# HTML 5 and JS Game Programming
# Introducing Programming Concepts

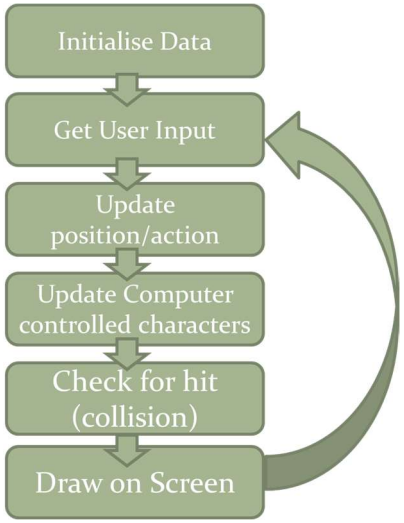
# Contents

## Programming Overview

All programs come down to a few basic fundamental processes:

- Input – Getting data from the user, whether that is from the keyboard, mouse, webcam or from a range of other devices.
- Output – Sending data back to the user through the screen, printer, speakers or again many other output devices
- Assignment – Data can be stored within the memory using a range of different types – Whole numbers, real numbers, characters etc
- Selection – Are the decision-making instructions the most common being the IF statement
- Iteration – The main reason for using computers, repetition! Sequences of instructions can be repeated over again.

Games tend to follow the same loop using a range of the above processes



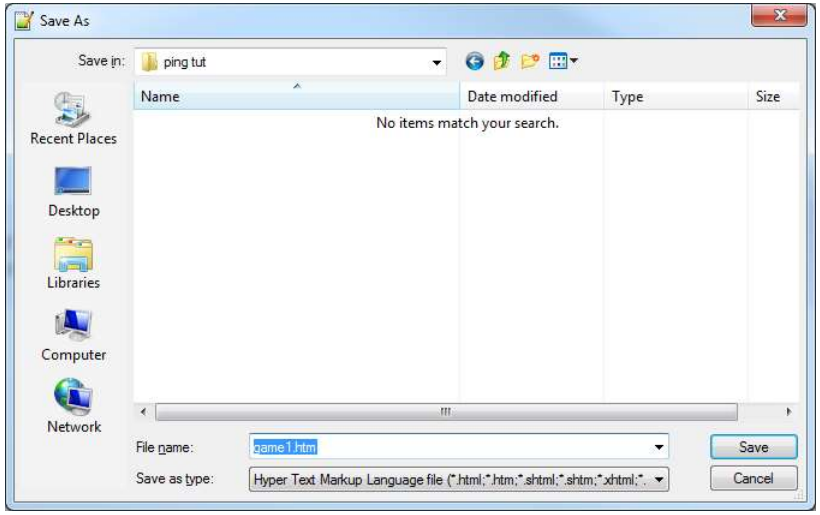

**Game On!**

## Setting the stage

First we need to make a simple webpage to host our game, the following code does just that!

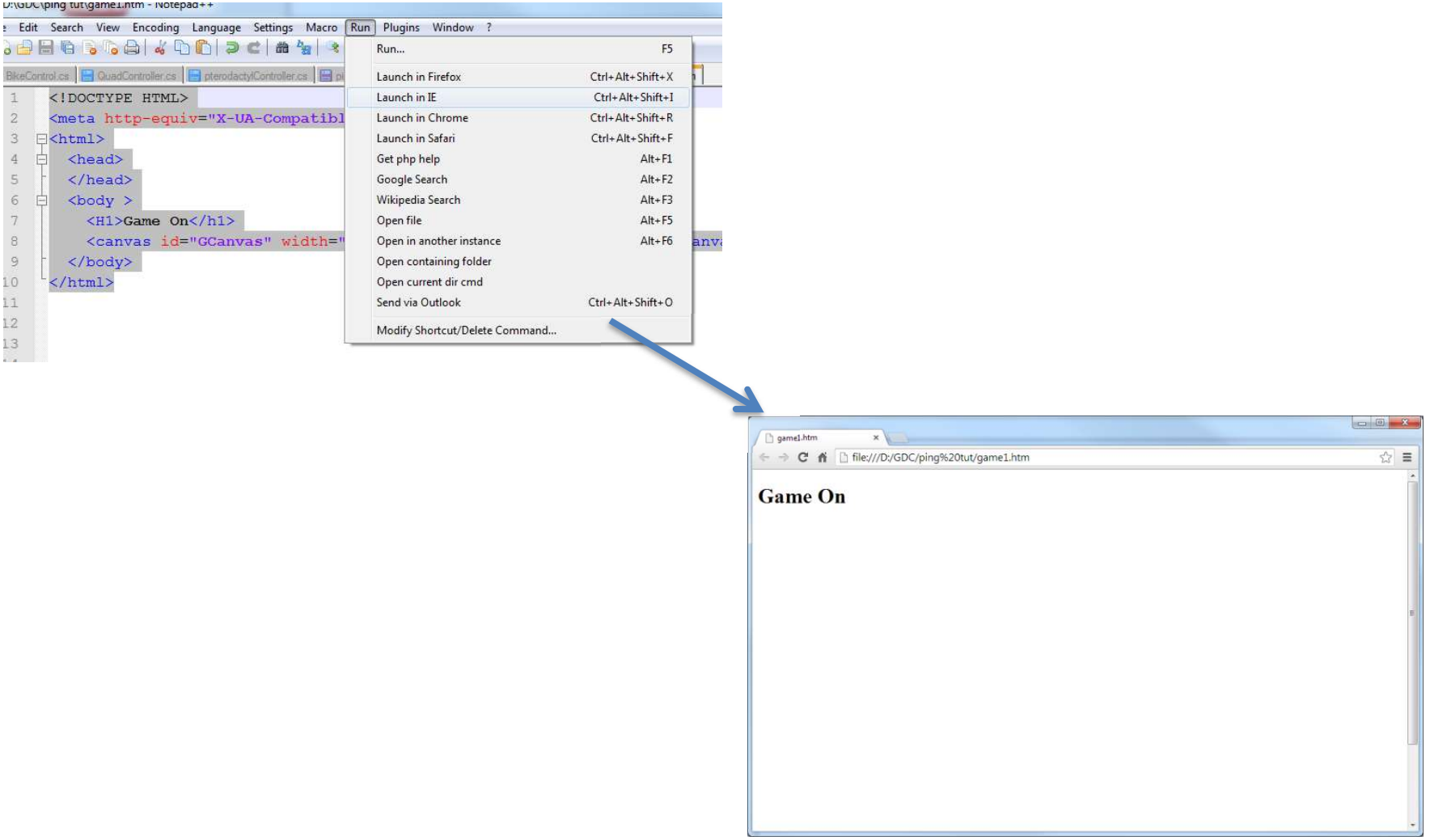
```
<!DOCTYPE HTML>
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<html>
  <head>
  </head>
  <body >
    <H1>Game On</h1>
  </body>
</html>
```

Let's save and test the simple webpage, go to File → Save as and call it game1.htm (***do not forget*** to add the ***.htm*** (or .html) or it will not work!)



## Running your game

If you are using Notepad++ you can simply go to the **Run** menu and select **Launch in IE** or whichever browser you prefer



## Creating the canvas

To get our game on we need to create a canvas using the HTML5 tag <canvas> and <script> tags to draw all of our game objects to.

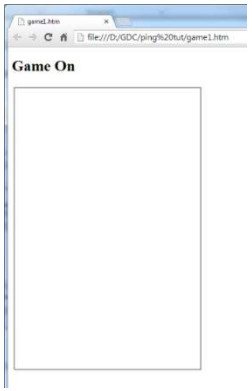
Add the blocked code

```
<!DOCTYPE HTML>
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<html>
  <head>
  </head>
  <body >
    <H1>Game On</h1>

    <canvas id="GCanvas" width="400" height="600">No HTML5 Support</canvas>
    <script>
        //Game canvas
        var canvas = document.getElementById('GCanvas');
        var GC = canvas.getContext('2d'); //GC = Game Canvas (what we draw on
        GC.strokeRect( 5, 5, 390, 590 );
    </script>

  </body>
</html>
```

New lines

Save and run just to check that you can see a rectangle on the screen.

*If it didn't work check your spelling and use of upper case and lower case letters.*



## Setting up the game functions

To make things easier later we are going to add a few functions now

1. Delete the GC.strokeRect line
2. Add the **onLoad** command inside the ***<body>*** tag
3. Add all of the lines following and including ***//Global Variables***

```
<body onload = setupGame()>
  <H1> Game On</h1>
  <canvas id="GCanvas" width="400" height="600">No HTML5 Support</canvas>
  <script>
      //Game canvas
      var canvas = document.getElementById('GCanvas');
      var GC = canvas.getContext('2d'); //GCC = Game Canvas (what we draw on

      //Global Variables

      //game functions
      function setupGame()
      {
          timerID = setInterval("mainGame()", 30); //run the mainGame
      }

      function gameLoop(){}
      function intro(){}
      function outro(){}
      function mainGame(){}

  </script>
</body>
```

The ***setupGame function*** tells our game to run the main game function at 30 frames per second (fps)

# Downhill Skiing

Let's make our game!!!

## Variables

First we need to know where we are going to draw our skier The computer needs to know where (position) and how big (size/scale)
Add all of the variables after the //Global variables comment[1]

```
        //Global Variables
        var screenWidth = 400; //must match how big the canvas is!
        var screenHeight = 600;

        var skierX = screenWidth/2;    //X Y Coordinates - start in the centre
        var skierY = screenHeight/2;


        //game functions
```

This code will place our 'skier' in the centre of the screen

If you save and run now, you will notice… it does nothing. That is because you haven't told it to draw anything yet

[1] Comments are there to help you, but the computer will ignore those lines. Comments can be written by adding //

## Drawing the 'Skier'

Go to the ***mainGame()*** function and the highlighted code inside of the braces (the curly brackets)[2]

```
function mainGame()
        {
            //DRAW ALL GAME STUFF

            //clear screen and draw outline
            GC.clearRect(0,0,screenWidth,screenHeight);
            GC.strokeRect(0,0, screenWidth,screenHeight);

            GC.fillRect(skierX, skierY, 20,20);

        }
```

- We are just using a cube to act as the skier for now!

Save and run ☺

[2] In programming we use brackets ()also known as parenthesis , square brackets [] and braces {}; Make sure you use the correct ones!

## Keyboard Input

In the global variables section after the skier coordinates we need to add Keyboard Listeners to check if a key is being pressed or released

```
        var skierX = screenWidth/2;    //X Y Coordinates - start in the centre
        var skierY = screenHeight/2;

        //Keyboard Input Listener
        var keyState = {};
        window.addEventListener('keydown',function(e){keyState[e.keyCode || e.which] = true;},true);
        window.addEventListener('keyup',function(e){keyState[e.keyCode || e.which] = false;},true);
```

In the ***mainGame*** Function we use if statement to check which keycode is being generated.

```
function mainGame()
        {
            //Get Player input
            if(keyState[37]){skierX-=10;}//left key
            if(keyState[39]){skierX+=10;}//right key


            //DRAW ALL GAME STUFF
```

Full list of Keycodes: https://www.cambiaresearch.com/articles/15/javascript-char-codes-key-codes

Save and run ☺

**Your Tasks**

- Make the skier move up and down
- Stop the skier from moving out of the screen
- Can you add a second player?

## Working with images

Images must stored in the same folder (or sub folder) as the webpage itself

We will be using a spritesheet (a collection of small images on one image)

In setupGame we need to create an image and load a png into it.

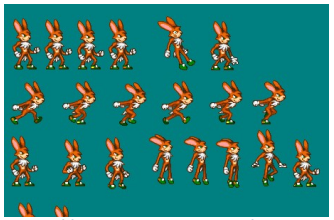
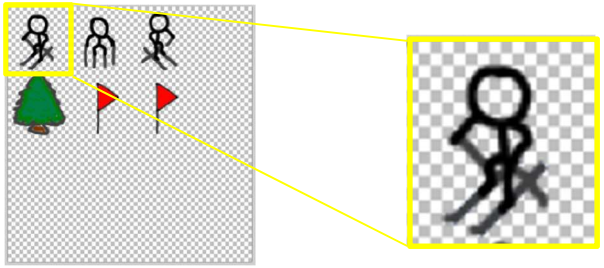


*Figure 1 https://opengameart.org/content/surge-of-opensurge-for-ultimate-smash-friends*

```
        //game functions
        function setupGame()
        {
            timerID = setInterval("mainGame()", 30); //run the mainGame

            //Create and load image (png)
            image1 = new Image();
            image1.src = "skier.png";
        }
```

In mainGame we can now draw a cropped part of the full image to the screen

```
    GC.fillRect(skierX, skierY, 20,20);
    GC.drawImage( image1, 0, 0, 64, 64, skierX, skierY, 64, 64 ); //image, source, destination (crop and scale)
```



Changing the first number will start drawing the image 64 pixels in.

```
GC.drawImage( image1, 64, 0, 64, 64, skierX, skierY, 64, 64 );
```

***Tasks:***

- ***Experiment with the numbers to see how it affects how the image is displayed on screen***
- ***Draw a different sprite when pressing left or right.***
- ***Add a background image that draw behind the skier***

## Scrolling Backgrounds

To make it look lie the skier is moving down hill we simply move a background image up! To make it look continuous we use 2 images.

1. Create 2 variable to hold the background y position (we don't need an 'x' position as it will be zero and will not change ). Add them after the skier coordinates

```
        var skierX = screenWidth/2; //X Y Coordinates - start in the centre
        var skierY = screenHeight/2;

        var bgd1Y = 0;
        var bgd2Y = 512;
```

*My image is 512 x 512 which is why the second background is set to start at 512

2. Create and load a background image in the setupGame function

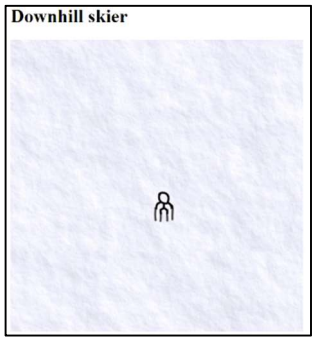
```
        //game functions
        function setupGame()
        {
            timerID = setInterval("mainGame()", 30); //run the mainGame

            //Create and load image (png)
            image1 = new Image();
            image1.src = "skier.png";

            bgdImage = new Image();
            bgdImage.src = "snow.jpg";

        }
```

3. Add the code to make the backgrounds move up the screen and automatically go to the bottom when they have gone off the screen

```
        function mainGame()
        {
              //Get Player input
               if(keyState[37]){skierX-=10;}//left key
               if(keyState[39]){skierX+=10;}//right key

               bgd1Y-=5;if(bgd1Y<-500){bgd1Y=500;}
               bgd2Y-=5;if(bgd2Y<-500){bgd2Y=500;}
```

4. Now to draw the backgrounds – After you have cleared the background, draw you images. The order you draw is very important

```
               //DRAW ALL GAME STUFF
               //clear screen and draw outline

               GC.clearRect(0,0,screenWidth,screenHeight);
               GC.strokeRect(0,0, screenWidth,screenHeight);
               GC.drawImage( bgdImage, 0,bgd1Y);
               GC.drawImage( bgdImage, 0,bgd2Y);
```

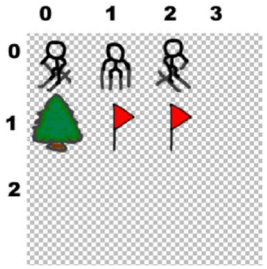



***The Game so far!***

***Tasks:***

- *Try removing one of the background images so you can see what is happening.*
- Change the speed of the scrolling backgrounds

## More Sprites




We want the sprite to change image based on which direction you are moving.

1. Back in ***Global Variables*** we need to add a new variable to store which number sprite we are using.

```
var skierX = screenWidth/2; //X Y Coordinates - start in the centre
var skierY = screenHeight/2;
var spriteNum = 1;
```

* spriteNum = 1 will give us the 2nd sprite

2. In the mainGame function we need to edit the drawImage line which draw the sprite. The Main sprite starts at 1*64 pixels across.

```
GC.drawImage( image1, spriteNum*64, 0, 64, 64, skierX, skierY, 64, 64 ); //image, source, destination
```

3. We want it to change if we are pressing left or right. We start by resetting the spriteNum to 1 each frame and then change if the key is pressed

```
function mainGame()
{
        //Get Player input
        spriteNum = 1;
        if(keyState[37]){skierX-=10; spriteNum = 0;}//left key
        if(keyState[39]){skierX+=10; spriteNum = 2;}//right key
```

## Trees

Trees are the enemy in this simple game, we want many trees to we will create an ***Array*** of trees. Each tree has an X and Y coordinate, Javascript lets us create a tree structure.

1. Again… Back in the Global Variables section, we create a simple variable (structure) called trees, we then use a for loop to add (push) 10 more trees on to out array of trees. We will randomly set the X coordinate and put them 128 pixels below each other.

```
        //Trees
        var trees=[{x:0, y:0}];
        for(t=0;t<10;t++)
        {
            var x = Math.floor((Math.random() * screenWidth) + 1);
            trees.push({x:(x), y:(t*128)});
        }
```

2. We need to draw the trees now. We already have the tree image on the spritesheet. We will again use a for loop to move through the array and draw each of our trees. Place the code just after we have drawn the skier

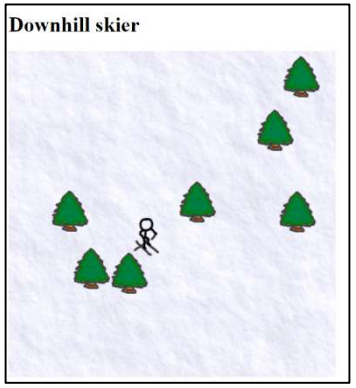
```
            GC.drawImage( image1, spriteNum*64, 0, 64, 64, skierX, skierY, 64, 64 );

            for(t=0;t<10;t++)
            {
                var tx, ty;
                tx = trees[t].x;
                ty = trees[t].y;
                GC.drawImage( image1, 0, 64, 64, 64, tx, ty, 64, 64 ); //image, source, destination (crop and scale)
            }
```

3. The Trees need to move.. yet again, one more for loop to update the position of each tree. This code needs go just have we have updated our scrolling background

```
        //Move Trees
              for(t=0;t<10;t++)
              {
                  trees[t].y-=scrollSpeed;
                  if(trees[t].y<0-32)
                  {
                      trees[t].x = Math.floor((Math.random() * screenWidth) + 1);
                      trees[t].y = 512+Math.floor((Math.random() * 100) + 1);
                  }
              }
```

4. Notice I have added a new variable called scrollSpeed to make it easier to make changes, like speeding the game up to make it harder

```
    //Background image coords - 2 allow for scrolling
        var bgd1Y = 0;
        var bgd2Y = 512;
        var scrollSpeed = 5;
```




***The Game so far!***

***Tasks:***

- *The tree disappear to soon!? Edit the code so they go completely off screen before resetting*
- Could you add different trees or other obstacles?

## Collision

Currently you can ski straight through the trees, we will now add a simple calculation to see if the skier is close to each tree.

1. In the move tree loop add the highlighted code. This code checks if you are close to a tree on both the X and Y axis

```
            //Move Trees
            for(t=0;t<10;t++)
            {
                trees[t].y-=scrollSpeed;
                if(trees[t].y<0-64)
                {
                    trees[t].x = Math.floor((Math.random() * screenWidth) + 1);
                    trees[t].y = 512+Math.floor((Math.random() * 100) + 1);
                }

                if((Math.abs(trees[t].x - skierX) <32 )&&  (Math.abs(trees[t].y - skierY)<32))
                {
                    alert("hit");
                }
            }
```

The ***Math.abs*** function makes any negative number a positive number

2. It simple results in an alert box popping up. The next section will deal with a game over.

## Intros and Outros

## Game Reset

## Score