COMP47390 – Assignment 5 – Breakout

Report

Carl McCann 12508463

After setting up the storyboard similar to how it is done in the specification, The first thing I coded was the settings TVC, as it seemed the least technical part, making it an easy introduction to the project, and also the game depends upon it to edit the number of balls and bricks and the paddle width.

SettingsTVC

After setting the generic table view controller connected to the second tab in the tab bar controller, to a custom SettingsTVC class, I implemented segmented controls and a slider within cells, and connected them to outlets and actions in this class. I also created a blank cell at the top of the view as without it the “Balls” header was placed in the status bar. Within these actions I updated NSUserDefaults to hold information regarding the game such as number of balls and bricks, and the slider width.

BreakoutViewController

I then set up the BreakoutViewController class which I used to control the game view in the first tab connected to the tab bar controller. In the storyboard I set up a score label and a label for the corresponding score, however I pass the second label to the BreakoutBehaviour class for it to handle. I created instances of BreakoutBehaviour and UIDynamicAnimator here too. I also set up gesture recognisers from the storyboard, and linked them to actions in this class. The tap gesture recogniser calls a function from BreakoutBehaviour called moveBalls() which is self explanatory and the pan gesture recogniser redraws the paddle view by taking the absolute value of your fingers position – half the paddle's width to centre the paddle with your finger. The paddle's boundary is also updated here using functions from BreakoutBehaviour. In viewDidLoad() some default settings are stored into NSUserDefaults, BreakoutBehaviour is added to the animator, and we reference BreakoutBehaviour's delegate which is used to send game over messages. I tried to resize the subviews upon rotation of the device, however my attempts failed in didRotateFromInterfaceOrientation(). viewWillAppear() calls a function called resetGame() which resets the score and updates its label, removes all boundaries and balls from the collider in BreakoutBehaviour. It also removes the balls,bricks and paddle from the view too. These are all readded according to the current values from settings via NSUserDefaults. Finally it passes a BrickView array, the number of balls and the score label to the behaviour using the function getBricksBallsAndScore(). Finally the function sendAlert alerts the user to a win or a loss via a message sent from the behaviour via its delegate.

BreakoutView

This class implements a UIView, that adds subviews for the paddle, balls, and bricks.

PaddleView, BallView, and BrickView

These classes represent simple UIViews that display CGRect's for the paddle, balls, and bricks. Each have their own colour and shape, such as the ball which has a cornerRadius attribute set to 2 to make it circular. They are added to BreakoutView.

BreakoutBehaviour

Firstly, there is a protocol setup in this file that allows the delegate to pass messages to BreakoutViewController. This class contains the collider, which the balls exist within, with the paddle and bricks being added as boundaries alongside the boundaries of the view. BallBehaviour represents how the ball will react to its environment such as rotation, how it bounces and how it will move. The function moveBalls generates random x,y values (always upwards) which are added to as linear velocities to the balls in the ballBehaviour, which are also in the collider. The function addBalls() takes an aray of BallViews from the BreakoutView and it adds them to ballBehaviour and collider. addBoundary() can take either a bezier path or a line, to create boundaries with ids. removeAllBoundaries() does exactly what it says on the tin and is used to clear all boundaries in preparation for the next game. Similarly, removeAllBalls() does this but for the balls. The function getBricksBallsAndScore() takes an array of BrickViews the number of balls and the scoreLabel from BreakoutView. Using these in the function collisionBehaviour() it can remove bricks from the super view which is animated with a simple fade, update the score label, and check when all the bricks are gone meaning a win. Similarly it can see when all the balls are gone due to collisions with the bottom border of the screen, meaning a loss. When it detects these events it uses the delegate to send an alert message to BreakoutViewController. The init() function simply adds the ballBehaviour and the collider to BreakoutBehaviour.

One thing I wasn't happy with was the fact that upon screen rotation the game doesn't resize appropriately, however a workaround exists where you can switch to the settings tab and back to the game tab, calling viewWillAppear(). This resets the game properly within the current screens boundaries, but this is not ideal. Also in the reset horizontal mode, the max number of bricks I set is very close to the paddle. I was unsure whether I should move the paddle lower, or lessen the max number of bricks, so I left it as it was.