COURSE: COMP 4478

ASSIGNMENT 1 – A BASIC 2P SQUASH GAME

NAME: DHAVAL THANKI

STUDENT NUMBER: 0871347

Objective

In this assignment, we are required to program a squash game within HaxeFlixel. The goal is to understand collision between sprites, limiting the movement of certain sprites, as well as begin including sounds and additional effects to enhance gameplay. Here is the link to the PlayState.hx code for easy access from this document.

All the variables

```
14  class PlayState extends FlxState
15  {
16      /* All moving sprites */
17      var playerTwo:FlxSprite;
18      var playerOne:FlxSprite;
19      var ball:FlxSprite;
20
21      /* Secondary sprites */
22      var barriers:FlxGroup;
23      var lPanel:FlxSprite;
24      var rPanel:FlxSprite;
25      var tPanel:FlxSprite;
26      var bPanel:FlxSprite;
27
28      /* All in game sounds */
29      var collisionSound:FlxSound;
30      var endGameSound:FlxSound;
31      var wallBounceSound:FlxSound;
32      var winnerSound:FlxSound;
```

```
34      /* All in-game text */
35      var _txtTitle:FlxText;
36      var _txtMessage:FlxText;
37
38      var scores:Array<Int> = [];
39
40      // Get the score value
41      var player1Score:Float;
42      var player2Score:Float;
43
44      // Create and add the text to the screen
45      var player1ScoreText:FlxText;
46      var player2ScoreText:FlxText;
47
48      var _player1ScoreText:FlxText;
49      var _player2ScoreText:FlxText;
```

Here is a screenshot to show all the variables used, mainly comprising of FlxSprites for the paddles, ball, and the box that imitates the squash court. We can also see the variables used to play sounds characterised by FlxSound. An array that logs the scores as there are collisions between either P1 ←→ ball for P1score and P2 ←→ ball for P2score.

```
6   import flixel.addons.display.FlxBackdrop;
7   import flixel.addons.effects.FlxTrail;
```

Making sure I make the necessary imports; I used the trailing effect for the ball and the backdrop for including a static background.

Loading in the sound and background elements

```
51      override public function create():Void
52      {
53          /* Adding a static background image */
54          FlxG.camera.bgColor = 0xFF666666; // standard backdrop
55          var bg:FlxSprite = new FlxSprite(0, 0);
56          bg.loadGraphic(AssetPaths.bg__png);
57          var bg:FlxBackdrop = new FlxBackdrop(AssetPaths.bg__png, 2, 0, true, false);
58          // add elements (static background) onto screen
59          add(bg);
60
61          /* loading soundfiles in */
62          collisionSound = FlxG.sound.load(AssetPaths.bpp__wav);
63          endGameSound = FlxG.sound.load(AssetPaths.tlr__wav);
64          wallBounceSound = FlxG.sound.load(AssetPaths.brrp__wav);
65          winnerSound = FlxG.sound.load(AssetPaths.zzp__wav);
```

Using the same method as previously employed in Exercise 1, I re-use the same code to load a static background sourced from google images, we also see the .wav sound clips, made using sfxr, are loaded in to be used when there are collisions, win/loss events, etc.

Sprite Creation

```
67          /* creation of sprite paddle/racket one */
68          playerOne = new FlxSprite(350, 450);
69          playerOne.makeGraphic(60, 5, FlxColor.LIME);
70          playerOne.immovable = true;
71          /* creation of sprite paddle/racket two */
72          playerTwo = new FlxSprite(150, 450);
73          playerTwo.makeGraphic(60, 5, FlxColor.CYAN);
74          playerTwo.immovable = true;
75          // add elements (player paddles) onto screen
76          add(playerOne);
77          add(playerTwo);
78
79          /* creation of sprite ball */
80          ball = new FlxSprite(200, 180);
81          ball.makeGraphic(10, 5, FlxColor.ORANGE);
82          /* setting behaviour attributes for sprite ball */
83          ball.elasticity = 1;
84          ball.maxVelocity.set(222, 222);
85          ball.velocity.y = 500;
86          ball.velocity.x = 500;
87          // Create trail
88          var trail:FlxTrail = new FlxTrail(ball);
89          // add elements (ball w/trail) onto screen
90          add(trail);
91          add(ball);
```

In the screenshot presented here, I create a sprite named playerOne, this is the paddle for P1, notice how the X coordinates are different for P1 and P2, this is to make sure the two paddles don't overlap on first start of the game, same idea for P2 in terms of creating the paddle.

After creating these sprites, I modify them to set some attributes, these include colour and size.

Now we move onto creating the sprite named ball, this will have more attributes set form here rather than in the update function. Set the size, the colour, elasticity and then add the velocity on the x and y directions. Lastly, add in a trail for the ball for extra effects during gameplay.

Add in the barriers to control collisions with the ball

```
96          /* creation and grouping of all barriers, i.e. simulating a squash court */
97          barriers = new FlxGroup();
98          lPanel = new FlxSprite(0, 0); // left side panel
99          lPanel.makeGraphic(5, 540, FlxColor.GRAY);
100         lPanel.immovable = true;
101         barriers.add(lPanel);
102         rPanel = new FlxSprite(550, 0); // right side panel
103         rPanel.makeGraphic(5, 550, FlxColor.GRAY);
104         rPanel.immovable = true;
105         barriers.add(rPanel);
106         tPanel = new FlxSprite(0, 0); // top side panel
107         tPanel.makeGraphic(550, 10, FlxColor.GRAY);
108         tPanel.immovable = true;
109         barriers.add(tPanel);
110         bPanel = new FlxSprite(0, 465); // bottom side panel
111         bPanel.makeGraphic(550, 10, FlxColor.TRANSPARENT);
112         bPanel.immovable = true;
113         barriers.add(bPanel);
114         // add elements (barriers) onto screen
115         add(barriers);
```

Fairly straightforward in terms of creating sprites as barriers by setting the (X, Y) coordinates and making sure that they are immovable, meaning the object cannot be moved with a collision, I group them together to make it easier to use them later on.

The scoresheet

```
117         // If the value was not initialised, default to zero
118         if (!(scores[0] > 0))
119             scores[0] = 0;
120         if (!(scores[1] > 0))
121             scores[1] = 0;
122
123         player1Score = scores[0];
124         player2Score = scores[1];
125
126         player1ScoreText = new FlxText(565, 25, 0, "P1 Score: ", 9);
127         add(player1ScoreText);
128
129         player2ScoreText = new FlxText(564, 225, 0, "P2 Score: ", 9);
130         add(player2ScoreText);
```

The usual FlxText to show these values on the game screen. The array is initialised to values of 0 just in case there were some garbage values that carried forward, each player is allowed one index position value and these are them incremented.

Game functionality

```
134         /* Make sure both players are stationary */
135         playerOne.velocity.x = 0;
136         playerTwo.velocity.x = 0;
137
138         /* Movement controls for P1 */
139         if (FlxG.keys.pressed.LEFT && playerOne.x > 10)
140         {
141             playerOne.velocity.x = -500;
142         }
143         else if (FlxG.keys.pressed.RIGHT && playerOne.x < 500)
144         {
145             playerOne.velocity.x = 480;
146         }
147         /* Movement controls for P2 */
148         if (FlxG.keys.pressed.A && playerTwo.x > 10)
149         {
150             playerTwo.velocity.x = -500;
151         }
152         else if (FlxG.keys.pressed.D && playerTwo.x < 500)
153         {
154             playerTwo.velocity.x = 480;
155         }
156         /* Hard reset button */
157         if (FlxG.keys.justReleased.R)
158         {
159             FlxG.resetState();
160         }
```

In this screenshot I set the controls for both players one and two, they're done simply by changing the velocity of the FlxSprite objects through an if statement based of the x position of the paddle. The letter '**R**' triggers a resetState() that will kick the game into a fresh run.

Implementation for collision physics (player and ball)

```
161         /* Add to score, play collision sound and show winner on screen text, all for P1 */
162         if (FlxG.collide(playerOne, ball))
163         {
164             collisionSound.play();
165             scores[0] += 1;
166             _player1ScoreText = new FlxText(615, 25, 0, "" + scores[0], 9);
167             add(_player1ScoreText);
168             if (scores[0] == 10)
169             {
170                 _txtMessage = new FlxText(200, 85, 0, "Congratulations P1!
171                                                       You Won!", 14);
172                 add(_txtMessage);
173                 winnerSound.play();
174                 FlxG.camera.fade(FlxColor.BLUE, .15, true);
175                 ball.velocity.y = 0;
176             }
177         }
178         /* Add to score, play collision sound and show winner on screen text, all for P2 */
179         if (FlxG.collide(playerTwo, ball))
180         {
181             collisionSound.play();
182             scores[1] += 1;
183 
184             _player2ScoreText = new FlxText(618, 225, 0, "" + scores[1], 9);
185             add(_player2ScoreText);
186             if (scores[1] == 10)
187             {
188                 _txtMessage = new FlxText(200, 85, 0, "Congratulations P2!
189                                                       You Won!", 14);
190                 add(_txtMessage);
191                 winnerSound.play();
192                 FlxG.camera.fade(FlxColor.BLUE, .15, true);
193                 ball.velocity.y = 0;
194             }
195         }
```

Here we see the implementation for the collision between both players and the ball, individually. The collision will play a sound, add a value to the score Array as well as display a message in the event that one of the players reaches a score of 10. As an extra bit of graphics, I added in colour coded variations to mean either a loss or a win, Blue for win, Red for loss.

Implementation for collision physics (barriers and ball)

```
196             /* Make sure sound will play upon each wall collision */
197             if (FlxG.collide(ball, tPanel))
198             {
199                 wallBounceSound.play();
200             }
201             if (FlxG.collide(ball, lPanel))
202             {
203                 wallBounceSound.play();
204             }
205             if (FlxG.collide(ball, rPanel))
206             {
207                 wallBounceSound.play();
208             }
209             /* End the game if ball was to hit bottom panel */
210             if (FlxG.collide(ball, bPanel))
211             {
212                 endGameSound.play();
213                 FlxG.camera.fade(FlxColor.RED, .15, true);
214                 _txtTitle = new FlxText(200, 85, 0, "     Whoops!
215                                                     Hit 'R' to reset!", 22);
216                 _txtTitle.screenCenter(FlxAxes.X);
217                 add(_txtTitle);
218                 ball.velocity.y = 0;
219             }
220         }
221     }
222
```

In this screenshot we see the last needed collision, collision between the bottom panel and the ball, this signifies that the game is lost and the only way to try again is to reset it, I could have implemented a switchState() here, I plan on including this for a much more polished game for the first project submission.

Testing the game