# PA4 – SpaceFrigate - Converter/Viewer

## 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: Rachel Micolichek

Date: 11/20/23

## Submission Details

Final ***Changelist*** number: 198256

Verified build: (•) Yes ( ) No

Number Tests Passed: 6

Required Configurations: DEBUG only

Discussion (What did you learn):

Based on the lecture material, I was able to get the converter working but I struggled with the unit tests. Specifically copying the VBO buffers / Texture into a buffer. I wasn't getting the correct MD5 value. If I checked locally, it had the expected value but because of how I was using memcpy, the value passed back wasn't right. Once I got that fixed, I was able to get the unit tests passing

## 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, 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

**No Adding files to this project**

- This project will work “as-is” do not add files…
- Grading system will overwrite project settings and will ignore any student’s added files and will returned program to the original state

**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 Dates

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** 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

- Convert a GLTF (glb – binary) into custom format
    - Use TinyGLTF
    - Use JSON files
- Google protobuf format
    - Place converted model with texture into ONE protobuf
    - Use protobuf file as the real-time asset in viewer
- Viewer
    - Display space frigate from protobuf file

## Assignments

- Directory has a unified Converter, Engine, Library structure
    - You are welcomed to use your own directory/project structure
    - Centralized directory
        - ProtoBuf - mesh, vbo, texture proto
        - Proto – lib, conversion applications
        - TinyGLTF - include /libs
        - Shaders – glsl files
        - Models – source models/texture to convert
        - Data – Engine's working directory
        - Math
            - *Replace src/include with student version*
        - PCSTree
            - *Replace src/include with student version*
        - File
            - *Replace src/include with student version*
        - Manager – source/include
        - Time – time src/include

- Two working projects
    - Engine
        - *Replace src/include with student version*
    - Converter
        - *Replace src/include with student version*

**STEP 1: -- Student libraries ---**

- Engine work and links after you add your libraries
- Add your src/include
    - Math– Engine library
        - *Replace src/include with student version*
    - PCSTree – Engine Library
        - *Replace src/include with student version*
    - File – Engine Library
        - *Replace src/include with student version*
- Compile and run Engine…
    - It should work

**STEP 2: -- Converter ---**

- Converter
    - Input:
        - /Models/
            - space_frigate.glb -- binary gltf
            - space_frigate.tga – texture external ref inside glb
            - space_frigate.fbx - not used original file
        - Converts into a protobuf file
            - Sample name: space_frigate.proto.azul
                i. Suggestion… can be anything
- Project links/compiles as-is
    - Add files as you want
        - .proto
        - classes
            - include / src
    - Do not include space_frigate.glb use the working directory
        - There is a example of the pathing already working
- This program converts GLTF file and generate the protobuf
    - Protobuf will be create and copied into the /Data runtime directory

**STEP 2: -- Engine ---**

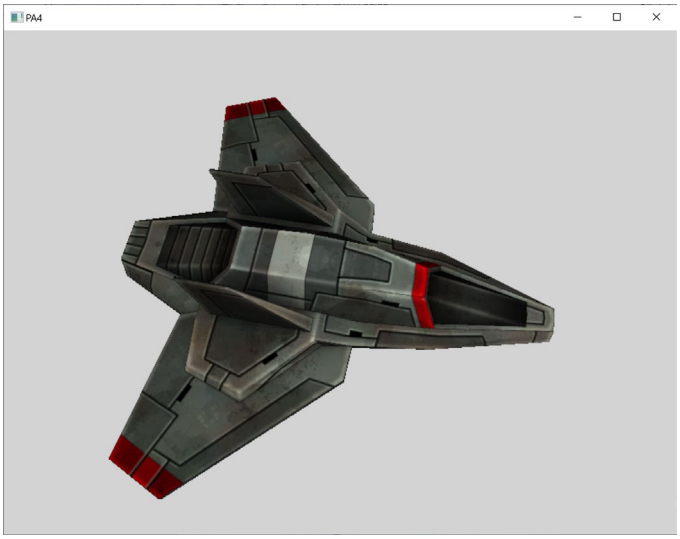
- _PA4_Engine
  - /Data/ ← input
    - Protobuf lives here: space_frigate.proto.azul
- All libraries are linked
  - Modify the code to load your space_frigate.proto.azul
    - Add/modify proto, src, include
  - Look at the lecture sample code
- You will need to modify Converter/Engine combo
  - Several times to get them working

**STEP 3: -- Unit Test ---**

- Quick validation – unit tests
  - Take your protobuf… extract the data into a buffer
- You complete the Verify class for the unit tests
  - Copy the VBO buffers / Texture into a buffer
  - Unit tests verifies the buffer
- Easy

Deliverables

- Converter working generating a single space_frigate.proto.azul
  - All data included inside that one protobuf – vbos, textures, info
- Viewer that displays the space ship




- Unit Tests verifying the data
  - 6 Tests pass
  - No memory leaks
  - Sample output:

```
****************************************
**      Framework: 3.84               **
**   C++ Compiler: 193732822          **
**  Tools Version: 14.37.32822        **
**    Windows SDK: 10.0.22621.0       **
**   Mem Tracking: enabled            **
**           Mode: x86 Debug          **
****************************************

--------------------------------
    Memory Tracking: start()
--------------------------------

------ Testing DEBUG ----------

 PASSED: VBO_INDEX_Test
 PASSED: MODEL_STATS_Test
 PASSED: VBO_NORM_Test
 PASSED: TEXTURE_BUFFER_Test
 PASSED: VBO_UV_Test
 PASSED: VBO_VERT_Test

  --- Tests Results ---

[x86 Debug] Ignored: 0
[x86 Debug]  Passed: 6
[x86 Debug]  Failed: 0

   Test Count: 6
 Indiv Checks: 38
         Mode: x86 Debug

-----------------

--------------------------------
    Memory Tracking: passed
--------------------------------
    Memory Tracking: end()
--------------------------------
```

## Validation

*Simple checklist to make sure that everything is submitted correctly*

- Make sure program build without errors or warnings
- Project should be able to run without crashing
- Set the _UnitTestConfiguration.h to the tests you want graded
    - DO NOT change the project setting to exclude any files
- Check
    - Does the model convert?
    - Does the engine run with protobuf model data?
    - Are textures included inside the protobuf?
    - Did you run the unit tests?