CS 4560

Fall Semester 2016

PerlSquad

Team Members:

Alex Mayle, Brian Reynolds

Favour Ogundare, Eric Keep, Robert Smith

Clients:

Dr. Danielle Dani, Emma Coleman,

Sarah Brown, Dr. Theda Gibbs

Teaching Assistant:

Yunyi Feng

Professor:

Chang Liu

Table of Contents

Requirements

Overview:

- Two of our clients, Sarah & Emma, are making a board game for their students meant to help them learn and understand how biomes work.
- Dr. Dani & Dr. Gibbs reached out to Dr. Liu and by extension, our team, to work on a computer game version to go alongside it.

Use Cases & Actors:

Use Cases:

- Adding Info to Game - Allows Teacher to add in their own scavenger hunt objects with info and image
- Editing Info in Game - Allows Teacher to remove or edit objects in the game
- Resetting Defaults - Resets objects in game to the defaults (what we set up)
- Game Start - Pick the number of players and start the game

Actors:

- Teacher - Classroom teacher who sets up game and does most of the clicking

- Class - Students who are learning from game/playing game

Use Cases Plus Narratives:

1. Use Case Name: Adding Info to Game
   Participating Actors:
   Teacher
   Pre-Condition:
   - Teacher has selected ‘Settings’ option.
   - Teacher has selected ‘Add Info’ option.

   Flow of Events:
   - Teacher adds images using ‘Add Photo’ button.
   - Teacher selects the biome the photo should go in.
   - Teacher selects the category of the photo (i.e. animal, precipitation, temperature, etc.).
   - Optionally teacher types fact about whatever is in the image in the ‘Biome Fact’ text box.
   - Teacher clicks ‘Save Changes’.

   Post-Condition:
   - Inserted info is added to the corresponding section of the game.

   Special Requirements:
   - Photo does not exceed maximum size.
   - Text does not exceed character limit.

2. Use Case Name: Editing Info in Game
   Participating Actors:
   Teacher
   Pre-Condition:
   - Teacher has selected ‘Settings’ option.
   - Teacher has selected ‘Edit Info’ option.

   Flow of Events:
   - Teacher selects the info they want to edit or delete.
   - Teacher edits or deletes the corresponding info (can individually edit each section or delete the entire thing).

- Teacher clicks ‘Save Changes’.

Post-Condition:

- Edits to info are applied.

Special Requirements:

- Must have at least one object per category per biome.

3. Use Case Name: Resetting Defaults

Participating Actors:

Teacher

Pre-Condition:

- Teacher has selected ‘Settings’ option.
- Teacher has selected ‘Edit Info’ option.

Flow of Events:

- Teacher Selects ‘Reset Default Game’.
- Text pops up warning about changes.
- Teacher confirms changes.

Post-Condition:

- Game is reset to default.

Special Requirements:

- None

4. Use Case Name: Game Start

Participating Actors:

Teacher

Class

Pre-Condition:

- Teacher has game customized to desire.

Flow of Events:

- Teacher selects number of players based upon how many teams Teacher wants to divide Class into.
- Teacher clicks “Start”

Post-Condition:

- Number of players is initialized

- Game board appears with player 1 selected and all scores displayed as 0.

Special Requirements:

- None

Technology Stack

Technology/Libraries:

- CreateJS (http://www.createjs.com/) - uses JavaScript and HTML5 Canvas
    - EaselJS (http://www.createjs.com/easeljs)
    - PreloadJS (http://www.createjs.com/preloadjs)
- Photoshop (http://www.adobe.com/products/photoshop.html) - image editing tool
- GIMP (https://www.gimp.org/) - image editing tool

Descriptions:

- EaselJS - Used to create simple graphics like lines and boxes for an HTML5 canvas.
    - Classes Used:
        - Bitmap
        - Container
        - DisplayObject
        - Graphics
        - Grapics.Circle
        - Graphics.Fill
        - Graphics.Rect
        - Shape
        - Stage
        - Text
- PreloadJS - Used to manage and co-ordinate the loading of assets.
    - Classes Used:
        - AbstractLoader
        - LoadManifest
        - LoadQueue
- Gimp/Photoshop - Used to crop/scale image assets for game.

Discussion

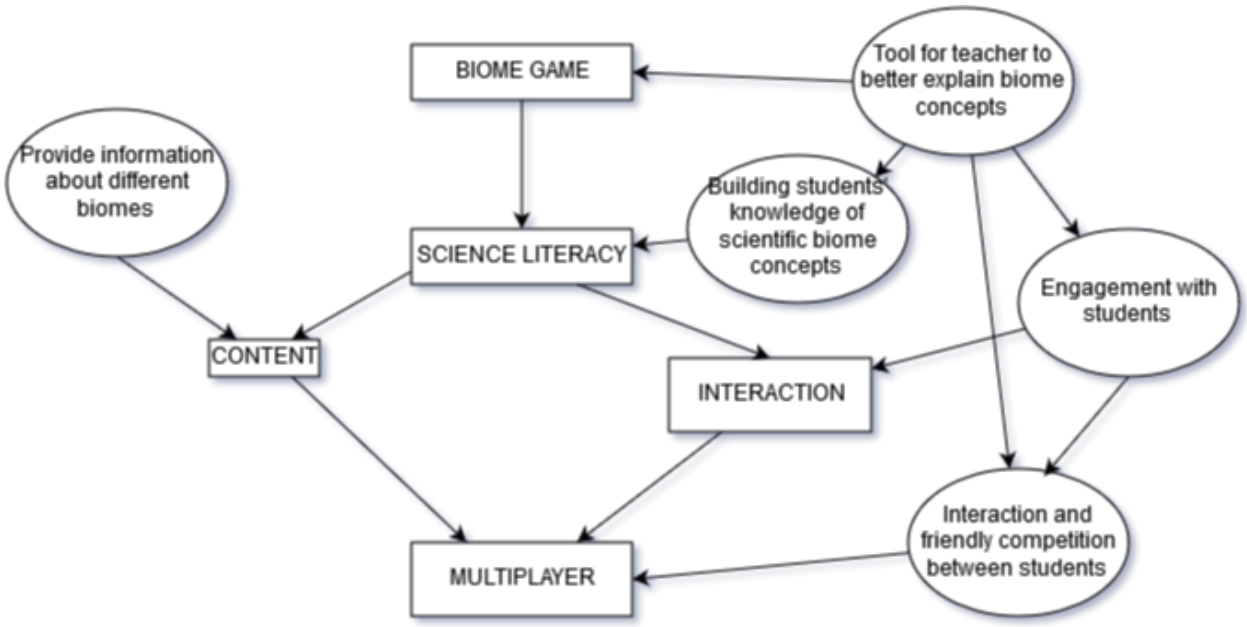
1. CreateJS was chosen because it was listed as one of the most popular choices for basic game design libraries. Also, it was well-documented and widely supported with multiple online demos. The libraries it contained simplified creating, filling, and updating stages with javascript.
2. Gimp/Photoshop were chosen because they provide methods to crop/resize and edit images used in-game.

Links:

- CreateJS: http://www.createjs.com/
- Gimp: https://www.gimp.org/
- Photoshop: www.adobe.com/Photoshop

Design

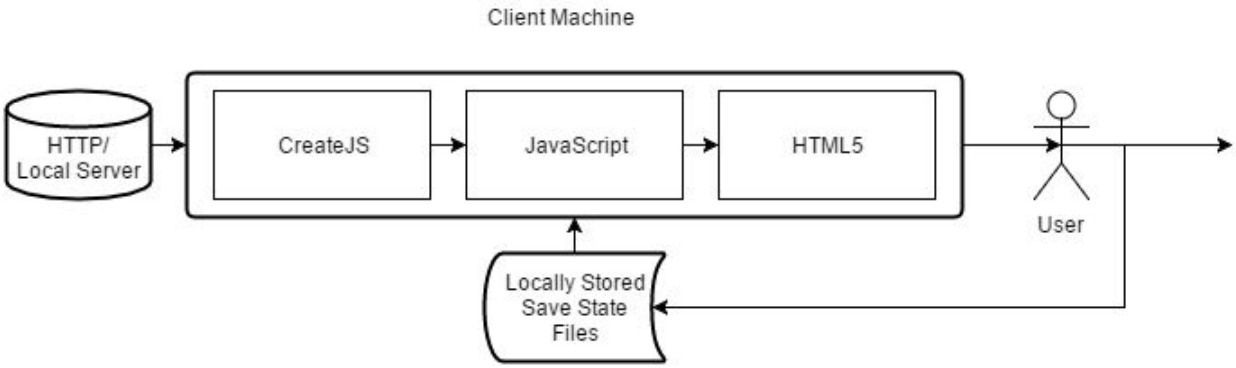
Purpose-Concept Map



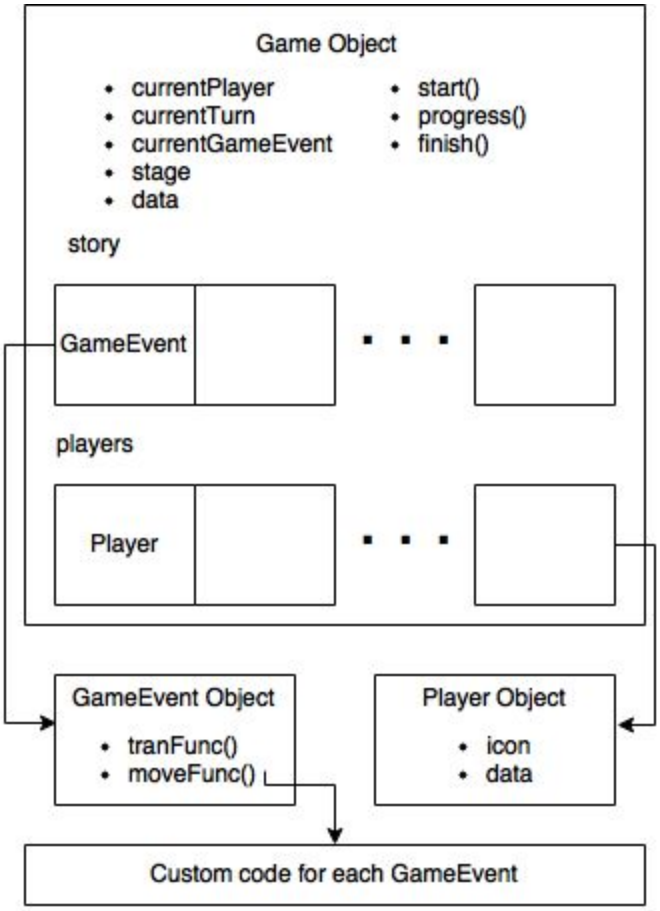
Purpose Concept Map: Rectangles = concepts, Ovals = Purposes

The goal is to build a fully functioning Biome Game. The game is designed for college students studying to teach middle school-aged and younger kids. Some of those college students might also take the game and use it in their classroom. It is intended to promote education through gaming, while providing a fun-filled, competitive and interactive experience.

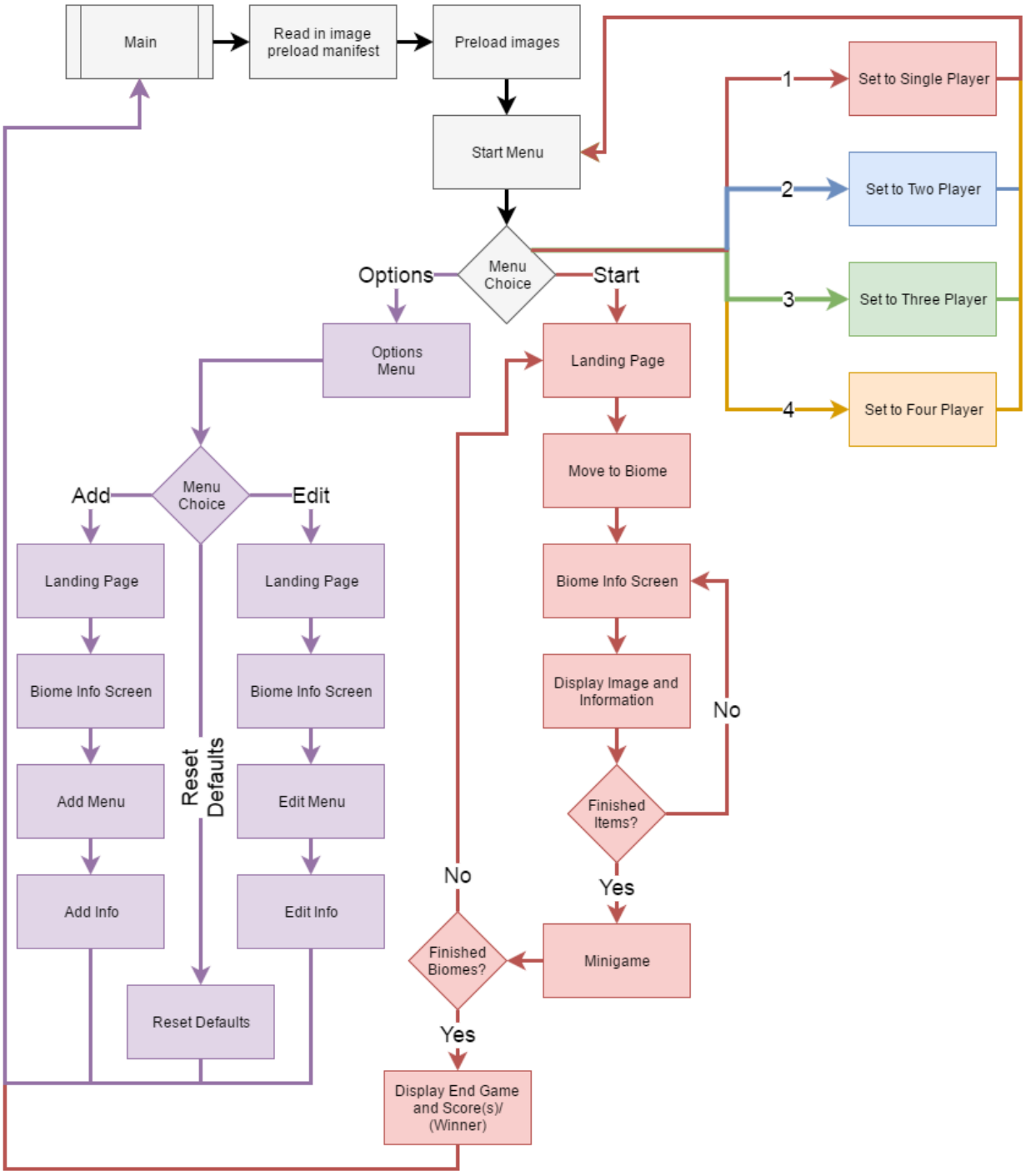
System Architecture



The game can be run on an HTTP web server or locally. Once the game starts running, the user will have the option to edit the game objects. Once the user hits "Start", the game will retrieve the images/item descriptions from their respective files and start the game. A frontend website could potentially be added in the future to host the game as well, but for now, it will just run locally.



Implementation Notes


Main
Read in image preload manifest
Preload images
Start Menu
Menu Choice
Options
Start
1
2
3
4
Set to Single Player
Set to Two Player
Set to Three Player
Set to Four Player
Options Menu
Menu Choice
Add
Edit
Reset Defaults
Landing Page
Biome Info Screen
Add Menu
Add Info
Landing Page
Biome Info Screen
Edit Menu
Edit Info
Reset Defaults
Landing Page
Move to Biome
Biome Info Screen
Display Image and Information
Finished Items?
No
Yes
Minigame
Finished Biomes?
No
Yes
Display End Game and Score(s)/ (Winner)

While the CreateJS libraries and the HTML5 Canvas provide a graphical platform, we needed some sort of game engine to provide core game constructs. These include abstractions for common game elements such as “players” and “turns,” as well as a means to progress through the game from start to finish.

The above image is a simplified model of the game engine architecture. Starting from the top, we see that the Game object provides what is intuitively a context for what is happening at any point in the game. It provides members and functions to retrieve information such as the current turn, active player, game progress and more. It also provides functions to manipulate the context of the game, such as those used to move a player forward or go to the next turn. Although Javascript is a prototyped language, meaning new members and functions can be added to the Game object at runtime to add additional details about the game’s context, it still offers the `data` object as a generic container for more game information.

The Game object also maintains an array of Player objects proportional to the amount of students playing. References to specific players may be retrieved such that the player’s score or other information may be associated with them at runtime. Furthermore, if the player is graphically represented in the game, a reference to their EaselJS Shape object is available for manipulation.

Most importantly, the Game object maintains a list of GameEvent objects. It is through this that the game engine provides facilities for a deterministic linear story game. Each GameEvent has two members, tranFunc() and actionFunc(). The game engine starts with the first GameEvent in the array, executes its two members functions and moves onto the next GameEvent until there are no more. At which point, Game.finish() is called to wrap up any loose ends. Game.start() is also available to put any code that needs to be executed before the first GameEvent. The tranFunc() and actionFunc() members of each GameEvent point to custom code that constitutes the actual story of the game. The tranFunc() member usually contains code that segues from one portion of the game to another, whether that be in a graphical or loading sense. The actionFunc() member contains what actually happens, including functions for handling user input, rendering the UI, and anything else.

To code a new portion of a game, create an object in its own file containing everything needed to facilitate the portion of the game. If data needs to be saved for a later GameEvent, store it in `game.data`. It is assumed a Game object named "game" is in scope at the time of execution. Then, add a GameEvent object to the Game object by adding code to it's `loadStory()` method as shown below.

The only requirement in the code is the `actionFunc()` member must eventually call `game.progress()` to move onto the next GameEvent at some point.

```
51
52      // Assign different code to different GameEvents
53      story[0] = new GameEvent(this.transition, this.startMenu);
54      story[1] = new GameEvent(this.transition, this.singleClick);
55      story[2] = new GameEvent(this.transition, this.scavHunt);
56      story[3] = new GameEvent(this.transition, this.singleClick);
57      story[4] = new GameEvent(this.transition, this.scavHunt);
```

Known Issues

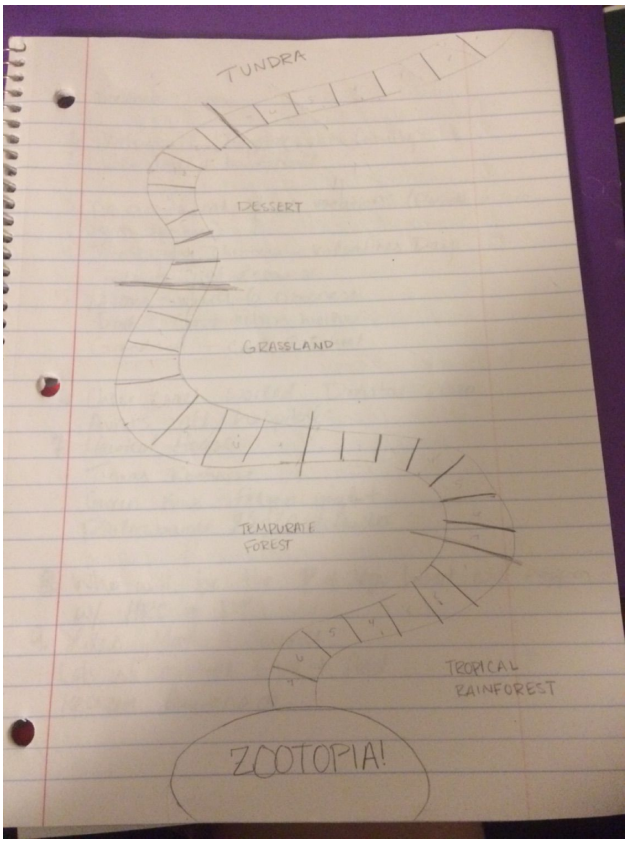
- Certain browsers don't play nicely with HTML5/Javascript (Firefox seems to work best so far).
- Even on Firefox, however, sometimes the javascript portions won't run on the first loading of the page, and the page must be refreshed

Links

Github: https://github.com/Sagerune/PerlSquad

CreateJS: http://www.createjs.com/

Appendix


TUNDRA
DESSERT
GRASSLAND
TEMPURATE FOREST
TROPICAL RAINFOREST
ZOOTOPIA!

Sarah Bown

This is a super rough sketch for our board. Basically it's just winding around the biomes. We have 7 big things that will work across the biomes: precipitation, average temperature, latitude, growth season (or example of plant life) and small, medium, and large examples of animal life. They are going to roll a die and each space will have a tab with some graphic indicating which of the seven things that space will have information on. So a paw print for the animals, a thermometer for the temp, rain drop for precipitation, etc. They'll flip the tab over, and it'll say something like "in the tundra one of the largest predators is a polar bear" or some other fun fact. We'll have a baggies with different pictures and they'll be asked to take the picture of the polar bear and add it to the tundra so they will build the biome as they go. Ideally we'll have 4 players for now, but it could really work with 1-4. Preferably at least 2.

==Project Info:

To develop a fun game that engages the player with science concepts, processes, and practices related to Biomes and integrates literacy specific functions.

Here are some useful links about Biomes:
http://www.ucmp.berkeley.edu/glossary/gloss5/biome/
http://kids.nceas.ucsb.edu/biomes/
http://earthobservatory.nasa.gov/Experiments/Biome/
http://www.geo.arizona.edu/Antevs/biomes/

Suggested game engine for kids:
Scratch: https://scratch.mit.edu/