CITY UNIVERSITY LONDON

INM376: Computer Graphics using OpenGL

Instructor: Dr. Eddie Edwards
Email: Philip.Edwards@city.ac.uk

Coursework: "Route"

Demo code submission 5 pm Sunday 28th March 2021

Demos in sessions Monday 29th or Wednesday 31st April 2021 (TBC)

Final code and report due 5 pm Sunday 26th April 2021

Submission Details:

You must upload the deliverables to Moodle by **Due 5 pm Sunday 26th April 2021**. Standard lateness penalties and extensions policy applies. This assignment constitutes **90% of the total marks for this module**.

There will be a **demo** of the projects in the last lecture/lab sessions on **Monday 29th or Wednesday 31st April 2021**. This will involve a 5 minute demonstration of your game/visualisation and accounts for **15% of the marks**. The code for the demo should be packaged as described below and submitted to Moodle by **5 pm Sunday 28th March 2021.**

The final report and software submission counts for the remaining **75% of the marks.**

There is a slightly different emphasis in the marking schemes for the demo and the final submission, with the demo having less emphasis on advanced techniqies. Marking in this document refers to the final submission.

Concept:

A number of games take place along a pre-programmed route. Examples include well-known driving simulators and racing games like Forza, Need for Speed, The Crew, Driver, as well as other games like AudioSurf, Thumper, Rez Infinite, etc. Taking inspiration from such games, you will implement a simple computer application (game / simulator) occurring on a route. The route must have a non-linear path that you will design programmatically using splines. Your game will provide first and third person views as the player moves along the route. The scene will be additionally be populated with interesting and colourful objects to make for a compelling experience.

CITY UNIVERSITY LONDON

Task:

Your task is to create a 3D graphics scene using OpenGL matching the theme described above. Your creation will include a route following a non-linear path on which the game or simulation will occur. There will be a series of camera views. You will create the scene using primitive shapes, as well as freely available downloaded mesh models. The scene will include computer-generated special effects and lights that illuminate the scene dynamically. The project can be implemented in a variety of ways. You are welcome to take some liberty with the theme as long as your project delivers against the requirements described below.

Technology:

Use the "OpenGL Template" provided on Moodle as the basis of your work. This should help provide the base functionality for the project.

You are to use OpenGL, C++, and GLSL for this coursework, using the Windows API. It is recommended that you use OpenGL 4.0 or higher, although you are permitted to use OpenGL 3.3 (if your development hardware does not support OpenGL 4.0). The rendering context must be configured as a **core context**. You are to program the game using Visual Studio 2017 or 2019. You are welcome to include other libraries as long as they are sufficiently packaged with both the source code and the final deployed executables. Deprecated fixed function programming through use of a compatibility context is prohibited and will receive **zero marks** – this is a module on modern OpenGL programming.

Spend some time familiarising yourself with the OpenGL template, as well as labs and code samples available via Moodle. Look at online tutorials on modern OpenGL, and refer to the books recommended for the module. At times you may wish to discuss the methods used with your colleagues. This is encouraged. The coursework is, however, **individual**. Plagiarism will be dealt with directly by the department's senior staff and may lead to course disqualification. Your work must be your own and you must cite any and all resources used.

Marking:

The coursework is marked out of 100%, with the final mark scaled to 75% for the module.

CITY UNIVERSITY LONDON

1. Deliverables -- Project report and source code (15%):

- Prepare a project report (maximum 15 pages) in Word or PDF format. (10%)
    - Overview of your project; a description of your concept and game or application.
    - Include a prototype sketch or annotated screengrab showing a top-down view of your scene.
    - Provide a table to identify all user interface (keyboard / mouse) controls.
    - For each section (Parts 2 - 5 below), list briefly the requirements that were implemented and relevant details on the implementation. If a requirement was skipped, say so.
    - List your scene's assets, including any websites where objects were downloaded – **URL**, **date** of download, and **license** for use. Reference the source of all meshes used in your coursework, including original files you designed (if relevant). Document any process used to convert to the meshes in the final deliverable. Reference any external source code used.
    - Include a discussion section reflecting on the project. Consider the strengths and weaknesses of the game implemented and what you have accomplished in the time provided. Also discuss what would be required to expand the project into a more complete game or simulation.
- Source code to be commented and follow a logical design, organisation, and coding style (i.e., use of classes). (5%)

2. Route and camera (25%):

- Route (15%)
    - Create a 3D non-linear centreline for your path, based on splines with C^1 or higher continuity.
    - Create primitives based on the centreline to generate a visualisation of the path/track on which the gameplay will occur.
    - Ensure your primitives have correctly oriented normals and texture coordinates. Render the route with appropriate texturing and lighting.
    - Demarcate the edge of the route. Some options include (1) use texture mapping (2) generating primitives (such as short walls or rings) or (3) placing objects like meshes at regular intervals along the path.

  - Note: the route must be generated programmatically, using OpenGL primitives. Building the path in 3D modelling software and loading it as a mesh is not permitted.

- Camera / viewing (10%)
  - The graphics template code allows the user to control camera movement in a "free view" mode. Retain this type of camera movement as an option (useful for debugging and visualisation).
  - Implement at least three of the following camera modes:
    1. First person: Add a keyboard control to place the camera so that it provides a first person view allowing the player to move along the route.
    2. Third person: Add a keyboard control to place the camera either in or behind the player (rendered as a mesh).
    3. Side view: Add a keyboard control to place the camera to the side of the player character (mesh) moving along the route.
    4. Top view: Add a keyboard control to place the camera so that it provides a top view of your scene, but follows the player moving along the route.
  - You will likely find it helpful to use a TNB frame for specifying the camera viewing geometry.

3. Basic objects, meshes, and lighting (30%)

- Basic objects (10%)
  - Create at least two different basic objects from primitives. Examples could include a cube, tetrahedron, torus, cylinder, cone, disk (other surfaces are possible). Render these objects using triangle primitive types (GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN). Note that the basic objects (plane, sphere) given in the template code do not count towards this requirement; nor do meshes loaded into the scene. However, shapes created in labs can count towards this section.
  - Apply appropriate texture coordinates and normals for your objects. Render these objects using texture mapping and lighting.
  - Transform the models to have an appropriate location, orientation and scale within the scene.

- Meshes (10%)

  - Load at least four different texture-mapped mesh models (with appropriate normals) to the scene using the model loader code provided in template code. Populate the scene with objects that match your theme. Note the mesh models (horse, barrel) already provided in the template do not count towards these four models.
  - Transform the models to have an appropriate location, orientation and scale within the scene.
  - Create parts of your scene using repeated mesh objects (e.g. rows of pickups, obstacles, etc.).

- Lighting (10%)
  - The OpenGL template has a general lighting for all objects. Implement an option to switch between bright and dark modes (e.g., turn the "sunlight" off and the spotlights on).
  - Create at least two spotlights that illuminate the scene in dark mode.
  - Give at least one of these spotlights a non-white colour.

4. Head's up display (HUD), gameplay, and advanced rendering (30%)

- HUD (5%)
  - Include a head's up display to provide information to user (time, score, speed, fuel, place, etc.).
  - Implement some interesting text rendering. Some ideas including changing the font, rendering font shadows, rendering 2.5D fonts.

- Gameplay (5%)
  - Implement control of the player position on the route using the mouse or keyboard. Animate the player movement.
  - Implement gameplay to make the game interesting to play. You're free to design the gameplay as you wish, but it should not be trivial to play. Some suggestions include:
    - Time-trial racing: The player seeks to get from the start to finish as fast as possible. Obstacles and/or speed-ups enhance gameplay.
    - Points: The player seeks to achieve as high a score as possible, by collecting pick-ups on the track. Obstacles may cause damage or lower score.

        - Shooter: The player defeats enemies by shooting at them.
    - Utilise basic collision detection using standard techniques (e.g., distance between player and object).

- Advanced rendering (20%)
    - Implement at least two advanced rendering techniques. At least one should involve shader coding. Some ideas for this include the following:
        - *Super easy*
            - Blinking (e.g. Lab 1; or rendering/not rendering using a timer)
            - Wobble
            - Fog
            - Camera shake
        - *Easier:*
            - Animation using `discard` in the fragment shader
            - Toon shader
            - Instanced rendering
            - Multi-texturing
        - *Harder:*
            - Blur (radial, regular, motion)
            - Particle animation
            - Environment mapping
            - Bloom
            - Shadows
            - Perlin (or related) noise
            - Mirror
            - Use of a geometry, tessellation, or compute shader

    We will cover most of these in lecture / labs. Pick techniques that are interesting and feasible for your work. Nearly all of these methods are described in the reference textbooks for the module. If you have a different technique in mind, please speak to the module leader (Eddie).

- In your report, be sure to describe the techniques implemented, and how they work.

5. Optional (unmarked)

- Include audio in your application to enhance the experience.

CITY UNIVERSITY LONDON

Presentation in Class (5 minutes)

- You are to present your work during the lab session on either **Monday 29th or Wednesday 31st March**. Clear explanations of your implementation will be expected.
- The presentation is to be done **in a Zoom class.** This is your opportunity to show everyone what you have accomplished, and should be a fun experience for all!
- The demo session will be marked on the same marking schedule as the report except that section 1 will cover presentation (30%) instead of the report/code as there is no submission
- This will be marked out of 100% and scaled to 15% for the module.
- **You should aim to have all the deliverables for the coursework completed by the time of the demo**.
- Feedback will be provided rapidly and give an indication of progress at that point.

Notes:

All source code should be commented. Clearly mark any code that you have written and any code that is externally produced. Use appropriate naming conventions and display an organised code strategy (header files etc).

If additional libraries are used, make sure that these are sufficiently packaged with both the source code and the final publicly deployed executables to allow the code to be **run on any lab machine**.

Deliverables:

ZIP file (note this must be less than 200MB!) uploaded to Moodle containing:

1. A folder of documentation in PDF/Word format.
2. A folder containing the source code.
    - The code must be at a stage where a simple 'clean, build & execute' will start your application. Code that does not compile and execute will receive a **FAIL**.
    - Do not include the .db file – this can be regenerated automatically by the compiler and it is a very large file
3. Folder for binary executables (release mode build sufficient).

- Executables must be in a runable state (**not requiring changes to the directory structure to point to resources/assets**). Test by double-clicking on your .exe from Windows Explorer.

Tips:

- Maximise your marks by making some attempt at every section.
- Install Visual Studio as soon as you can. Explore the template code provided and any demos given in class.
- Google any of the terms you are unfamiliar with, and include the keywords “OpenGL 3.3 tutorial” or “OpenGL 4.0 tutorial” after it. Put aside time each week to cover material and practice coding frequently. Build up a good portfolio of example code for you to keep and explore best practices.
- Start early! It is recommended that you begin the coursework following Lab 3 and make consistent progress each week. Take advantage of Reading Week to make great progress on the project.