CS324 Coursework Assignment WebGL/GLSL 2023

Shan E Ahmed Raza, Rob Jewsbury

November 2023

1 Introduction

Your coursework assignment task is to write a graphics program using WebGL and GLSL. You should use what you have learnt in the practical laboratories and lectures. Note that the program **must** run on the department's Linux machines on Google Chrome browser and you should test this out before you submit your code for assessment.

Your task is to create a video game. Here is an example [1] as inspiration. The demo code can be downloaded here [2].

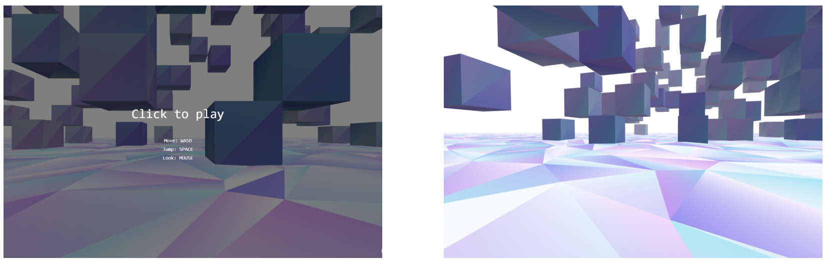

Figure 1: Screenshots of the inspiration demo.

You can modify the gameplay as you see fit, so feel free to use your creativity! Please refer to Section 2 for detailed requirements.

You can use any of the materials from the labs or adopt external libraries. Please note that any code that is borrowed from other sources ***has to be acknowledged*** in the list of references and in the code itself. Your game ***should not require*** the assessor to download any extra software or library. However, you are allowed to use any software available on the department machines. We expect that you will need to spend about a week (20-30 hours) to complete the task(s).

Please try to complete the essential functionalities (section 2.1 to 2.4) in the first instance and work on the cosmetics and desirable features (section 2.5) later on.

If you have any questions, please get in touch with the module faculty. Please check the FAQ page (https://moodle.warwick.ac.uk/mod/forum/view.php?id=2153014) before submitting any new questions. Good luck and have fun!

2 Requirements

2.1 Code Implementation (40 points) - Essential

Game Implementation

All of the following functionalities must be implemented:

1. **Levels:** You must implement two uniquely designed environments. At the very least, environments are considered unique if they look distinctively different. Each level must have a distinct texture.
   (a) Environments generated using procedural algorithms will not be considered to be uniquely designed unless the environments are built using different units.
   (b) Altering parameters within the same environment will not be considered as a new level (such as lightning conditions, movement speed, etc.)

2. **Light Sources:** The game must have the following light sources implemented:
   (a) Ambient light must be implemented.
   (b) Must have at least two distinct light sources other than ambient light in each level.

3. **Menu:** The game must have a main menu that can be operated using a mouse. The main menu must contain an instruction section to explain clearly how a player can control the game.

4. **Camera:** The game must contain at least 2 camera views. This can be achieved either by using 2 different cameras or a single camera with multiple views. Action games with a mouse-controllable viewpoint automatically pass this requirement.

Code Linting

Your code and workspace must be properly structured, formatted, documented and checked for errors. Your code should be properly refactored and structured into different files where necessary. Functions and methods must be properly commented and have their intended purpose explained appropriately. You can consider using linters and code formatting tools to help you with structuring of the code.

2.2 Blender Model(s) Design (20pts) - Essential

You must create at least one 3D model using Blender and include it in your game. Please provide the original .blend file and also export your model in a glTF format in the game (i.e., .glTF or .glb). In your report (see 2.4) you should add at least 4 screenshots of the intermediate stages of the creation of your 3D model.

We require a minimum level of complexity for the Blender model(s), comparable to what we have seen during the lectures and tutorials. If you make use of blueprints, add these in the references. Elements that contribute to the assessed quality of the Blender model are: the model opens and renders (with Eeve or Cycles) without errors; the model is made of Quads and triangles only; there are no multiple stacked polygons; there are no open edges; unique and meaningful names are given to the objects, materials, and textures; textures -if used - are properly applied with appropriate seams and with a consistent scale to avoid discrepancies in the rendered texture resolution; the minimum texture resolution is 512 $\times$ 512 and the model is centred at the origin.

Note: Animations (e.g., rigging a character and making use of inverse kinematics) are not a requirement for this coursework. However, if these are present they will be considered when awarding bonus points.

2.3 Game Design (20pts) - Essential

The general quality of the graphics produced, and the ease of use of the solution will be evaluated. It is desirable for all of your Blender models and the game functionality to work cohesively. We encourage you to make a game that also has a creative design or gameplay.

2.4 Report (20 points) - Essential

You should submit a report that clearly explains your implementation and the work conducted for code implementation and design of the blender model. Summarize all the **main points within 1500 words**.

The report must satisfy the following criteria:

1. The first section must contain clear and precise instructions so that a 3rd party can run the code. If we can not run the code based on your instructions, we will apply a 50% penalty on the points scored for the report.

2. The main portion of the report (not including the instruction section and references) must be strictly within 1500 words.

3. Please note that the report should be a complete and cohesive document, properly structured with clear section headings.

2.5 Bonus Points (max 20 pts)

To improve the game quality and user experience, it might be desirable for your game to include extra features. Please see below a non-exhaustive list:

- Sound, e.g., play a sound every time the player scores a point etc.;
- Heads-Up Display (HUD), visualise on screen useful info for the player using transparency e.g., health bar, points, remaining lives etc.;
- Animations;
- Collision Detection, to detect collisions between different objects e.g., using Raycaster;
- Physics e.g., gravity;
- Environment Maps can be used to add realism to the scene.

Please note that these points cannot be scored unless the game meets all the essential requirements (section 2.1 to 2.4). Therefore, we suggest that you focus on implementation of essential requirements first.

3 Collaboration and Plagiarism

As for all pieces of assessment you undertake, the submission must be your **own** work. You may have been working in pairs in the practical labs, but this piece of work must be done individually. Discussion of ideas with your colleagues is OK, but the final piece of work must be your own.

Also, you must not copy without attribution from work you may have found online. Using any code that is given in the laboratory tutorial examples is permitted.

We will use anti-plagiarism software to look for copying between submissions, with external sources and past submissions. Any breaches of the rules will incur a severe penalty.

4 Deadlines and Submission Formats

The deadline for this coursework is **12pm Monday, 8th January 2024**.

Please submit a **single** ZIP file with the following structure:

- Solution: Folder containing your original source files (`.html`, `.js`, `.css`)
- Extra Libraries: Anything useful to your project that you have not developed yourself.
- Assets: Anything used in your game that is not HTML, CSS, or JavaScript goes in here: textures, 3D models, fonts, sounds, and so on.
- PDF of a supporting document that describes your code.

5 Some useful resources

- Games Database, it might give you some ideas on what game you may want to implement.
- Fullerton Tracy, *Game Design Workshop*, 2014, (available through Warwick Library): Book on how to create a game.
- Three.js homepage.
- G. Moioli, "Introduction to Blender 3.0: Learn Organic and Architectural Modeling, Lighting, Materials, Painting, Rendering, and Compositing with Blender", Mar 2022.

References

[1] THREEjs. Controls pointer lock example. [Online]. Available: https://threejs.org/examples/?q=contr#misc_controls_pointerlock

[2] THREEjs. Example codes on GitHub. [Online]. Available: https://github.com/mrdoob/three.js/tree/master/examples

Shan E Ahmed Raza, Rob Jewsbury
November 2023