SOFTWARE DEVELOPMENT Technician Project B - Maze Game

PREPARED FOR

BCS

PREPARED BY

Siphiwe Manda

18 December 2020

Dear Examiner,

Re: Enclosed documentation for the compilation of my apprenticeship

This maze game has covered the following competency of my apprenticeship: design, construction, testing and documentation.

The game itself revolves around a sheep trying to escape 'Olde World Farms'. He has fences in his way that reduce his life, but eating carrots and peppers increase it.

The application was built using javascript implementing HTML canvas API and using node js to serve a static server and is also hosted on Heroku
https://oldeworle-mazegame.herokuapp.com/

Yours Truly,

Siphiwe Manda

1. Project Overview

A maze game designed and implemented for “**Olde World Phunne”** which customers can get access to by visiting a website.

2. Analysis

The first risk involved with the project was the time allowed for development (a week). Within this time was required a working beta version of the maze, design considerations and documentation, handling of unforeseen errors and deployment issues as well as test scenarios or unit testing. It was important that the code covered the most important parts of the rubric and could be easily shown to scale up.

In order to show that the project can be scaled up, the code needed to be very modular and data needed to be easy to read in.

The gameplay would go *User lands on the page - > user presses space to start -> user makes moves → user lands on the end screen.*

4. Design

Game Play

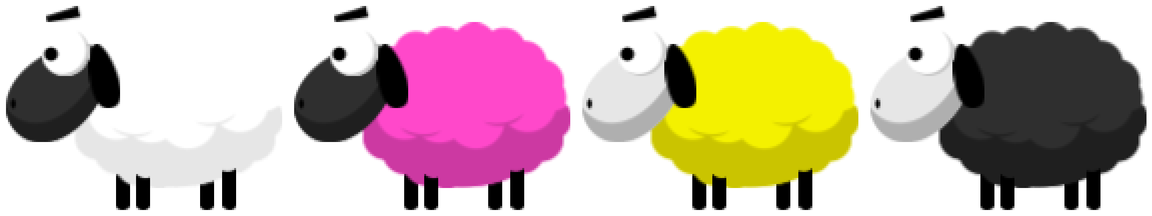
A sheep is attempting to escape a farm. The sheep will have to get from the start of the maze, the top left corner, and find his way to a hay bundle which will transport the sheep and player to the next level.

How to play

- Use the arrows on the keyboard to navigate the sheep around the obstacles
- Walk over the carrots and peppers to gain a life
- Navigate to the hay while trying to keep as many lives as possible
- Lives are indicated at the right-hand corner of the screen

Characters

There are four playable characters which are different colours of sheep, although as this is a beta version the other characters have not yet been implemented. Currently, the only playable character is the pink sheep and he/she will be initialised on starting the game

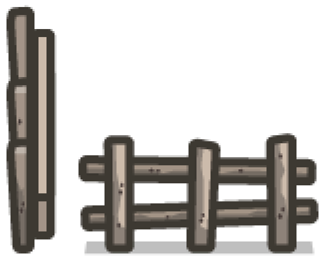


Background

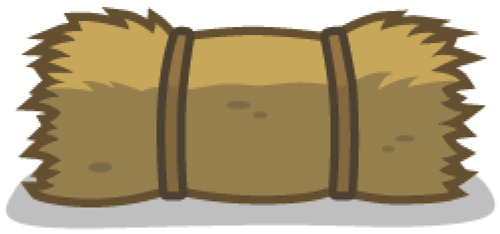
The background is made of simple green grass tiles

Obstacles

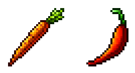
The sheep's obstacles come in the form of fences that he will need to navigate around, bumping into a fence will lose the player one life



The goal of the sheep is to get to the hay barrel that will spawn randomly within the maze



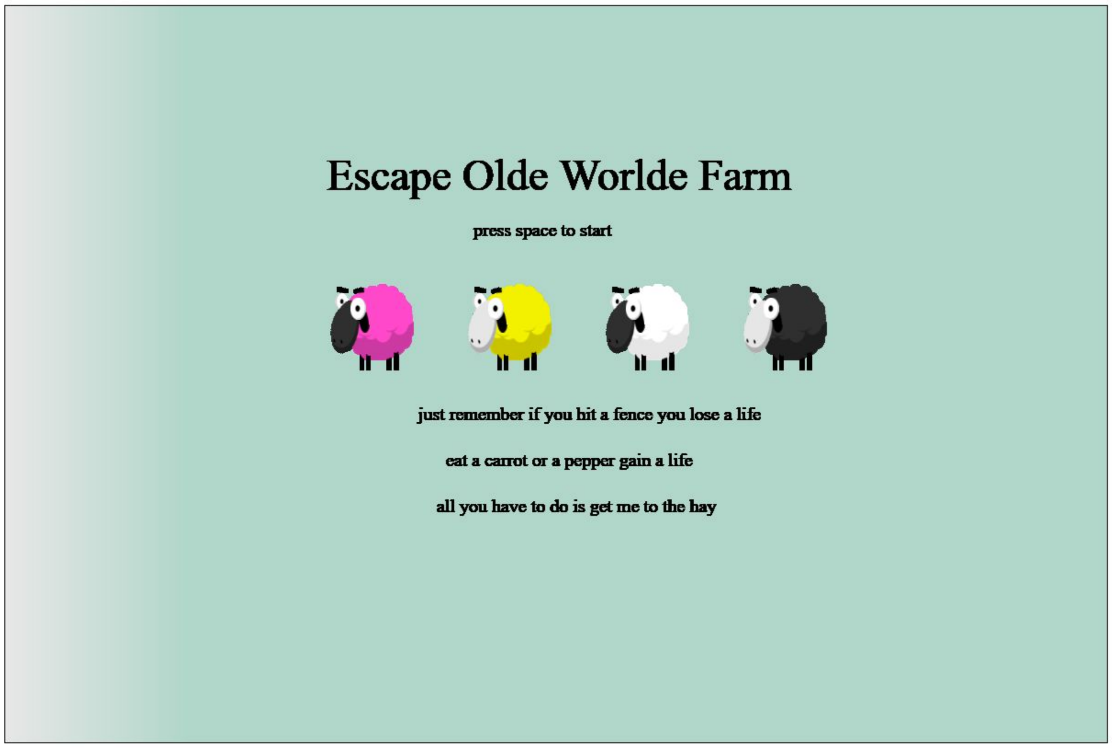
The other two maze attributes are peppers and carrots which will gain the player one life



Game playing area

The game is to be played on a website and the game playing area is a width of 1200pixels and a height of 800px

Start Screen



Escape Olde Worlde Farm

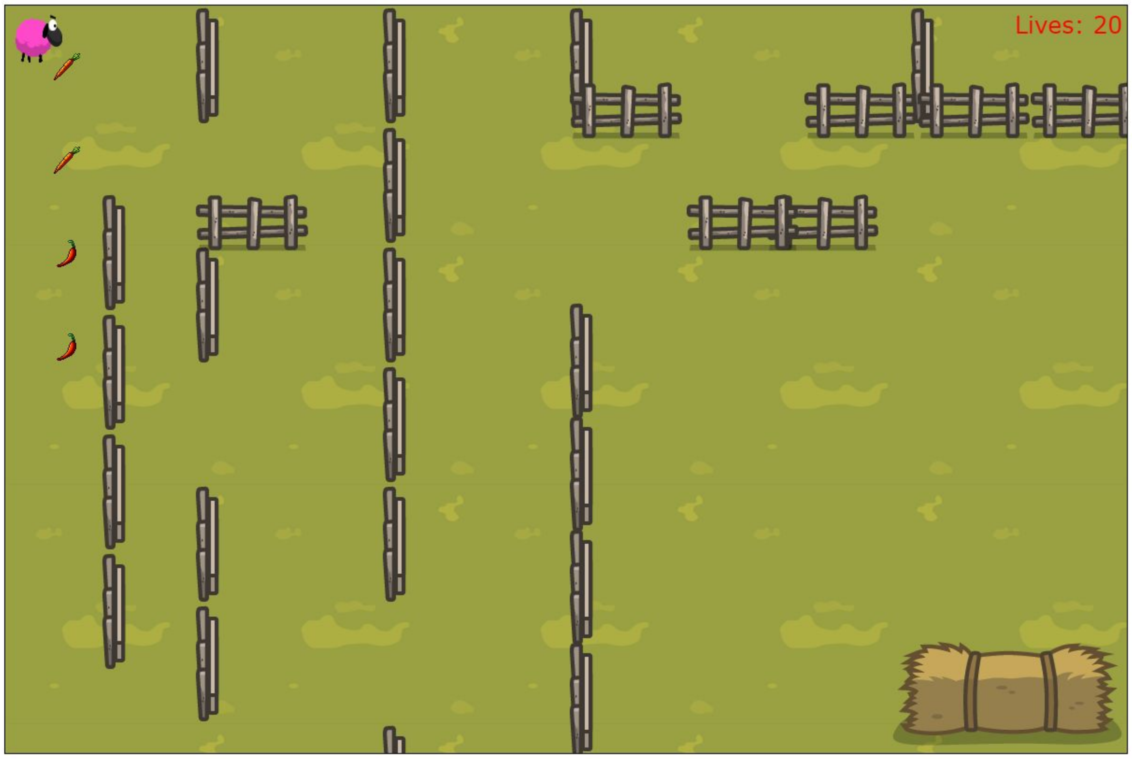
press space to start

just remember if you hit a fence you lose a life

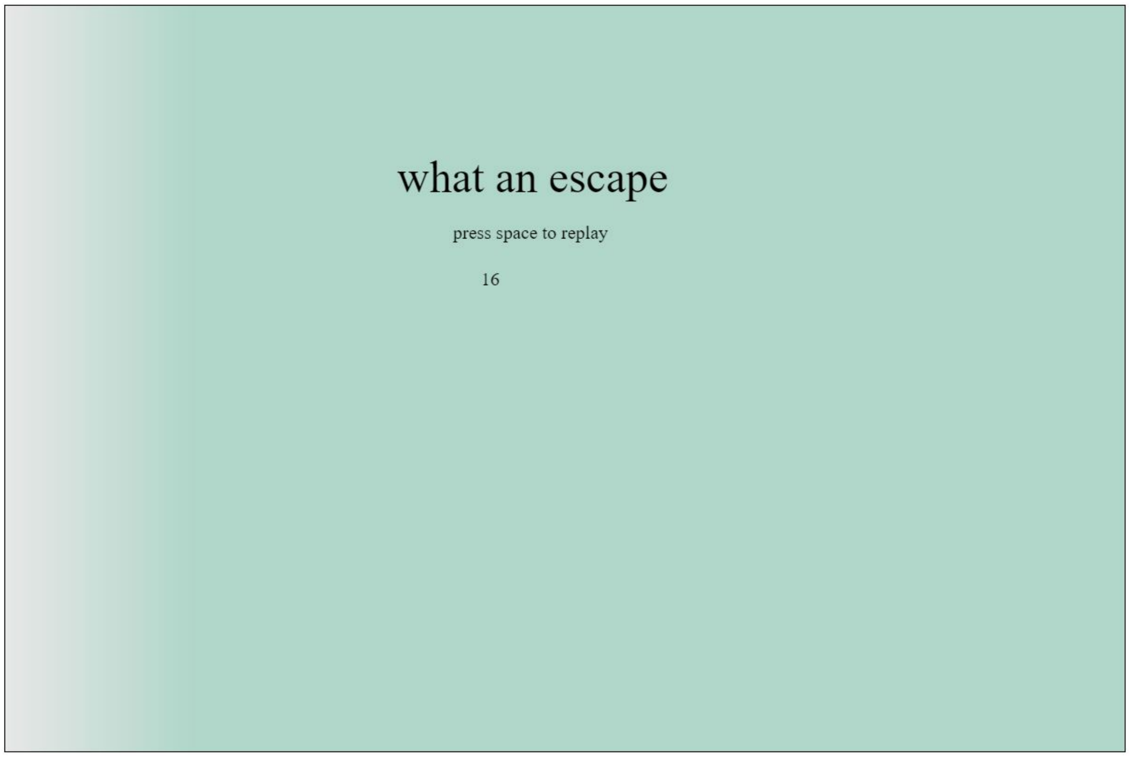
eat a carrot or a pepper gain a life

all you have to do is get me to the hay

Game Screen



End Screen



Data

Game objects are read in from JSON files, this allows the game to be easily changed by adding or removing JSON files and works as an easy way to create and control configuration files. Examples of these can be found in the project code at src/data

Assets

Game assets were acquired from Craftpix.Net for which I bought a licence, further information can be found here https://craftpix.net/file-licenses/

5. Code

For hosting the application locally please read the README.md that can be found within the root directory of the project code.

The game is written in javascript using HTML canvas API and is hosted using express to sever a static website with node js.

The code has been commented to make the functions visible and easy to understand and javascript coding conventions have been followed by referencing (https://google.github.io/styleguide/jsguide.html)

6. Testing

Testing was done using mocha, the test script is located at test/Test.js

Tests can be run in the terminal navigating into the project's root directory and running

npm test

There are currently six unit tests that are checking the functionality of the collision method

This method checks whether a sheep has collided with a fence, using mock JSON data in order to pick where the sheep and fences would be. As part of continuous integration, these tests are also run every time the application is deployed

Test One

```
it('should check that a sheep and a fence did not
collide', function () {
expect(CollisionFencevsSheep(500,400,fence)).to.be.true
});
```

This passes in the x and y coordinates of the sheep and the x and y coordinates of the fence. The method returns true if there was no collision.

Test Two

```
it('should check that a sheep and a fence collided',
function () {
expect(CollisionFencevsSheep(100,200,fence)).to.be.fals
e
});
```

This passes in the x and y coordinates of the sheep and the x and y coordinates of the fence. The method returns false if there was a collision.

Test three

```
it('should check that the sheep was in the bounds of
the game', function () {
expect(CollisionFencevsSheep(400,400,fence)).to.be.true
});
```

This passes in the x and y coordinates of the sheep and the method returns true if the sheep is within the bounds of the game

Test four

```
it('should check the sheep was outside the bounds of
the game', function () {
expect(CollisionFencevsSheep(1300,0,fence)).to.be.false
});
```

This passes in the x and y coordinates of the sheep and the method returns false if the sheep is outside the bounds of the game

Test five

```
it('should check that a sheep collided with a fruit and
return 0', function () {
   assert.isNumber(fruitCollision(200,200,fruit), "0")
});
```

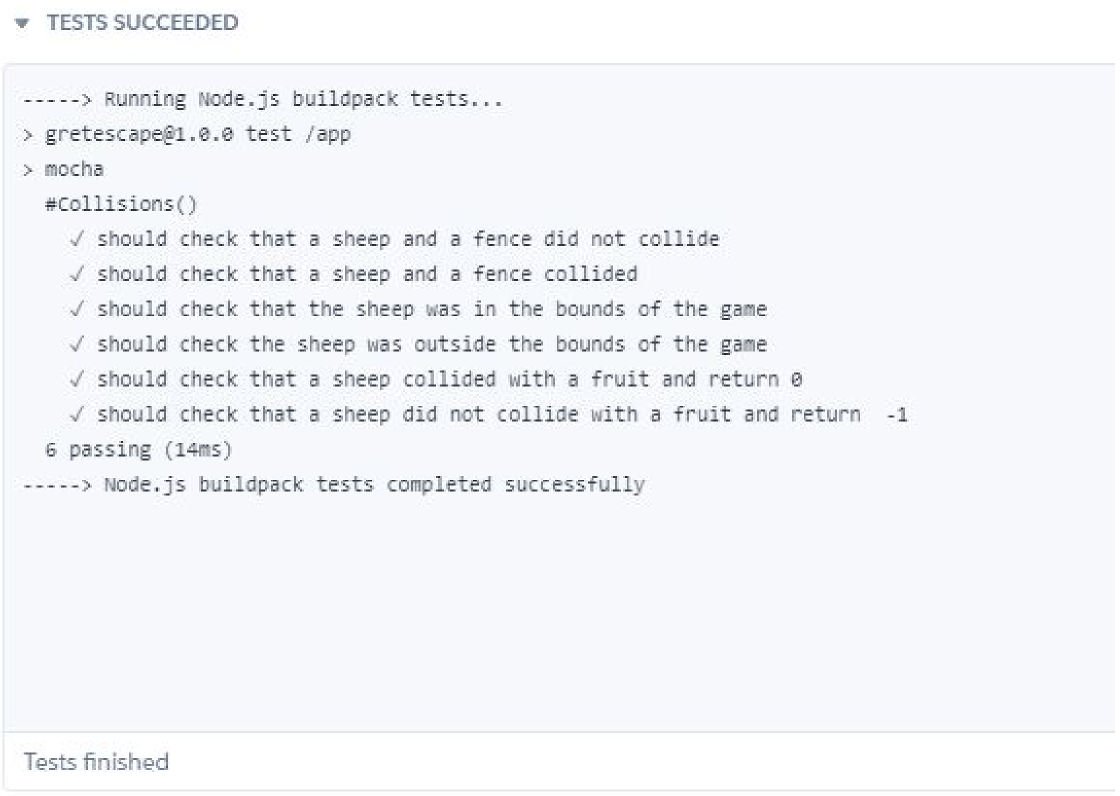
This method returns 0 if the sheep and fruit collide passing the x and y of the sheep and the fruit. The test expects the return value to be zero

Test six

```
it('should check that a sheep did not collide with a
fruit and return  -1', function () {
   assert.isNumber(fruitCollision(600,600,fruit), "-1")
});
```

This method returns -1 if there is no sheep/fruit collision passing the x and y of the sheep and the fruit. The test expects the return value to be -1

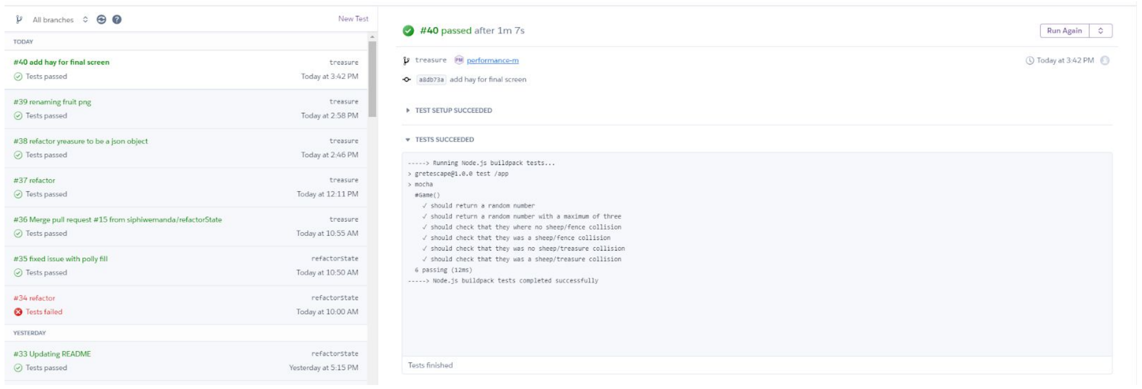
Test results



TESTS SUCCEEDED

```
-----> Running Node.js buildpack tests...
> gretescape@1.0.0 test /app
> mocha
  #Collisions()
    √ should check that a sheep and a fence did not collide
    √ should check that a sheep and a fence collided
    √ should check that the sheep was in the bounds of the game
    √ should check the sheep was outside the bounds of the game
    √ should check that a sheep collided with a fruit and return 0
    √ should check that a sheep did not collide with a fruit and return  -1
  6 passing (14ms)
-----> Node.js buildpack tests completed successfully
```

Tests finished



TODAY

#40 add hay for final screen — treasure
Tests passed — Today at 3:42 PM

#39 renaming fruit png — treasure
Tests passed — Today at 2:58 PM

#38 refactor yreasure to be a json object — treasure
Tests passed — Today at 2:46 PM

#37 refactor — treasure
Tests passed — Today at 12:11 PM

#36 Merge pull request #15 from siphiwemanda/refactorState — treasure
Tests passed — Today at 10:55 AM

#35 fixed issue with polly fill — refactorState
Tests passed — Today at 10:50 AM

#34 refactor — refactorState
Tests failed — Today at 10:00 AM

YESTERDAY

#33 Updating README — refactorState
Tests passed — Yesterday at 5:15 PM

#40 passed after 1m 7s

treasure performance-m — Today at 3:42 PM
a8db73a add hay for final screen

TEST SETUP SUCCEEDED

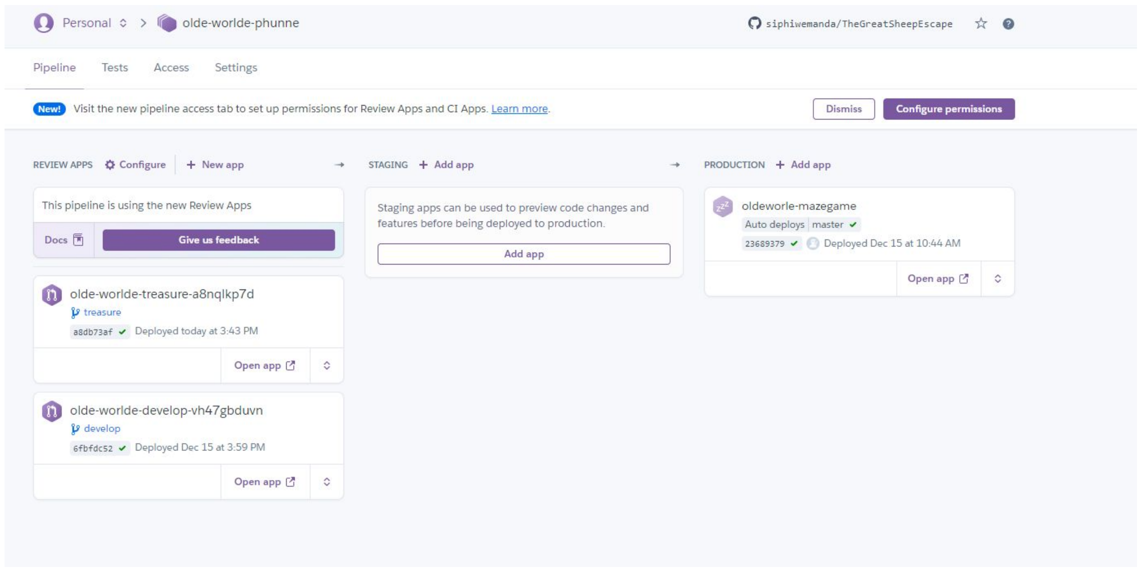
TESTS SUCCEEDED

```
-----> Running Node.js buildpack tests...
> gretescape@1.0.0 test /app
> mocha
  #Game()
    √ should return a random number
    √ should return a random number with a maximum of three
    √ should check that they where no sheep/fence collision
    √ should check that they was a sheep/fence collision
    √ should check that they was no sheep/treasure collision
    √ should check that they was a sheep/treasure collision
  6 passing (12ms)
-----> Node.js buildpack tests completed successfully
```

Tests finished

7. Deployment

The application is deployed to Heroku using a pipeline for the develop branch and the master branch where ‘V1’ version is hosted


Personal > olde-worlde-phunne   siphiwemanda/TheGreatSheepEscape

Pipeline Tests Access Settings

New! Visit the new pipeline access tab to set up permissions for Review Apps and CI Apps. Learn more. Dismiss Configure permissions

REVIEW APPS Configure + New app

This pipeline is using the new Review Apps
Docs Give us feedback

olde-worlde-treasure-a8nqlkp7d
treasure
a8db73af ✓ Deployed today at 3:43 PM
Open app

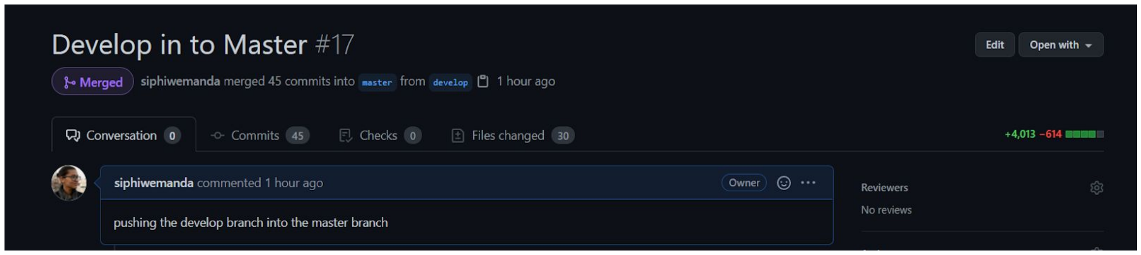
olde-worlde-develop-vh47gbduvn
develop
6fbfdc52 ✓ Deployed Dec 15 at 3:59 PM
Open app

STAGING + Add app

Staging apps can be used to preview code changes and features before being deployed to production.
Add app

PRODUCTION + Add app

oldeworle-mazegame
Auto deploys master ✓
23689379 ✓ Deployed Dec 15 at 10:44 AM
Open app


Develop in to Master #17 Edit Open with

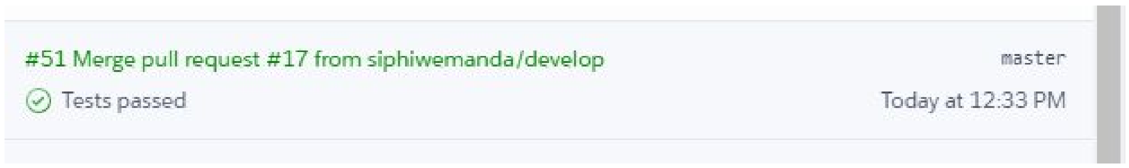
Merged siphiwemanda merged 45 commits into master from develop 1 hour ago

Conversation 0 Commits 45 Checks 0 Files changed 30 +4,013 −614

siphiwemanda commented 1 hour ago Owner

pushing the develop branch into the master branch

Reviewers
No reviews


#51 Merge pull request #17 from siphiwemanda/develop master
✓ Tests passed Today at 12:33 PM

8. Bugs

- Error in the console 'regenerator-runtime' is not defined if the window is refreshed with the development tools window open (seen in the chrome browser). If this happens please close to the development console and restart the application, or close the development console and reload the window.
- The score will continue reducing on the end screen if the player keeps pressing an arrow key
- The 'hay' and the 'fruit' occasionally flash of the screen
- The starting screen is looping over causing the starting text to look blurry, clearRect is not being called
- Collision around the sheep is sometimes larger and before the sheep is visibly in contact with a fence this causes the score to reduce before the user see contact with an obstacle

- Hay does not randomly spawn

9. Missing Functionality and V2

- Adding additional mazes, this should be a simple process as the code is very modular and would simply require adding more JSON files and allowing the state to read in more screens when a sheep has got to the hay (end zone) this would involve adding a method that controls which JSON file is being read in this method could be called by the game class when drawing the playing screen to randomly generate mazes
- The hay is hardcoded in instead of appearing randomly on the screen, the hay needs to randomly spawn anywhere on the map. This could be achieved by adding a function that creates random x and y coordinates and then passing those coordinates to the Draw.Hay() function. These coordinates would also need to be passed to the game class constructor as they would be needed to end the game.
- Animate the direction in which the sheep is walking instead of just facing right, this would mean updating the sheep draw function. Adding a sprite sheet that contains all the walking directions of the sheep. And then changing the index of the cell being drawn depending on what button the user pressed.
- Animate the sheep on the loading screen for better aesthetics
- Add an animate sheep to the end screen depending on the results of the game a ‘dizzy’ sheep if lost or a ‘happy’ sheep if won. This would involve updating the end screens draw function and adding an animation loop.