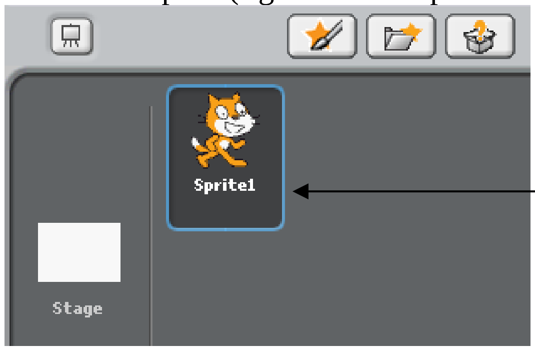
SIMPLE TRACK GAME WORKSHEET

Task 1

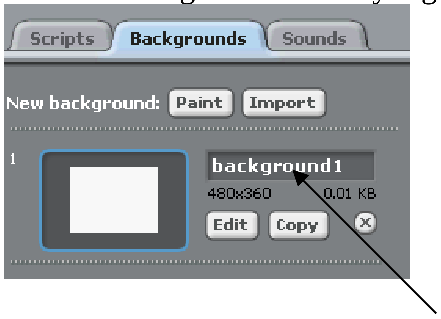
- Load Scratch software
- Delete Sprite (right click on sprite and select delete)



Mouse Right-click and select delete

Task 2

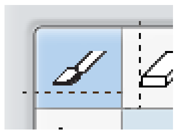
- Select Stage if not already highlighted and the press the background tab.



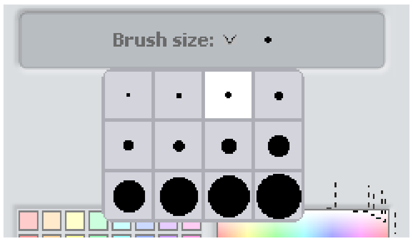
Change background name to “track1” and then press edit.

HOW TO USE THE PAINT EDITOR

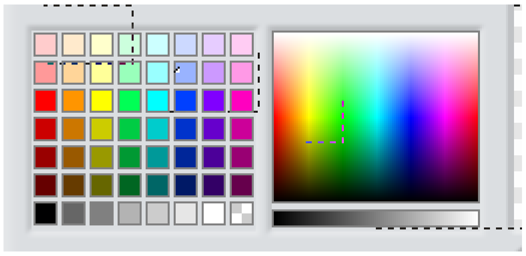
The simple way to make the track is using the paint brush tool.



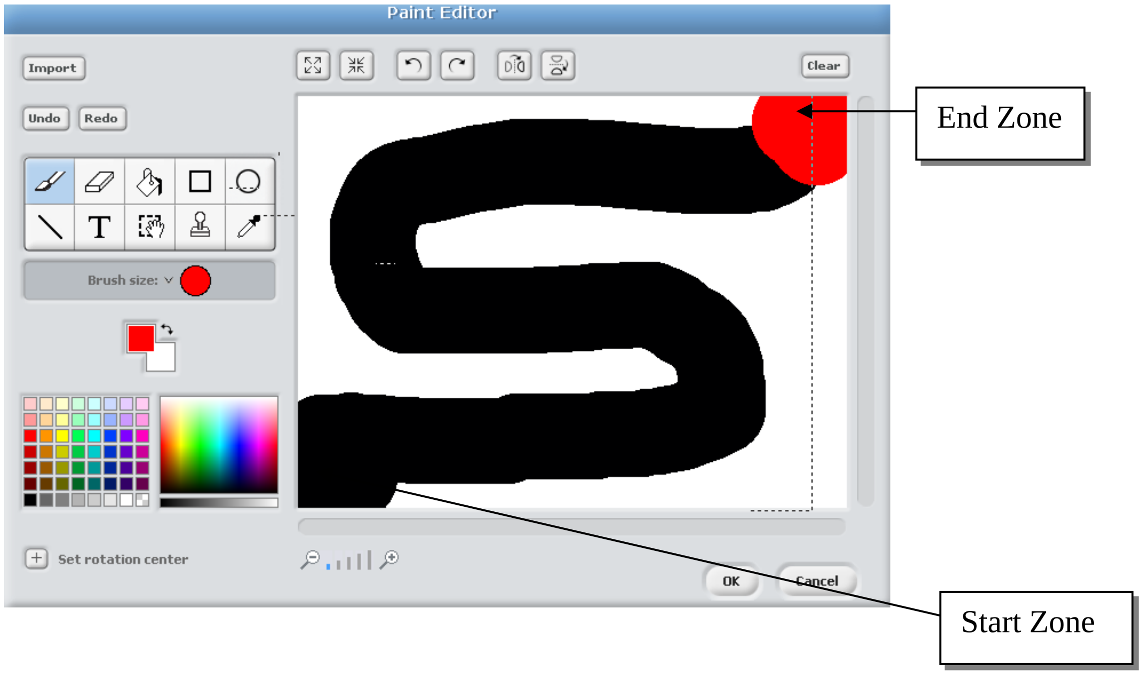
Make the brush size its largest by pressing the small down arrow next to brush size and then select the largest one.



To change the colour of your pen chose one of the colours in the colour grid.



Then draw your track to look something like below:



If you make any mistakes you can use the option – which goes back a step at a time.

If you want to start again use the button.

TRACK REQUIREMENTS

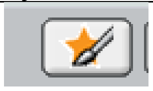
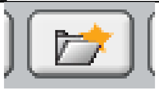
- Start Zone (same colour as track path)
- End Zone (different colour)

Once the track has been completed press OK.

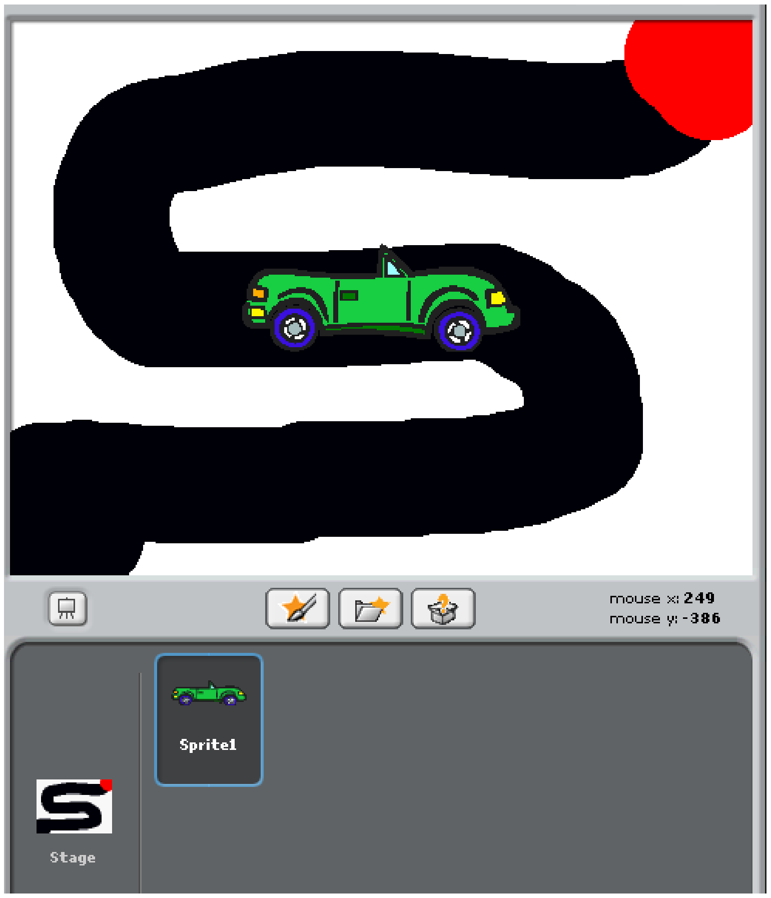
Task 3

Now we need to create a ball or similar object to run around the track. This can be anything you want, such as a ball, car, face, picture, etc. These are called “sprites”

You have two options :

Create your own sprite	Use a pre - created sprite
 **This icon will be located below your track** • Design your sprite in the paint editor • Make sure the outside colour is different from the track path or end zone colours • Press OK to save the sprite	 • **This icon will be located below your track** • Choose a folder and select a sprite • Make sure the outside colour is different from the track path or end zone colours • Press OK to select your sprite

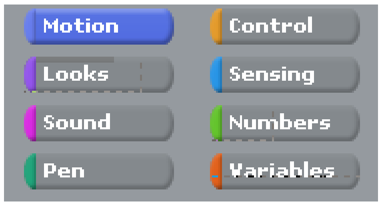
This worksheet has opted for a pre-created sprite from the transportation folder and selecting “car1”



Now we need to get the car to move!

Task 4

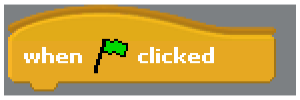
This is where the programming begins.



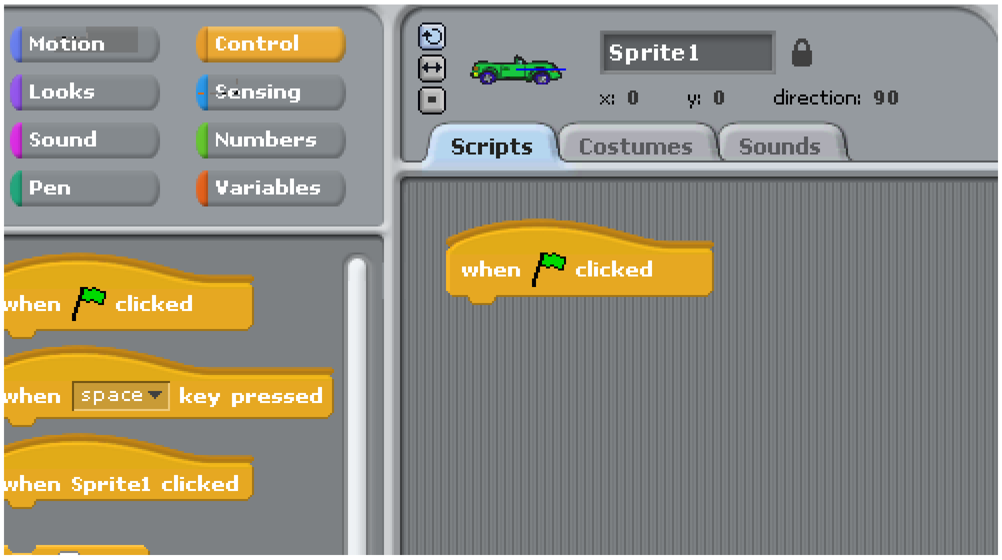
These give your key choices on what you would like the program to do.

Motion	Controls the movement, direction and position of the sprite.
Looks	Controls size, speech, colour, and different costumes for the sprite.
Sound	Controls pre-recorded and user made sounds and volume levels.
Pen	Controls the trail and pen colours – (not used that much)
Control	Main controls for the program, starting, ends, loops, pauses, etc.
Sensing	User controls such as mouse or keyboard presses.
Numbers	Number controls, such as random number generator
Variables	Allows additional programming (advanced).

Programs always have to start with a command of the "green flag". This is in the Control menu.



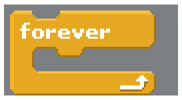
You need to drag this into the scripts area. Just left click and drag over.



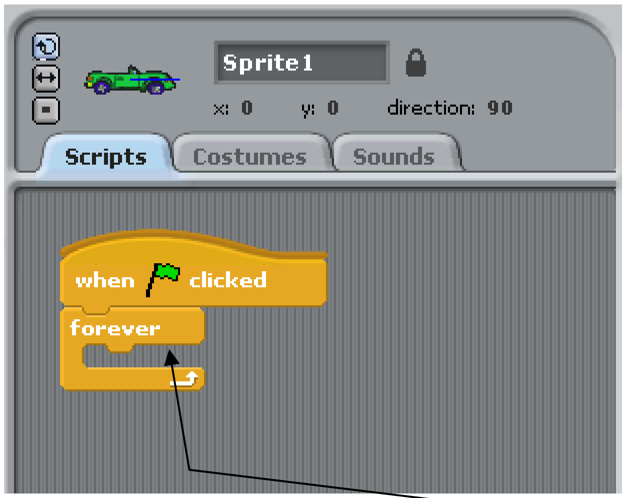
As the computer is looking for user interaction (interaction is when the person playing the game, the “user” is communication with the computer – they interact together).

When interaction is needed the program needs to continuously check for any interaction, so the program has to go over and over to check for any interaction, such as a key press. Therefore the program needs to use a “forever” statement.

This statement will run over and over again awaiting a user instruction.



This again is in the control menu and is dragged over and connects to “green flag”.

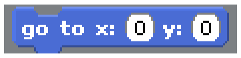


As you can see there is now a gap in the forever statement.

This is where the program looks for interaction with the sprite.

However before this we need to sort where the sprite starts and what it looks like.

The sprite needs to start at the start point. Therefore we need to position the sprite in the correct location.

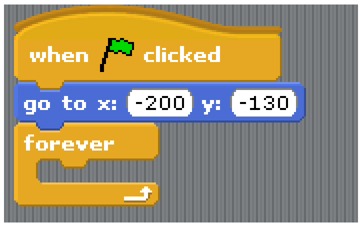
The menu controls the positioning of the sprite, so select this menu and use the  statement and drag over to between the green flag and forever loop statement.

Now to get the co-ordinates, place the mouse over the position you want to start the sprite and the co-ordinates will be show on the bottom right of the track.

You then need to add the co-ordinates to the statement, by mouse clicking the numbers on the statement.

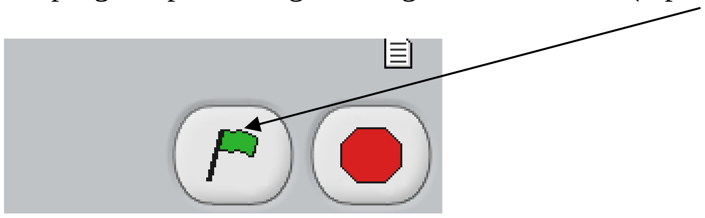
If your start zone is on bottom left your co-ordinates would be about -200 and -130.

(Please note these are negative numbers)



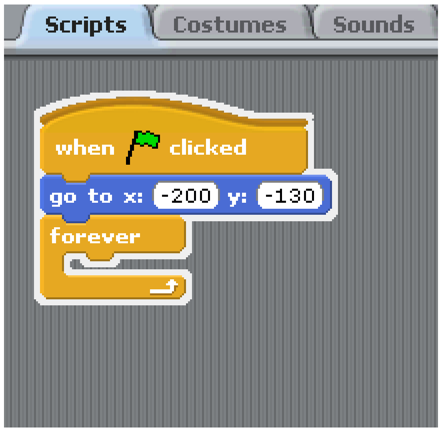
```
when [flag] clicked
go to x: -200 y: -130
forever
```

We can now check the program to see if it starts the sprite in the right place. To run the program press the green flag above the track (top right of the screen).



This runs the program so far and should position your sprite in your correct location. If not the re-check your x,y co-ordinates and try again.

Once you have finished checking press the red button to stop the program from running (as it is still running the forever loop – you can see a program is running by the white border around the program:



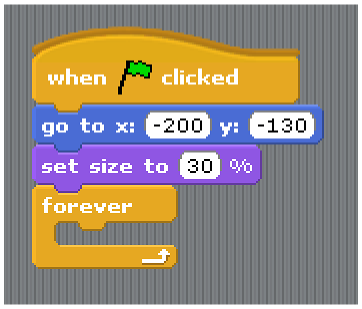
```
when [flag] clicked
go to x: -200 y: -130
forever
```

Once the red octagon is pressed the white border is removed.

If you sprite is too big (or even too small) like this example, you can change the size of the sprite.

Therefore the Looks menu controls is required.

Select the set size to 100 % and position between the green flag and forever loop.

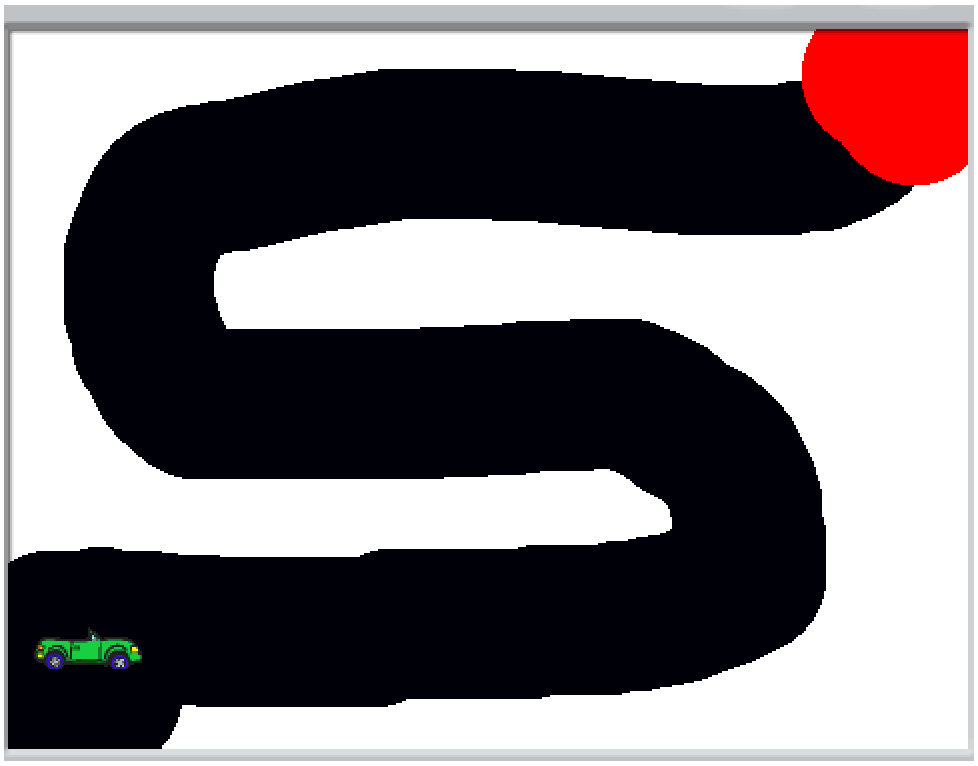


```
when [flag] clicked
go to x: -200 y: -130
set size to 30 %
forever
```

I have needed to change the example to 30%, but try with various sizes to get the right size for your sprite. Each time you change press the green flag.

Once you have the right size remember to stop the program from running.

At the end of your task you should now have a track and sprite looking ready for the game to start.



Task 5

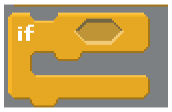
Human interaction.

This simple track requires 6 possible actions. The first four are up, down left and right key presses.

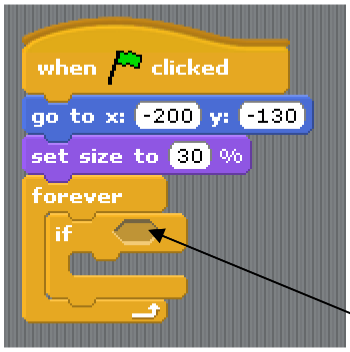
The next two are either failing the track by touching the area off the track – in this example it is white. The final outcome is touching the end zone – red in this case.

These 6 choices are governed by what are known as IF statements. This means the computer program checks an IF statement and if it is true do this.

There are various types of IF statements, but this track requires the basic IF statement, found in the Control menu.



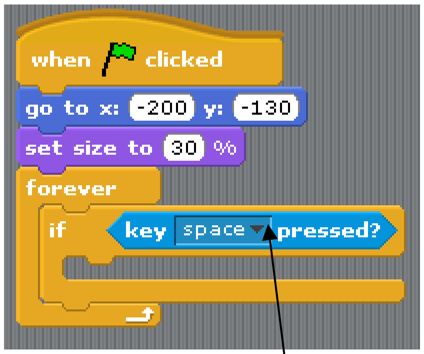
This fits in between the forever loop as the “IF” needs to be continuously checked.



```
when clicked
go to x: -200 y: -130
set size to 30 %
forever
    if
```

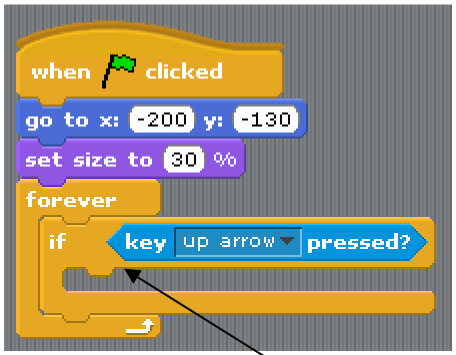
The statement requires a hexagon shaped command. As the program needs to “sense” a human interaction these commands are in the Sensing menu. As you can see many of these options are hexagonal shaped. We need the key space pressed? option.

This drags over to the IF statement and fits into the hexagonal area.



```
when [flag] clicked
go to x: -200 y: -130
set size to 30 %
forever
    if key space pressed?
```

Using the pull down menu choose “UP ARROW”

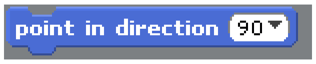


```
when [flag] clicked
go to x: -200 y: -130
set size to 30 %
forever
    if key up arrow pressed?
```

Now we need to fill in here with what we want the car to do. There are two things.

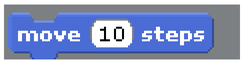
- Point the sprite the right way
- Move the sprite in that direction

The two commands are :



```
point in direction 90
```

and

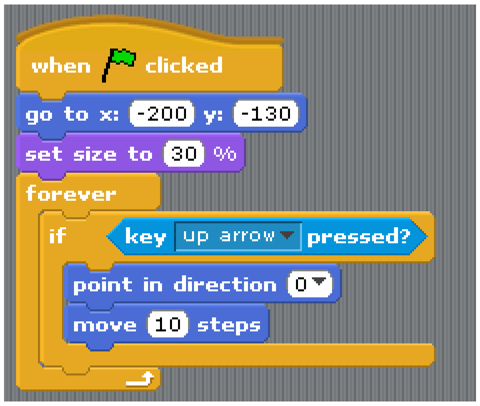


```
move 10 steps
```

You can find out for yourself where these commands are (HINT: They are colour coded).

You must ensure that you point in the right way (use the drop down list on the command).

Your program should now look like this



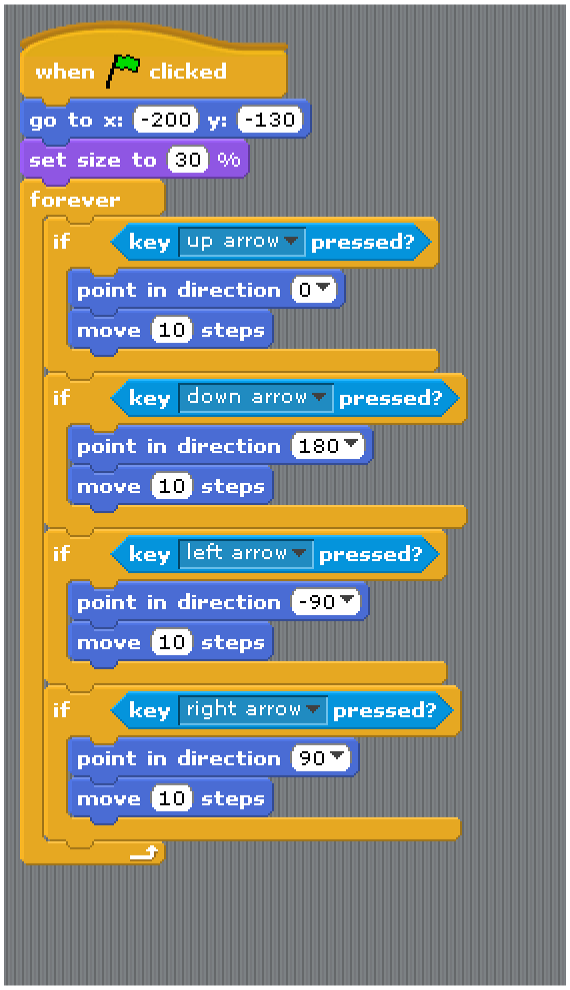
```
when [flag] clicked
go to x: -200 y: -130
set size to 30 %
forever
    if key up arrow pressed?
        point in direction 0
        move 10 steps
```

To check press the green flag and check what happens when you press the up arrow. Once you have checked then don't forget to stop the program with the red octagon.

There is one change that you need to make. Don't worry if you cannot see it, it will become more obvious later. If you see the change then just go ahead and change it.

You now need to create 3 more IF's, for down, left and right. Each IF statement needs to be within the forever loop, but not inside any other IF statement.

Your program should now be like this:



```
when [green flag] clicked
go to x: -200 y: -130
set size to 30 %
forever
    if key up arrow pressed?
        point in direction 0
        move 10 steps
    if key down arrow pressed?
        point in direction 180
        move 10 steps
    if key left arrow pressed?
        point in direction -90
        move 10 steps
    if key right arrow pressed?
        point in direction 90
        move 10 steps
```

Now try the program out again (green flag) and try to run your sprite around the track without touching the edge. You need to ask yourself if it is possible to get around the track – is it too easy or too hard.

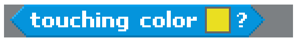
You have some options such as making the sprite smaller or change the steps count to lower, which means the sprite will not move as far on each key press.

Once you are happy with your program you can move on to the next task.

Task 6

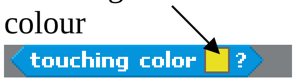
We have two more IF statements to create.

The first is to get to the end point.

We use another IF statement and use the  statement within the IF.

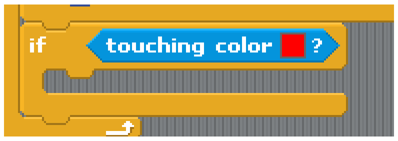
```
touching color ?
```

To change the colour to the colour you have used in your end zone, left click over the colour



```
touching color ?
```

Your mouse pointer will change to an eye dropping tool. Just move the mouse to the colour on your end zone and left click again.

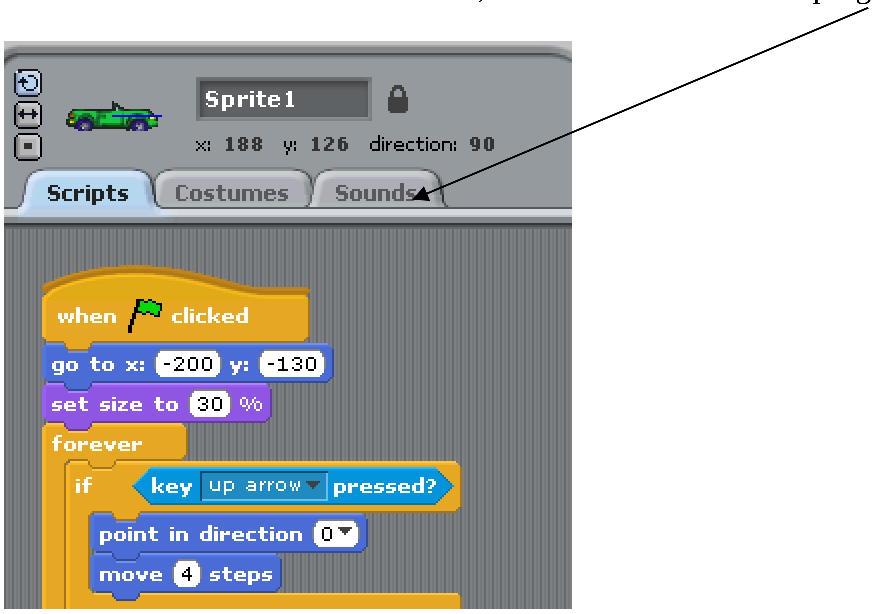


```
if touching color ?
```

Now we need to do something when the player touches the end zone. For simplicity (as this is a simple track game) we will play a sound.

Sounds can be pre-recorded such as “WELL DONE YOU HAVE FINISHED MY GAME”, but for now we will use a sound already recorded.

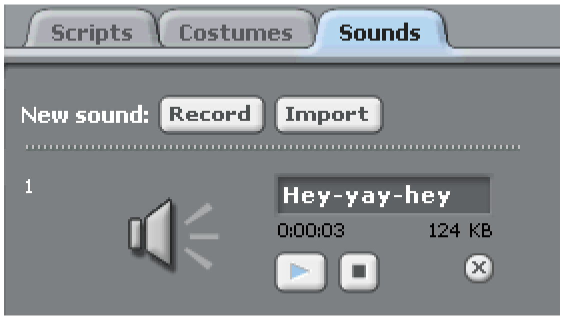
You will need to select the sound tab, which is above the main program:



```
when clicked
go to x: -200 y: -130
set size to 30 %
forever
    if key up arrow pressed?
        point in direction 0
        move 4 steps
```

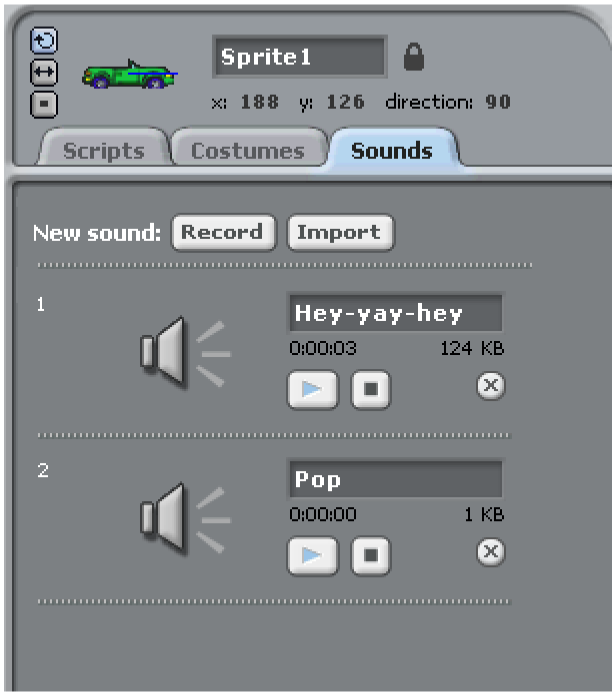
Click the tab and select “import”

Choose a sound you think will be suitable – I have chosen "Hey-yay-hey" from the vocals folder. Then press OK.

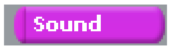
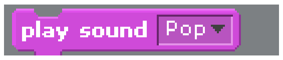


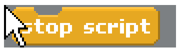
We will also need a sound for if the player touches away from the track – game over.

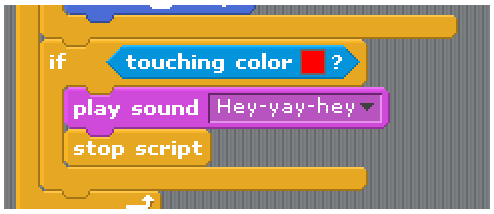
So press the import button and select another sound you think is appropriate. I have chosen "Pop" from the effects folder. Press OK and you should have two sounds ready to go.



You now need to go back to the scripts tab.

Using the  menu select the  and insert in the IF (touching colour) statement and select the correct sound from the drop down list.

We also want the game to end there so you will need to find the following statement  and place in after the "play sound" command. (This statement is a bit harder to find, but remember they are colour coded).

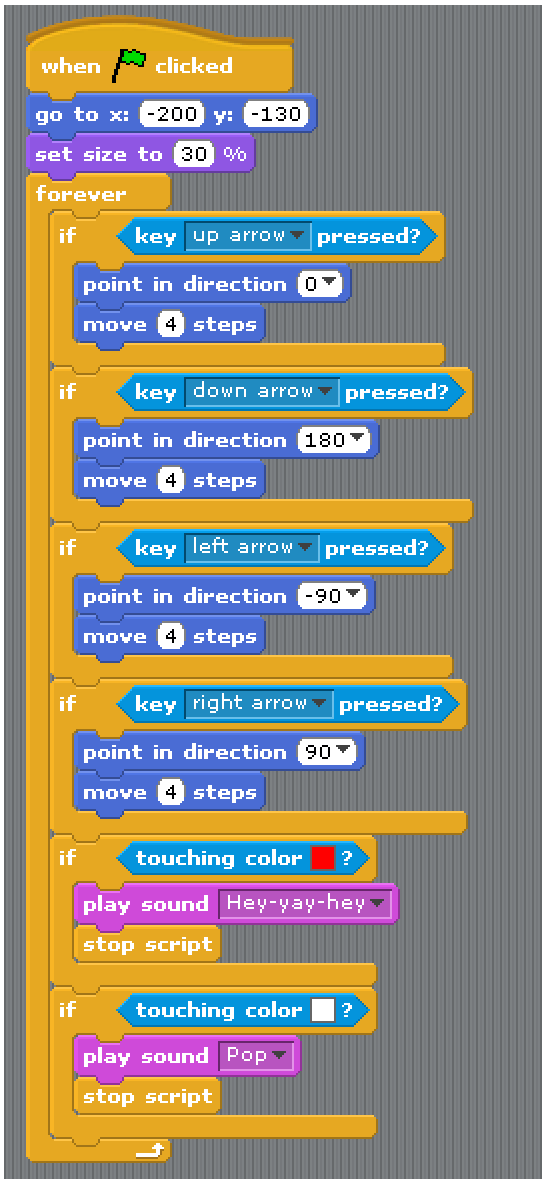


```
if <touching color 🟥 ?>
    play sound Hey-yay-hey
    stop script
```

The final IF statement is for when a player fails to stay on the track.

Create an IF statement similar to the last with the correct sound.

Your program should look like this:



```
when clicked
go to x: -200 y: -130
set size to 30 %
forever
    if <key up arrow pressed?>
        point in direction 0
        move 4 steps
    if <key down arrow pressed?>
        point in direction 180
        move 4 steps
    if <key left arrow pressed?>
        point in direction -90
        move 4 steps
    if <key right arrow pressed?>
        point in direction 90
        move 4 steps
    if <touching color 🟥 ?>
        play sound Hey-yay-hey
        stop script
    if <touching color ⬜ ?>
        play sound Pop
        stop script
```

Now press the green flag to try.

You can also play in full screen mode by pressing the button. It is located below your testing area on the left hand side.

To revert back you need to press the button.

WELL DONE YOU HAVE JUST CREATED YOUR FIRST GAME

Once you have tested your program you might want to make some changes:

- Multiple levels – more difficult every time
- Multiple players (at the same time) to make the game a race
- Moving obstacles
- Changing the map each level
- Making the ending better
- Making the sprite better
- Making the game faster (or slower)
- Have a score based system
- Have a timer based game

Make a completely different game, such as pac man, snakes, or even an educational game.

You can also create movies or presentations using programming.

You need to decide whom you are designing your games for.

- Reception / Infant children
- Junior level children
- Teenagers
- Adults
- Experienced “gamers”

By understanding who you think will play the game you can then plan on what type of came you wish to create. Try to imagine sitting in there shoes.