Koleman Nix
Brent Baumgartner
Morgan Locks

My Little Saga

Prototype Features

Our prototype has a character with free motion on a tile background. The camera is constrained by the edges of the map; follows the character unless he's near the edges. We also have a system for notifications and dialogues that autosize to the size of the text provided. We extended the basic collision detection to handle proximity collisions so that we can do actions "near" objects. Speaking of actions, we allow the user to do actions on objects he's near - a class keeps track of the current actionable item and the player can press "z" to action that item; in this case just removing the rock he's near.

You can use our prototype at https://my-little-saga.herokuapp.com/ . To run it locally, run npm start (or node server.js) and open localhost:5000.

Implementation

The camera panning was done within Prototype.js, which extends Game. The move method is where the magic is - feast your eyes.

ActionManager – Singleton, keeps track of the current actionable object. In the prototype's case, it has a reference to the nearest rock (set through the focus method. It also has a map of strings to functions so that it can call the function corresponding to the current actionable object. We use a map of strings to functions instead of attaching callbacks so that events can be serialized into the Game Map objects. It also draws a triangle over the actionable object to show that it's actionable.

Toast - Display a message to the user on the top left of the screen. Can be customized with optional params for size, position, and content. Automatically sizes to content's size. That last sentence was a bitch and a half to implement, let me tell you.

ToastManager - Queues Toasts. Will only display one at a time, and will swap it out after provided amount of milliseconds.

MapGenerator – generates json objects representing Game Maps. Each map has two keys: foreground and background. Each key corresponds to a 2D array of tile names.

MapReader – takes the output of MapGenerator and parses it. Can make ajax calls to pull levels stored elsewhere on the server.

QuestManager – handles all the events of the app. Extends EventManager. Right now, listens for collisions and proximity-collisions. When it gets collision events, it resets the hero's position so that it's outside the object it collided with. On proximity collision events, it shows a toast with a humorous message then selects that object as the current actionable item in ActionManager.

Game Vision

The prototype fits into the game like this:
We plan to have the character navigate through a tiled version of Iceland, at least the Southwestern regions of Iceland where most of the stories take place, following some of the actual events and places from the real medieval sagas.

On this journey, there will be mountains, tundra, rivers and lakes, enemies, friendly critters, and interesting challenges and battles. The point of the game is to explore this world Iceland, doing things and becoming more powerful. As you do things, you also accumulate "reputation points". These represent your progress through the game. At 100 reputation points, you will have the option to "beat the game" by fighting the King of Norway and putting him and his tyranny in his place.

The prototype sets us up to realize the vision of this game by implementing all the basic mechanics we'll need:

1) Generating and loading maps / areas
2) Basic character movement
3) Proximity detection to trigger events such as battles or conversations
4) An "interaction engine" that allows us to program unique interactions between the hero and all the objects in the world, however many we decide to add.
5) A "Toasting" engine, a.la Android, that allows us to display some text to the user when they discover a new area, or interact with anything.

It's now, we believe, simply a question of creating the world and populating it with content.