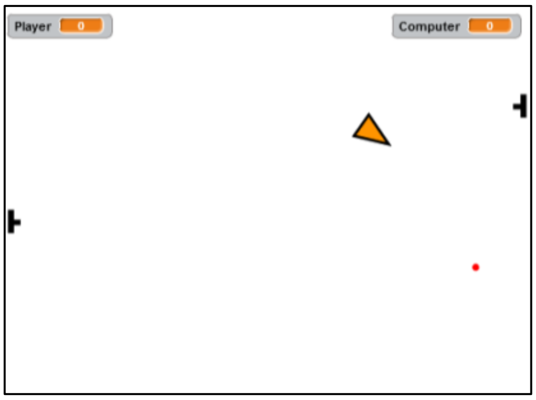
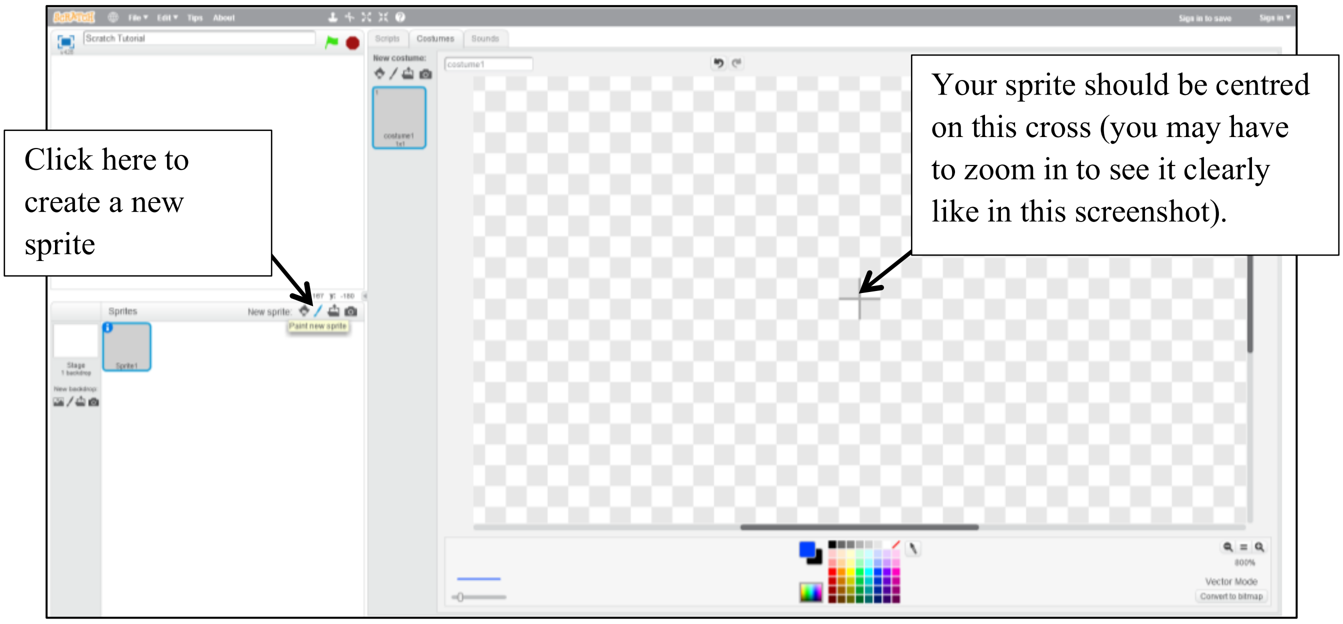
Scratch Tutorial

We will use Scratch to create a game with the following rules:

- The player controls a spaceship. Pressing *up* makes the spaceship accelerate, pressing *down* makes the spaceship decelerate, press *left* makes the spaceship turn left, pressing *right* makes the spaceship turn right, and pressing *space* makes the spaceship fire a rocket.
- On each side of the screen the computer will control enemy spaceships. They will move vertically and randomly fire rockets across the screen.
- If the play gets hit by a rocket the computer's score will increase by 1. If the player shoots an enemy spaceship the player's score will increase by 1.



Creating Sprites

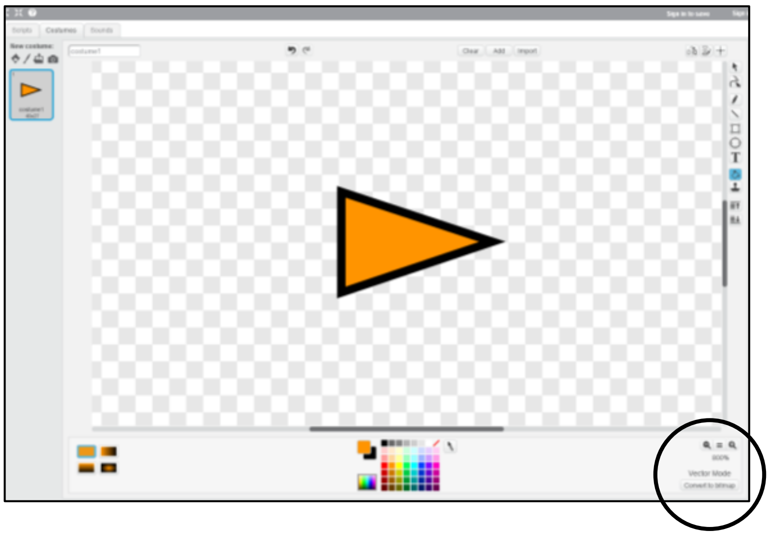

Click here to create a new sprite

Your sprite should be centred on this cross (you may have to zoom in to see it clearly like in this screenshot).

Draw a triangle to represent your spaceship. The spaceship should be pointing to the right.

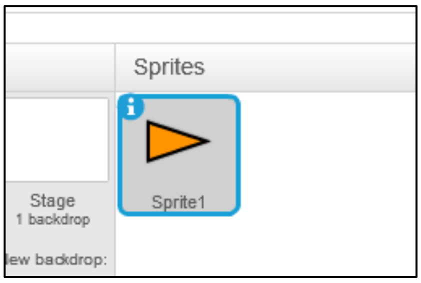
You may need to zoom in to get a better view.

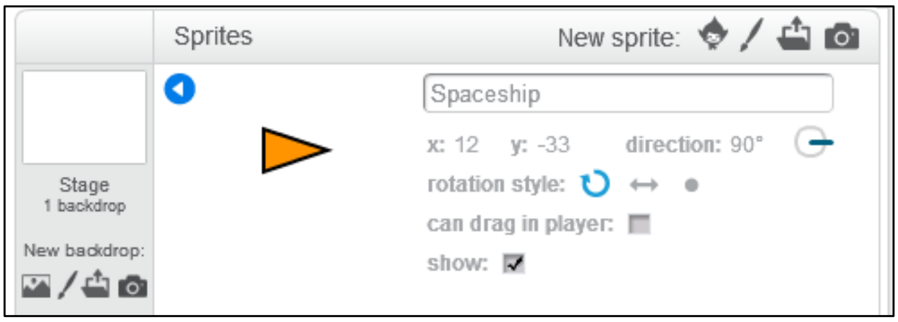
Note that the quality of the sprite looks better if you create it in *vector mode* (circled in the screenshot on the right).



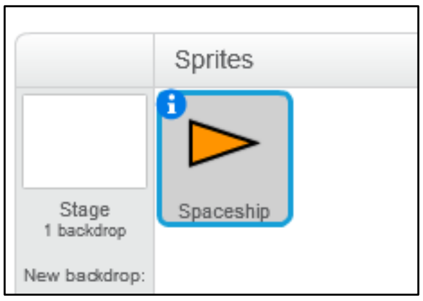
The default name of the sprite will be *Sprite*.

1. To change this into something more descriptive click on the blue and white *i*.
2. Change the name of the sprite to *Spaceship*.
3. Then click on the blue and white arrow to go back.



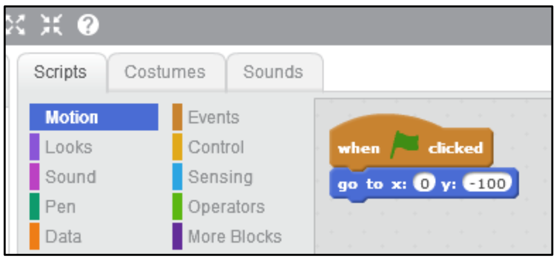


Spaceship
x: 12 y: -33 direction: 90°
rotation style:
can drag in player:
show: ☑



Controlling Sprites - Positioning

When the game starts we will position the spaceship just below the centre of the screen. Click on the *Scripts* tab and enter the following code:



```
when [flag] clicked
go to x: 0 y: -100
```

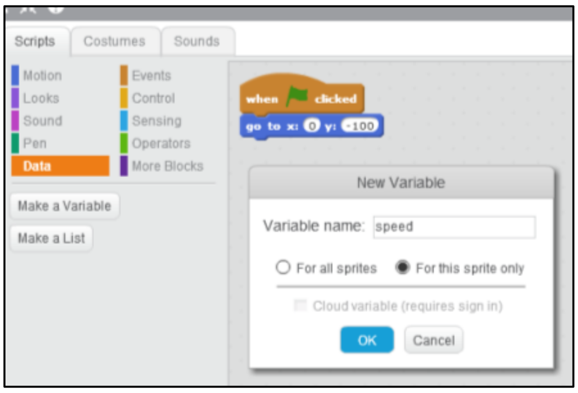
This tells Scratch to position the spaceship at coordinates (0,-100) when the green flag is clicked to start the game.

Note that the scripts are colour coded to make finding them easier.

We need a way to control the speed of the spaceship, and the direction in which it is facing.

Variables

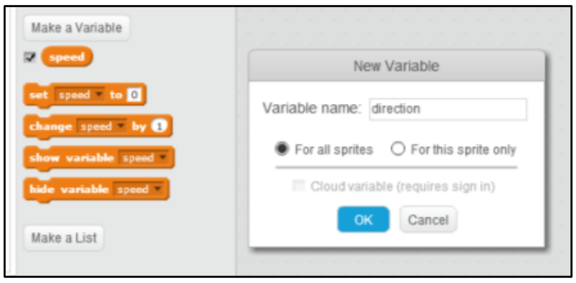
Click *Make a Variable* from the *Data* menu and create a variable called *speed* for this sprite only (this means only the spaceship will be able to access this variable). This will be used to control the speed of the spaceship.



```
when [flag] clicked
go to x: 0 y: -100
```

Variable name: speed
For all sprites [✓ For this sprite only]

Create another variable called *direction* for all sprites. This means that other sprites will be able to access it and read and change its value. This will be used to control the direction in which the ship is facing, and also the direction at which it will fire its rockets.

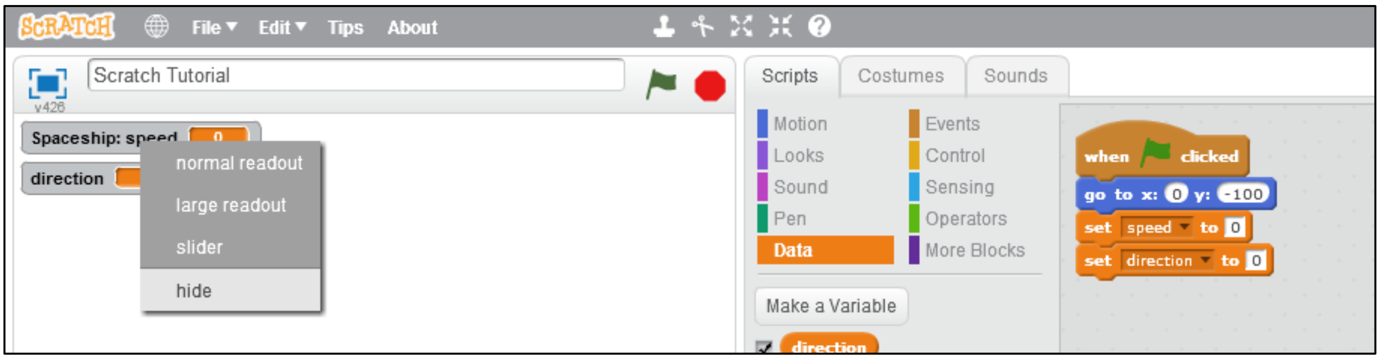


Variable name: direction
[✓ For all sprites] For this sprite only

When the game first starts the value of *speed* should be 0 and the value of *direction* should be 0.

Modify your existing script to make this happen, as in the screenshot below.

Notice that the values of the variables are displayed on the screen. To hide these values right-click on them and select *hide*.



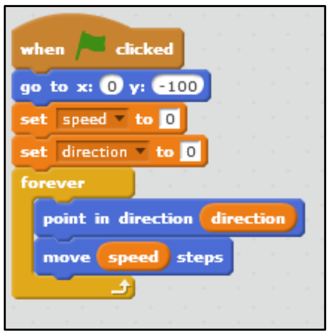
```
when [flag] clicked
go to x: 0 y: -100
set speed to 0
set direction to 0
```

Controlling Sprites – Moving and Rotating

To animate the motion of a sprite we can use a loop which will repeat what is inside it indefinitely.

Modify your code as in the screenshot on the right. This will continuously point the spaceship in the direction of the variable *direction*, and move it forward by *speed* pixels.

If you click the green flag the spaceship should now be facing upwards. However, it will not move because the value of speed is still equal to zero.



```
when [flag] clicked
go to x: 0 y: -100
set speed to 0
set direction to 0
forever
    point in direction (direction)
    move (speed) steps
```

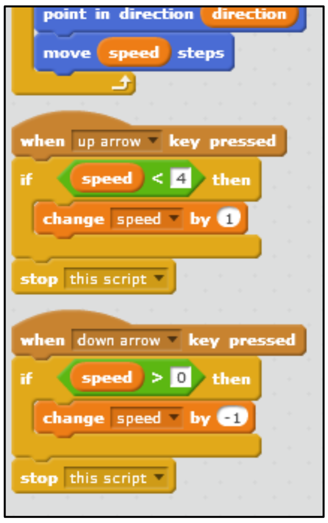
Reacting to User Input

When the up arrow is pressed we need the spaceship to accelerate i.e. we need the value of *speed* to increase.

When the down arrow is pressed we need the spaceship to decelerate i.e. we need the value of *speed* to decrease.

We need to be careful that we do not increase the speed by too much otherwise the spaceship will be uncontrollable. We also need to make sure that when the speed is zero, the ship stops decelerating.

Modify your code as in the screenshot on the right. Notice that this sets the maximum value of *speed* to 4. The *stop* function is not mandatory. However, it makes the code look neater as it marks the end of this particular script. If it were not there and we went back to look at the code in the future, we may think that we have accidentally deleted some code since the script doesn't end in a neat way. Make sure you only stop *this script* and not *all* or *other scripts in sprite*.

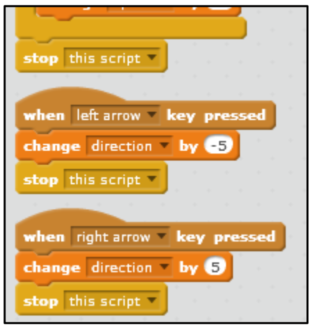


```
    point in direction (direction)
    move (speed) steps

when up arrow key pressed
if < (speed) < 4 > then
    change speed by 1
stop this script

when down arrow key pressed
if < (speed) > 0 > then
    change speed by -1
stop this script
```

To test to see if everything is working, click the green flag to start the game and hold down the up arrow. The spaceship should move forward. Holding the down arrow should make it slow down.

When the left and right arrows are pressed we need the spaceship to turn left and right.

Modify your code as in the screenshot on the right. You should now be able to steer the spaceship. Test to see if it works by clicking the green flag and controlling the spaceship.

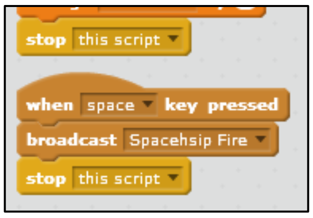


```
stop this script

when left arrow key pressed
change direction by -5
stop this script

when right arrow key pressed
change direction by 5
stop this script
```

Broadcasting

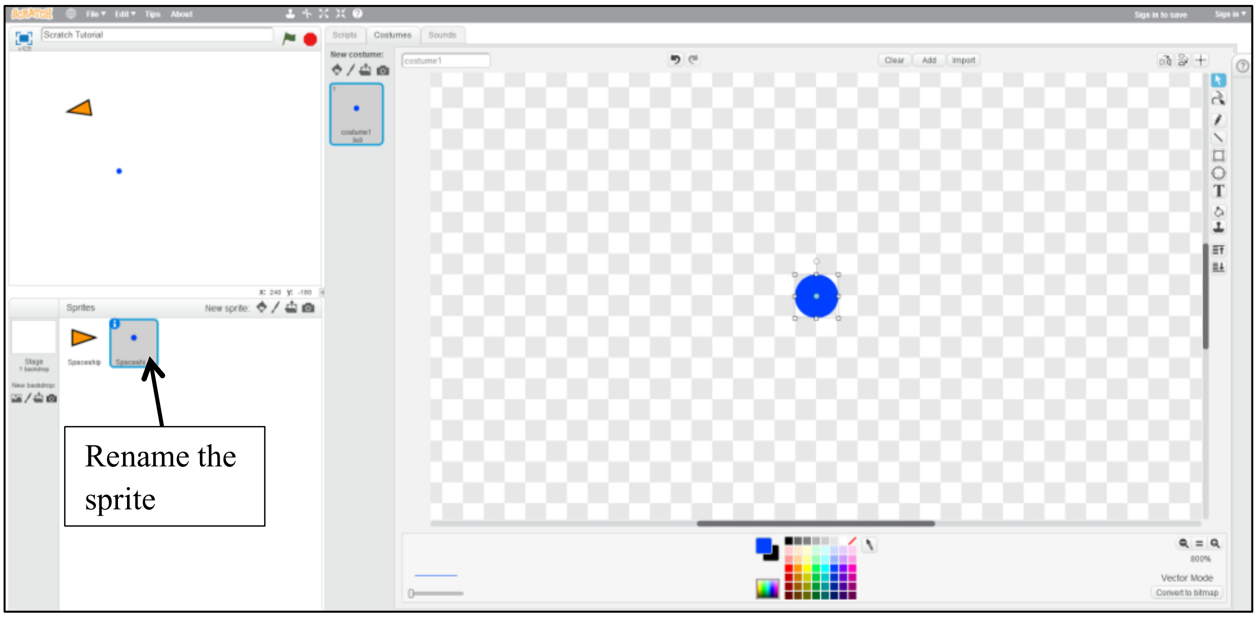
There are often many ways to make something happen in Scratch. To make our spaceship fire when we press *space* we will use the *broadcast* function. We do not have to use this to make the spaceship fire, but now is a good time to see how this function works.

Modify your code as in the screenshot on the right. Note that you should create a new message (by clicking the black downwards arrow) and call it something descriptive such as *Spaceship Fire*.



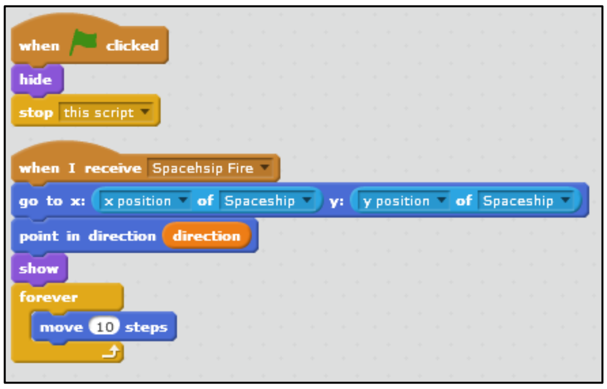
```
stop this script

when space key pressed
broadcast Spacehsip Fire
stop this script
```

Create a new sprite and draw something to represent your rocket. It can be as simple as a circle. Remember you should draw your sprite in the centre of the window, on top of the gray cross. It is a good idea to rename your sprite into something more descriptive. Rename it to *Spaceship Rocket*



When the game first starts the rocket should be hidden. It should only appear when *space* is pressed. When the message *Spaceship Fire* is broadcast the rocket should appear at the same position as the spaceship, and move forward in the same direction as the spaceship is facing.

Click on the *Scripts* tab of this new sprite and add code as in the screenshot on the right.



```
when 🏴 clicked
hide
stop this script

when I receive Spacehsip Fire
go to x: x position of Spaceship y: y position of Spaceship
point in direction direction
show
forever
    move 10 steps
```

Now test the script by clicking the green flag to start the game, moving and rotating the spaceship, and firing the rocket.

Notice that you can change the speed of the rocket by changing how much it moves inside the forever loop.

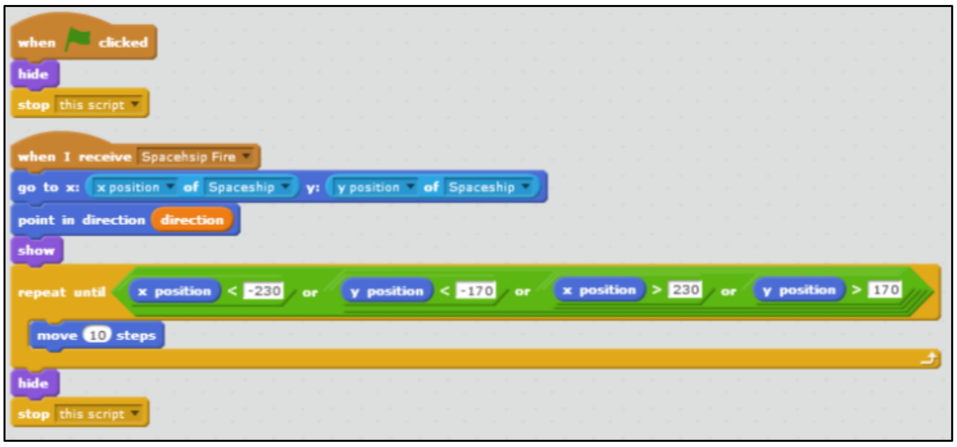
Troubleshooting

When testing you should notice two things:

1. When the rocket reaches the edge of the screen it will just stay there and still be partially visible.

2. If you press space at any time the rocket will move back to where the spaceship is and fire again. This does not look good if the rocket has not completed its previous journey as it will just disappear mid-journey to be fired again.

To fix problem 1, *change* the code according the screenshot on the right.



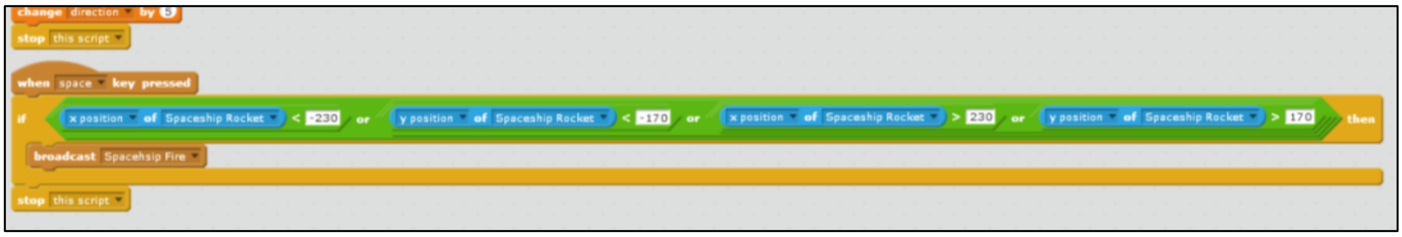
```
when 🏴 clicked
hide
stop this script

when I receive Spacehsip Fire
go to x: x position of Spaceship y: y position of Spaceship
point in direction direction
show
repeat until x position < -230 or y position < -170 or x position > 230 or y position > 170
    move 10 steps
hide
stop this script
```

This only moves the rocket if it hasn't reached the edge of the screen.

When it does reach the edge of the screen it is hidden.

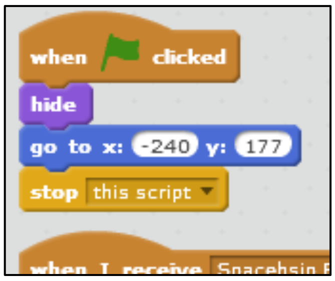
Test the code to see if it works.

To fix problem 2 we need to go back to the scripts for the spaceship. Change the code for when space is pressed to the following:



```
change direction by 5
stop this script

when space key pressed
if x position of Spaceship Rocket < -230 or y position of Spaceship Rocket < -170 or x position of Spaceship Rocket > 230 or y position of Spaceship Rocket > 170 then
    broadcast Spacehsip Fire
stop this script
```

This means that *Spaceship Fire* will only be broadcast if the rocket is currently at the edge of the screen. However, this will cause another problem. When the game first starts the rocket will not necessarily be at the edge of the screen. To fix this, go back to the scripts for the rocket and change to the code on the right.

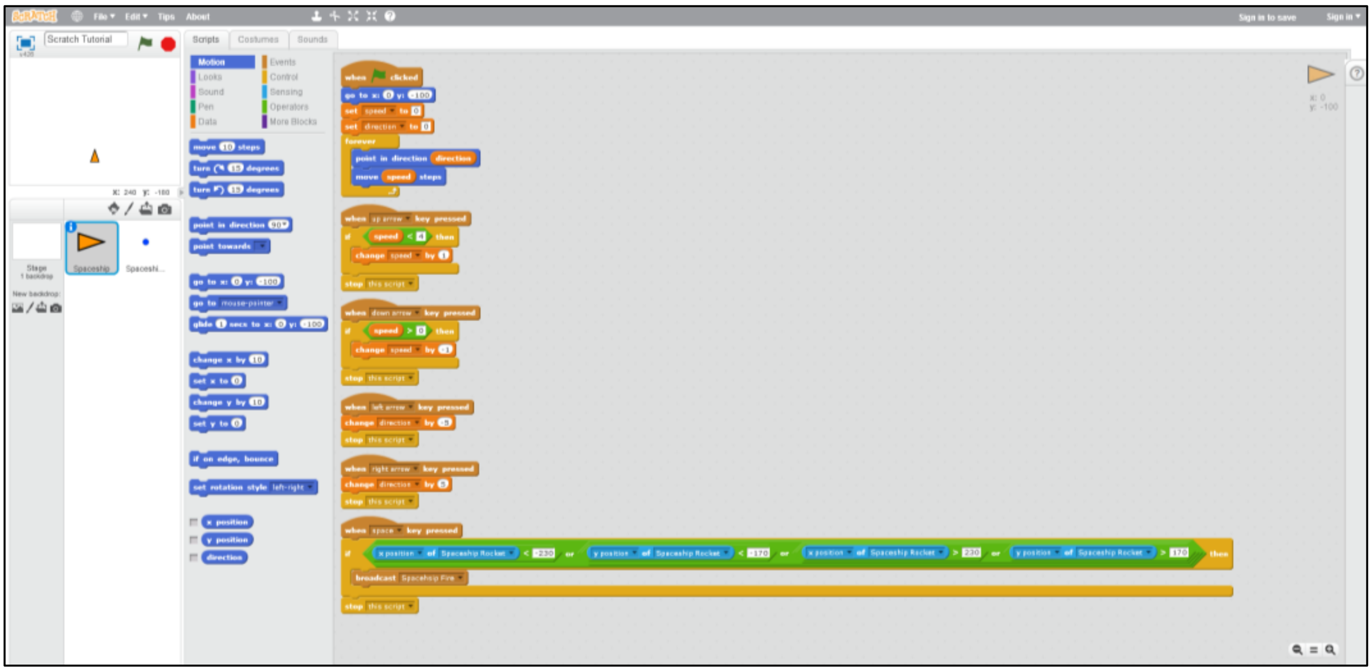


```
when [flag] clicked
hide
go to x: -240 y: 177
stop this script
when I receive Spacehsip F
```

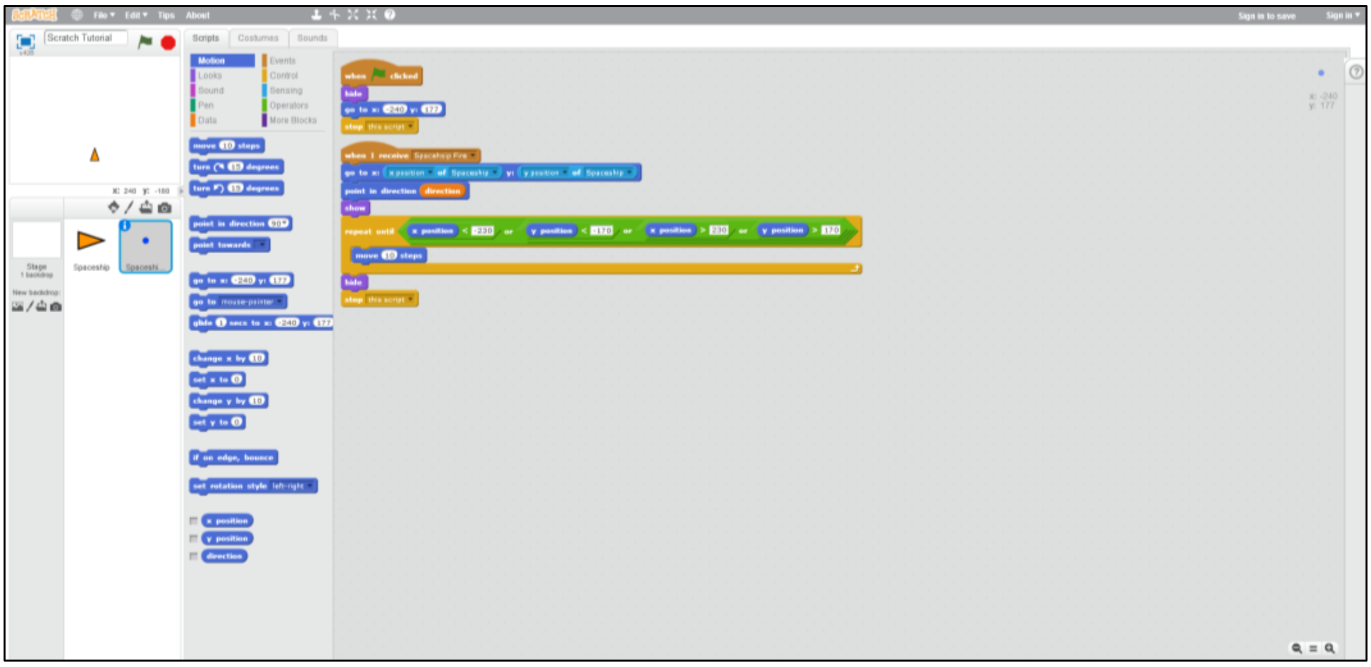
This will place the rocket at the edge of the screen when the game begins. Note that the value of the *y*-coordinate is not important.

Test the game to check that it has fixed the problems we had.

The code for your two sprites should now look similar to the following:



```
when [flag] clicked
go to x: 0 y: -100
set speed to 0
set direction to 0
forever
    point in direction (direction)
    move (speed) steps

when up arrow key pressed
if <(speed) < 4> then
    change speed by 1
stop this script

when down arrow key pressed
if <(speed) > 0> then
    change speed by -1
stop this script

when left arrow key pressed
change direction by -5
stop this script

when right arrow key pressed
change direction by 5
stop this script

when space key pressed
if <(x position of Spaceship Rocket) < -230 or (y position of Spaceship Rocket) < -170 or (x position of Spaceship Rocket) > 230 or (y position of Spaceship Rocket) > 170> then
    broadcast Spacehsip Fire
stop this script
```



```
when [flag] clicked
hide
go to x: -240 y: 177
stop this script

when I receive Spacehsip Fire
go to x: (x position of Spaceship) y: (y position of Spaceship)
point in direction (direction)
show
repeat until <(x position) < -230 or (y position) < -170 or (x position) > 230 or (y position) > 170>
    move 10 steps
hide
stop this script
```

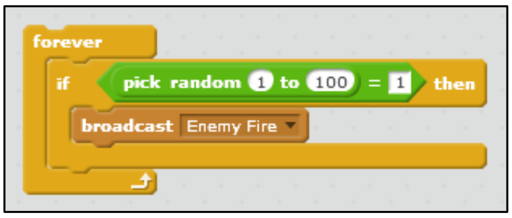
Putting it all Together

Use the skills you have learned to add an enemy to your game on the left side of the screen that obeys the following rules:

- The enemy can only move vertically. If the player is above the enemy it will move up; if it is below it will move down
- The enemy will randomly fire a rocket horizontally across the screen.
- If the enemy's rocket hits the player the computer's score will increase by one; if the player's rocket hits the enemy the player's score will increase by one (use variables for the scores).
- Add another enemy to the right side of the screen that obeys the same rules. To save time you can duplicate the sprites you created for the first enemy and alter the code slightly. Give this enemy a different speed to the other enemy to make the game more interesting (and challenging).

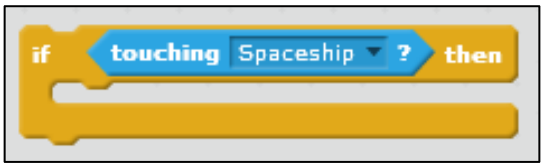
Hints:

To randomly fire a rocket you can use code similar to the screenshot on the right. To fire more or less frequently, adjust the range of numbers from which the random number is chosen. You will also need to make sure that you don't encounter the same problems we encountered when firing our rocket (you should only fire if the previous shot has reached the edge of the screen).



```
forever
    if pick random 1 to 100 = 1 then
        broadcast Enemy Fire
```

To determine whether a rocket has hit a spaceship use the function below. You will need to check this every time you move the rocket to a new position.



```
if touching Spaceship ? then
```