

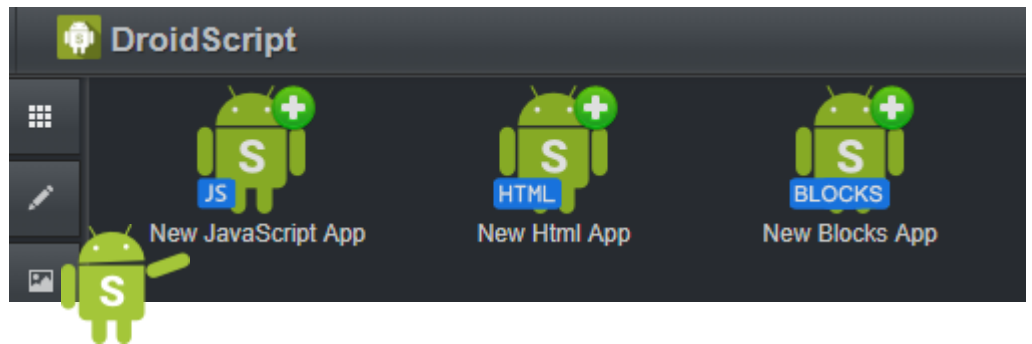
DroidScript Gameview

Tutorial - Flappy Dino Part 1

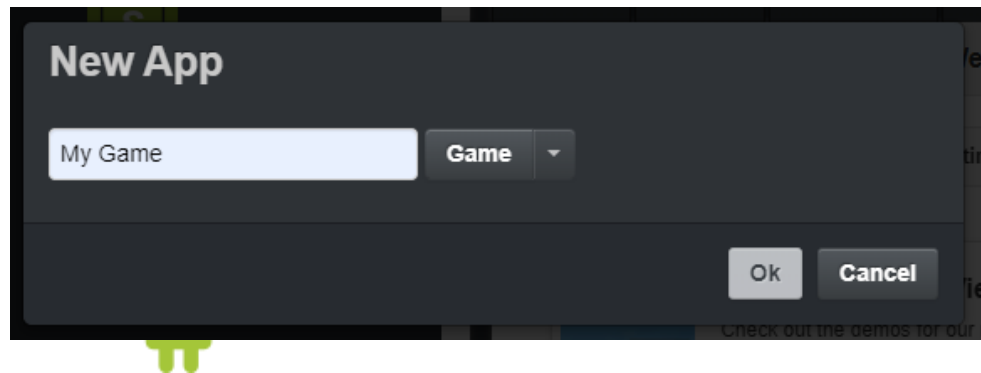


Setting up a New Game – From the Web Editor:

First you will need to launch DroidScript and connect to the Wi-Fi editor. From here open the apps view and select 'New Javascript App'.



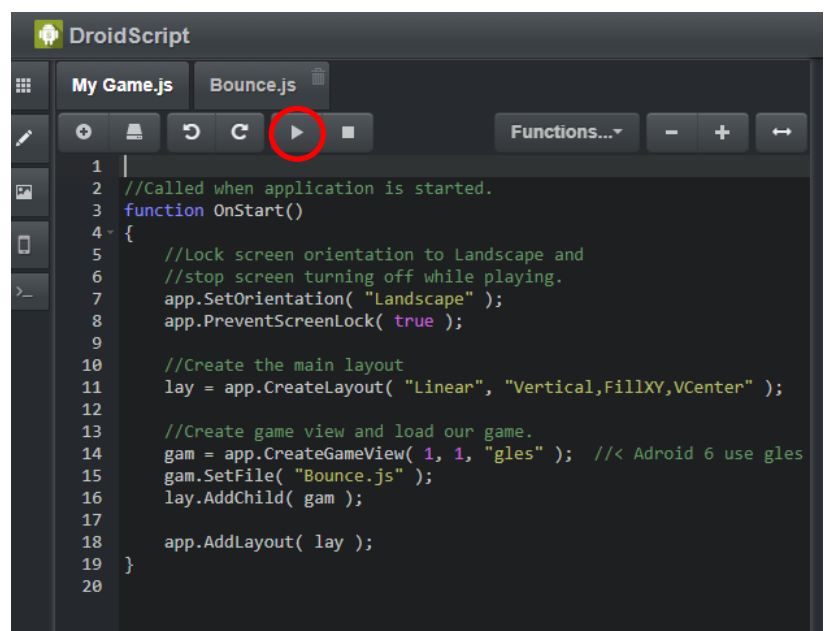
This will bring up a box where you can choose the name for your app and choose the type. Make sure you select 'Game' from the drop down menu.



Once you click OK, your new app will be opened. There will be two files in a Game app, the first is used to set up the game in DroidScript (called My Game.js) and the second is the actual game code (Bounce.js).

When you create a new game, there is an example game in the file called bounce.js.

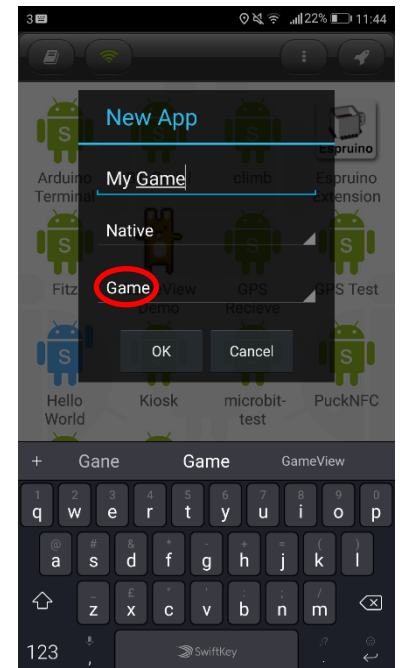
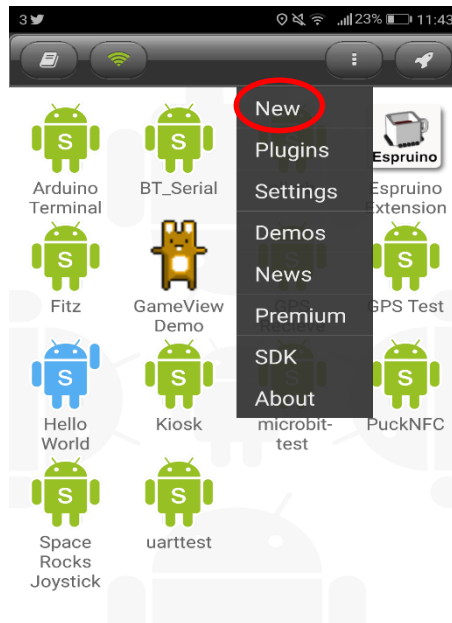
You can test this by pressing the play button at the top of the code window.



Setting up a New Game – From the Mobile App:

Click the menu button in the top right-hand corner and select “New”.

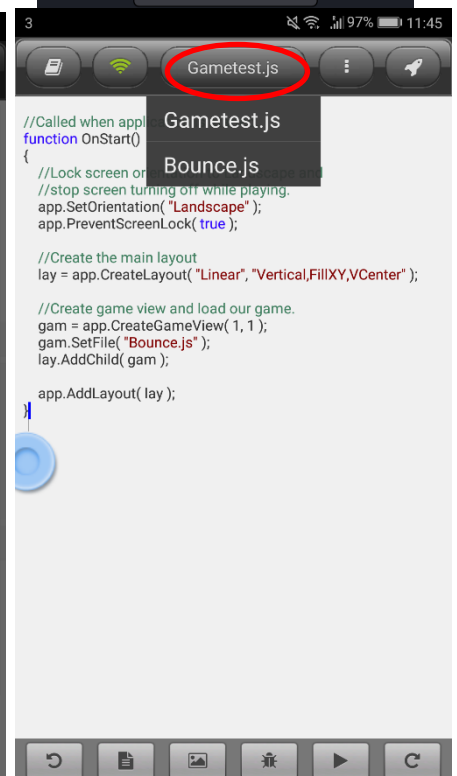
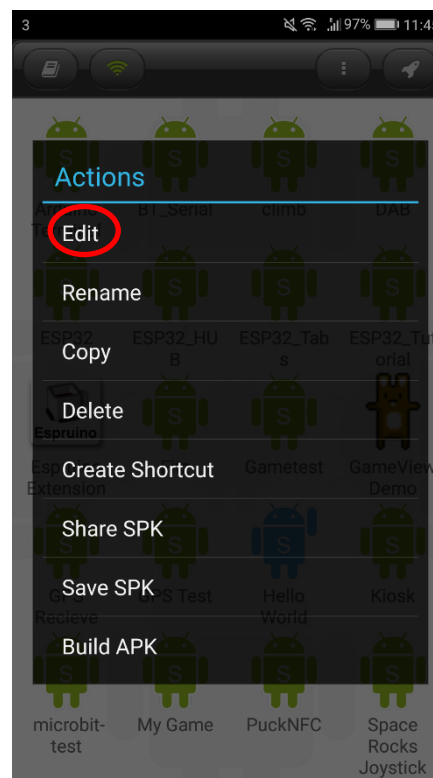
When the “New App” dialog appears, Give your new game a name and make sure that the drop down menus show “Native” and “Game”. You can then click OK and your game will be created.



The app will then take you back to the home screen where you should be able to see the icon for your new app. Long press on this icon to bring up the Actions menu. Select the first option “Edit”.

The editor will then open where you will see the code for a demo app. This can either be edited or deleted. In every Game type of app, there are two script files. These can be switched between by pressing the button with the file name at the top centre.

To run the demo game in the app, the play button is at the bottom right hand corner.



Setting up a New Game – Creating new game files

For the remainder of this tutorial we will assume you are using the web editor.

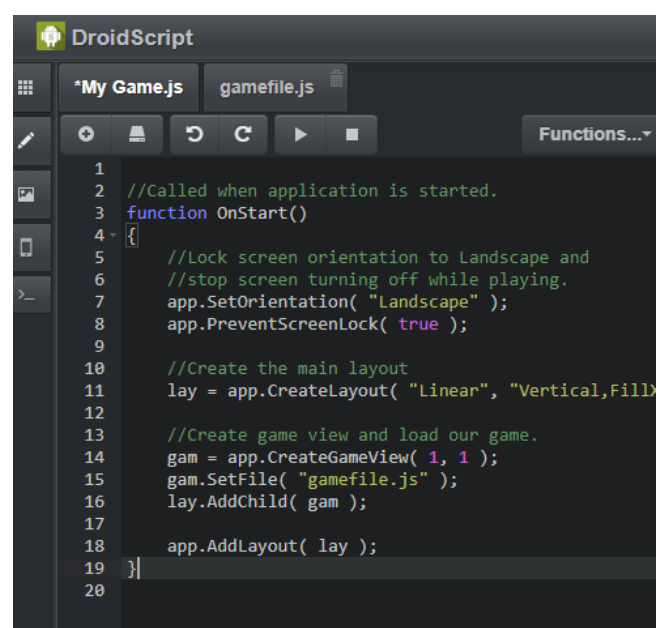
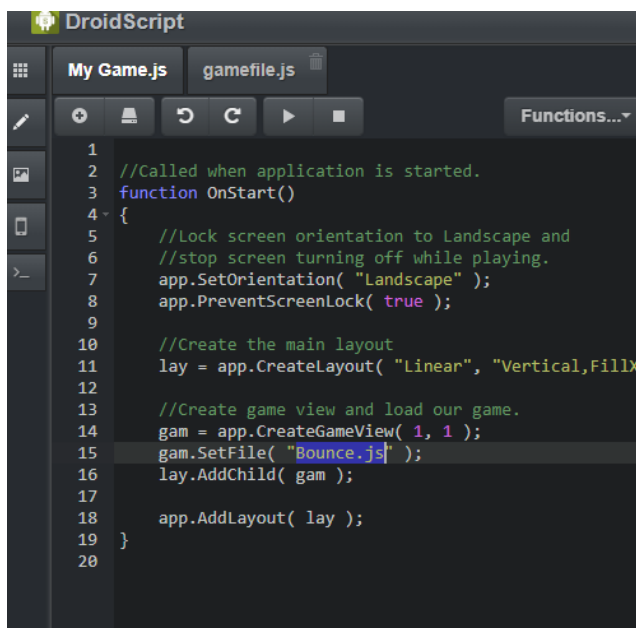
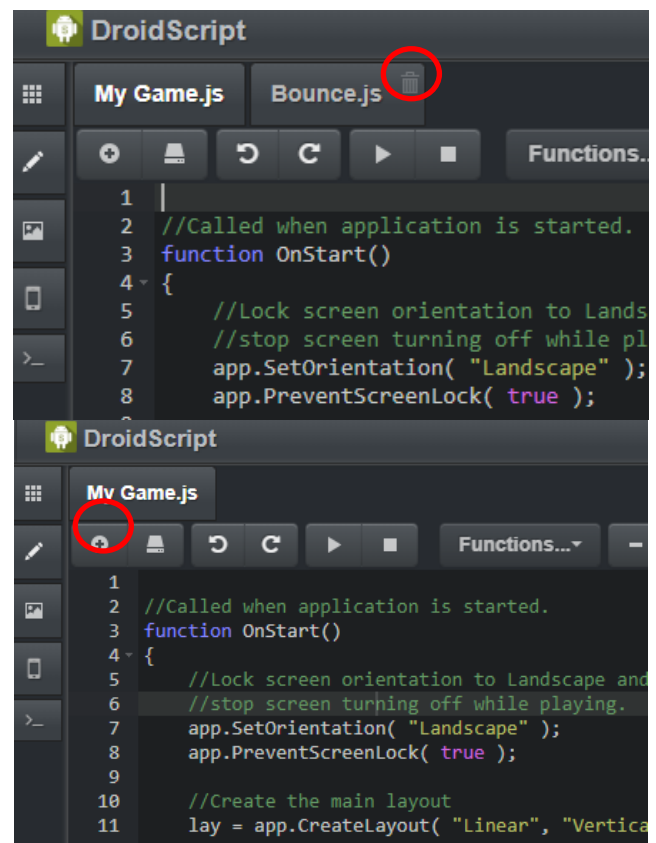
Now you have created your new game and played with the demo script, we need to modify it so we can start building our own game.

At the top you will see two files, one with the game name and another called Bounce.js. We don't need the Bounce.js file any more as we will create our own new game file so press the trash can symbol on the tab to delete it. A popup will ask if you're sure, click OK.

Next we need to add our own file. To do this click on the "+" symbol on the left hand side. This will bring up the "New File" popup. Give your file a name and press "OK".

A game will always need these two files. The first is the main program that creates the app and launches the game code, and the second is the file that actually contains the game code.

The last thing we need to do to set up the game is change the name of the game file in the first script from "bounce.js" the name of our new file. Here it is "gamefile.js". We are now ready to start building a game! All of the following code will be written in the game file.



Lesson 1 – Anatomy of the Game

OnLoad, OnReady, OnAnimate

Every game requires three main functions in order to run:

OnLoad() – This is the first thing will run when the game is launched, this is where graphical objects, sounds, backgrounds and physics should be created.

OnReady() – This is executed when OnLoad() function has finished running. Here the graphical objects, backgrounds, sounds etc. are added to the GameView with their appropriate settings. Once everything has been set up, calling the 'gfx.Play()' function will start the game running.

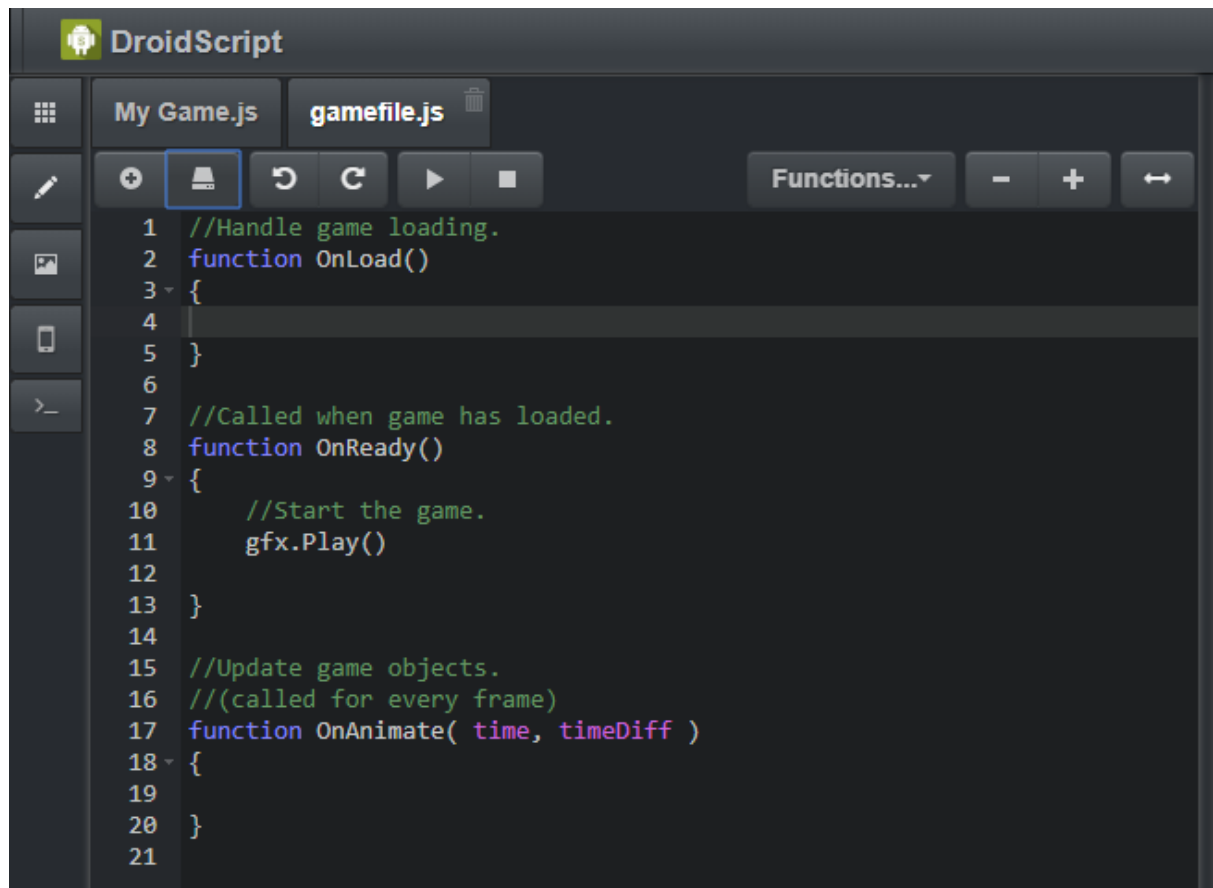
OnAnimate() – This function is repeatedly called, around 60 times per second. This is where you update the position of backgrounds and graphical objects.

Copy this code into the game file you created. If you run the app now by pressing the play button, there should be no errors but nothing will happen as there is nothing in the functions yet.

```
//Handle game loading.
function OnLoad()
{
}

//Called when game has loaded.
function OnReady()
{
    //Start the game.
    gfx.Play()
}

//Update game objects.
//(called for every frame)
function OnAnimate( time, timeDiff )
{
}
```



```
DroidScript
My Game.js  gamefile.js
+  [ ]  ↺  ↻  ▶  ■  Functions...  -  +  ↔
1  //Handle game loading.
2  function OnLoad()
3  {
4
5  }
6
7  //Called when game has loaded.
8  function OnReady()
9  {
10     //Start the game.
11     gfx.Play()
12
13 }
14
15 //Update game objects.
16 //(called for every frame)
17 function OnAnimate( time, timeDiff )
18 {
19
20 }
21
```

Lesson 2 – Adding Assets

To build a game we will need images for the background and characters (Sprites) as well as sound effects.

To add these, click the “Assets” button on the left-hand side.

From there you can add images and sounds by pressing the upload button. Find the files on your computer, select them all and press

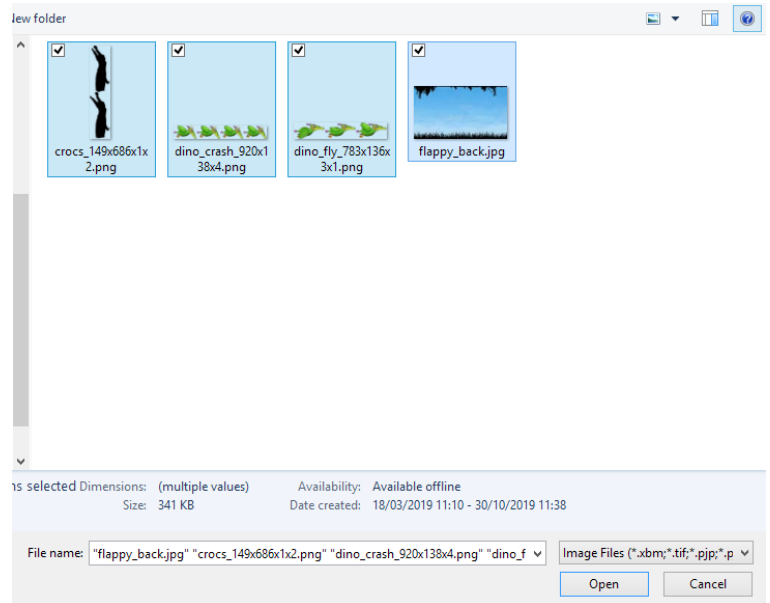
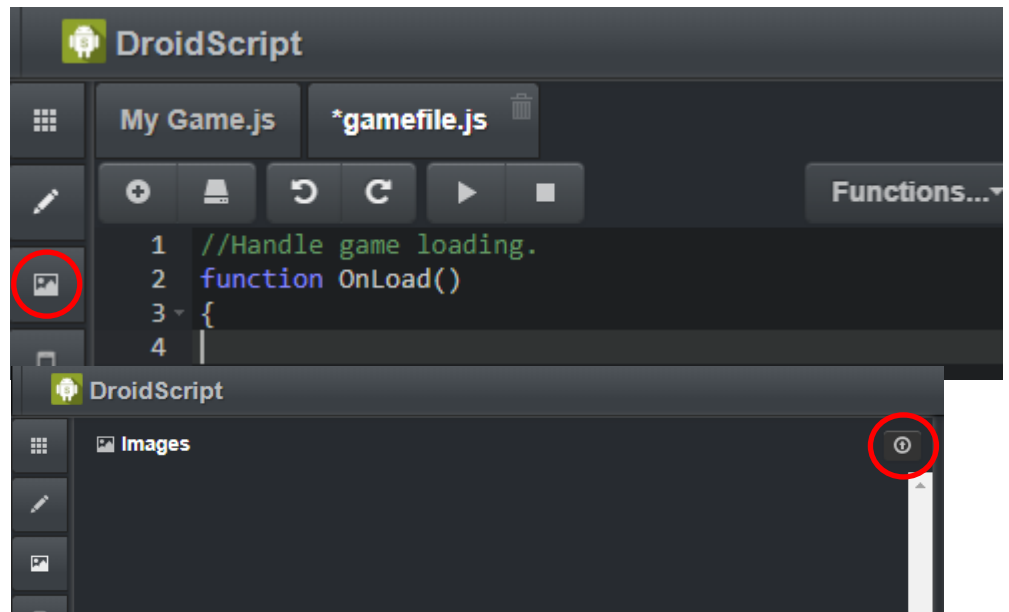
“Open”.

Next do the same for the sound files.

All the files for this tutorial will be available to download.

The files needed are listed below and can be found at:

<http://androidscript.org/tutorials/assets/>



<input type="checkbox"/>	Name	Type	Size
<input checked="" type="checkbox"/>	flappy_back.jpg	JPEG image	129 KB
<input checked="" type="checkbox"/>	crunch.mp3	MP3 Format Sound	36 KB
<input checked="" type="checkbox"/>	game_over.mp3	MP3 Format Sound	70 KB
<input checked="" type="checkbox"/>	jungle.mp3	MP3 Format Sound	261 KB
<input checked="" type="checkbox"/>	crocs_149x686x1x2.png	PNG image	12 KB
<input checked="" type="checkbox"/>	dino_crash_920x138x4.png	PNG image	97 KB
<input checked="" type="checkbox"/>	dino_fly_783x136x3x1.png	PNG image	105 KB

Lesson 3 – Creating Backgrounds

`gfx.CreateBackground()`, `gfx.AddBackground()`

The next step to creating a game is choosing, creating and adding a background. This requires two lines of code, one in the `OnLoad()` function like this:

```
//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )
}
```

And one in the `OnReady()` function which adds the previously created background to the `GameView`, like this:

```
//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Start the game.
    gfx.Play()
}
```

The file name must match the name of the Assets we added in the previous step. If you try running the app now, you should see the background image appear.



Lesson 4 – Scrolling Backgrounds

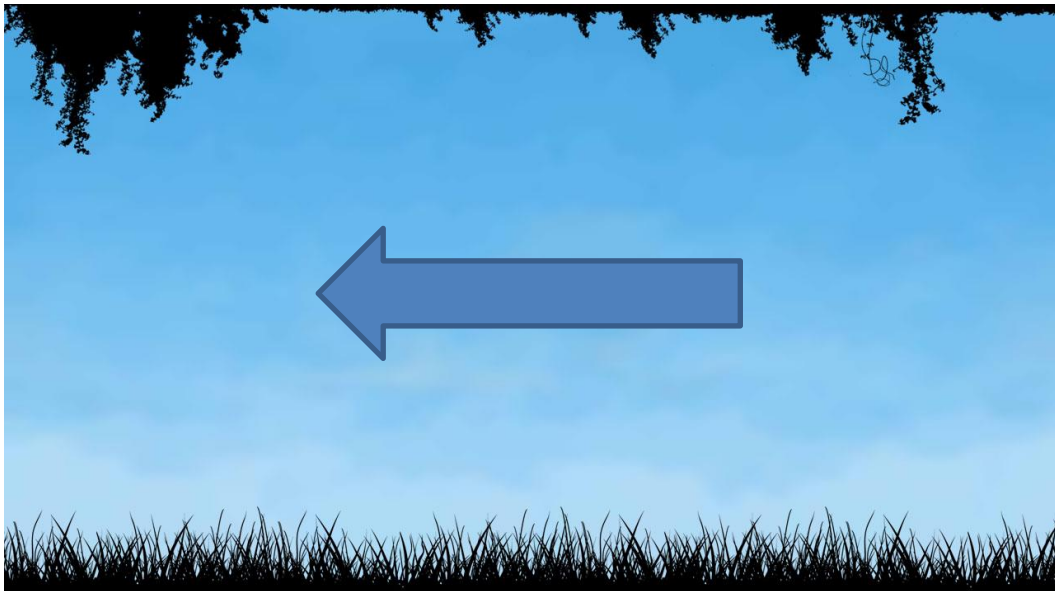
`img.Scroll()`

Some games will use a still background, but most will use a scrolling background to give the impression of the player moving through an environment. The `Scroll(x, y)` function is given two values, of the amount moved in each direction.

```
//Update game objects.  
//(called for every frame)  
function OnAnimate( time, timeDiff )  
{  
    //Make background slide to the left a small amount.  
    sky.Scroll( -0.002, 0 )  
}
```

All size and position values when building a game are given as fractions of the total screen width; this is so that it can be played on any device regardless of the screen size and orientation. Here the sky image is scrolled to the left by 2% of the total screen width, each time the frame is updated.

Add the above code to the “OnAnimate” function and if you run the app, this time you should see the background scrolling to the left.



Lesson 5 – Adding Characters (Sprites)

gfx.CreateSprite(), gfx.AddSprite()

Characters in the game are known as sprites. These need to be added in the same way as the background, created in the OnLoad() function and added to the GameView in the OnReady() function.

The numbers after 'dino' when adding the sprite show the x and y position that the image will appear on the screen. These positions are given as fractions of the screen width and height away from the top left hand corner. This corner would have a position of 0,0.

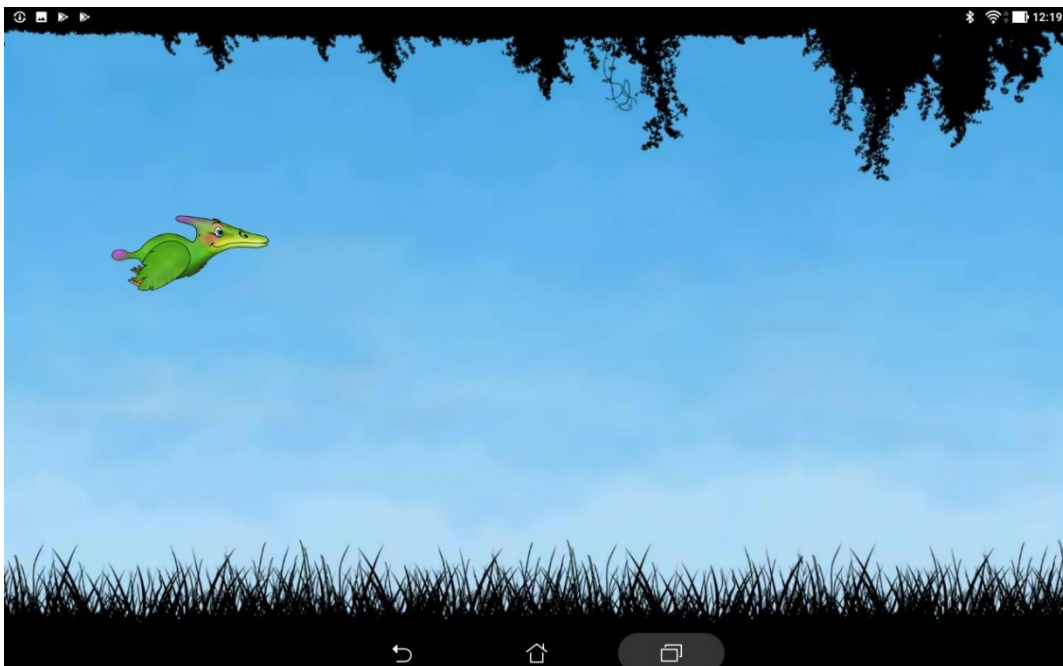
```
//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )

    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )
}

//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Add dino image
    gfx.AddSprite( dino, 0.1, 0.3, 0.15 )

    //Start the game.
    gfx.Play()
}
```



Lesson 6 – Sound Effects

`gfx.CreateSound(), snd.Play()`

Sounds effects and game music are also created in the `OnLoad()` function. When you want to play the sound, you use `.Play`. If the sound is game music it can be called in the `OnReady()` function, or if it is a sound effect you would call it in the `OnAnimate()` function. Here is an example of game music:

```
//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )

    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )

    //Create sounds.
    jungle = gfx.CreateSound( "Snd/jungle.mp3" )
}

//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Add dino image
    gfx.AddSprite( dino, 0.1, 0.3, 0.15 )

    //Play looping background sound.
    jungle.Play(true)

    //Start the game.
    gfx.Play()
}
```

Lesson 7 – Character Animation – Sprite Sheets

Sprite Animation

Characters that have repeating movements can be created using 'Sprite Sheets'. These are images that are made up of a series of smaller pictures that are cycled through repeatedly.



The game engine knows how to split this image up using the details in the file name. For example the sheet shown here has the filename: 'dino_fly_783x136x3x1' The numbers after the name state that the total image size is 783x136 and the frames are split into 3 columns and 1 row.

```
//Called when game has loaded.
function OnReady()
{
  //Set background.
  gfx.AddBackground( sky )

  //Add dino image
  gfx.AddSprite( dino, 0.1, 0.3, 0.15 )

  //Play looping background sound.
  jungle.Play(true)

  //Start animations
  dino.Play( 0, 0.25 )

  //Start the game.
  gfx.Play()
}
```

If the filename has no extension (.png) the game engine will assume the image is a sprite sheet.

Lesson 8 – Character Animation – Movement

We can now add a second sprite to make an enemy in the game.

```
//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )

    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )
    croc = gfx.CreateSprite( "Img/crocs_149x686x1x2", "enemies" )

    //Create sounds.
    jungle = gfx.CreateSound( "Snd/jungle.mp3" )
}

//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Add dino image
    gfx.AddSprite( dino, 0.1, 0.3, 0.15 )
    //Add enemy sprites.
    gfx.AddSprite( croc, 0.8, 0.55, 0.12 )

    //Start animations
    dino.Play( 0, 0.25 )
    croc.Play( 0, 0.05 )

    //Play looping background sound.
    jungle.Play(true)

    //Start the game.
    gfx.Play()
}
```

Character Movement

To animate your character, you will need to change its position each time the frame updates. In the example below the 'croc' object is shifted by 0.4% of the screen width to the left (-x direction).

An alternative and faster way to write 'croc.x = croc.x - 0.004' would be: 'croc.x -= 0.004'.

```
//Update game objects.
//(called for every frame)
function OnAnimate( time, timeDiff )
{
    //Make background slide to the left a small amount.
    sky.Scroll( -0.002, 0 )

    //Slide croc to left.
    croc.x = croc.x - 0.004
}
}
```

Lesson 9 – Putting it All Together (So Far)

Before we are ready to move onto the next feature, we need to test what we have learnt so far. The following code includes all the features covered so far and should produce a flapping dino in front of a scrolling background. A croc will also appear and move across the screen when the app is first run.

The 'gfx.Play()' function is what sets the whole game rolling.

So far the player does not have any control and there is no way of detecting collisions between the dino and croc. This will be covered in the next lesson.

```
//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )

    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )
    croc = gfx.CreateSprite( "Img/crocs_149x686x1x2", "enemies" )

    //Create sounds.
    jungle = gfx.CreateSound( "Snd/jungle.mp3" )
}

//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Add dino image (with physics)
    gfx.AddSprite( dino, 0.1, 0.6, 0.15 )

    //Add enemy sprites.
    gfx.AddSprite( croc, 0.8, 0.55, 0.12 )

    //Start animations
    dino.Play( 0, 0.25 )
    croc.Play( 0, 0.05 )

    //Play looping background sound.
    jungle.Play()

    //Start the game.
    gfx.Play()
}

//Update game objects.
//(called for every frame)
function OnAnimate( time, timeDiff )
{
    //Make background slide to the left a small amount.
    sky.Scroll( -0.002, 0 )

    //Slide croc to left.
    croc.x = croc.x - 0.004
}
}
```

Lesson 10 – Basic Collisions

To make this a game there must be a way to win or lose. Here the game is lost if the player collides with an enemy or the ground. To detect the collision between two objects, the simplest method is to detect when two images on the screen overlap. This is achieved using the 'IsOverlap()' function.

We also need to add the sound effect that plays when the dino collides with a croc and change the sprite sheet to the crashed dino.

To test this you can change the Y position of the dino in "OnReady()" from 0.3 to 0.5 meaning the dino will start lower and crash into the croc as it comes past.

```
//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Add dino image
    gfx.AddSprite( dino, 0.1, 0.5, 0.15 )
    //Add enemy sprites.
    gfx.AddSprite( croc, 0.8, 0.55, 0.12 )
}
```

```
//Global variables
var gameOver = false

//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )

    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )
    croc = gfx.CreateSprite( "Img/crocs_149x686x1x2", "enemies" )
    crash = gfx.CreateSpriteSheet( "Img/dino_crash_920x138x4" )

    //Create sounds.
    jungle = gfx.CreateSound( "Snd/jungle.mp3" )
    crunch = gfx.CreateSound( "Snd/crunch.mp3" )
}

function OnAnimate( time, timeDiff )
{
    //Make background slide to the left a small amount.
    sky.Scroll( -0.002, 0 )

    //Slide croc to left.
    croc.x = croc.x - 0.004
    if( gfx.IsOverlap(dino, croc, 0.03) && !gameOver )
    {
        Crash()
    }
}

//Crash our character and end the game.
function Crash()
{
    //Switch sprite sheets and change play speed.
    dino.SetSpriteSheet( crash )
    dino.Play( 0, 0.1 )

    //Play crunch sound and prevent scroll.
    crunch.Play()

    //Set gameover state.
    var gameOver = true
}
```

Lesson 11 – Touch Control

Next we need a way to avoid crashing into the croc. The OnControl() function can be used to detect when the player touches the screen. If the following function is added to the bottom of the game code, every time the screen is touched, the dino moves up the screen.

```
//Handle screen touches and key presses.  
function OnControl( touchState, touchX, touchY, keyState, key )  
{  
    //Increase dino's upward velocity when screen touched.  
    if( touchState=="Up" && !gameOver ) dino.y-= 0.03  
}
```

If we try the game now, we can avoid the croc but there is nothing to bring our dino back down again. To make the game more fun, add this line to the OnAnimate function:

```
//Update game objects.  
//(called for every frame)  
function OnAnimate( time, timeDiff )  
{  
    //Make background slide to the left a small amount.  
    sky.Scroll( -0.002, 0 )  
  
    //Slide croc to left and move dino downwards.  
    croc.x = croc.x - 0.004  
    dino.y+= 0.001  
  
    if( gfx.IsOverlap(dino, croc, 0.03) && !gameOver )  
    {  
        Crash()  
    }  
}
```

Now our dino will drift downwards towards the ground making it harder to avoid the crocs.

Lesson 12 – Scoring

Now we can avoid the croc we should do two things: add more crocs and keep track of how many we have avoided. To do this we can detect when the croc has gone off the edge of the screen and reset its position to the other end of the screen, and when this happens, add 1 to a score variable.

To do this first we need to add some text to the game to display the score. There are two font files, (Desyrel.png and Desyrel.xml) that must be saved in the same folder as the images. You will need to select the file type as “All Files” otherwise you will not be able to find it.

The score variable must be outside the function – making it a ‘global variable’ which means any part of the program can see its value.

```
//Global variables
var gameOver = false
var score = 0
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )

    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )
    croc = gfx.CreateSprite( "Img/crocs_149x686x1x2", "enemies" )
    crash = gfx.CreateSpriteSheet( "Img/dino_crash_920x138x4" )

    txt = gfx.CreateText( score.toString(), 0.1, "Img/Desyrel.xml" )

    //Create sounds.
    jungle = gfx.CreateSound( "Snd/jungle.mp3" )
    crunch = gfx.CreateSound( "Snd/crunch.mp3" )
}

//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Add dino image
    gfx.AddSprite( dino, 0.1, 0.5, 0.15 )

    //Add enemy sprites.
    gfx.AddSprite( croc, 0.8, 0.55, 0.12 )

    //Start animations
    dino.Play( 0, 0.25 )
    croc.Play( 0, 0.05 )

    //Add score text
    gfx.AddText( txt, 0, 0 )

    //Play looping background sound.
    jungle.Play(true)

    //Start the game.
    gfx.Play()
}
```


This next section of code makes the crocs reappear at the right edge of the screen once they have disappeared and adds one to the score each time this happens.

```
//Update game objects.
//(called for every frame)
function OnAnimate( time, timeDiff )
{
    //Make background slide to the left a small amount.
    sky.Scroll( -0.002, 0 )

    //Slide croc to left and move dino downwards.
    croc.x = croc.x - 0.004
    dino.y+= 0.001

    if( gfx.IsOverlap(dino, croc, 0.03) && !gameOver )
    {
        Crash()
    }

    //If croc has gone off left side of screen.
    if( croc.x < -croc.width )
    {
        //Move the croc back to the right hand side.
        croc.x = 1

        //Scale croc with a random value.
        var height = 0.4 + Math.random()*0.4
        croc.SetSize( null, height )
        croc.y = 1 - croc.height
        score++
        txt.SetText( score.toString() )
    }
}
```

Lesson 13 – Game Over!

To make more of a dramatic end to the game, a simple way to add a game over screen is to add a 'game over' image and a game over sound effect to the assets.



```
//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )
    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )
    croc = gfx.CreateSprite( "Img/crocs_149x686x1x2", "enemies" )

    gameover = gfx.CreateSprite( "Img/gameover.jpg", "gameover" )

    crash = gfx.CreateSpriteSheet( "Img/dino_crash_920x138x4" )
    txt = gfx.CreateText( score.toString(), 0.1, "Img/Desyrel.xml" );
    //Create sounds.
    jungle = gfx.CreateSound( "Snd/jungle.mp3" )
    crunch = gfx.CreateSound( "Snd/crunch.mp3" )
    end = gfx.CreateSound( "Snd/game_over.mp3" )
}
```

All we then need to do is add the game over image over the top of everything else when the dino crashes. To do this we create a new function called "GameOver()" do add the image and play the sound and call this function 1 second after the dino crashes.

The "setTimeout()" function calls another function after a specified delay in milliseconds.

```
//Crash our character and end the game.
function Crash()
{
    //Switch sprite sheets and change play speed.
    dino.SetSpriteSheet( crash )
    dino.Play( 0, 0.1 )

    //Play crunch sound and prevent scroll.
    crunch.Play()
    setTimeout(GameOver, 1000)
    //Set gameover state.
    gameOver = true
}

function GameOver()
{
    gfx.AddSprite( gameover, 0, 0, 1, 1 )
    //Play 'game over' sound after half a second.
    end.Play( false, 500 )
}
```

Everything so far!

By now we should have a very simple playable game. The full code is given here showing everything we have learned so far:

```
//Global variables
var gameOver = false
var score = 0
//Handle game loading.
function OnLoad()
{
    //Create background (and stretch it to full screen).
    sky = gfx.CreateBackground( "Img/flappy_back.jpg", "stretch" )

    //Create graphical objects.
    dino = gfx.CreateSprite( "Img/dino_fly_783x136x3x1", "dinos" )
    croc = gfx.CreateSprite( "Img/crocs_149x686x1x2", "enemies" )
    crash = gfx.CreateSpriteSheet( "Img/dino_crash_920x138x4" )
    txt = gfx.CreateText( score.toString(), 0.1, "Img/Desyrel.xml" )

    gameover = gfx.CreateSprite( "Img/gameover.jpg", "gameover" )

    //Create sounds.
    jungle = gfx.CreateSound( "Snd/jungle.mp3" )
    crunch = gfx.CreateSound( "Snd/crunch.mp3" )
    end = gfx.CreateSound( "Snd/game_over.mp3" )
}

//Called when game has loaded.
function OnReady()
{
    //Set background.
    gfx.AddBackground( sky )

    //Add dino image
    gfx.AddSprite( dino, 0.1, 0.5, 0.15 )
    //Add enemy sprites.
    gfx.AddSprite( croc, 0.8, 0.55, 0.12 )

    //Start animations
    dino.Play( 0, 0.25 )
    croc.Play( 0, 0.05 )

    //Add score text
    gfx.AddText( txt, 0, 0 );

    //Play looping background sound.
    jungle.Play(true)

    //Start the game.
    gfx.Play()
}
```

(Continued on the following page)

```

/Update game objects.
//(called for every frame)
function OnAnimate( time, timeDiff )
{
    //Make background slide to the left a small amount.
    sky.Scroll( -0.002, 0 )

    //Slide croc to left and move dino downwards.
    croc.x = croc.x - 0.004
    dino.y+= 0.001

    if( gfx.IsOverlap(dino, croc, 0.03) && !gameOver )
    {
        Crash()
    }

    //If croc has gone off left side of screen.
    if( croc.x < -croc.width )
    {
        //Move the croc back to the right hand side.
        croc.x = 1

        //Scale croc with a random value.
        var height = 0.4 + Math.random()*0.4
        croc.SetSize( null, height )
        croc.y = 1 - croc.height
        score++
        txt.SetText( score.toString() )
    }
}

//Handle screen touches and key presses.
function OnControl( touchState, touchX, touchY, keyState, key )
{
    //Increase dino's upward velocity when screen touched.
    if( touchState=="Up" && !gameOver ) dino.y-= 0.03
}

//Crash our character and end the game.
function Crash()
{
    //Switch sprite sheets and change play speed.
    dino.SetSpriteSheet( crash )
    dino.Play( 0, 0.1 )

    //Play crunch sound and prevent scroll.
    crunch.Play()
    setTimeout(GameOver, 1000)
    //Set gameover state.
    gameOver = true
}

function GameOver()
{
    gfx.AddSprite( gameover, 0, 0, 1, 1 )
    //Play 'game over' sound after half a second.
    end.Play( false, 500 )
}

```

End of Part 1, see Part 2 for more advanced game features such as physics!