```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Text;
 4  using Microsoft.Xna.Framework;
 5  using Microsoft.Xna.Framework.Graphics;
 6  using Microsoft.Xna.Framework.Input;
 7
 8  using Microsoft.Xna.Framework.Content;
 9  using Android.Gestures;
10  using Microsoft.Xna.Framework.Input.Touch;
11
12  namespace GameDemo.Shared
13  {
14      class Player : Sprite
15      {
16
17          bool isDead;
18          bool isDucking;
19
20          #region Animations
21          AnimationPlayer playerAnimate;
22          Animation ducking;
23          Animation running;
24          //Animation dying;
25          Animation jumping;
26
27          #endregion
28          // Constants for controling horizontal movement
29
30          public Vector2 Velocity
31
32          {
33
34              get { return velocity; }
35
36              set { velocity = value; }
37
38          }
39
40          public Vector2 Position
41          {
42              get { return position; }
43              set { position = value; }
44          }
45
46
47          private float startingYpos;
48          Vector2 velocity;
49
50
51          // Constants for controlling vertical movement
52
53          private const float DuckTime = 500F;
54
55          private const float JumpLaunchVelocity_Y = -40;
56
```

```
57          private const float JumpLaunchVelocity_X = 40;
58
59          private const float GravityAcceleration = 3;
60
61          //private static float timeSinceJump = 0;
62
63          private float airdrag = 1;
64
65          TimeSpan elapsedTimeForAnimation= TimeSpan.Zero;
66
67          bool isOnGround;
68
69          public bool IsDead
70          {
71              get
72              {
73                  return isDead;
74              }
75              set { isDead = value; }
76          }
77
78          public bool IsOnGround
79          {
80
81              get
82              {
83                  return isOnGround;
84              }
85              set { IsOnGround = value; }
86          }
87
88          // define the margin of the collision
89          private Rectangle localBounds;
90
91          /// <summary>
92
93          /// Gets a rectangle which bounds this player in world space. No use
              for the moment
94
95          /// </summary>
96
97          public Rectangle BoundingRectangle
98          {
99              get
100             {
101                 // smaller hitbox for ducking
102                 if (isDucking)
103                 {
104                     int left = (int)Math.Round(position.X -
                        playerAnimate.Origin.X) ;
105
106                     int top = (int)Math.Round(position.Y -
                        playerAnimate.Origin.Y)+30 ;
107                     return new Rectangle(left, top, (int)((float)
                        playerAnimate.Animation.FrameWidth*scale-(int)
                        collisionOffset.X),
```

```
108                         (int)((float)playerAnimate.Animation.FrameHeight*scale
                        - (int)collisionOffset.Y));
109                 }
110                 else
111                 {
112                     int left = (int)Math.Round(position.X -
                        playerAnimate.Origin.X);
113
114                     int top = (int)Math.Round(position.Y -
                        playerAnimate.Origin.Y)-20;
115                     return new Rectangle(left, top, (int)((float)
                        playerAnimate.Animation.FrameWidth*scale - (int)
                        collisionOffset.X),
116                         (int)((float)playerAnimate.Animation.FrameHeight*scale
                        - (int)collisionOffset.Y));
117
118                 }
119             }
120         }
121
122         public Player(Texture2D texture, Vector2 position, Point frameSize,
              int totalFrames, Vector2 collisionOffset, Point currentFrame, Point
              sheetSize, Vector2 speed,float scale)
123             : base(texture, position, frameSize, totalFrames, collisionOffset,
                   currentFrame, sheetSize, speed,scale)
124         {
125
126
127             startingYpos = position.Y;
128             isOnGround = false;
129             isDucking = false;
130             isDead = false;
131         }
132
133         public Player(Texture2D texture, Vector2 position, Point frameSize,
              int totalFrames, Vector2 collisionOffset, Point currentFrame, Point
              sheetSize, Vector2 speed, int secondsperFrame,float scale)
134             : base(texture, position, frameSize, totalFrames, collisionOffset,
                   currentFrame, sheetSize, speed, secondsperFrame,scale)
135         {
136             localBounds = new Rectangle((int)position.X, (int)position.Y,
                frameSize.X+(int)collisionOffset.X, frameSize.Y+(int)
                collisionOffset.Y);
137             startingYpos = position.Y;
138         }
139         public void LoadContent(ContentManager contentManager)
140         {
141             playerAnimate = new AnimationPlayer();
142             running = new Animation(contentManager.Load<Texture2D>
                ("horse_running"), 60, 11, true);
143             jumping = new Animation(contentManager.Load<Texture2D>
                ("horse_jump"), 20, 16, false);
144             ducking = new Animation(contentManager.Load<Texture2D>
                ("ducking1"), 40, 11, true);
145
146         }
```

```
147
148         /// <summary> TOUCH IN MONOGAME
149         /// https://gregfmartin.com/2017/12/27/monogame-working-with-touch/
150         /// </summary>
151
152
153         public override void Update(GameTime gametime)
154         {
155
156             var gesture = default(GestureSample);
157
158
159                 // for some reason in debug its false before the first touch even though we enabled some gestures
160             while (TouchPanel.IsGestureAvailable)
161             {
162                 gesture = TouchPanel.ReadGesture();
163                 TimeSpan time = gesture.Timestamp;
164
165                 // JUMPING
166                 if (gesture.GestureType == GestureType.Flick && gesture.Delta.Y<0)
167                 {
168                     if (isOnGround)
169                     {
170                         playerAnimate.PlayAnimation(jumping);
171                         velocity.Y = JumpLaunchVelocity_Y;
172                         velocity.X = JumpLaunchVelocity_X;
173                         isOnGround = false;
174                         isDucking = false;
175                     }
176                 }
177                 // DUCKING
178                 if(gesture.GestureType == GestureType.Tap )
179                 {
180                     if (isOnGround)
181                     {
182                         isDucking = true;
183                         playerAnimate.PlayAnimation(ducking);
184                         elapsedTimeForAnimation = TimeSpan.Zero;
185                     }
186                 }
187                 // SLOWING DOWN
188                 if (gesture.GestureType == GestureType.Flick && gesture.Delta.Y > 0)
189                 {
190                     velocity.X -= 5;
191                 }
192             }
193
194             #region ducking_handling
195
196             if (isDucking )
197             {
198                 if(elapsedTimeForAnimation==TimeSpan.Zero)
199                 velocity.X += 10F;
```

```
200                 elapsedTimeForAnimation=elapsedTimeForAnimation.Add ↵
                      (gametime.ElapsedGameTime);
201                 velocity.X += 1;
202             }
203 
204             if(elapsedTimeForAnimation.Milliseconds> DuckTime)
205             {
206                 isDucking = false;
207                 playerAnimate.PlayAnimation(running);
208                 elapsedTimeForAnimation = TimeSpan.Zero;
209             }
210 
211             #endregion
212 
213             // change the velocity of the player due to factors
214             velocity.Y += GravityAcceleration;
215             velocity.X -= airdrag;
216 
217             // update player position according to his velocity
218             position.Y += velocity.Y;
219             position.X += velocity.X;
220 
221             // So the player doesnt fall off the screen
222             if (position.Y >= startingYpos )
223             {
224                 position.Y = startingYpos;
225                 velocity.Y = 0F;
226                 isOnGround = true;
227                 if (isDucking)
228                 {
229 
230                 }
231                 else playerAnimate.PlayAnimation(running);
232 
233             }
234             // Limit the player to certain bounds of the screen so he doesnt ↵
                  run off
235 
236             if(position.X < 130 )
237             {
238                 airdrag = 1;
239                 position.X = 130;
240                 velocity.X = 0;
241             }
242             if(position.X> 1300)
243             {
244                 position.X = 1300;
245                 velocity.X = 0;
246             }
247 
248             // Limit player speed
249 
250             if (velocity.X < -8)
251             {
252                 velocity.X = -5;
253             }
```

```
254             if(velocity.X>10 && isOnGround)
255             {
256                 velocity.X = 10;
257             }
258 
259 
260         }
261         public override void Draw(GameTime gametime, SpriteBatch
              spritebatch,float scale,SpriteEffects spriteEffects )
262         {
263             // because at first there is no animation
264             if (playerAnimate.Animation == null)
265                 playerAnimate.PlayAnimation(running);
266             if (isDucking)
267             {
268                 // Lower the sprite because its a bit smaller when ducking
269                 position.Y += 20;
270                 playerAnimate.Draw(gametime, spritebatch, position,
                      spriteEffects, scale+0.1F);
271             }
272             else
273             playerAnimate.Draw(gametime, spritebatch, position,
                  spriteEffects,scale);
274 
275         }
276 
277         public override Rectangle collisionRect()
278         {
279             return BoundingRectangle;
280         }
281 
282 
283     }
284 }
285 
```