F21 MA1 - 3D Modelling and Animation

Flex Assignment

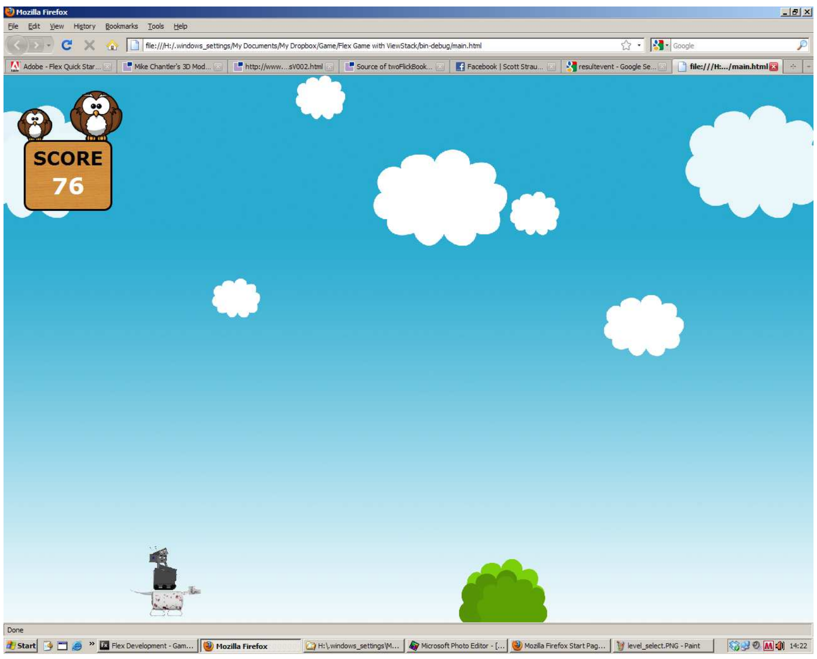
Joint Report

Scott Straughan & Thomas Clulow

Executive Summary

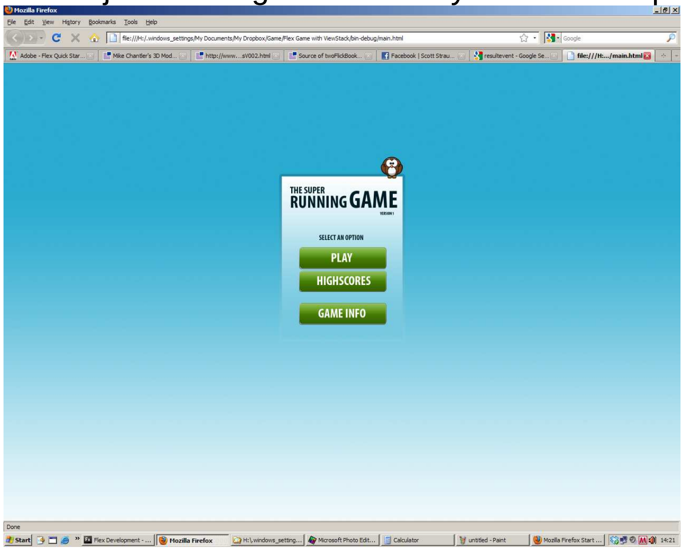
You may be surprised to hear that 'Dog Jumping Over Obstacles Game!' is not actually as terrible as it sounds.

Challenge your reactions, timing and wits in this fast-paced side-scrolling jump action game. You take control of a robot riding on top of a dog and your goal is to stay seated for as long as possible by jumping clear of any obstacles that may be in your path.

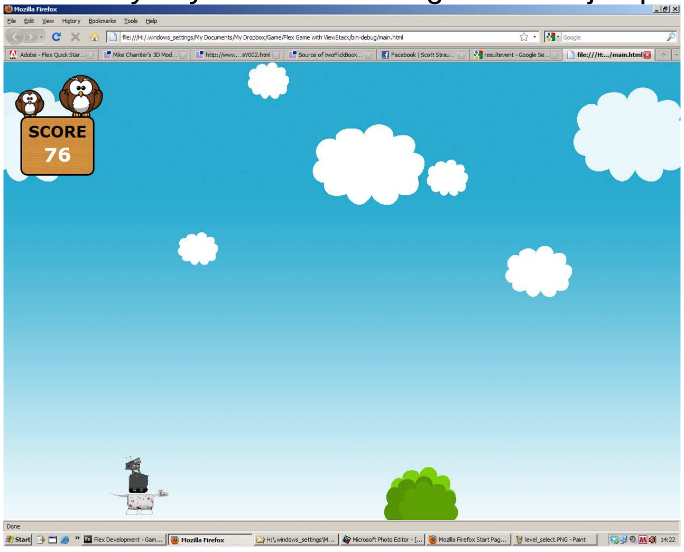


Game Objectives and Play

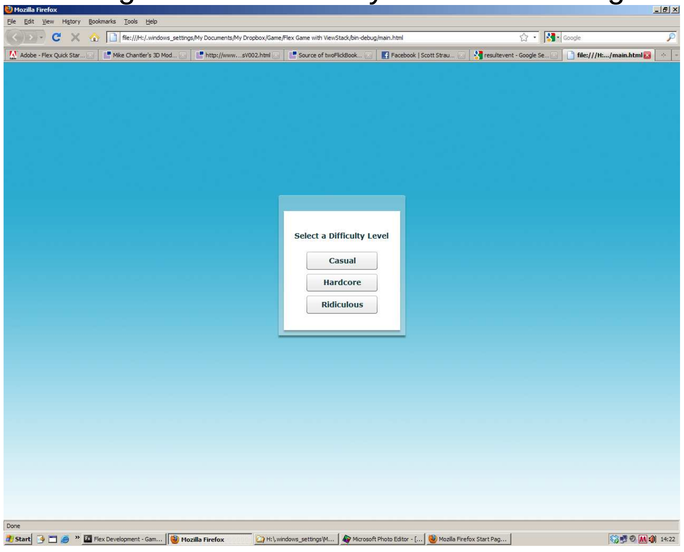
The object of the game is to stay seated on top of your dog for as long as possible.



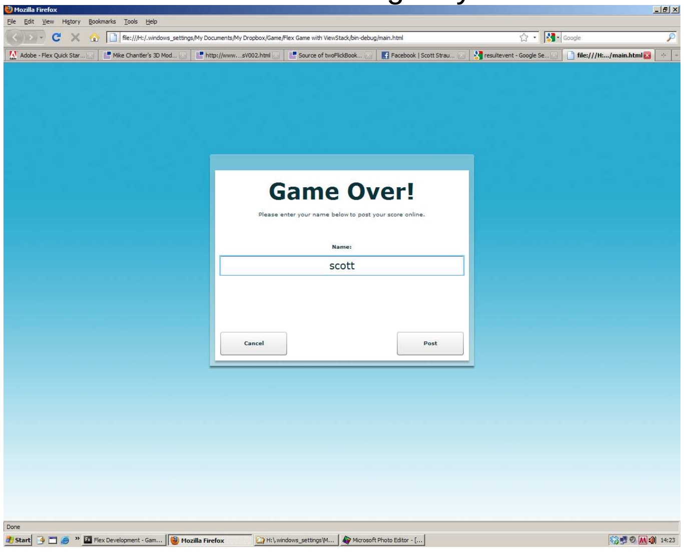
Press any key to make the dog and rider jump over oncoming obstacles.



Selecting different difficulty levels will change the frequency of the obstacles.



The score increases as long as you remain seated on your dog.

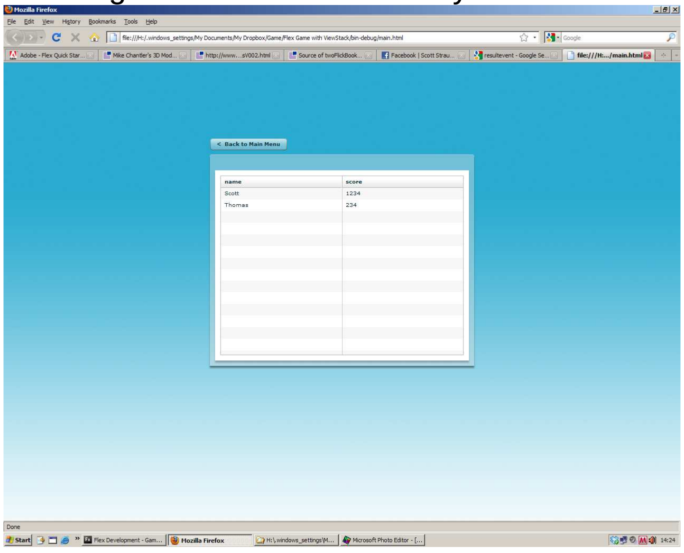

Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///H:/.windows_settings/My Documents/My Dropbox/Game/Flex Game with ViewStack/bin-debug/main.html
Cancel
Post
Done
Start
Flex Development - Gam...
Mozilla Firefox
Microsoft Photo Editor - [...
14:23

Game Over!

Please enter your name below to post your score online.

Name: scott

Colliding with an obstacle ends your current run and your high score is recorded.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///H:/.windows_settings/My Documents/My Dropbox/Game/Flex Game with ViewStack/bin-debug/main.html
< Back to Main Menu
Done
Start
Flex Development - Gam...
Mozilla Firefox
Microsoft Photo Editor - [...
14:24

name	score
Scott	1234
Thomas	234

Game Development

Requirement was to make a simple flash game which utilises second order motion. The game also had to include two animated characters each with two different animated actions. The game also had to include a scoring mechanism which allows the player's high score to by recorded in a database.

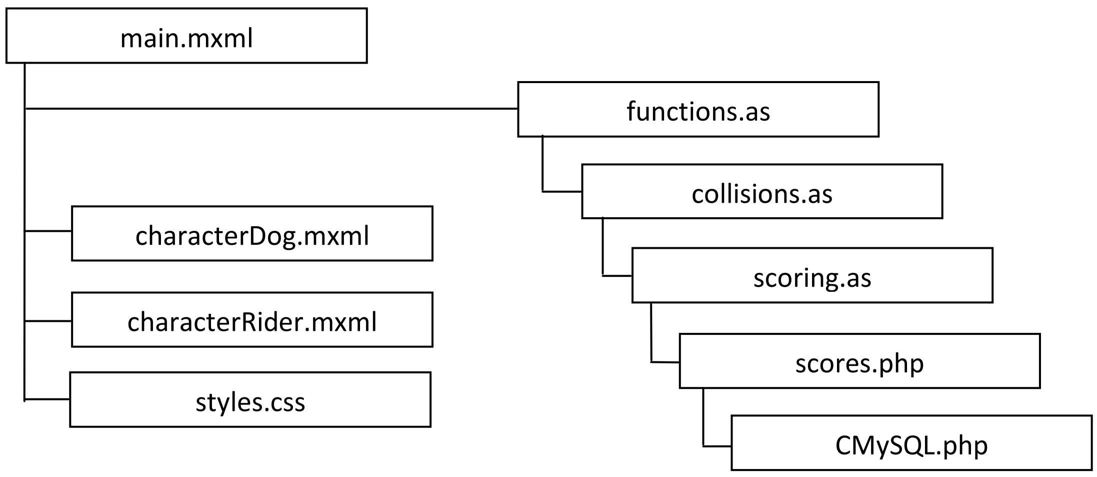
Our initial game idea was to make a 2D side scrolling game in which the player must jump over obstacles located in the level's background. The user would have been able to choose between one of two characters to use in the level and selecting a different difficulty would change the speed at which the background scrolls behind the player.

During the early stages of development, we realised that scrolling the entire game's background required us to make a large amount of artwork. As a result, we decided to use a repeating background for the sky to give the impression of forward movement. For obstacles, we made the game generate a new 'bush' using the same image every time an obstacle was required.

To provide the different difficulty levels we decided to control the frequency of the random generation of obstacles rather than the speed of the entire game. This provided more flexibility for tweaking the difficulty levels based on user testing.

In the final stages of development, it became apparent that the ability to choose which character to play as was a very under used feature. As a result, we decided it would be more interesting and creative to make one of the characters ride on the back of the other character. The animations were tweaked and extra 'bounce' was added to the rider to make it visually obvious that the characters were not physically attached to each other.

Overall Design



main.mxml
This is main layout file for the game. It contains all the components, images and layouts for every interface screen in the game. All interface screens are constructed inside a 'canvas' tag and then stored in a 'ViewStack' element. Switching between interface screens is accomplished by switching the active canvas in the ViewStack using the provided ViewStack function.

characterDog.mxml
This is the mxml file for the Dog character. It contains the animation frames for its running and jumping actions as well its own logic for second order motion. It also contains a 'jump' function to handle the necessary user interaction.

characterRider.mxml
This is the mxml file for the Rider character. Most of the code is the same as the Dog character but the logic for second order motion has been tweaked to make it visually obvious that the two characters are not physically connected. The bounce factor for the Rider is slightly greater than the Dog and this allows the Rider to 'bounce' on the Dog's back after every jump.

functions.as
This is the main functions file for the game. It contains various event handlers and also contains code necessary for the game to function. This code could be contained inside the main.mxml file but was put into a separate file to maintain readability.

collisions.as
Contains functions used to detect when collisions between objects. Also contains functions that execute when a collision is detected.

scoring.as
Contains functions used to check the player’s score as well as functions used to store the high score in a database.

styles.css
Contains style information for the game’s background.

scores.php, CMySQL.php
Contains php functions to post scores to a MySQL database.

Joint Conclusions

Overall, the game satisfies all of the requirements set out in the project specification and the finished game is very similar to the original design concept. Very few compromises were made.

One of the better aspects of the game is the artwork and the look and feel of the menu screens. Together these give the game a very polished feel.

One of the aspects of the game that could have been improved upon is the jumping mechanic. Currently, pressing the jump key simply causes your character to jump a set distance into the air. Ideally, the height of the jump would be directly proportional to the amount of time the jump key is held down. This would allow for small ‘hops’ to be made over single obstacles and longer ‘leaps’ to be made over multiple obstacles.

One final improvement to the game that could be made would be the inclusion of animations that execute when a collision is detected. Currently, when a collision is detected, the game immediately ends and the high score screen is displayed. Ideally, the characters would tumble around the screen for a short period before the game ends.

Individual Report

Scott Straughan

Summary of Role

In general, my role was to create the "background" code that handles the interactions between the player and the 2D world. The following sub roles were assumed:

- **Event Handling**
  In order for our game to work, we needed a way of checking when specific events occurred in the game. These events included when a player collided with an obstacle or such things as simple redrawing the frames to the screens.
- **Graphics**
  Another role was to create the graphics for the game.
- **2D World Dynamics**
  Our game featured a scrolling background and also the generation of obstacles. I coded this into the game using previously acquired knowledge of background tiling and timers.
- **Scoring System**
  Also, I integrated the scoring system to allow a player to post as well as review previously created scores.
- **My Character**
  I also created my character including the frames used to animate it and the functions used to handle it.

Modules

- **collisons.as**
  this file contained the functionality to check if collisions occurred as well as to take action when a collision did occur.
- **scoring.as**
  this file contained the functionality to handle the updating of scores to the internet as well as requesting them.
- **functions.as**
  this file contained some functions written by me, mainly for animating the background and the generation of obstacles.
- **scores.php, CMySQL.php**
  this contained functions which allowed the score to be stored in a MySQL database.
- **characterRider.mxml**
  the function of this module was to represent my character in the game. The characters main roles included jumping and riding.

What I Gained from this Module

I gained a lot of experience in how to layout a game allowing two developers to work in tandem. Originally, I decided a basic layout for the game but it was later change by my team member after discovering fatal flaws in its structure.

I have also developed an understanding of how difficult it is to perform collision detection. Our game features a very simple “box” collision detection system but I could see it being incredibly difficult to detect collisions on objects that are of different shapes accurately.

Individual Report

Thomas Clulow

Summary of Role

In general, my role was to create the “front end” code, the overall structure of the program and the scoring system. The following sub roles were assumed:

- **Interface Layout**
  Create different screens for tasks such as selecting a difficulty level and recording a high score. All interface elements were laid out in a single mxml file.
- **Scoring System**
  Also, I integrated the scoring system to allow a player to post as well as review previously created scores.
- **My Character**
  I also created a 3D model of my character, animated two different actions for it and included all necessary code for second order motion in the character’s mxml component.

Modules

- **main.mxml**
  this file contains all the interface layout code for every screen in the game’s interface. All screens are contained in a ViewStack element.
- **functions.as**
  I wrote some basic button press functions in here to allow navigation around the game’s user interface.
- **scoring.as**
  This file contains functions to record a player’s score, upload it to the database as well as read existing scores from the database.
- **characterDog.mxml**
  This mxml component contains all code relating to the control and display of my character in the game.

What I Gained from this Module

This module provided useful experience on what it is like to program as a team. It taught several useful skills relating to time and resource management as well as numerous technical skills relating to version control.

Mainly, this coursework task has shown me how difficult it can be to manage one codebase when it is constantly being updated by multiple programmers, sometimes simultaneously.

It has also taught me a lot about the mxml format and how user interfaces are put together using Flex.