UFCF9M-30-2 Game Engine Programming

Gold Feedback

TEAM NAME: Red

VIDEO: <https://youtu.be/O4eUBSAjY5I>

FEEDBACK:

**15% Implementation of & Documentation for Pipeline & Game Data Files**

A level editor has been produced, with a novel interface but I still hold that it might have a steep learning curve. You level data file gives pretty much a full description of a basic level, but you are let down by not being able to string together a complete playthrough and by the odd hard-coded element: thumb nails for levels (ought to have been a naming convention for an image based on the level name) as well as hard-coded names for the textures for entities in the game.

Mark: 10/15

**40% Implementation of Game Engine**

Whilst not completely feature complete from a gameplay standpoint (no power-ups, multiple levels but not playable through, not clear the tank interacts differently with the environment) and the menu system could perhaps have been made more data driven. A range of quite sophisticated systems have been implemented here, and then well used to create a reasonably good gameplay experience. A rather nice touch is making the Scarle engine into a library (some years I do this myself) to allow its systems to be used in the editor.

Mark: 28/40

**10% TDD & GDD**

Whilst covering the essential elements for each document they could have gone into more detail, and discussed more your development process. It does read a bit like preliminary thoughts that weren’t much expanded upon. Although, suggestions for how work would continue make sense and show clear understanding of where things would go (much better to say that than what you hadn’t implemented).

Mark: 6/10

**10% Discussion for Arcade + Vanilla Machine Builds**

Comes more from gameplay perspective but does highlight a range of things that needed to be taken into account. Really also needed a bit more on the technical aspects: runtime libraires for C++ and DirectX?

Mark: 5/10

**5% Use of repository and other collaboration tools**

Majority of commits appear to be suitably atomic, but you really need to have made use of the commentary box as well as the names for each commit to give just a bit more detail as to what it carries out.

Mark: 3/5

**Total MARK: 52/80**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Student Name | Student ID | W. Alpha /10 | W. Beta/10 | Final Weight /20 | W. Gold / 80 | Total Mark |
| Stephen Rayment | 18034264 | 8.1 | 9.6 | 20  ~~25~~ | 52 | 69.725 |
| Zack Collins | 19027000 | 6.5 | 9.2 | 20  ~~24~~ | 52 | 67.7 |
| Csongor-Zsolt Horosnyi | 18029633 | 4.9 | 2 | 2 | 5.2 | 12.075 |
| Jamie Winfield | 19014899 | 6.5 | 8.8 | 19  ~~23~~ | 49.4 | 64.7 |
| Samuel Badman | 17025835 | 6.5 | 10 | 21  ~~26~~ | 54.6 | 71.1 |

Given the low weighting for Horosnyi, I have rescaled the weightings for the rest of the members of the team as though only 80 units were being distributed.

**Group mark distribution**

Each group will have a number of points to distribute amongst team members, according to their perceived overall contribution to the project. The overall mark for the project will be scaled according to this distribution of points, to make up each student’s individual mark for the module. The number of points allocated for a group will be 20 \* number of students in the group.

Individual student marks are determined based on the formula:

Ms = Ps / 20 \* Mg

Where Ms is the student’s mark, Ps is the points given to the student by the team, and Mg is the overall mark given to the group.

**For example:**Group A consists of 5 students, who will have 100 points to distribute amongst the team members.

Students 1, 2 and 3 are perceived to have contributed equally to the project, while student 4 has put in much more work, and student 5 much less. The team distribute their marks as follows:

1. 20 points

2. 20 points

3. 20 points

4. 30 points

5. 10 points

When marked, the project receives an overall mark of 65%. This mark is scaled as follows, for each student:

1. 20 / 20 \* 65% = 65%

2. 20 / 20 \* 65% = 65%

3. 20 / 20 \* 65% = 65%

4. 30 / 20 \* 65% = 97%

5. 10 / 20 \* 65% = 32%

**Please note:** Group weightings are intended to allow teams to reflect the reality of their development practice throughout the project. However, the module leader reserves the right to adjust or otherwise moderate the metric and/or weightings submitted in the event of exceptional group circumstances occurring.