City Builder Game technical document

team #31- Code Connoisseurs

In this document, we will introduce important files/folders, classes/packages in the repository to help future developers hands on quickly.

For frontend (game development): files are under Unity/CityBuilder folder
For backend (server, node.js development): files are under Server folder

Builds folder

- Linux
- Mac
- WebGL
- WebGL_remote_server
- Windows

Server folder

- server.js
- package.json

Naming convention for the following folders:
    General user-related files (User…)
    Inventory-related files (Item… or Inventory…)
    Map-related files (Map…)

- routes/
  - url for get/post requests
- controllers/
  - functions for handling requests from the client side and interacting with the database
- models/
  - definition of schema for the database (MongoDB)
- test/
  - unit test for server

Unity/CityBuilder folder

- Assets
  - Scenes
    - LoginScene, MainScene (for game)
  - Scripts
    - Login/logout functions:
      - user login/signup, logout related files (in UI folder):
        - Login

      - Logout
- Map functions:
  - generate map, save/load map to server
    related files:
    - MapDataManager: top-level functions for the map: generate, save, load
    - MapTile: tracks the occupancy of the tile as well as what object is placed on top.
    - SerializeHelper: serializable map data structures for communication with the server
    - SaveFile: send requests to the server, callback MapDataManager functions
- Inventory functions:
  - live update inventory information to server
    related files:
    - InventoryManager: top-level functions for collecting and storing inventory information. Inventory information is saved in a dictionary-of-dictionary format: {key - category: value - inventoryItem}, where inventoryItem is a class that contains name, quantity and itemID of a distinct item.
    - InventoryList:
      - list of all prefabs in the game, used for redrawing the game map
    - InventorySerializeHelper: serializable inventory data structures for communication with the server
    - InventoryToServer: send requests to server, callback InventoryManager functions
  - UI functions (in UI subfolder)
    related files:
    - UI/ItemUI: functions to manage inventory UI buttons in the game. It includes the name, quantity and itemID retrieved from the game inventoryManager and enables real-time quantity updates on the game screen.
    - UI/Login:
    - UI/Logout:
    - UI/MenuManager: Manages which UI components for the various inventory categories and the inventory itself should be displayed based on the user's clicks.
    - UI/ObjectMenuManager: The menu that shows up when the user clicks on a building. Contains the button functions.
- Resource functions:
  - live update resources information to server
    related files:
    - ResourceDataManager: works with inventory manager to interact with server for resource information
  - harvesting functionality
    related files:

                - HarvestManager: script that keeps track of number of occupied harvesters and available harvesters
                - HarvestSystem: implements harvesting natural resources in a 3x3 grid, giving resources based on natural resources harvested
        - Game functions:
            - CemaraController: moves the camera
            - CloudManager: spawns clouds outside the camera’s viewing range over time
            - CloudsOnLoad: spawns and deletes the clouds that spawn during the transition between the login scene and the game scene
            - CursorManager: Manages which cursor sprite should be used
            - PointerDetector: Finds the closest colliding tile to the pointer and returns its position
            - InputManager: Uses raycasting to find what tile the cursor is currently on
            - PlacementSystem: hosts all the functions for managing objects on the map. This includes placing continuous objects such as roads, spawning, placing, deleting, rotating, and selecting buildings.
            - UtilitiesManager: singleton that facilitates utility buildings to update the allocated utility count for neighboring houses.
            - Road: determines what road model to be used for a road object depending on how many roads are adjacent to it.
    - Animations:
        - Enable the transition of pop-up and close window during the game
        - manages the states of the different inventory UIs and makes transitions when they need to be opened/closed.
    - Models:
        - Uniquely-designed Game Objects models for displaying in the game, including Buildings, Decor and Roads subfolders
        - Made using MagicaVoxel. Most models are 80x80x80 and imported at 0.25 scale.
    - Prefabs:
        - Configured Game Objects for the project to reuse
    - Tests
        - Unit tests for Unity functions