Jankscape

4/3/2017

Colin Gidzinski
Kyle Leng

Overview

In an effort to reminisce about old games of my childhood i've decided to combine the conceptual idealizations of two games that I have invested a vast amount of time in my life (Runescape and Rust). These games have been distilled into a 2D Tile based multiplayer game. Although many aspects are still missing the core functionality remains as a good proof of concept and starting point for further development (Summer 2017). It was decided that a game would be a proper choice for this final project as it meets all the requirements and thoroughly demonstrates not only being able to interacting multiple components with each other but also requires a very thorough knowledge of exactly how they operate and coexist in a 3-Tier internet model.

The website is also HTTPS secured since it is accepting passwords for users.

Goals

The goals of this game are to create a multiplayer world that can handle a large number of simultaneous players on the same server.

Each player will be required to login to their own account and have all their data stored in a database when not in playing.

Each Player can move and interact with the world around them by attacking enemies or destroying parts of the environment. Each user will also be able to interact with and each other.

When a player interacts with anything it should be visible to all players in the current area.

Completion and Code Style

In an attempt to keep the game code clean many features were implemented in a reusable and modular style. This can be witness by the Zone loading implementation and how map items and enemies are implemented.Although this resulted in much cleaner and efficient code it ended up requiring much longer to develop.

Unfortunately due to the extended time required to code clean and efficient modules some features had to be cut. I wish to work further on this game throughout the summer so instead of rushing and writing poor trash code. It is felt that this was the right choice as the implementations that were done are higher quality and not just written haphazardly in an attempt to shoehorn a feature in. The features that were cut where Player VS. Player combat and Items/Equipment.

Specifications

The front end of Jankscape is built with HTML, CSS and javascript for the front end. The game itself is almost entirely rendered on a Canvas element with small select items such as the console and playerlist being standard list html elements.

The backend server is based on a nodejs environment with socket.io for client/server communications and mongodb to store user login data and persistent in-game player data.

The client side Javascript code ended up being significantly longer than the server code. This is due to the map rendering engine and player controls. Although longer the server side was significantly more difficult to design and program (greater weight) as it had to be quick, efficient and reliable for use among multiple players at the same time in different zones. Adding a multiplayer component to the game increase the difficulty greatly in comparison to having the game be single player. Every function and action had to be strategically chosen and programmed with multiplayer in mind.

Individual Features

I. Login Authorization and Initial Data

When the user opens the game they are required to login to their account. Currently 2 test accounts have been setup and chosen from the login menu. When a user attempts to login they are verified against the data stored in the database. If the user exists then data about the player (Health, Name, Location) are sent back and used to render and place the user on the gameworld. (Open web console to see data retrieved from database on login)

II. Multiple Zones and Map Generation

On server start each Zone is loaded from a json file into a Zone object which is then added to a MultiMap object called "Zones". The data for each map is based on an array of X size comprised of 32*32 pixel squares. Each square is drawn in accordance to the value in the corresponding coordinate position.

Each Zone is made up of 5 parts:

1. textureMap: Identifies which base texture to display on each coordinate (Grass,Water,Dirt,Sand)

2. overlayMap: Identifies which overlay to place on top of the textureMap (Trees). This map is used to display non interactive elements of the map
3. actionMap: Used to determine if a player can walk on a coordinate or not.
4. Entities: Interactable objects that are removed permanently when destroyed/
5. defaultEntities: Interactable entities that can respawn after a certain time has elapsed

III. Multiple Users

When a user is loaded into a Zone, they are provided with the entire map layout which includes the textures and the entities. When this map is updated only the entities are resent to ensure the user doesn't need to request the entire map data again.

IV. Environment Interactions

Environmental interactions are based on the user attacking or using an object in the Zone which is then relayed to the server. If the action was valid the required changes are made to the entity and the entity map is then relayed to all players in that current Zone.

V. Server and Client Tick

The client is based on a 60FPS update tick which is gives a smooth refresh rate for gameplay. The server runs on the same 60FPS update tick for response to all user interactions.

Additionally there is an additional 1 Second tick which is used for respawning entities and moving enemies.

VI. Enemies

Enemies are based on Items with the addition that they have a chance to heal and move on each 1 Second server tick. Currently they do not harm the user as they have been tamed by the coders.

VII. Player interaction

Currently the only player interaction is the addition of not being able to be in the same coordinate as another player.

VIII. Client/Server interaction

When a user performs an action it is sent to the server as a request. The server then checks to see if the action is valid or if there is an error. If the action was valid the results are then sent to all players in the same zone as the original player who performed the action.

Operation

1. To run the game you must have NodeJs installed.
2. Go to the mmorpg/mmorpg folder (second mmorpg folder)
3. Run “npm install”
4. Run “node server.js”
5. Open localhost:8005
6. Select one of the test users to login
7. WASD to move (Around World or Map Change Pad)
8. Left Mouse to Attack (Rock, Grass, Enemies)
9. Right Mouse to Use (Chest)

Will not work offline