Group 31 Code Connoisseurs - Unity subteam D2 report

Ricky Wen, Sirui (Ariel) Chen

1. summary of decisions and options

Game Engine: This subteam holds responsibility for designing and making the game. For the game engine, we have chosen to use Unity. Our partner had specified for us to use either Unity or Unreal Engine for game development, and so we carefully considered these 2 choices before making the decision. Unreal Engine is more suited for games with high quality graphics and can allow heavy computing. However, our game is a web based low-poly game that does not demand high quality graphics and should be able to run in a web browser. To this end, Unity is a much better option. Unity is a slightly lighter engine and can support high end graphics if needed, but was sufficient for the purposes of this project. One member in the team has prior experience with Unity game development, which will aid team members to master essential techniques. Since Unity is much more well-documented, with many tutorials and forums online, it is also easier for our members to learn given the short timeframe.

Art Direction: Given the short timeframe for development and a lack in expertise in technical art, we have chosen a low poly voxel art style. This art style is less time consuming but still conveys the visual feel we want the game to have. MagicaVoxel is the tool that we chose to use to create the models. It is a simple 3d voxel editor that is easy to use and integrate into the Unity workflow. We will utilize things like shaders and post processing to make the game still look good despite having low detail models.

Game Design: The game that our partner envisioned and wanted us to make is a city building game at its core, with an AR twist. Therefore, we needed to design the core gameplay elements for a city building game. We decided to make the player focus on increasing a specific parameter, population. The player will need to supply their population with different services such as electricity, water etc, while balancing the sustainability of the city development, something the partner requested. For example, if the user builds a coal plant to provide electricity for their populace, the coal plant will cause air pollution, which will negatively impact the community (reflected in a decrease in population capacity for housing). To combat these negatives, the player will have to invest in sustainable development such as using green energy solutions like wind turbines or solar panels. We have made various revisions and conducted extensive discussions with the partner to ensure that we are on the same page and that the game we are making will convey the ideas the partner wants the game to. Note that only some functionalities outlined here are implemented in D2.

UI: For UI, we designed and implemented the inventory menu specific to this deliverable. There are some design decisions that we made for the inventory menu. For the viewing format of the menu, we choose to make it a scroll view instead of paging turning with buttons, which is more user-friendly and easy to operate during the game. For the menu item display, we choose to have an image icon with its name in the text. The image icon will help users visually recognize each menu item while the text helps distinguish similar items. For image icons, we purposely get them from prefabs(a special type of component that allows fully configured GameObjects to be saved in the Project for reuse) rather than models. This is because the prefabs contain extra features, for example the house has trees around it, and also accompanied with better colors. The prefab also shows special effects such as the

water effect in the sewage plant that is done through a shader, which is separate from the models. To this end, we believe image icons from prefab will provide a better visual experience to our users.

2. **individual contributions**

Ricky: Set up the unity project. Implemented a basic map composed of differently colored tiles. Implemented a water shader. Implemented a building system in which players can place, delete, rotate, and move objects, all via keyboard inputs (Ariel implemented UI to aid some of these functionalities). Made the current building models. Implemented the road placing system, which allows users to place roads by clicking and dragging on the map. Roads will automatically change their model (1-4 way) based on the adjacent roads.

Sirui (Ariel) designed and implemented the inventory system with a functional GUI display. For now, the inventory UI contains buttons for some existing game objects such as houses, water power stations and coal power stations. These buttons are connected to the existing game logic that Ricky implemented. They enable the user to select a game object from the inventory menu by clicking on the corresponding button, and then place it on the map. The inventory UI also utilizes a scrolling view, which aids users to check all inventory objects when the page cannot display all at once. (4 actual items and 2 placeholders in the inventory to test scrolling for now) Each inventory object has a corresponding subscript number indicating the amount of objects currently in the inventory. This is planned to connect to server and database for displaying real time numbers for later phases. Besides implementation, Sirui (Ariel) also contributes to organize tasks/to-do lists for the team, test game functionalities and suggest possible solutions for detected bugs, and also coordinate with other subteam in terms of possible game designs for better connection later.

3. **Details and instructions**

To download and test the game, clone the repository and open the builds folder. Run the corresponding executable for your operating system. We've included Windows, MacOS, Linux, and WebGL.

Features to test:

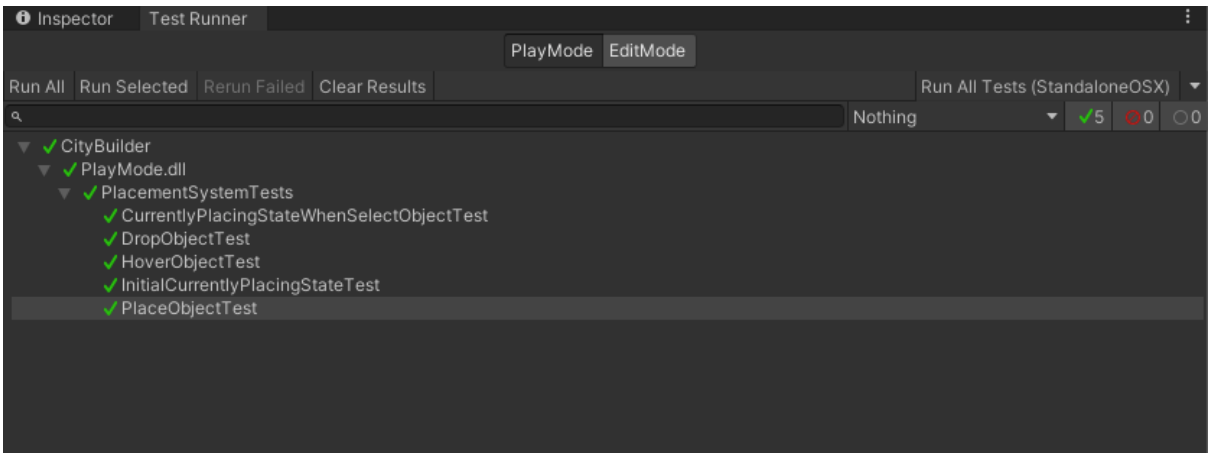
- Zoom in/out: Move your move to the map and scroll up/down to zoom in/out of the whole game scene.
- Move Camera: To look around different parts of the map, use WASD.
- Inventory scrolling: Move your mouse to the inventory menu and scroll up/down to browse all items in the inventory (4 actual items and 2 placeholders in the inventory to test scrolling for now)
- Placing objects: Click an icon button in the inventory menu on the left. A game object instance should be hovered with the mouse moving. Then, move your mouse and click on the map to place the object. Notice that if you hover the object to a vacant map tile, the tile will turn green indicating that the object can be placed to the current tile. Otherwise, the tile will be red and you will not be able to place it on that tile.
- Deleting objects: Click on an object on the map, then press the delete button on your keyboard. Note: deleting objects will be implemented via UI in the future, instead of requiring keyboard inputs
- Rotating objects: Click on an object on the map, then press the left or right arrow keys to rotate. Note: rotating objects will be implemented via UI in the future, instead of requiring keyboard inputs

- Moving objects: Click on an object on the map, then move the cursor to the newly desired position and press on the map again.
- Placing roads: Press the R key on your keyboard. Then, hold and drag your cursor on the map to place roads.
- To close the application, use alt-f4 (we will implement an exit button in the future)

Note: roads are not treated as objects and currently cannot be deleted or moved. That feature will be implemented at a later date

How to run the unit tests:

To run our unit tests, you would have to download Unity and UnityHub softwares. Clone the repository and open *Unity/CityBuilder* folder in Unity. Go to *Window->General->TestRunner* and click *RunAll* in PlayMode. You should see all tests pass like the following:



4. Application /Deployment

For D2, we focus on the following user story: *As someone who wants to design a city/town layout, I want to be able to select an item from my inventory and place it at a desired position on the map in the city builder game, in order to carry out my city plans and create a simulation of the city I want to design.*

As a user, you should be able to access this user story following our instruction above (in part 3).