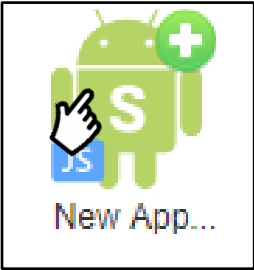
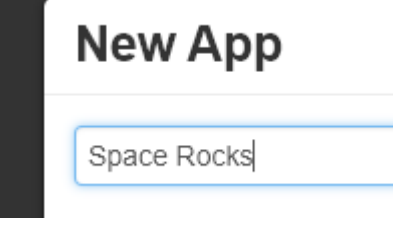
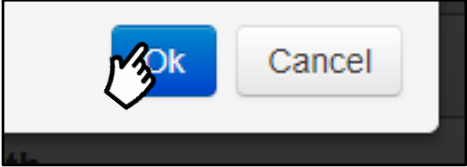


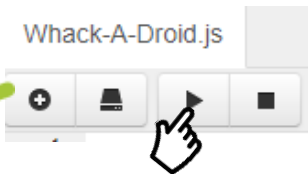
Space Rocks



Welcome to the Space Rocks Tutorial! Read on to learn about **at Images, Backgrounds, Intervals, timers, plugins, the micro:bit, Bluetooth controls, explosions and displaying a score**

Step 1 – Create your new App

<p>Create a new JavaScript App</p> 	<p>Call it Space Rocks</p> 	<p>Click OK</p> 
---	---	--



If you **Run** your new App.

You should see it says **Hello** on the screen.

Press back arrow or the 'Esc' key to exit the app.

Step 2 – Set the Score



Find the word **"Hello"** and change it to **0**.

Add a **new line** of code below the line that says 'Set textSize'.

Your code should now look like this!

```
//Called when application is started.
function OnStart()
{
    //Create a layout with objects vertically centered.
    lay = app.CreateLayout( "linear", "VCenter,FillXY" );

    //Create a text label and add it to layout.
    txt = app.CreateText( "0" );
    txt.SetTextSize( 32 );
    txt.SetTextColor( "yellow" );
    lay.AddChild( txt );

    //Add layout to app.
    app.AddLayout( lay );
}
```

Step 3 – Find some Images....



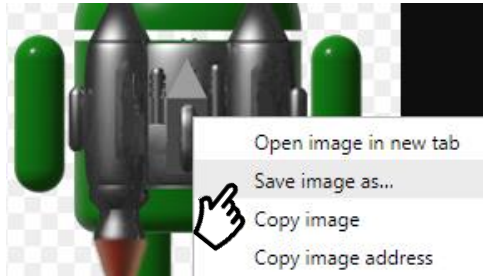
We need to download some **images** for our game

They can be found at - <http://androidscript.org/tutorials/SpaceRocks/Images>

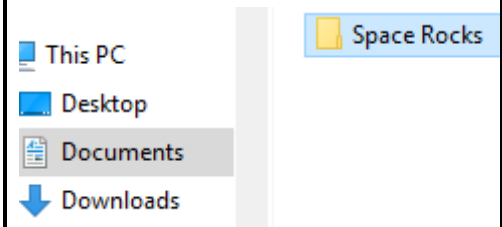
Click on player.png

Icon	Name
[DIR]	Parent Directory
[IMG]	baddy.png
[IMG]	player.png
[IMG]	rock.png
[IMG]	space.jpg

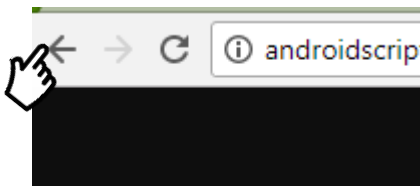
Right Click on the Picture and then click Save Image as...



Click OK



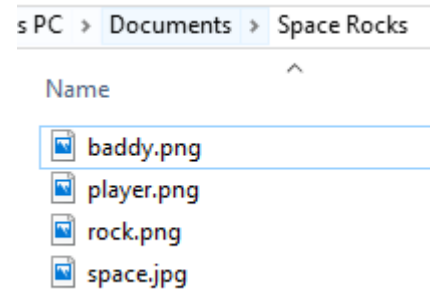
Click Back on your Browser



Now do the same for baddie.png, rock.png and space.jpg

Icon	Name
[DIR]	Parent Directory
[IMG]	baddy.png
[IMG]	player.png
[IMG]	rock.png
[IMG]	space.jpg

Until you have them all saved!

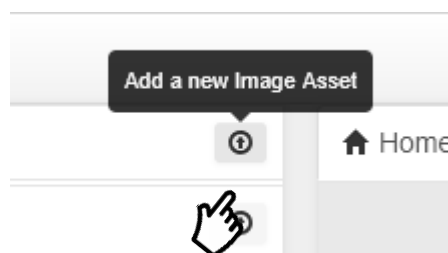


Now we need to **add them to our project**.

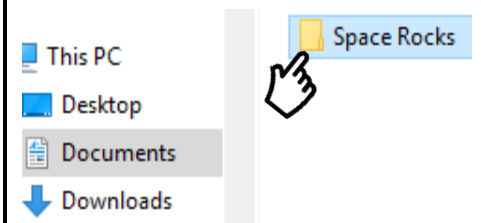
Click on the icon to go to the Assets section



Right Click on the Picture and then click Save Image as...



Find the folder, add the player, rock and space Images



You should now see them in the **Images** section



Step 4 – Change the Background

Add a new **SetBackground** line below the line that says **'CreateLayout'** like this:-

Now run your app.



```
function onStart()
{
    //Create a layout with objects vertically centered.
    lay = app.CreateLayout( "linear", "VCenter,FillXY" );

    lay.SetBackground( "Img/space.jpg" );
}
```

Step 5 – Add a Player

Find the word **"Linear"** in your program and change it to **"Absolute"**.

Type these 3 lines below the line that says **'AddChild'**

Your complete code should now look like this:-



```
function onStart()
{
    //Create a layout with objects vertically centered.
    lay = app.CreateLayout( "Absolute", "VCenter,FillXY" );

    lay.SetBackground( "Img/space.jpg" );
    //Create a text label and add it to layout.
    txt = app.CreateText( "Hello" );
    txt.SetTextSize( 32 );
    lay.AddChild( txt );

    //Create a player image.
    imgPlayer = app.CreateImage( "Img/player.png" );
    lay.AddChild( imgPlayer );

    //Add layout to app.
    app.AddLayout( lay );
}
```



Now **Run your App**, You should see a player Image.

Step 6 – Connect to the BBC micro:bit



Before we add the code to our DroidScript project, we need to set up the micro:bit to send messages to our device via Bluetooth.

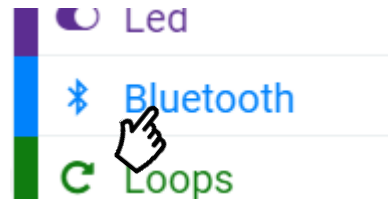
First you need to go to <https://microbit.org/code/>

At the top of the page, in the Section about **JavaScript Blocks Editor**, click **Let's Code**

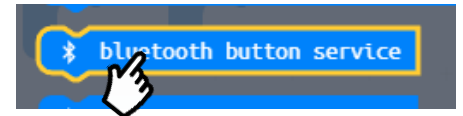
JavaScript, along with peer-to-peer radio. F



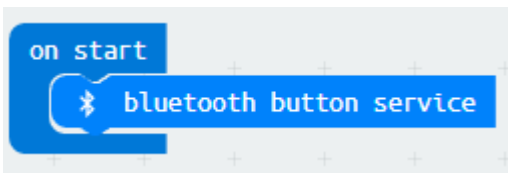
Once the editor loads open the **Bluetooth** block selection



Select the **Bluetooth button service**.



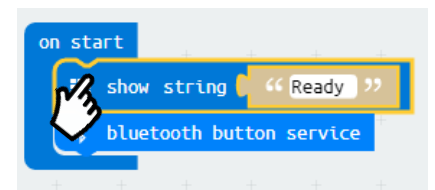
Drag it into the **on start** block in your editor



Inside the **Basic** blocks, grab the **show string** command



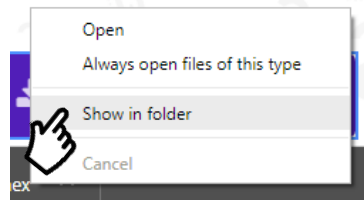
Drag it above the Bluetooth block in your **on start** function and change **Hello** to **Ready**



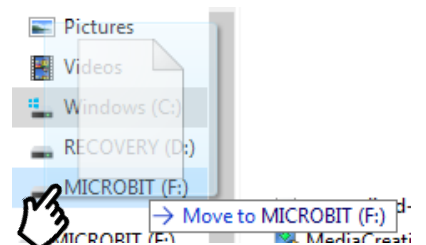
Now **connect your micro:bit with a micro USB** to your computer, and **download your script**.



Right click on the download and click **Show in folder**.

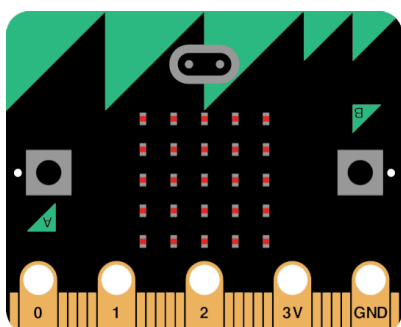


Drag the file onto your **micro:bit**



Now we are all set up to code the button presses into our App.

Head back over to the wi-fi Editor for the next part!

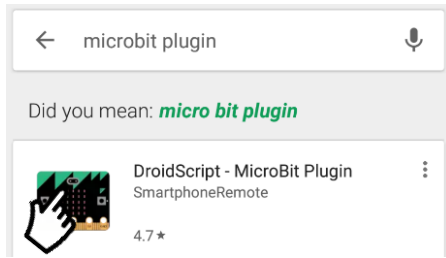


To use the **BBC micro:bit**, we must load a plugin, so add the following line to the very top of your code:-

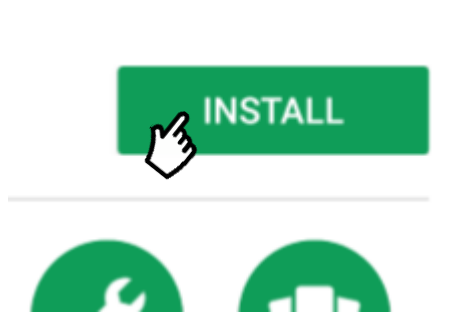
```
app.LoadPlugin( "MicroBit" );
```



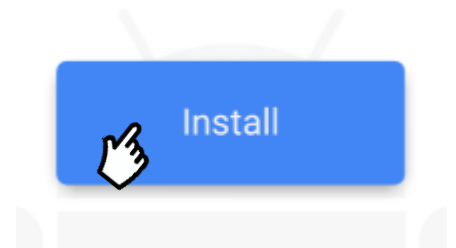
Open **Google Play Store** on your device, and search for **microbit plugin**



Install and **Open** the plugin App.



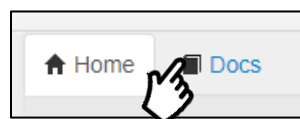
Press **Install**



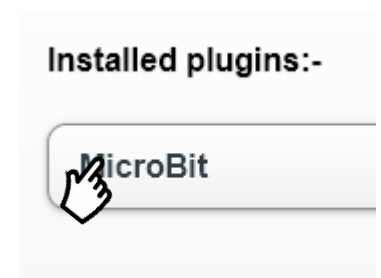
Restart DroidScript on your device. Once it has reopened press this **icon** in the top right of the Editor



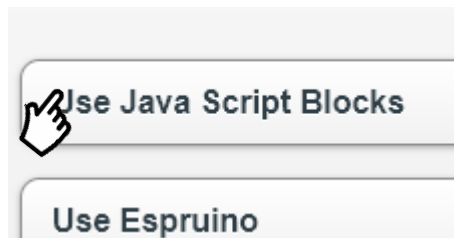
Open the **Docs** to the **Plugin Section**



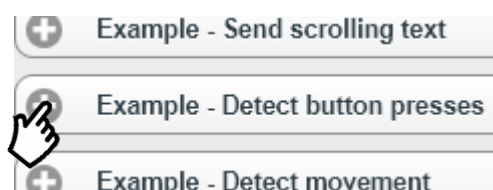
Click on the **micro:bit Plugin**



Click the **Use JavaScript Blocks** Section



Locate the **Detect button presses** section



Highlight and Copy the bold section

```
app.AddLayout( lay );  
  
microbit = app.CreateMicroBit();  
microbit.SetOnConnect( OnConnect );  
microbit.SetOnButton( OnButton );  
microbit.Scan();  
}  
  
function OnConnect()
```



Paste the text into your code below the line that says 'AddLayout' (use the button at the top of the screen to paste).

Your code should now look something like this:-

```
//Add layout to app.  
app.AddLayout( lay );  
  
microbit = app.CreateMicroBit();  
microbit.SetOnConnect( OnConnect );  
microbit.SetOnButton( OnButton );  
microbit.Scan();  
}
```



Important: Please make sure you re-pair with your micro-bit each time a new hex file is installed and then press the reset button after the pairing tick is shown (the reset button is on the back of the micro:bit).

To pair with your micro-bit, hold down buttons A and B at the same time and briefly press the reset button (on the back) while still holding down A and B. You should then see the word 'Pairing' appear on the LEDs. You can then pair with the micro-bit using the Android bluetooth settings page.



Now go back to the same **micro:bit** example and copy the OnConnect and OnButton functions from the example and **paste them into your code at the very bottom.**

You should now have **two functions** at the bottom of your code like this:

```
function OnConnect()  
{  
    microbit.SetOnButton( OnButton );  
}  
  
function OnButton( name, state )  
{  
    txt.SetText( name + " : " + state );  
}
```



Now run your app. You should be able to connect to the **micro:bit** when it appears in the scan list.

Once connected, try **pressing the micro:bit buttons**. What happens?

Step 7 – Take Control.

Now we are going to use the **setInterval** function to create a timer that **updates the screen every 20 milliseconds** (so we can move our player around).

Change the **OnConnect** function to look like this:-

```
function OnConnect()
{
    microbit.SetOnButton( OnButton );

    y = 0.7; x = 0.4; direction = 0;
    updateTimer = setInterval( Update, 20 );

    app.SetDebugEnabled( false );
}

function OnButton( name, state )
{
    if( state==1 )
    {
        if( name=="A" ) direction = -1;
        else if( name=="B" ) direction = 1;
    }
    else direction = 0;
}

function Update()
{
    x += direction * 0.01;
    imgPlayer.SetPosition( x, y );
}
```

Now let's make our **player move when the buttons are pressed.**

Change the **OnButton** function to look like this:-

We get 2 pieces of info back from the button...

It's **name**, either **A** or **B**.

And the **state**, **0** or **1**. This tells us whether the button was pressed down or released.

What will happen to the variable direction when the buttons are

Then we need to create a **function** called 'Update' which will be called by the timer every **20 milliseconds**.

Add a this new function at the very bottom of your program:-

Run your app and see what happens when you press the buttons

Step 8 – Check your Code



Your complete code should now look something like this:-

```
app.LoadPlugin( "MicroBit" );

//Called when application is started.
function OnStart()
{
    //Create a layout with objects vertically centered.
    lay = app.CreateLayout( "Absolute", "VCenter,FillXY" );

    lay.SetBackground( "Img/space.jpg" );

    //Create a text label and add it to layout.
    txt = app.CreateText( "0" );
    txt.SetTextSize( 32 );
    lay.AddChild( txt );

    //Create a player image.
    imgPlayer = app.CreateImage( "Img/player.png" );
    lay.AddChild( imgPlayer );

    //Add layout to app.
    app.AddLayout( lay );

    microbit = app.CreateMicroBit();
    microbit.SetOnConnect( OnConnect );
    microbit.SetOnButton( OnButton );
    microbit.Scan();
}

function OnConnect()
{
    microbit.SetOnButton( OnButton );

    y = 0.7; x = 0.4; direction = 0;
    updateTimer = setInterval( Update, 20 );

    app.SetDebugEnabled( false );
}

function OnButton( name, state )
{
    if( state==1 )
    {
        if( name=="A" ) direction = -1;
        else if( name=="B" ) direction = 1;
    }
    else direction = 0;
}

function Update()
{
    x += direction * 0.01;
    imgPlayer.SetPosition( x, y );
}
```


Step 9 – Add Some Rocks

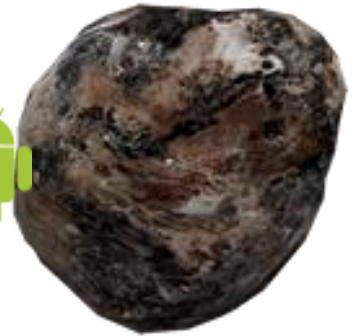


Now we need to **create another timer** to **generate rocks** for our player to dodge, so add the following code to your `OnConnect` function just **below the line** that says 'SetInterval':-

```
updateTimer = setInterval( Update, 20 );  
  
rocks = [];  
rockTimer = setInterval( AddRock, 1000 );
```

Tip: When typing the name of a known variable try writing the first couple of letters and then pressing the <Alt> and <Space> keys at the same time. It will give you a list of variable names to choose from.

Create a **new function at the bottom of your code** called `AddRock` that will get called by our **rockTimer every 1000 milliseconds** like this:-



```
function AddRock()  
{  
    var size = 0.1 + 0.1 * Math.random();  
    var imgRock = app.CreateImage( "Img/rock.png", size );  
    rocks.push( imgRock );  
  
    imgRock.x = Math.random();  
    imgRock.y = 0;  
    imgRock.SetPosition( imgRock.x, imgRock.y );  
  
    lay.AddChild( imgRock );  
}
```



Now **run your app** and see what happens.

Step 10 – Rock Fall



Next we need to make the **rocks start falling downwards**, so edit the 'Update' function to look like this:-

Then **Run your App**. Do the rocks fall?

What would happen if we added 0.02 to rock.y instead?



```
function Update()
{
    x += direction * 0.01;
    imgPlayer.SetPosition( x, y );

    for( r in rocks )
    {
        rock = rocks[r];
        rock.y += 0.01;
        rock.SetPosition( rock.x, rock.y );

        if( rock.y > 1 )
        {
            lay.DestroyChild( rock );
            Arr.remove( rocks, rock );
        }
    }
}
```

Step 11 – Collision



At the moment the rocks just pass straight through our player, so we need to make this more interesting by detecting when **a rock collides (overlaps) with our player image**.

Type the following code into the 'Update' function just **below the line** that says 'rock.SetPosition':-

```
rock.SetPosition( rock.x, rock.y );

if( rock.IsOverlap( imgPlayer, 0.03 ) )
{
    imgPlayer.Explode();

    app.ShowPopup( "Game Over" );
    clearInterval( updateTimer );
    clearInterval( rockTimer );
}

if( rock.y > 1 )
```

Now **run your app** and see what happens when your player touches a rock.



Step 12 – Update the Score

Add this line of code to the `OnConnect` function **after the line** that says `setInterval`:-



```
score = 0;
```

```
score++;  
txt.SetText( score );
```



Add these lines to the 'Update' function just **below the line** that says `'Arr.remove'`:-

Now **run your app**. You should see the score change when rocks pass by.

Step 13 – Play the Game and Improve it.



You've made it the end of our tutorial.

We have looked at **Images, Backgrounds, Intervals, timers, plugins, the micro:bit, Bluetooth controls and displaying a score**

Now play the game for a while and enjoy your hard work! While your playing though there are a few more things to think about. If you want to add some more stuff see the next page for our **Bonus Tasks**

- Is the game too easy or too hard? See if you can work out how to make it harder or easier by changing some of the numbers.
- Can you work out how to change the rock size?
- How about changing the speed of the rocks?
- Can you make your player larger?

Congratulations you are finished.

Well done!

Bonus Tasks

Task 1 – Spin the Rocks

We need to create a property to store the spin angle of each rock, so add the this line of code to the 'AddRock' function, below the line that says 'imgRock.y ='



```
imgRock.y = 0;  
  
imgRock.angle = 0;
```

```
rock.SetPosition( rock.x, rock.y );  
  
rock.angle += 10;  
rock.Rotate( rock.angle );
```

Next we need to make the rock rotate a little each time the screen is updated, so add the following two lines of code to the 'Update' function, just below the line that says 'rock.SetPosition'.



Now run your app. Do you see the rocks spinning?
Try adjusting the amount added to the angle and see what happens.
Can you work out how to make the spin rate of each rock random?

Task 2 – Use the Motion Sensor

1. Go back to the MicroBit plugin documentation.
2. Find and expand the 'Detect movement' example.

See if you can work out how to use the motion sensor to move your player instead of the buttons.

