Introduction

Get excited, it’s time to start planning your very own 3D game! The project will occur in two phases: planning, and implementation. The planning phase will help us ensure that each student is responsible for a significant piece of functionality in the final, as well as to ensure that each group is prepared to code their project. The implementation phase is designed to keep you working at a steady pace so your end result is a game you’re proud of.

For a more detailed description of what we are expecting out of your final project, see the final1 through final5 sections at the end of this handout.

Planning Phase

Final Idea - Due Mar. 21

Every student must complete an idea proposal. This is a short (roughly 1-page) document split into two sections: engine feature and game idea.

For the engine feature section, you will need to take a look at the section at the bottom of this handout. You should pick out the ones you think are essential for your game and tally up the number of points that each features is worth. Each group member will be responsible for at least 5 points worth of engine features, so this should give you an idea of how many group members are required to make your project come to fruition.

For the game idea section, briefly describe a game concept and explain how your engine features fit into this game. We are requiring that every student comes up with his / her own game idea. Even if you already have a group, you must write your own idea proposal. There are a number of reasons this is required. One is that explaining a game idea is a good exercise in game design. Another is that another group may like the idea and decide to implement it, even if you don’t actually want to.

In class the day after turning in your idea, each student will be given 90 seconds to pitch their game idea to the class and respond to questions. Use this as an opportunity to convince people that your project is worth working on.

Final Group - Apr. 4

After presenting your ideas, you will have two weeks (over spring break) to form a group and decide on the engine features and gameplay for your final project.

For this week, you will meet with the Game Engines Board Of Elders (GEBOE) to discuss your project and ensure that it is both feasible and a reasonable amount of work. This includes a more detailed explanation of the game you plan on implementing, as well as a specific breakdown of how each group member will satisfy their 5 points worth of engine features.
Final Design - Due Apr. 4

Every final project group must complete a design proposal. This document must be split into three sections: engine features, engine integration, and game. Make sure to list all members of your group at the top.

The engine features section describes each major engine feature you plan on implementing, with the name of the group member implementing that feature. We're expecting about a paragraph describing each feature, including dependencies it has on other systems and what, if any, libraries it requires.

The engine integration section describes how your engine features fit together into a cohesive game engine. Feel free to use diagrams here - the more thought you put into this, the less work you will have next week when you complete your initial implementation. While it is not required, we recommend at least writing interfaces for this section so you know what contracts you will be working with.

The game section describes the game that is built upon your engine. Make sure to give an example of how a level / short session of your game might play. Also make a note of any external resources (3D models, textures, etc.) you are planning on using here.

After handing in your design, you will be contacted by one of the TA staff who will server as your mentor for the project. This TA will (hopefully) have experience with the engine features you want to implement, and will be responsible for making rubrics for each week of the project.

Implementation Phase

Introduction

The final project is unlike previous assignments in that you are expected to create a polished final result, as opposed to a simple demo of an engine. Your engine should be built for the technical needs of your game. This is very important; if you spend 2 of your 5 weeks implementing an engine feature that doesn’t fit into your game, it will be very difficult to complete a satisfactory game.

We recommend using a private BitBucket git repository for source control. Groups of 3 or more will be required to use source control, but we strongly recommend that everyone, including solo acts, use source control. Note that any source control solution you use must be private so members of other final project groups cannot read your code.

Final 1 - Due Apr. 11

In the first week, you will get your project up and running, primarily working on your engine features. You must have a working source control system in place. You will likely have a long design meeting with your group members to discuss and document code structure, component interfaces, libraries you will be using, and to refine gameplay ideas. After that you should begin your engine feature implementation. The implementation may use code from any group members’ implementations of projects so far this semester.

The formal requirements for your handin will be determined with your mentor TA, but if you want
a rough idea of what to expect:

- All engine components must at least be fully stubbed out, meaning interfaces and empty
  methods exist. This includes full integration of any libraries being used. In addition, we
  strongly recommend beginning engine feature implementation as well, but not much will be
  required.
- The project must run without crashing, but can be just a black window.

Final 2 - Due Apr. 18

In the second week, you will finish your engine features and create a basic demo for them. Most
engine features can be tested independently of others. For example, you can test your AI and
physics even if your networking isn’t quite completed yet. In addition to finishing engine features,
you will implement the most basic parts of your game (like player motion), though it will likely not
result in a fun game. All games will be playtested in class this week.

Expect requirements to look like the following:

- Engine features completely implemented and mostly integrated.
- Basic gameplay implemented but not necessarily “fun”.

Final 3 - Due Apr. 25

By the end of the third week, you will have a rough game implementation that can be played and
completed that is ready for public playtesting. Each group member will need to playtest at least 5
people who are not in the class and write down some notes for each playtesting session (like an open
beta test). Public playtesting will likely be done through some combination of asking other CS
students in labs, bringing people into the SunLab, and running playtesting elsewhere on a laptop.

Regardless of how you playtest, we recommend making a /contrib project so that any student can
run it from their own account. You’ll have to do this for final5, so getting a headstart is a good
thing. If you’ve never made a /contrib project before, ask the TA’s and we’ll point you in the
right direction.

Your requirements for this week will likely be:

- Gameplay refined to the point of “playability”. Every game feature may not be implemented,
  but a player can play for a few minutes and enjoy the experience.
- 5 public playtesting signatures per group member with notes for each playtester.

Final 4 - Due May 2

Although this isn’t the final week of the project, your game will be mostly complete by the end of
this week. With plenty of playtesting feedback in your hands, you will polish the gameplay and
iron out any kinks left from week 3. You will also gather another 5 playtests per group member,
and hopefully use this feedback to finish up the polish.
In addition to finishing your gameplay, you will create and give a postmortem presentation in which you discuss 5 things that went well and 5 things that didn’t during the development of the game. These can be about the final product or the process of creating it.

Requirements will look like:

- Polished gameplay. A person should be able to play your game with every planned feature implemented, and it should “feel” good.
- 5 public playtesting signatures per group member with notes for each playtester.
- A postmortem presentation powerpoint, handed in separately under the assignment postmortem as a ppt or pptx.

Final 5 - Due May 9

Your final project is complete! We are requiring that you make a demo video at least one minute long that shows off your game, so you can show it off whenever you want. Since videos will be too large to hand in, you will hand in a file called video.txt containing a link to the original high-quality video file (e.g. vimeo, google docs, public dropbox, but not youtube), along with a final copy of the game, including all code and assets used. In addition, you will have to put your game in /contrib so that future generations of Brown CS students will be able to see your work!

The final set of requirements are:

- A link to a video demo of the final project that is at least 60 seconds long
- The name of your /contrib script
- Additional polish (optional)
- Cleaned up and documented code (optional)

To record a video on the department, you can run:

```
ffmpeg -f x11grab -r 24 -s WxH -i :0.0+X,Y -vcodec libx264 -vpre lossless_ultrafast -threads 0 video.mkv
```

This will record a video that is W by H pixels starting at (X, Y). Department machines will have trouble recording at realtime rates, especially if your recording area is large. You may want to slow your game down (divide your timestep by 4), record in slow motion, then speed up the video 4x afterwards. To speed up a video (96fps / 24fps = 4x), you can run:

```
ffmpeg -r 24 -i video.mkv -sameq -r 96 video2.mkv
```

Handing In

Hand in the entire directory tree for your project, including both your engine and game code. You must also include a README file that describes how to verify each requirement, and an INSTRUCTIONS file that describes how to play your game, as specified in the Global Requirements.
To hand in, run `cs1972_handin finaln` from the top level directory of your project (which should be where your Qt pro file is), where `n` is the checkpoint you are handing in. **Please do not hand in the build files from your project.**

There is one exception: in week 4, you must hand in your postmortem separately by running `cs1972_handin postmortem`.

**Engine Features**

As previously mentioned, each group member will be responsible for implementing 5 points worth of engine features for the final project. **If you are taking the course for a capstone, you will be responsible for 7 points worth of engine features.** For each of the below features, the associated point value may be lowered if you decide to use a library to implement that feature. We understand that learning and integrating a library can be difficult, but because this class focuses on implementing your own game engine, we want to encourage and reward those who implement their features from scratch.

In addition to the above requirement, **each group member must be responsible for at least one 3+ point feature.**

The following is the point breakdown for possible engine-features:

- Networking - 5 points
- Sound - 2-3 points
- Deferred lighting - 4 points
- Particles - 1 point
- Portals - 3 points
- Other advanced graphics effects (shadows, toon shading, etc...)\(^1\) - 1-5 points
- Rotational physics of convex polyhedra - 5 points
- Spatial acceleration data structures - 2 points

In addition to the above, we offer points for “exceptional use” of certain engine features that are more dependent on how they are used game-side. Most, but not all, of these features must be implemented engine-generic like the above, but the bulk of their work will be in how they are used game-side. The TA staff holds the right to determine how many points are earned for the following depending on how they are implemented and used, so be sure to confirm with your mentor TA what is expected of you:

- Embedded scripting - 3-5 points
- Skeletal animation - 2-4 points
- Rigged mesh animation - 3-5 points

\[^1\]See the CS123 final project specification for more inspiration
• Advanced AI - 1-3 points
• Polished UI toolkits - 1-2 points
• Procedural content generation - 1-3 points
• Data persistence - 1 point
• High scores system - 1-3 points
• Level editor - 1-5 points