Assignment 09

The canvas assignment is **here (https://utah.instructure.com/courses/512907/assignments/5250596)** .

The files that you need can be downloaded **here (https://utah.instructure.com/courses/512907/files/82232212/download?wrap=1)** .

Requirements

- Create a human-readable Effect file format
- Update AssetBuildSystem.lua to handle Effects
  - In addition to the functions that you already know about (that you had to implement for meshes) you will also have to add a `RegisterReferencedAssets()` function for Effects
- Create an EffectBuilder project that reads in a human-readable Effect file and writes out a binary Effect file
  - You will need to update your solution with the **provided cRenderState files (https://utah.instructure.com/courses/512907/files/82232212/download?wrap=1)** so that you can use the render state functions in your EffectBuilder just by #including the cRenderState.h file (I have changed the render state bit functions to be defined inline)
- Convert your hard-coded effects into human readable files and add them to the Asset Build System
  - Remove the shaders that your effects use from AssetsToBuild.lua
    - (If you have correctly implemented the RegisterReferencedAssets() function for Effects then the shaders will automatically get built)
  - (You should still explicitly list your vertex input layout shader because it isn't referenced by any effects, but no other shaders should be listed)
- Update your run-time code to load the binary Effect files instead of using your hard-coded ones
  - You should be able to figure out how to do this based on previous Mesh assignments and our discussion in class
- Your write-up should:
  - Show us a human-readable Effect file
  - Show us the binary version of the same Effect file (a screenshot from a hex editor is fine if it's readable)
    - Explain how your run-time code knows where the second path in the file starts
    - Tell us whether the paths you store are relative to $(GameInstallDir) or $(GameInstallDir)/data. Explain why you made this decision. (You must explain the advantages and disadvantages of your choice or else you will lose points.)
  - Show us how you extract the two paths at run-time
  - (You should still show a screenshot to keep your readers interested even though nothing is different visually)

Submission Checklist

- Your write-up should follow the **standard guidelines for submitting assignments (https://utah.instructure.com/courses/512907/pages/submitting-assignments)** and the **standard guidelines for every write-up (https://utah.instructure.com/courses/512907/pages/write-up-guidelines)**

- There should be no shaders listed in the AssetsToBuild.lua file except for your vertex input layout shader
    - If you delete $(TempDir) and build assets then all of the other shaders should be built (because they are referenced by effects)

Finished Assignments

- Yitong Dai (https://yzd0014.wixsite.com/dyt1205/blog/eae6320-effect-file)

Details

RegisterReferencedAssets()

- This function must examine the effect file and register the two shaders that the effect references
- The function will get a relative path to the source effect as input. That means your implementation should look something like the following:

    -
      ```
      NewAssetTypeInfo( "effects",
        {
          RegisterReferencedAssets = function( i_sourceRelativePath )
            local sourceAbsolutePath = FindSourceContentAbsolutePathFromRelativePath( i_sourceRelat
      ivePath )
            if DoesFileExist( sourceAbsolutePath ) then
              local effect = dofile( sourceAbsolutePath )
              -- EAE6320_TODO Get the source shader paths from the effect table and then do somethi
      ng like:
              RegisterAssetToBeBuilt( path_vertexShader, "shaders", { "vertex" } )
              RegisterAssetToBeBuilt( path_fragmentShader, "shaders", { "fragment" } )
            end
          end
        }
      )
      ```

Binary Effect Files

- The shader paths that are in your human-readable source Effect file should be source shader paths, but the shader paths that are in your binary built Effect file should be the paths to the binary shaders in the install directory. That means that you will have to convert them in your EffectBuilder tool.
    - There is a `ConvertSourceRelativePathToBuiltRelativePath()` utility function in AssetBuildLibrary that can be used to do this (it will use the Lua implementation of this function defined in AssetBuildFunctions.lua)
- Even though the paths themselves are text, when they are part of a binary file the priority you need to keep in mind is what the best format is for reading them in at run-time. Some things to consider:
    - You only need the paths while loading the file
        - (In other words, you don't need to store them. When you extract them from the loaded binary data the pointers can point into that binary data as long as you only use those pointers during load-time.)
    - NULL termination
        - You don't technically need to terminate the strings with a NULL character. However, in order for char arrays to be used as strings at run-time a terminating NULL is necessary. If your file

        doesn't include a terminating NULL then you will have to add one at run-time, and that is expensive. Since a terminating NULL is only one byte it is definitely worth adding to the file so that the run-time can use the data directly as a string.
    - How does the run-time know where the fragment shader path starts?
        - One option is to calculate this at run-time. (Can you think of how?)
        - Another option is to add this information to the binary file
            - If you do this then the size of the file increases and you have to extract the information at run-time, but you save the run-time cost of having to calculate where the fragment shader path starts
            - How small can you make the information that tells where the fragment shader path starts?
    - What should the path look like?
        - In the Asset Build System the paths are relative to $(GameInstallDir)data/, but at run-time when you load a file the current working directory is $(GameInstallDir). That means that you have two options:
            - Store the paths relative to $(GameInstallDir), and at run-time add a data/ prefix to them after extracting them
            - Add the data/ prefix at build-time
        - One option decreases file size but increases run-time cost. The other option increases file size but decreases run-time cost. Which do you think is better?

Optional Challenges

- Can you make your shader files more platform-independent?
    - You can use the preprocessor to make your shader files mostly platform-independent
    - You should be able to declare constant buffers in a platform-independent way
    - You should be able to make the main() function's implementation (everything within the curly braces) platform-independent
    - You probably need to keep the input and output declarations platform-specific