use Adobe Reader to complete

(Type in fields)

PA5 – Converter

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: [✓ Yes] No

Name:	Robert Grier
Date:	11/11/23

Submission Details

Final ***Changelist*** number:	199051
Verified build:	[✓ Yes] No
YouTube Link:	https://youtu.be/jlm6jhYMiWo
Required Configurations:	Any - just need one that works

Discussion (What did you learn):

I learned how to modify my converter to handle large meshes, multi-mesh, and other variations of glb files. I also learned how to load them into the engine quickly. I added bounding sphere's calculated in the converter as well as texture md5 hashes.

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.cpp, *.h
  - *.vcxproj, *.vcxproj.filters, *.vcxproj.user, CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Points will be deducted if minimum is not reached

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level ALL …
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work "as-is".
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
  - It will not be graded and you get a 0

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
    - No automatic containers or arrays
    - You need to do this the old fashion way - ***YOU EARNED IT***

Leave Project Settings

- Do NOT change the project or warning level
    - Any changing of level or suppression of warnings is an integrity issue

Simple C++

- No modern C++
    - No Lambdas, Autos, templates, etc…
    - No Boost
- NO Streams
    - Used fopen, fread, fwrite…
- No code in MACROS
    - Code needs to be in cpp files to see and debug it easy
- ***Exception:***
    - implicit problem needs templates

Leaking Memory

- If the program leaks memory
    - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
    - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
    - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
    - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
    - All files must be active to get credit.
    - Better to lose points for unit tests than to disable and lose all points

UnitTestConfiguration file (if provided) needs to be set by user

- Grading will be on the UnitTestConfiguration settings
    - Please explicitly set which tests you want graded… no regrading if set incorrectly

Due Date

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***<u>Submission Report</u>*** and commit to perforce
  - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
  - Fill out the form and discussion for full credit.

Goals

- Write a standalone model converter
  - GLTF models with textures exports to ***YOUR*** custom Google ProtoBuff format.
  - Command line commands allowing the ability be used in a Batch file
  - No loose TGAs… only proto.azul files
- Export at least ***4 models*** with large polygon count
  - GLTF models with textures, verts, norm, uv, index data

Assignments

1. **Write a standalone model converter**
   - Stand-alone runtime time converter
     - Takes GLTF models with textures and exports to ***YOUR*** custom Google ProtoBuff format.
     - Jedi Modification (optional)
       - Add command line commands allowing the ability be used in a Batch file or a python file
       - Hint: See the Basics9 from CSC461 on parsing
   - You are allowed to hard-code
     - If you need to make "hard coded" modifications in your conversion tool
     - Its OK… sometimes it's hard to have a general-purpose solution at first

   - Converts data to your runtime ProtoBuff format
     - VBO runtime format
       - Verts, norms, uv, index buffers
     - Texture
       - Texture needs to be embedded into ProtoBuff
       - Texture needs to be RAW no texture conversion in Engine
       - ***Store the MD5 hash for each texture***
     - Bounding sphere
       - ***Calculate bounding sphere and store data into ProtoBuff***
     - Any necessary data miscellaneous data

        - Num verts, name, version number of converter, etc…
- Export at least ***4 models*** with large polygon count
    - One from each of the categories
        - Group A – small poly count
        - Group B – large poly count
        - Group C – multi-mesh
        - Group D - Student's favorite models
    - Group D - models
        - We will have a Group_D directory in /Common
            - Students can push any model they would like to use
            - Share your findings of models – push into perforce
        - Find a GLTF – binary version GLB with texture
            - If you can only find a FBX format…
                - you can convert it with the FBX2GLB.exe
        - Required elements on model
            - Vert, norm, uv, index buff with a Texture
            - Needs to be more than 1K verts in count
            - Texture needs to be of type png or tga

2. Game engine needs to be modified to read your custom binary file

- Engine should be able to load file data once
    - Creating temp buffers for file
    - Load the data (Fields, VBOs & Textures) into graphics memory
        - Remember you need to embed your raw data from the texture into your archived data set
        - No conversion of texture is happening in engine
    - Free temp buffer

3. Video

- ***Record the demo*** **(in your Viewer)**
    - Video demo of key aspects of your code (code review) and show your converter in action.
        - Make sure each model is visible in your engine
            - Either drive the camera or spin the models
                - Show the complete model with textures
        - Please make it easy to convert the models…
            - Add a batch file or have a function that can convert your models
            - This way its repeatable
    - Place video on YouTube, place link in PDF

Validation

- Make sure program build and run without crashing
    - Converter
    - Game Engine
- Submit the data to perforce
    - Any source model data and texture that you used
    - Exported data of the model
    - AzulConverter.exe
- YouTube
    - Movie recording – showing the converter and viewer

Hints

- Do this assignment by iterating and slowly growing your project
    - You won't be able to finish this assignment in one day - Start now

Troubleshooting

- Focus Input format
    - Get input format working first
        - Command line
    - Make sure you can load and display the format
    - Modify and change as needed
- Focus on converter – follow the ***wooden_crate*** demos from lecture
    - Create data into an internal temporary format
        - Print the verts
        - Print the data…
    - Next convert the data to your input format
- Extra step that REALLY helps
    - Create a standalone program that reads the input binary format
    - Read and print that to a file
    - This way you can visualize your data.

BABY STEPS!