CS 4560

Spring 2017

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 3

Overview 3

Use-cases & Actors..... 3

Narratives..... 4

Technology Stack 6

Libraries/Descriptions 6

Discussion 6

Design 8

System Architecture 9

Implementation Notes 14

Known Issues 15

Build Instructions.....16

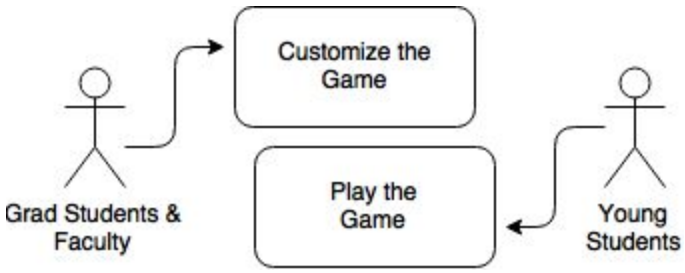
Additional Links 17

Appendix 18

Requirements

Overview:

- Game must effectively teach young students about biological biomes
- Be useful in our clients' graduate programs as a teaching tool of their own.



Use Cases & Actors:

Use Cases:

- Game: Young students will play the game, progressing through the biomes and learning along the way.
- Customization: Our clients and their graduate students will customize the game with content as an exercise for designing effective teaching tools.

Actors:

- Young Students: Those who actually play the game

- Clients & Grad Students: Customize the game as an exercise in their graduate programs, lead by our clients.

Narratives:

1. Use Case: Game
   Participating Actors: Young Students
   Pre-Condition:
   - Game is pulled up in a web browser

   Flow of Events:
   - For each of the biomes
     - Students will peruse the "info screen," containing information regarding the temperature, animals, etc of each biome
     - Students will proceed after they have acquired the knowledge by pressing an "OK" button
     - They will participate in a "scroll game" where they are challenged to click the logical choice of animals as they scroll around the screen, based upon the knowledge gathered at the "info screen."
     - Student will receive feedback, play again, or go to the next biome.

   Post-Condition:
   - Student can play the whole game over or not, probably depending on the instruction of the supervising teacher

2. Use Case Name: Customization
   Participating Actors: Clients and Grad Students
   Pre-Condition:
   - Game pulled up in a web browser wit the start menu open

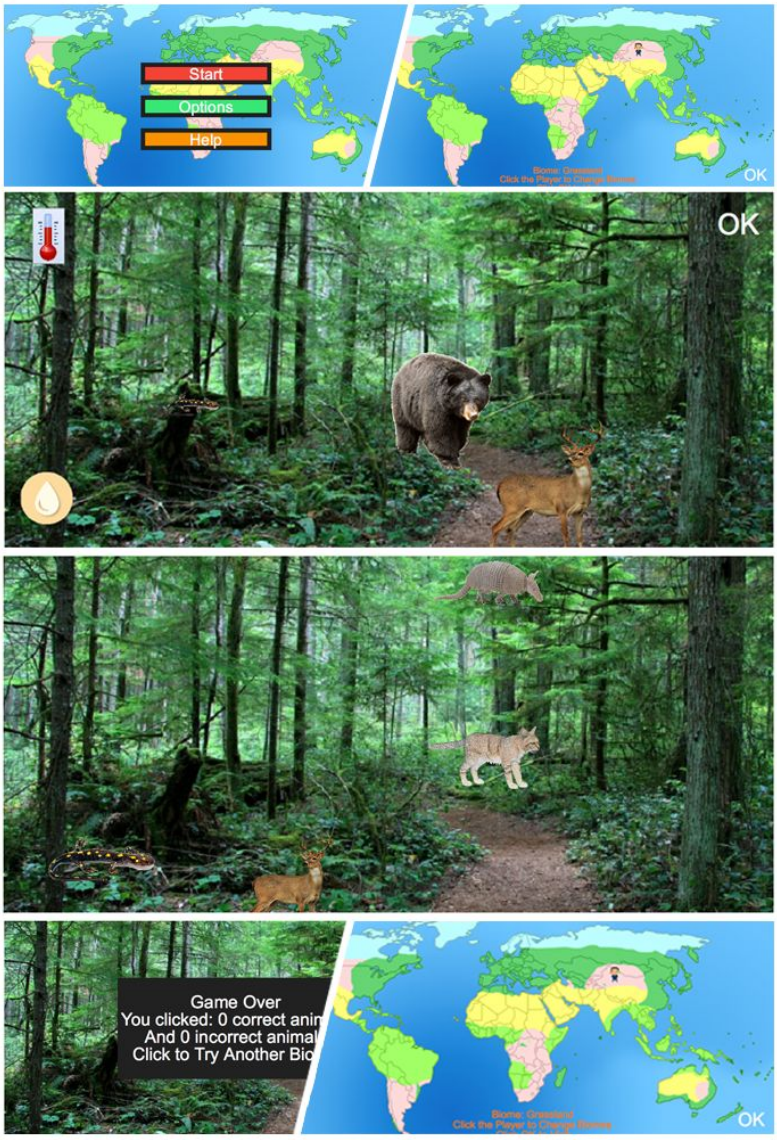
   Flow of Events:
   - Teacher will be able to graphically manipulate each biome to their liking or choose a manifest with changes implemented from a previous customization session.

- Teacher can add/remove images to/from a biome s well as edit the scaling, coordinates, and text of each image.

Post-Condition:

- The changes made by the user will be reflected in the game upon playing, and a manifest will be saved so that it may be chosen at a subsequent time.

Below is a storyboard depicting the walkthrough of the game. This will be the sequence of events that the young students will perform while playing the game.



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/Photoshop) - image editing tool
- GIMP (https://www.gimp.org/) - image editing tool
- PHP (http://php.net/downloads.php)\
- HTML5/CSS - Standard supported by multiple web browsers
- JavaScript (https://javascript.com) - basis of graphics
- FileSaver.JS (https://github.com/eligrey/FileSaver.js/)
- Blob.JS (https://github.com/eligrey/Blob.js/)
- jQuery (https://jquery.com)

Descriptions:

- EaselJS - Used to create simple graphics like lines and boxes for an HTML5 canvas.
- PreloadJS - Used to manage and co-ordinate the loading of assets.
- Gimp/Photoshop - Used to crop/scale image assets for game.
- PHP - Used for uploading images to the server
- JS - Used for HTML5 canvas
- FileSaver.JS - Saves blobs as text files
- Blob.JS - implements blob interface on browsers that do not support it
- jQuery - Simplifies functions and removes cross-browser issues

Discussion

1. CreateJS was chosen because it was listed as one of the most popular

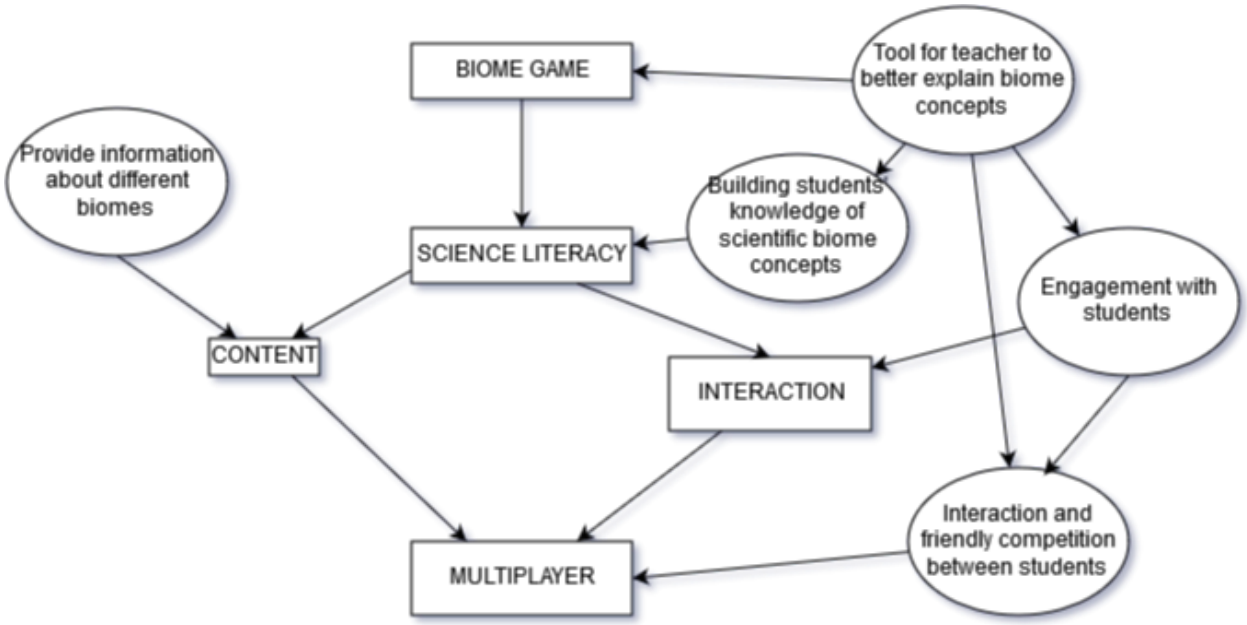
   choices for basic game design libraries. Also, it was well-documented and

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.
3. Please see the folder “artifacts/rationale” in our master branch, which explains our decision more thoroughly.

Design

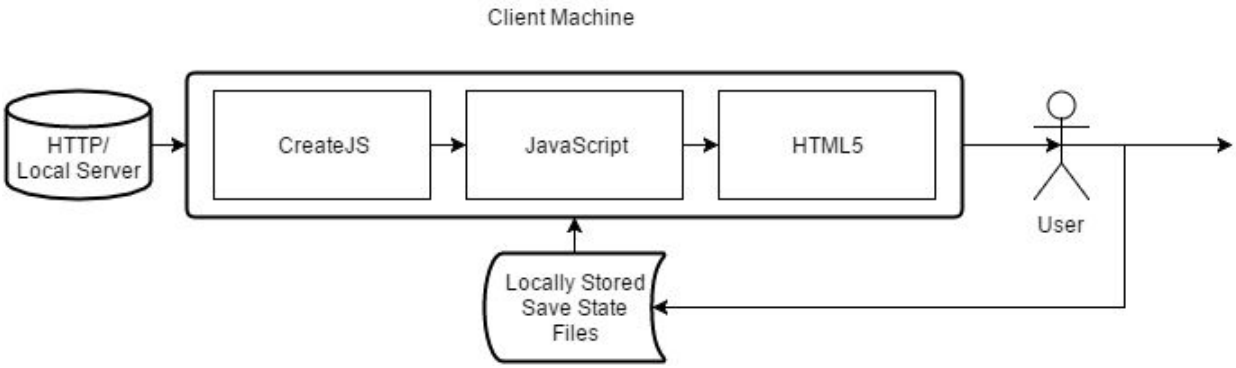
Purpose-Concept Map



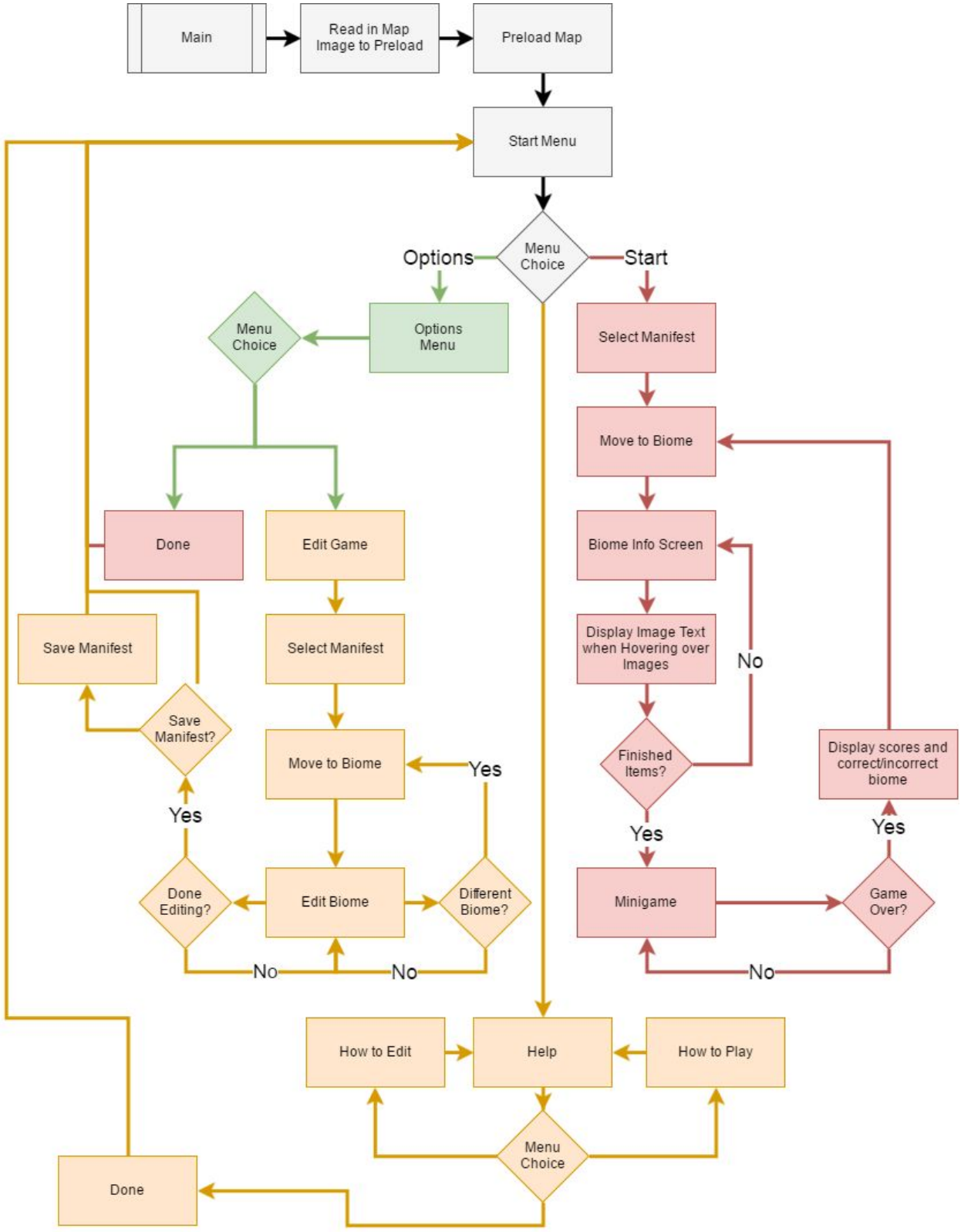
Purpose Concept Map: Rectangles = concepts, Ovals = Purposes

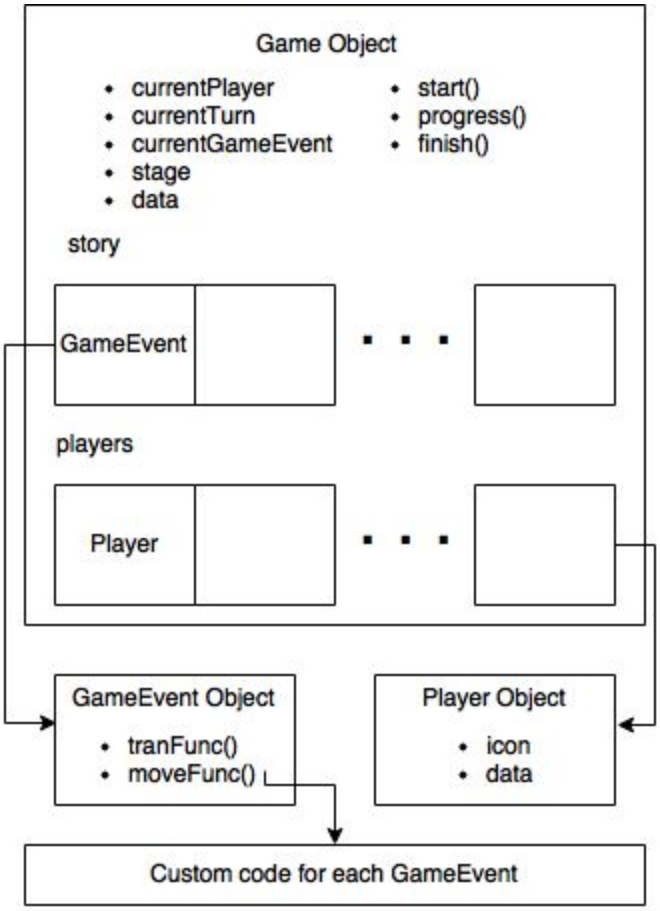
The goal is to build a fully functioning Biome Game. The game is designed for college students studying to teach middle school-aged and younger kids. Some of those college students might also take the game and use it in their classroom. It is intended to promote education through gaming, while providing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


Main
Read in Map Image to Preload
Preload Map
Start Menu
Menu Choice
Options
Start
Options Menu
Menu Choice
Select Manifest
Move to Biome
Done
Edit Game
Biome Info Screen
Save Manifest
Select Manifest
Display Image Text when Hovering over Images
No
Save Manifest?
Move to Biome
Yes
Finished Items?
Display scores and correct/incorrect biome
Yes
Yes
Yes
Done Editing?
Edit Biome
Different Biome?
Minigame
Game Over?
No
No
No
How to Edit
Help
How to Play
Menu Choice
Done



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.

Implementation Notes

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.

```
/**
 * @function this.loadStory
 *  Assigns GameEvent objects their custom code
 */
this.loadStory = function() {
    /** Assign different code to different segments GameEvents */
    story[0] = new GameEvent(eventStartMenu);
    story[1] = new GameEvent(eventMoveAroundEarth);
    story[2] = new GameEvent(eventInfoScreen);
    story[3] = new GameEvent(eventScrollGame);
};
```

Known Issues

- Game can no longer be hosted locally.
- Can upload images to server but there is no way to delete them from the server without direct access
- With the current manifest setup, the manifest has to be shared with everyone and individually selected on each computer for multiple people to play a non-default version of the game.
- No way to flip or rotate images.
- No text to audio or way to add audio.
- No way to add biomes.
- No way to replace images for temperature/precipitation.

Build Instructions

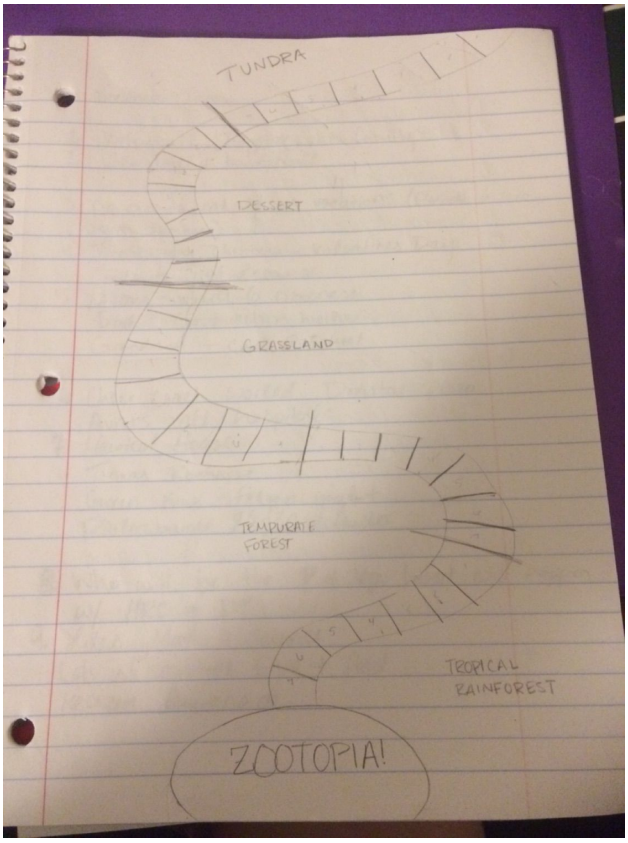
- Download files from github repo at https://github.com/Sagerune/PerlSquad
- Download and install a php client http://php.net/downloads.php (optional, only for full customization)
- Run game from “index.html” in the downloaded file

- Alternatively, you can just play the game with full features at biomegame.online

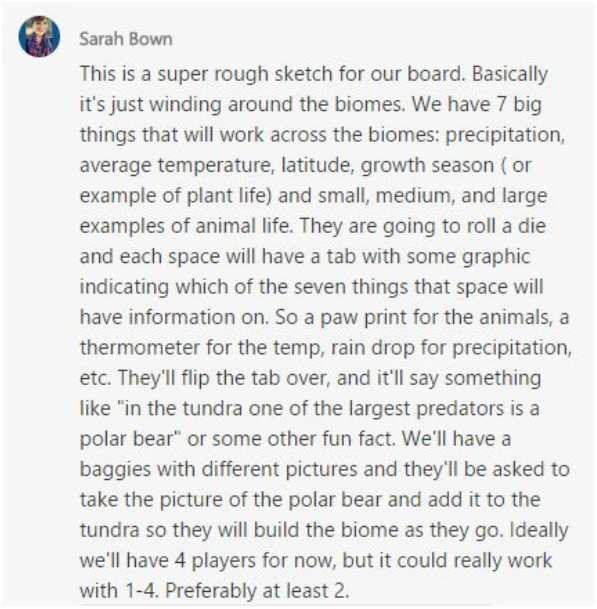
Links

Github: https://github.com/Sagerune/PerlSquad
CreateJS: http://www.createjs.com/
EaselJS: http://www.createjs.com/easeljs
PreloadJS: http://www.createjs.com/preloadjs
Photoshop: http://www.adobe.com/photoshop
Gimp: https://www.gimp.org/
PHP - http://php.net/downloads.php
JavaScript - https://www.javascript.com/
FileSaver.JS - https://github.com/eligrey/FileSaver.js/
Blob.JS - https://github.com/eligrey/Blob.js/
jQuery - 3.1.1 - https://jquery.com/

Appendix



The image above is a rough sketch that was provided by our student clients, demonstrating the appearance and flow of their board game in an attempt to inspire ideas for our web-based game. Directly below is a summary/explanation of the sketch and some of the game rules provided by one of the clients.



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/

Above are some resources provided by Dr.Liu and/or the clients for more information regarding the project and its concepts.

QR Code for website