use Adobe Reader to complete

(Type in fields)

Milestone 2 – Graphics System

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:	06/11/23

Submission Details

Final ***Changelist*** number:	191429
Verified build:	[✓ Yes] No
Required Configurations:	2: { Debug/Release | x86 }
YouTube Link:	https://youtu.be/3o6tAZnavw8

Discussion (What did you learn):

I think I have everything working. I have game objects using graphics objects (4-5 shapes), I have 4 cameras, 4 textures, and lighting with different directions and colors. I also used my code for the math, file, and pcstree.

In my demo, I showed that I have no memory leaks. I have 5 meshes each using different shaders or textures. We can move the camera and swap between cameras. I'm using the PCS tree for keeping track of the scene and showed different rendering modes. I also have all five managers: camera, texture, mesh, shader, and game object.

I do think my texture manager could be fleshed out a little bit more.

YouTube Process

- Record the YouTube demo
  - You need to record with commentary
  - Suggestion: ***<u>OBS</u>*** screen capture
- Record the desktop (enough to show your directory and the visual studio and output)
  - Show your directory in recording
    - Launch the visual studio (double click solution)
  - Show off relevant parts of the code with commentary
  - Launch and run the demo
    - Play the demo and add your commentary in real-time
  - Watch your video
    - Verify that video clear and can you hear the commentary with audio.
- Note:
  - Expectation 5-10 min recording length
- Publish your YouTube recording
  - Make sure it is accessible without any login or permission to play
  - It can be private but not restrictive to play by anyone with the link
- Submit your code to perforce to the appropriate PA directory
  - Verify it

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
    - 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, *.suo,
  - *.vcxproj, *.vcxproj.filters, *.vcxproj.user
  - *.cpp, *.h
  - 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

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

- Create a standalone Graphics system

Assignments

1. **Basic features:**
   a. **Game Objects (with Graphics Object)**
      - Management System
        1. Create/Destroy game objects
      - Transformation
        1. Transform complex operations, into one resulting world matrix
      - Pipe several matrix transformation together
        1. Per instance
      - Change states
        1. Each object controls it's respective OpenGL states

   b. **Camera**
      - Camera controls
        1. Cleanly adjust/set attributes
        2. Move cameras
        3. Frustum(View) calculations
      - Management system - 4-5 different camerass
        1. Support multiple camera
        2. Switch between cameras

   c. **Texture**
      - Support texture on graphical objects
      - Swap texture on same object
      - Support and set all the controls for the texture in a texture object
        1. These are defaulted but should have an interface to change
           a. min/max filters
           b. Clamping/wrapping mode

   d. **Lighting**
      - Support different types of lighting
        1. Accomplished by supporting for different shaders
      - Allow each object to have different lighting parameters
        1. (color, direction)

   e. **Libraries**
      - Need to use YOUR custom libraries (6 libraries in total)
        1. Math, File, PCSTree <-- use your own code
        2. Supplied libraries Manager, Time, DxTex

2. Required demo features

a. Need to show ***NO memory leaks***
- Keep the original memory tracking system in place
- Make sure there is no leaking
    1. Show the start and ending Memory banners in demo (output window)

b. Draw at ***least 4*** or more different primitive objects
- Cube (box) counts as one of them
- You need to add at least 3 more
    1. Need to contain textures and drawn with lighting
- Look around for these... they are out there as simple data
    1. Torus, cylinder, sphere, cube
    2. Create your own simple model
        a. Simple shapes are allowed
            i. Cross
            ii. Diamond
            iii. Sphere
        b. You can share models
- Can be small or large in vertex count
- Should have texture, normals, verts, colors for each mode

c. ***Instancing*** capability
- Rendering multiple graphic objects at:
    1. Different locations
    2. Different transformations (complex transforms…)
    3. Different lighting attributes
- Render ***at least*** 4 instances for each of the 4 primitive objects
    1. (that's ***minimum*** of 16 objects 4 of each type).
    2. Typically, students have 30-50 objects

d. ***Moving the camera***
- Driving the camera through the scene, 4-5 different cameras
    1. By keyboard
    2. (optional) Splines or data driven pathway would be cool

e. ***Draw the objects with your Mesh***
- Vertex Buffer holds - verts(pos), norms, uv, colors
- Each object should be independent to the texture.
    1. This allows you to swap it in runtime

g. ***Show different rendering modes***
- Should have at LEAST 4-5 different shaders
    1. Different lighting modes
        a. Wireframe, FlatTexture, LightTexture, VertexColor
        b. Look at the others…

h. ***Scene Graph***
- Hierarchy Scene using the ***PCSTree*** to arrange and manage the scene
- Transformation,
    1. Display is all based off this scene graph (PCSTree is that role)
    2. Culling will be done next quarter

i. ***Complex attribute support***
- Camera Manager - make sure you do this
    1. Support multiple cameras
        a. Creating and destroying specific cameras
    2. Transitions
        a. Cut Scene or moving between cameras
- Texture manager - make sure you do this
    1. Register and manage multiple textures
    2. Create / destroy textures
        a. Reference counting system the number of objects using specific textures
        b. Free resources only if the texture reference count is zero
- Mesh Manager - make sure you do this
    1. Support multiple mesh
        a. Creating and destroying specific models
        b. Clean up model during shutdown
- Shader Manager – make sure you do this
    1. Support multiple Shaders
        a. Creating and destroying specific Shaders
        b. Clean up model during shutdown
- Game Object Manager - make sure you do this
    1. Support multiple game objects
        a. Creating and destroying specific Game Objects
        b. Clean up model during shutdown

3. ***Record the demo***
    a. Fill out the submission report
        - Listing all the features completed and working
        - Listing of all the features not completed
        - Link to YouTube movie
    b. Video
        - Need a 5-10 minute video demo of your project
            1. Show case the features you completed
            2. Demo and add commentary of your project
            3. This is to show case your work
                a. Be honest with what is working and not working
        - Post video to YouTube
            1. Use any video capture tool you
                a. Many free ones
                b. Start discussion thread on options
        - Do not record the whole desktop
            1. Restrict your recording to the area of interest
                a. Code editor to show code
                b. Window to show working demo
                c. Saves space on movie
        - Audio
            1. Test your audio
                a. Make sure it is loud enough and easy to understand
            2. Don't be nervous,
                a. Everyone is awkward and weird in their own unique way
                b. You listen to me, that's strange and goofy

Validation

- Submitted project to perforce correctly
    - Is the project compiling and running without any errors or warnings?
    - 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?
- Submitted the YouTube link to perforce?

Hints

Most assignments will have hints in a section like this.

- Focus on one feature at a time
  - Check- in to perforce
  - Work on next
- Time is your enemy, baby steps are key
  - Incremental development!
- Please
  - Draw diagrams to help you understand