IN3026
Milestone 1
Report

Table of Contents

This report is kept short, to be looked at in combination with other documentation such as the README.md and design/development md docs. **These are not critical to the report unless otherwise stated or referred to.**

Game Idea

A strategy game prototype inspired by The Aether mod for Minecraft, as well as a combination of strategy games I play. The target style is to have a hyper-realistic low-poly game of floating islands, connected by bridges where the player will expand and build their citadel. The enemy will come in waves in attempt to destroy buildings and make the game harder. The name "Skyward Citadel" was chosen as it fits the theme of the game.

A full explanation of the design, feature exploration and design choices can be found in documentation, which can be accessed via the README.md.

Milestone 1: Implementation

A lot of Milestone 1 focuses on implementing the base core and structure of the game.

The following has been implemented so far:

- **text_renderer (engine)**
  - Updated to make use of multiple fonts.
  - Generalised code engine-side, to now use by default the 2D shader (unless explicitly passed in).
- **camera::perspective_camera (engine)**
  - Added an isometric mode without altering previous camera setup in case it might be useful. This could be expanded by making the choice up to the game implementation, but for now I've left it out since I only intend to use the isometric camera.
- **Rock primitive (engine)**
  - A primitive shape randomly generated based on inputs: radius, subdivisions, roughness.
  - This means that different unique rock formations can be made by creating multiple rocks.
- **game_state_manager (game)**
  - Created a state manager with states for controlling the current state of the game.
  - This will tie in with other managers to send events like **key_pressed_event** to each layer, which will handle keys differently.
- **controlled_layer (game)**
  - An abstraction on top of the engine layer, which is controlled and can change states using the game state manager.
  - Will be further expanded to contain any other managers controlled by the game in a more centralised and generic manner.
- **layers/main_menu (game)**
  - Implemented a button system to select a given option from the menu.

    - Will be further expanded with support for mouse, and potentially controller.
- **layers/game_play (game)**
    - Main game layer which will define all assets
    - Currently it holds 3 meshes, but all of them have been proven to render and be compatible except MTL files
        - A TODO has been made for this, unless the assets change
    - It also holds one primitive type – a rock, since it was relatively hard to find anything fitting into the game theme
- **Asset Preparation (assets)**
    - Prepared music tracks
    - Prepared some basic models, textures and assets
        - The kaykit-medieval-hexagon pack fits very well with the game, so I might replace all models with those in favour for a game design in that style

Assets Used

All assets are referenced in the README.md under the "Usages" heading, with details of the type, source and licence details.

Dependencies Used

Only the engine has been used with minor alterations to simplify certain aspects of code and stylise them to work better for the given game. This means that the only dependency is the AGT engine/template, including all its original dependencies.