Interactive Multimedia

Unity Project Brief

Final Project Delivery: week 13

Deliverables Submission Schedule:

Semester week	Date	Time	Component	Weighting
Week 6	25/10/2024	5pm	Team formation	N/A
Week 9	15/11/2024	10pm	Alpha version	20%
Week 13	09/12/2024	9am	Final submission Video ***Demo***	40%

1 Project Objective

In **groups of 2-3**, and using the Create-With-Code tutorials as a foundation, you are to design and develop an ORIGINAL interactive, 3D multimedia system using Unity.

This project is an opportunity for students to showcase their understanding of the topics covered in this module. This may be achieved by informing the practical skills learned with the theoretical knowledge gained in the lectures. This document provides a general overview of the project.

In undertaking this project, students are reminded of the software development lifecycle. Every software project has several stages before the implementation stage and begins at a research and planning stage.

2 Acceptable project ideas

All ideas for interactive multimedia systems will be considered. It is best to run your idea past your lab tutor to be sure.

All projects will use the game mechanics from one or more of the Create-With-Code tutorials. In this way, students can use the code from one of these Units and build upon that through the addition of features to make a complete game. Most marks will be awarded for unique implementation of existing game mechanics and the addition of unique features.

3 Project Requirements

The IMM system should follow good game-design and ludological principles, in that:

- The player has one or more clear goal to achieve
- The player is appropriately incentivised to progress within the game.
- The game becomes progressively more difficult so that the beginning should be straightforward for most people to complete, but it should require some practice and skill to complete the final parts of the game (so the player has a reason to want to come back to play it several times …)

NB: Groups of 3 are expected to undertake a larger workload than groups of 2. See details later in this document.

The IMM system must demonstrate the following:

- Multiple scenes with UI elements to navigate between scenes (main menu/instructions/level 1/game lost etc.). Using appropriate HCI design.
    - Menu should also include a link to code Repo on GitHub.com.
- Detecting 'collisions' between the player & objects, or objects & objects
    - Additional marks are awarded for additional complexity, such as different actions depending on what is being 'carried' at time of collision (e.g., if carrying a key and hit door, then door opens, otherwise if not carrying a key then the player is notified that they need to pick up a key before opening the door).
- Unity animations
- Unity particle system
- Appropriate use of prefabs
- Appropriate use of the Instantiate(), Destroy() and SetActive() methods
- Variety of different data types, including Arrays and Lists.
- Audio, including all of the following:
    - Background sound (e.g. music)
    - Sound effects (e.g., crash, shoot, jump
    - Event based sounds (e.g., button click, game over)
- Collecting points/ score / picking up objects
    - Visually display the current game state with a Heads-Up Display (HUD)
    - Additional marks will be awarded for graphical display (depending on the approach used)
- The game should be of a HIGH PRODUCTION QUALITY and have a consistent audio-visual 'theme'.
- USP - "Unique Selling Point", although your game can be inspired by other games it should be unique.

Some general rules of thumb:

- Follow the *'Labs'* in Units 1 – 5 in the Create-With-Code tutorials. These will guide you through the project process. E.g., it is best to begin programming functionality before adding fancy graphics.
- **Under no circumstance** should students develop a game based on a tutorial or tutorial series! Following a tutorial, step-by-step, showcases zero understanding of the task-at-hand. In addition, if this is not referenced correctly then it is plagiarism.
    - Students are encouraged to learn from tutorials but are not permitted to submit the product of a tutorial or tutorial series. Students are, however, permitted to use any materials from the Create-with-Code series.
- You may use media assets from other sources if these are clearly declared in your 'sources' document.
- All scripting assets must be your own or from Create-with-code. Any code from a third-party source will attract little or no marks and will be assumed plagiarised if cited incorrectly. Therefore, be sure to reference everything correctly.
- Sources file: You need to declare **ALL** 3rd party assets as follows (inc. 3D assets, audio etc…):
    - The name of the asset as it appears in your source files
    - The original name of the asset
    - The location the asset came from (e.g. URL / magazine disk etc.) and/or a copy of the original file. Note that 'Google images' is not a specific enough reference. I need exact links.
    - A statement of whether the asset is used unchanged from the source, or if you have adapted it, what adaptations you have made.

4 Deliverable 1: Alpha Version

The Alpha version of your game will be a rough and unfinished version of your game. However, all the main game mechanics will be present.

What should the alpha look like?

Games will look different at Alpha stage depending on the type of game under development. The alpha is an early version of the system that may not contain all the features that are planned for the final version. Although, it is expected that most of the game mechanics will be functional in the alpha version. Students should submit something like what might be outputted from 'Lab 4' in the Create-with-Code series (but, of course, you should not be copying Lab 4, rather you should be creating the basic functionality for your own game).

Don't worry if your Alpha version does not look too complete or well-polished, it is not supposed to at this stage. What is important at alpha stage however, is that the functionality is there, e.g., character movement, collision events, scoring and lives (though perhaps not the HUD).

Alpha versions will be submitted via GitHub and a playable WebGL build MUST be created and published on Unity Play.

NOTE: be sure not to overwrite your Alpha code when developing the final game version. Create a new GitHub repository or branch.

DELIVERABLE 1 SUBMISSION: SUBMIT A LINK TO YOUR PUBLIC GITHUB REPO, AND A LINK TO THE WEBGL VERSION ON UNITY PLAY. BE SURE TO TEST THAT THESE LINKS WORK AND THAT REPOS ARE SET TO PUBLIC. IF THE FILES CANNOT BE ACCESSED, THEN THEY CANNOT BE GRADED.

IMPORTANT: DO NOT EDIT THE GITHUB REPO FOR THE ALPHA VERSION AFTER THE SUBMISSION DEADLINE.

5 Deliverable 2: Final Submission / completed game

The final submission will be a fully playable game with all the components listed on pages 3 & 4 of this document. Note that marks for the final game version are granted for:

1) Work completed **since** the Alpha submission (if you submit the same as your Alpha, you will gain very little marks for this deliverable)
2) Completeness of game development
3) Application of theory (from the lectures)
4) Game features

6 Project Submission

Note: project components that are to be submitted during the semester must also be submitted at the end of the semester (week 13). Only one submission per group is required.

Semester week	Date	Time	Component	Weighting
Week 6	25/10/2024	5pm	Team formation	N/A
Week 9	15/11/2024	10pm	Alpha version	20%
Week 13	09/12/2024	9am	Final submission Video ***Demo***	40%

Submit the Alpha version as described above.

Final Delivery

THE FOLLOWING FILES MUST BE SUBMITTED WITH THE FINAL PROJECT SUBMISSION (WEEK 13):

- ☐ A LINK TO THE ALPHA PROJECT ON GITHUB
- ☐ A LINK TO THE ALPHA PROJECT GAME BUILD ON UNITY PLAY (WEBGL BUILD)
- ☐ A LINK TO THE FINAL PROJECT ON GITHUB
- ☐ A LINK TO THE FINAL PROJECT GAME BUILD ON UNITY PLAY (WEBGL BUILD)
- ☐ SOURCES.TXT
- ☐ POST ALL FILES AND LINKS ON BRIGHTSPACE
- ☐ DEMO VIDEO (VIA A SEPARATE LINK ON BRIGHTSPACE)

METHOD OF SUBMISSION:

MARKS FOR THE GAME DEVELOPMENT WILL BE AWARDED BASED ON HOW YOUR PROJECT IS SUBMITTED. IF THE GAME IS SUBMITTED INCORRECTLY, THEN THE PROJECT CANNOT BE CORRECTED. GIVEN THIS, STUDENTS SHOULD CONFIRM THAT THEIR FILES HAVE BEEN SUBMITTED CORRECTLY. NOTE: CORRECT SUBMISSION OF WORK IS THE RESPONSIBILITY OF ALL STUDENTS IN THE GROUP.

GIVE YOURSELF THE BEST CHANCE OF GAINING A STRONG GRADE BY INCLUDING EVERYTHING IMPORTANT IN THIS SUBMISSION.

DO NOT EDIT YOUR GITHUB REPO AFTER THE SUBMISSION DEADLINE.

6.1 Demo Video

Instead of a live demonstration, which would require an explanation and defence of your work in-person, you are required to submit a recorded video 'screencast' with voice-over recording. In the recording, you will demonstrate your game play and source code, while arguing how well it has met all the project criteria.

*IMPORTANT NOTE: the video demo is **compulsory** and will be the primary asset used to grade your project work. Students who do not submit a demo, will not receive a grade.*

- *The video demo **max** duration is 15 mins. Please do not go beyond this.*
- *All group members must contribute to the demo, i.e., I should hear all voices.*
- *I have received videos in the past with no voice-over. To avoid this, I recommend that you playback and watch your video before submitting to avoid any technical (or other) errors.*

The video must include a demonstration of the gameplay:

- By you playing the game (not necessarily the entire game, just important parts)
- By your discussion of the source code. It might be an idea to flip back and forward between various aspects of the code and discuss it in relation to aspects of the game play.
- In both cases refer to your code via the project rubric provided on Brightspace, so that your demo/defence video forms a strong argument for how your project meets the various requirements of the brief. *Indeed, if I was demonstrating this project, I would use the rubric as a checklist for my demo.*

Be certain to include the following in your video Demo:

1. Identify CLEARLY what code is yours and what is not yours.
2. Do not simply provide a video of game play. You must provide explanations of how you achieved that gameplay through code.
3. Remember, the goal of this demonstration is to show that you know and understand the code implemented in your game… so explain what your code is doing. If there's a complex piece of code, and you don't explain it, I'll assume that's because you don't understand it.
4. Use the grading rubric to help guide your demo. For example, there is one item under 'Scripting' that awards marks for use of arrays, lists, iteration, conditionals and so on… so show where you do that in the code.

6.2 Video Capture and Format

I recommend using the screen share and record facilities in MS Teams to capture your demo.

You can use any screen recording software but be sure that your video is coded in a commonly used codec such .avi, .mpeg, see https://www.videolan.org/vlc/ for further formats.

Brightspace has an upload limit, so be sure your video is less than that. You can reduce your video size by reducing the quality, aspect ratio and/or the FPS. Do this in the settings of your screen capture software (but be sure that your code is still readable in the video). You may also choose to use a video converter to convert your video file to .mp4 or some other format that does a good job at compressing.

If you are having a real problem with the size of your video, then just pop it onto OneDrive, or a video service such as YouTube, and upload the link to Brightspace.

7 Penalties

1. Not submitting a project component will result in a grade of 0% for that component.
2. Non-submission of the final (video) project demo (week 13), by a group member, will result in a grade of 0% for the final submission, for that group member.
3. Late submission will result in a grade of 0% for that component.
    a. Project files edited after the submission deadline will be marked as late submissions.
    b. I recommend that you submit a day (or more) early to avoid technical difficulties.
4. Unreadable files will achieve a grade of 0%
5. EMAIL submissions will NOT be accepted
6. A project that is simply the result of following a tutorial will achieve a grade of 0%
7. Plagiarised work will be awarded 0%

8 Marking criteria

The general criteria upon which the project will be assessed are:

- Originality
    - different from, and lots of 'value added' to, the tutorial work
- Technical challenge
- Completeness
- Correctness
- Code Quality
- Application of theoretical learning to the project

A full grading rubric is available on Moodle

9 Helpful advice

Note: The following is NOT a comprehensive checklist of items to consider for your project. Please refer to the notes for more detailed information.

- Follow an iterative approach (see Software Development LifeCycles)
  - Begin with the most fundamental game interactions and then add layers of functionality and visuals to that.
  - Make a record of your iterations as proof that you followed an iterative approach.
- Evaluate the various aspects of your game at each iteration.
- After deciding on a concept, consider all aspects of the design
  - Game play design
    - Is it appropriate for your target age group?
    - What are the incentives
    - Goals
    - Is it interesting/fun
    - Is it too hard, too easy?
    - Evaluate this
    - Etc...
  - Level design
    - Can the user navigate appropriately?
    - Are the chosen assets suitable for the game?
    - Evaluate this
  - HUD design
    - How will the HUD look?
    - Begin this at the beginning of the project. Don't leave until the end.
    - Understand how you will implement this in code
    - Linking to gameplay to a visual display could be worth a lot of marks
    - Evaluate this
  - Interactions
    - All interactions should be well considered
    - Be sure to offer examples of all event based programming that we covered in the labs.
    - Events can turn on a light or sound, change a value in the HUD, play an animation, trigger another script etc…
    - Be sure to include as many examples as possible.
    - Most interactions will trigger multiple events.
    - Evaluate these
  - Audio design
    - Audio is extremely important to the ambience of a game and is straightforward to implement.
    - Consider what sounds you would like to play for each interaction.
    - Audio can be sourced online for free. Simple google 'sound effects'.
    - Evaluate this
- Proof
  - Proof may be offered in the form of a document, design diary, screen grabs etc…