```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using System.Threading.Tasks;
5  using GameDemo.Shared.Menu;
6  using Microsoft.Xna.Framework;
7  using Microsoft.Xna.Framework.Graphics;
8  using Microsoft.Xna.Framework.Input;
9  using Microsoft.Xna.Framework.Input.Touch;
10 using SQLite;
11
12 namespace GameDemo.Shared
13 {
14     class SpriteManager : Microsoft.Xna.Framework.DrawableGameComponent
15     {
16
17         SpriteBatch spriteBatch;
18         Player player;
19         double score;
20         List<BackgroundSprite> spriteList = new List<BackgroundSprite>();
21         List<Enemy> enemies = new List<Enemy>();
22
23         public const float groundLevel= 800;
24         public const float airLevel = 560;
25
26         public TimeSpan previousSpawnTime;
27         public TimeSpan enemySpawnTime;
28         float timeonscreenMS;
29         Random random;
30
31         float rotationAngle = 0;
32
33         Enemy e;
34         int current_enemy;
35
36         public static GameState gameState;
37         MainMenu menu;
38         SpriteFont sf;
39         string name="";
40
41
42         public SpriteManager(Game game) : base(game)
43         {
44
45             random = new Random();
46             gameState = GameState.mainMenu;
47             menu = new MainMenu();
48
49         }
50
51         protected override void LoadContent()
52         {
53             spriteBatch = new SpriteBatch(Game.GraphicsDevice);
54
55             // Scales and offsets of the player and enemy animation boundaries
56             float playerscale = 2F;
```

```
57              float enemyscale = 3;
58              Vector2 offset_player = new Vector2(100, 70);
59
60              sf = Game.Content.Load<SpriteFont>("font");
61
62              // Load the buttons
63              menu.LoadContent(Game.Content, new Size(1080, 1920));
64
65              // Load the player,background and enemies
66              player = new Player(Game.Content.Load<Texture2D>("horse_running"),
                    new Vector2(130,groundLevel-99), new Point(184, 116), 11,
                  offset_player, new Point(1, 1), new Point(3, 4),new Vector2
                  (0,0),50,playerscale);
67              player.LoadContent(Game.Content);
68              spriteList.Add((new BackgroundSprite(Game.Content.Load<Texture2D>
                  ("parallax-mountain-bg"), new Vector2(0, 0), new Vector2
                  (Game.Window.ClientBounds.Height,
                  Game.Window.ClientBounds.Width), 0, Color.White, false)));
69              spriteList.Add((new BackgroundSprite(Game.Content.Load<Texture2D>
                  ("montain-far"), new Vector2(0,-200), new Vector2
                  (Game.Window.ClientBounds.Height,
                  Game.Window.ClientBounds.Width), 0,Color.Green,false)));
70              spriteList.Add((new BackgroundSprite(Game.Content.Load<Texture2D>
                  ("mountains"), Vector2.Zero, new Vector2
                  (Game.Window.ClientBounds.Height,
                  Game.Window.ClientBounds.Width), 0, Color.Yellow,false)));
71              spriteList.Add((new BackgroundSprite(Game.Content.Load<Texture2D>
                  ("foreground-trees"),new Vector2(0,groundLevel-300), new Vector2
                  (1920, 300), -1, Color.ForestGreen,true)));
72              spriteList.Add(new BackgroundSprite(Game.Content.Load<Texture2D>
                  ("mountainGround"), new Vector2(0, groundLevel), new Vector2
                  (200, 200), -10, Color.Gray,true));
73              spriteList.Add(new BackgroundSprite(Game.Content.Load<Texture2D>
                  ("mountainTile"), new Vector2(0, groundLevel+200), new Vector2
                  (200, 180), -10, Color.Gray,true));
74              enemies.Add(new Enemy(ENEMY_TYPES.AXE,Game.Content.Load<Texture2D>
                  ("battleaxe-sheet"), new Vector2(2000, airLevel+50), new Point
                  (16, 16), 8, Vector2.Zero, new Point(1, 1), new Point(8, 1), new
                   Vector2(5, 0), 100, enemyscale+0.5F));
75              enemies.Add(new Enemy
                  (ENEMY_TYPES.SKELETON,Game.Content.Load<Texture2D>("useful
                  skele"),new Vector2(2000,groundLevel-32*enemyscale), new Point
                  (32, 32), 5,new Vector2(10,0), new Point(1, 1), new Point(5, 1),
                   new Vector2(5,0),100,enemyscale));
76              enemies.Add(new Enemy(ENEMY_TYPES.BOULDER,
                  Game.Content.Load<Texture2D>("boulder2"), new Vector2(2000,
                  groundLevel-140), new Point(76, 73), 1, Vector2.Zero, new Point
                  (1, 1), new Point(1, 1), new Vector2(8, 0), 10, 1.5f));
77              enemies.Add(new Enemy(ENEMY_TYPES.GHOST,
                  Game.Content.Load<Texture2D>("ghost"), new Vector2(2000,
                  airLevel), new Point(25, 35), 10, Vector2.Zero, new Point(1, 1),
                   new Point(10, 1), new Vector2(5, 0), 100, enemyscale));
78              base.LoadContent();
79          }
80
81          public override void Update(GameTime gameTime)
```

```
82
83          {
84              if (gameState == GameState.mainMenu)
85              {
86                  score = 0;
87                  menu.Update();
88              }
89
90
91              if (gameState == GameState.enterName)
92              {
93                  if (KeyboardInput.IsVisible)
94                  {
95                      gameState = GameState.viewLeaderboards;
96                  }
97                  // respawn the enemy
98                  e.Spawn(gameTime);
99                  // revive the player
100                 player.IsDead = false;
101                 // reset the score
102
103                 player.Position = new Vector2(130, groundLevel - 99);
104                 return;
105             }
106
107             if (gameState == GameState.viewLeaderboards)
108             {
109                 menu.Update();
110             }
111
112             if (gameState==GameState.inGame)
113             {
114                 score += 2;
115
116                 // Update player
117                 player.Update(gameTime);
118
119                 // Update all background sprites
120                 foreach (BackgroundSprite s in spriteList)
121                 {
122                     s.Update(gameTime, 1920);
123                 }
124
125
126                 // Handle the spawning of enemies  (Currently: 1 enemy per
                      screen)
127                 if (gameTime.TotalGameTime - previousSpawnTime >
                      enemySpawnTime)
128                 {
129                     current_enemy = random.Next(0, 4);
130                     e = enemies[current_enemy];
131
132                     timeonscreenMS = (1400 / (e.Speed.X * 60))*10;
133                     previousSpawnTime = e.Spawn(gameTime);
134
135                     int spawnSeconds = random.Next(6, 7); // random should be
```

```
                    a member of the class
136                 enemySpawnTime = TimeSpan.FromSeconds(spawnSeconds);
137
138             }
139
140             e.Behaviour(gameTime, ref timeonscreenMS);
141             e.Update(gameTime);
142
143             // If the player hits an enemy show the keyboard
144             if (Collide(e))
145             {
146                 player.IsDead = true;
147                 if (!KeyboardInput.IsVisible)
148                     NewKeyboard();
149                 gameState = GameState.enterName;
150                 return;
151             }
152
153         }
154
155         base.Update(gameTime);
156     }
157
158
159
160     public override void Draw(GameTime gameTime)
161     {
162         spriteBatch.Begin(SpriteSortMode.Deferred, null, null, null, null, ↵
               null, Game1.screenScale);
163
164         if (gameState == GameState.mainMenu)
165         {
166             foreach (BackgroundSprite s in spriteList) { s.Draw(gameTime, ↵
                  spriteBatch, 1920); }
167             menu.Draw(spriteBatch);
168
169         }
170
171         if (gameState == GameState.enterName)
172         {
173             spriteBatch.End();
174             return;
175         }
176
177         if(gameState == GameState.viewLeaderboards)
178         {
179             spriteBatch.Draw(Game.Content.Load<Texture2D>("leaderboards"), ↵
                   new Rectangle(720, 100, 500, 900), Color.White);
180             List<Scores> scores = Database.getAllScores();
181
182             for (int i = 0, offset = 330; i < scores.Count; i++)
183             {
184                 Scores s = scores[i];
185                 offset += 90;
186                 spriteBatch.DrawString(sf, i+1 + ".    " + s.Name + "     " ↵
                    + s.score, new Vector2(800, offset), Color.Crimson);
```

```
187                 }
188
189                 menu.Draw(spriteBatch);
190             }
191
192             if (gameState == GameState.inGame)
193             {
194                 if (player.IsDead) { GraphicsDevice.Clear(Color.Black);
                      spriteBatch.End(); return; }
195
196                 rotationAngle -= 0.2F;
197
198
199                 // Draw backgroud
200                 foreach (BackgroundSprite s in spriteList)
201                 {
202                     if(s.Loops)
203                         s.Draw(gameTime, spriteBatch, 1920);
204                     else
205                     s.Draw(gameTime, spriteBatch);
206                 }
207
208                 // Draw player
209                 Rectangle rect = player.collisionRect();
210                 //spriteBatch.Draw(Game.Content.Load<Texture2D>
                      ("dummytexture"), rect, Color.AliceBlue);
211                 player.Draw(gameTime, spriteBatch, player.Scale,
                      SpriteEffects.FlipHorizontally);
212
213                 spriteBatch.DrawString(sf, "Score: " + score, new Vector2
                      (1600, 200), Color.Beige);
214                 // Draw current enemy
215
216                 //spriteBatch.Draw(Game.Content.Load<Texture2D>
                      ("dummytexture"), e.collisionRect(), Color.AliceBlue);
217                 if (e.Type == ENEMY_TYPES.BOULDER)
218                     e.DrawRotating(gameTime, spriteBatch, e.Scale,
                        SpriteEffects.None, rotationAngle);
219                 else if (e.Type == ENEMY_TYPES.GHOST)
220                     e.Draw(gameTime, spriteBatch, e.Scale,
                        SpriteEffects.FlipHorizontally);
221                 else
222                     e.Draw(gameTime, spriteBatch, e.Scale,
                        SpriteEffects.None);
223
224             }
225             spriteBatch.End();
226         }
227
228         protected bool Collide(Enemy enemy)
229         {
230             return player.collisionRect().Intersects(enemy.collisionRect());
231         }
232
233         public async void NewKeyboard()
234         {
```

```
235             await ShowKeyboard();
236         }
237         private async Task ShowKeyboard()
238         {
239             await Task.Run(async () =>
240             {
241                 var result = await KeyboardInput.Show("enter your name",
                      "name", "", false);
242                 if (null != result)
243                 {
244                     //your method to set text goes here
245                     name = result;
246                     if (name == "")
247                     {
248                         name = "ANONYMOUS";
249                     }
250                     Scores score_player= new Scores();
251                     score_player.Name = name;
252                     score_player.score = score;
253                     Database.SaveScore(score_player);
254 
255                 }
256             });
257         }
258 
259     }
260 }
261 
```