

ED1 File Packer Lab

# Objective

This lab requires you to write the functionality to load the directory of a PAK file, load a file’s content from the PAK file to a buffer, and supplementary functions. The end result of this lab will be the loading of texture assets from a PAK file to be used in the game engine. You will need a PAK file with assets to load to complete the assignment, one is provided to you on the student VFILER, a PAK generator is also supplied if you want to create your own.

# Implementation

You will be completing the following functions in the EDUtilities project -> PakFile filter -> PakFile.cpp

**ClearPak** –clears any internal memory and close any open file streams, this function will be called when the game engine is being shut down

**LoadDirectory** – opens a stream to a pak file and load its directory into memory, this function will be called when the game engine is initializing

**IsInPak** – checks the loaded directory to see if the file exists, this function will be called by the asset manager to see if it should bother loading a texture from the PAK, you can also use it in **ExtractAsset** to make sure a given asset is in the PAK

**LoadAsset** – loads a file from the PAK into a buffer, if an asset is in the PAK the asset manager will call this function to load the asset

**ExtractAsset** – extracts a file from the PAK to the file system, to invoke this functionality pull down the input console (~) and use the following, “ExtractAsset filename.ext”. When writing this function you must use a fixed size buffer to read / write with, at first you should do it the easier way using a large buffer. To test fully try to extract all 3 assets from the provided the PAK, CompanionCube.dds, jeepbmp.dds, and VZ\_Robot\_Texture.dds, the default path is the FSGDGame folder, this is a debugging function that will be invoked by the user

**FindAssetName** – you do not need to write this function (unless you want to be a pro), it is largely a string manipulation function (look into string tokens). You will need to call it from **IsInPak**, **LoadAsset**, and **ExtractAsset**, it simply removes folders from a path name e.g. “//folder1//file.txt” becomes “//file.txt”.

# Rubric

Commenting should be appropriate and used where necessary.

Your completed lab should be warning free.

Your completed lab should be error/crash free, a submission that crashes will result in a 0%.

Your project should be free of memory leaks.

Failure to follow directions will result in the loss of additional points.

|  |  |
| --- | --- |
| Points | Description |
| 10 | ClearPak |
| 15 | LoadDirectory |
| 10 | IsInPak |
| 30 | LoadAsset |
| 35 | ExtractAsset |

# Hints

* Check the CPP Reference site for specifics on how STL containers and objects work if you need more information on them
* In the ExtractAsset function you must use a small fixed size buffer, this means you cannot just call the LoadAsset function to complete this function.

# Submissions

Lab is due at **the end** of the lab period. Have a lab instructor grade your assignment; you still need to turn in the assignment on VFILER, grades will be posted on LMS. If you are off campus, late assignments can be turned in through LMS.

This lab must be turned in using the .zip file format using as *LastName.FirstName.lab\_name.zip*.

To create the zip file, run the “make submission folder.bat” file. It will create a folder one level higher called “zip\_this\_and\_turn\_in,” leave it named that. Zip it, rename the zip file and turn it in on VFILER