. The highest number that can be defined is 255; adding Values for Locations 1-8.

Once you have built your array, it may appear like this example: **01101011** which is Binary Code for the number 107. 107 is entered in one of the data items; <item data="107">. When the Game Engine looks at this code, it reverses the code back into Binary and now has the specific functions the object can perform.



The Commands and Exposures are best defined by the following chart.

BLITZKRIEG COMMAND SIZE (BIT ARRAY) / EXPOSURES (BIT ARRAY)												
UNIT TYPE =		INFANTRY										
				BIT ARRAY VALUES								
NOTES	X	COMMAND	SIZE	D1	D2	D3	D4	D5	D6	D7	D8	VALUE
	X	MOVE TO	1	1	1							1
	X	ATTACK UNIT	2	2	2							2
	X	ATTACK OBJECT	3	4	4							4
	X	SWARM TO	4	8	8							8
	X	LOAD	5	16	16							16
		UNLOAD	6	32	0							20
	X	ENTER	7	64	64							40
	X	LEAVE	8	128	128	0						80
	X	ROTATE TO	9	0	1	1						100
	X	STOP	10	0	2	2						200
	X	PARADE	11	0	4	4						400
		PLACE MINE	12	0	8	0						800
		CLEAR MINE	13	0	16	0						1000
		GUARD	14	0	32	0						2000
	X	AMBUSH	15	0	64	64						4000
		RANGE AREA	16	0	128	0	0					8000
		ART BOMBARDMENT	17	0	0	1	0					10000
		INSTALL	18	0	0	2	0					20000
		UNINSTALL	19	0	0	4	0					40000
		PARADROP	23	0	0	64	0					400000
		RESUPPLY	24	0	0	128	0	0				800000
		REPAIR	25	0	0	0	1	0				1000000
		BUILD FENCE BEGIN	27	0	0	0	4	0				4000000
		ENTRENCH BEGIN	28	0	0	0	8	0				8000000
		USE SPYGLASS	30	0	0	0	32	0				20000000
		TAKE ARTILLERY	32	0	0	0	128	0	0			80000000
		DEPLOY ARTILLERY	33	0	0	0	0	1	0			100000000
		PLACE ANTITANK	34	0	0	0	0	2	0			200000000
	X	DISBAND FORMATION	35	0	0	0	0	4	4			400000000
	X	FORM FORMATION	36	0	0	0	0	8	8			800000000
	X	FOLLOW	40	0	0	0	0	128	128	0		8000000000
REINFORCE		RESUPPLY HUMANS	44	0	0	0	0	0	8	0		80000000000
	X	ENTRENCH SELF	46	0	0	0	0	0	32	32		200000000000
		REPAIR OBJECT	48	0	0	0	0	0	128	0	0	800000000000
ENGINEER		BUILD BRIDGE	49	0	0	0	0	0	0	1	0	1000000000000
		RESUPPLY MORALE	50	0	0	0	0	0	0	2	0	2000000000000
	X	STAND GROUND	51	0	0	0	0	0	0	4	4	4000000000000
MEDICAL		HEAL INFANTRY	52	0	0	0	0	0	0	8		8000000000000
		FILL RU	55	0	0	0	0	0	0	64	0	40000000000000
		CAPTURE ARTILLERY	57	0	0	0	0	0	0	0	1	100000000000000
		COMMAND SIZE =	51	223	71	0	0	140	32	4	0	
				BIT ARRAY VALUES								

This is a view of a Binary Code Function / Bit Array Calculator that I created to make it easier to determine the specific values required in the correct array locations for the object I was creating. When I wish to define the Commands, I simply place an "X" in the 2<sup>nd</sup> column from the left, and the calculator reports what values I need to place into each of the <item data=" "> locations. This is much easier than trying to experiment. I might mention, not all functions are possible for all objects and some combinations cause strange things to happen to the object. Example: A tank cannot tow artillery and still be defined as a tank; it becomes a truck that looks like a tank.

You might notice that not all binary codes are used within each D# section. D3 is the 3<sup>rd</sup> <item data=" "> location in the Command Array. D1 and D2 do use all of the binary codes, while D3 through D8 do not. Why is this? I can tell you through my own research that these codes were set aside for a specific use, but for some reason, not used or at least not documented. I have explored this issue over the years and have concluded that some undocumented codes do in fact serve a purpose, but not anything that I want to disclose at this time. Perhaps when I complete my research, I can announce some useful Commands that were not documented by the original authors. All of these codes are deeply embedded within the game source code and cannot be changed or manipulated, so trying to define the unknown has been very time consuming and challenging.

Look at the SIZE Column. This number represents the Objects basic Command Function.

**Examples:**

Infantry = 51 or Stand Ground  
Transport = 44 or Resupply Humans  
Engineer = 49 or Build Bridge  
Medical = 52 or Heal Infantry

The <item data=" "> values simply define the other functions that are possible for that object.

The **Value** Column is the number which is expressed inside of the Resource Editor if that program is used to define the Commands. This number is re-defined into hex code, or what we refer to as machine code or machine language. So this one simple section of the **1.xml** has a mixed bag of information, expressed as numbers in three forms of mathematics, everyday numbers that we use, binary code and hexadecimal code.

This section of the **1.xml** is where most modelors have problems with their objects because there is not a handbook or college level textbook that teaches you the way to solve this issue. Now you have some of the answers at your fingertips.

When you build the **Objects Bit Array**, you have a value for each of the <item data=" "> locations. If there is no value for a location, then the number is 0 (zero). If the last location or locations are zero, they are not defined within the Command Array. Using </BitArray> followed by </Commands> tells the game engine that all of the data for that object has been defined.

I had defined an Infantry Unit in the Calculator so the Command Section will look like this:

```
<Commands Size="51">  
<BitArray>  
<item data="223"/>  
<item data="71"/>  
<item data="0"/>  
<item data="0"/>  
<item data="140"/>  
<item data="32"/>  
<item data="4"/>  
</BitArray>  
</Commands>
```

[illegible]

<AcksRefs><item>sounds\ack\allies\tank3-multipurpose</item></AcksRefs><AcksRef></AcksRef>

You can however change the `HorizontalRotationSpeed` and the `Constraint` data, but you must be careful to not provide a number which is out-of-bounds for the Object. The out-of-bounds threshold is different for every model or object, so you can't really compare several and then just throw one in.

```
<Platforms>
<item VerticalRotateSpeed="1" HorizontalRotationSpeed="1" ModelPart="1" GunCarriageParts="-1" FirstGun="0"
NumGuns="1"><Constraint Min="0" Max="0"/><ConstraintVertical Min="0" Max="0"/></item>

<item VerticalRotateSpeed="1" HorizontalRotationSpeed="6.4" ModelPart="2" GunCarriageParts="-1" FirstGun="1"
NumGuns="2"><Constraint Min="-6.28319" Max="6.28319"/><ConstraintVertical Min="0" Max="0"/></item>
</Platforms>
```

And now we are the GUNS. Here we can change the 'Ammo' carried and the 'Weapon' defined. We cannot change the 'Shootpoint' or 'Direction'. We can change the 'ReloadCost', 'Recoil' and the 'RecoilLength'. You cannot change the 'ModelPart'. Guns 1 and 2 are the 7\_62mm\_browning\_m1919a4 machine gun. We can select different machine guns from the weapons folder. The Main Gun is the 76mm\_m1a1. We can change this to another gun, but make certain that it makes sense. You would not replace this 76mm with a 105mm since the barrel length of the 105mm is much shorter. But you could change the 76mm to a 17pdr, which is also a 76mm. The 17pdr was the Main Gun used in the British Firefly Sherman.

```
<Guns>
<item Priority="1" Ammo="3000" Direction="65535" ReloadCost="1" ShootPoint="4" Recoil="0" RecoilLength="0"
RecoilTime="50" ModelPart="1" RecoilShakeTime="500"
RecoilShakeAngle="0"><Weapon>7_62mm_browning_m1919a4</Weapon></item>

<item Priority="1" Ammo="3000" Direction="65535" ReloadCost="1" ShootPoint="8" Recoil="0" RecoilLength="12.5664"
RecoilTime="50" ModelPart="2" RecoilShakeTime="500"
RecoilShakeAngle="0"><Weapon>7_62mm_browning_m1919a4</Weapon></item>

<item Priority="0" Ammo="97" Direction="65535" ReloadCost="1" ShootPoint="9" Recoil="1" RecoilLength="15.8914"
RecoilTime="50" ModelPart="3" RecoilShakeTime="500" RecoilShakeAngle="0"><Weapon>76mm_m1a1</Weapon></item>
</Guns>
```

Here are the objects Armor values. Notice you have the Minimum and Maximum Values represented in thickness. You can change these, but I suggest that you keep this pretty reasonable. It is quite possible to build a super-tank that can repulse almost any heavy gun and impervious to attack. But what is the point of an easy killer with easy victories? Using real life data is very useful here.

```
<ArmorLeft Min="50" Max="150"/>
<ArmorRight Min="50" Max="150"/>
<ArmorTop Min="22" Max="30"/>
<ArmorBottom Min="18" Max="27"/>
<ArmorFront Min="150" Max="200"/>
<ArmorBack Min="50" Max="150"/>
```

Past this, there is very little that can be changed. Most of this data is specific and defined in the 3D Software when the Model (object) was created.

```
<AABBCenter x="0.0905523" y="1.33224"/>
<AABBBHalfSize x="24.2538" y="51.9237"/>
```

```
<ExhaustPoints>
<item data="6"/>
<item data="7"/>
</ExhaustPoints>
```

```
<DamagePoints><item data="5"/></DamagePoints>
```

```
<PeoplePoints/>
```

```
<TowPoint2D x="0" y="0"/>
<HookPoint x="0" y="0"/>
<FrontWheel x="0" y="0"/>
<BackWheel x="0" y="0"/>
<EntrancePoint2D x="0" y="0"/>
<PeoplePoints2D/>
<AmmoPoint2D x="0" y="0"/>
```

```
<Gunners>
<item><data/></item>
<item><data/></item>
<item><data/></item>
</Gunners>
```

```
<EffectDiesel>smokediesel1</EffectDiesel>
<EffectSmoke>smokeflame1</EffectSmoke>
<EffectWheelDust>dustmove1</EffectWheelDust>
<EffectShootDust></EffectShootDust>
<EffectFatality>expfatality1</EffectFatality>
<EffectEntrenching></EffectEntrenching>
<EffectDisappear>brakedown2</EffectDisappear>
```

```
JoggingX Period1="1" Period2="0.3" Amp1="1" Amp2="0.6" Phase1="0" Phase2="2.1"/><JoggingY Period1="0.7" Period2="0.2"
Amp1="0.6" Amp2="0.4" Phase1="1.9" Phase2="0"/><JoggingZ Period1="1" Period2="0.3" Amp1="1" Amp2="0.6" Phase1="0"
Phase2="2.1"/>
```

You can change the Sound Files and the Death Craters.

```
<SoundMoveStart>Tank11_custom_start</SoundMoveStart>
<SoundMove>Tank11_custom_cycle</SoundMove>
<SoundMoveStop>Tank11_custom_stop</SoundMoveStop>

<DeathCraters>
<item>objects\terraobjects\death_hole\verybig\01\1</item>
<item>objects\terraobjects\death_hole\verybig\02\1</item>
<item>objects\terraobjects\death_hole\verybig\03\1</item>
</DeathCraters>
```

This is the End of this 1.xml. The Game Engine reads the </RPG> followed by </base> and moves to the next object file. All of the specific data which defines this object now resides within the Game Data-Base.

```
</RPG>
```

```
</base>
```

I suggest that you study several **1.xml** files from various objects as you begin to create or modify the **1.xml** file for a specific purpose. You don't want to define something that was impossible or improbable. If you plan on releasing some projects into the Public, it won't be accepted very well if the object isn't based upon known and applied physics and mathematics.

Remember the Sherman Tanks created in Part 2? I created the **M4A2(75)** which was based on the **M4A3(75)**. Now that I have all of the basic knowledge at my fingertips, I can truly create a new tank. I can create a new skin specific to that tank. I can now create a new **1.xml** file which defines the parameters of the new tank. Then I can create a new tank folder which holds all of the tanks files. If I place this tank folder into my Game Data folder I can use the new tank in the game once I index the new Tank in the **objects.xml**. And since I know where everything must go, I have observed the correct file structure. And finally, I can share my new tank with anyone else because I can create a new **Tank.pak** file that can be sent to anyone.

So where most books and manuals have a conclusion, this one is quite different. There is no ending. Rather this is the beginning for you. Now it is your turn to write new chapters, create new objects, and become the teacher for others.

I was greatly inspired to write this **BK "HOW TO" Tutorial** by the vast number of new BK Gamers who wished to learn the games secrets. Here I have tried to expose those secrets, and what we have found is nothing more than a buildable open source file program that is quite managable and changeable. There are truly very few secrets to Blitzkrieg. Many before me and many after me have or will discover additonal information and possibilities. We are only limited by our imagination.

If you find errors contained within the text or you wish to see something addressed in more detail, feel free to contact me. And I would sure appreciate hearing from you on whether this helped you.

Thank you for taking the time to read this Document. I truly hope that it has taught you something and you will find the information useful. It has been my pleasure to present it to you and I wish you the best of luck in your quest to build a better Blitzkrieg.

Major Pain