use Adobe Reader to complete

(Type in fields)

Sprint – Graphics Progress

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:	05/27/23

Submission Details

Final ***Changelist*** number:	188486
Verified build:	[✓ Yes] No
Number Tests Passed:	does not apply
Required Configurations:	2: { Debug/Release | x86 }

Discussion (What did you learn):

It was so cool to get to use our code from the previous PAs in the final assignment. I was able to use my Math, File, and PCStree for this sprint. I was able to utilize what I learned in GAM 370 to make my movement demo. At first, I had all of the objects moving together. So I had to update each object separately so that they could move individually.

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

Allowed to Add files to this project

- This project will work "as-is" do not add files…

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

- Add PCSTree code/library to the project
- Create a GameObjectManager
    - Use PCSTree
    - You can attached object in a hierarchy
- Create a Pyramid model
    - Look at Cube as an example
- Movement
    - Place several cubes and pyramids on the screen
    - Have the spin and move in interesting way
        - Do the work through transformation matrices
- Reach goal
    - Create a manager class
        - Create the library for this functionality
    - Manage Camera, Models, Shaders

Assignments

1. Add PCSTree code/library to the project
    a. Take your PA5 code and add it to the project
    b. Look at the File system/library for ideas
    c. Make sure your forward/reverser iterators work
    d. Do a simple smoke test in main() to test functionality

2. Create a GameObjectManager
    a. Use PCSTree
        - To create a “holder” for GameObjects
        - Place PCSNode as a base clas to GameObjects
        - Have the manager hold a pointer to the root tree
    b. GameObject Manager functionalit
        - Should be able to Update() all objects

- Should be able to Draw() all objects
- Use iterators to help (forward)

3. Create a Pyramid model
   a. Look at Cube as an example
   b. Create a Pyramid model… see notes on coordinates
      - Follow the pattern
      - Debug and verify in debug and release
      - Should be pretty easy

4. Movement
   a. Place several cubes and pyramids on the screen
   b. Have the spin and move in interesting way
      - Do the work through transformation matrices
      - You are going to need specialized GameObject data to create World matrix
      - Look at the demo on movement from week7

5. Reach goal
   a. Create a manager class
      - Create the library for this functionality
   b. Manage Camera, Models, Shaders

Validation

Simple checklist to make sure that everything is submitted correctly

- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
  - Is all the code there and compiles “as-is”?
  - No extra files
- Is the project leaking memory?

Hints

Most assignments will have hints in a section like this.

- This assignment is pretty easy.. if you don’t understand look at all the code drops and understand the different changes beween the directories in the lecture notes
  - Use a good Difference Tool like Araxis Merge