```
1  #region Using Statements
2  using System;
3  using System.Threading.Tasks;
4  using Android.OS;
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 #endregion
13
14 namespace GameDemo.Shared
15 {
16     /// <summary>
17     /// This is the main type for your game.
18     /// </summary>
19     public class Game1 : Game
20     {
21         GraphicsDeviceManager graphics;
22         SpriteBatch spriteBatch;
23         SpriteManager spriteManager;
24
25         // scales the game to any aspect ratio and resolution of a screen
26         public static Matrix screenScale = Matrix.Identity;
27
28         public Game1()
29         {
30
31
32             graphics = new GraphicsDeviceManager(this);
33
34             Content.RootDirectory = "Content";
35
36             graphics.IsFullScreen = true;
37
38             graphics.ApplyChanges();
39
40
41             // add the handler corresponding to the display orientation
42             if (this.Window.CurrentOrientation== DisplayOrientation.Portrait)
43             this.Window.ClientSizeChanged += WindowSizeChange;
44             else
45             {
46                 // no need to change anything
47             }
48
49         }
50
51         /// <summary>
52         /// Allows the game to perform any initialization it needs to before
               starting to run.
53         /// This is where it can query for any required services and load any
               non-graphic
54         /// related content.  Calling base.Initialize will enumerate through
```

```
            any components
55          /// and initialize them as well.
56          /// </summary>
57          protected override void Initialize()
58          {
59              // TODO: Add your initialization logic here
60              spriteManager = new SpriteManager(this);
61              Components.Add(spriteManager);
62
63              TouchPanel.EnabledGestures = GestureType.Tap  | GestureType.None |
                    GestureType.Flick;
64
65              base.Initialize();
66
67          }
68
69          /// <summary>
70          /// LoadContent will be called once per game and is the place to load
71          /// all of your content.
72          /// </summary>
73          protected override void LoadContent()
74          {
75              // Create a new SpriteBatch, which can be used to draw textures.
76              spriteBatch = new SpriteBatch(GraphicsDevice);
77
78              //TODO: use this.Content to load your game content here
79          }
80
81          /// <summary>
82          /// Allows the game to run logic such as updating the world,
83          /// checking for collisions, gathering input, and playing audio.
84          /// </summary>
85          /// <param name="gameTime">Provides a snapshot of timing values.</
              param>
86          protected override void Update(GameTime gameTime)    // gameTime is
              actuallly used because every processor speed is different so we need
               the time that HAS PASSED during the running of the game to code
87          {
88              if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
                    ButtonState.Pressed)
89                  this.Exit();
90              if (SpriteManager.gameState == GameState.over)
91              {
92                  Process.KillProcess(Process.MyPid());
93
94              }
95
96              // For Mobile devices, this logic will close the Game when the
                    Back button is pressed
97              // Exit() is obsolete on iOS
98              // TODO: Add your update logic here
99              base.Update(gameTime);
100         }
101
102         // if the resolution is changed (we are working with a 1920,1080 view)
103         public void WindowSizeChange(object sender, EventArgs e)
```

```
104         {
105             var bw = GraphicsDevice.PresentationParameters.BackBufferWidth;
106             var bh = GraphicsDevice.PresentationParameters.BackBufferHeight;
107             screenScale = Matrix.Identity * Matrix.CreateScale(bw / 1080, bh / 
                    1920, 0f);
108         }
109 
110         /// <summary>
111         /// This is called when the game should draw itself.
112         /// </summary>
113         /// <param name="gameTime">Provides a snapshot of timing values.</
              param>
114         protected override void Draw(GameTime gameTime)
115         {
116             GraphicsDevice.Clear(Color.Black);
117 
118             //TODO: Add your drawing code here
119             spriteBatch.Begin(SpriteSortMode.Deferred, null, null, null, null, 
                    null, screenScale);
120 
121             spriteBatch.End();
122             base.Draw(gameTime);
123 
124         }
125     }
126 }
127 
```