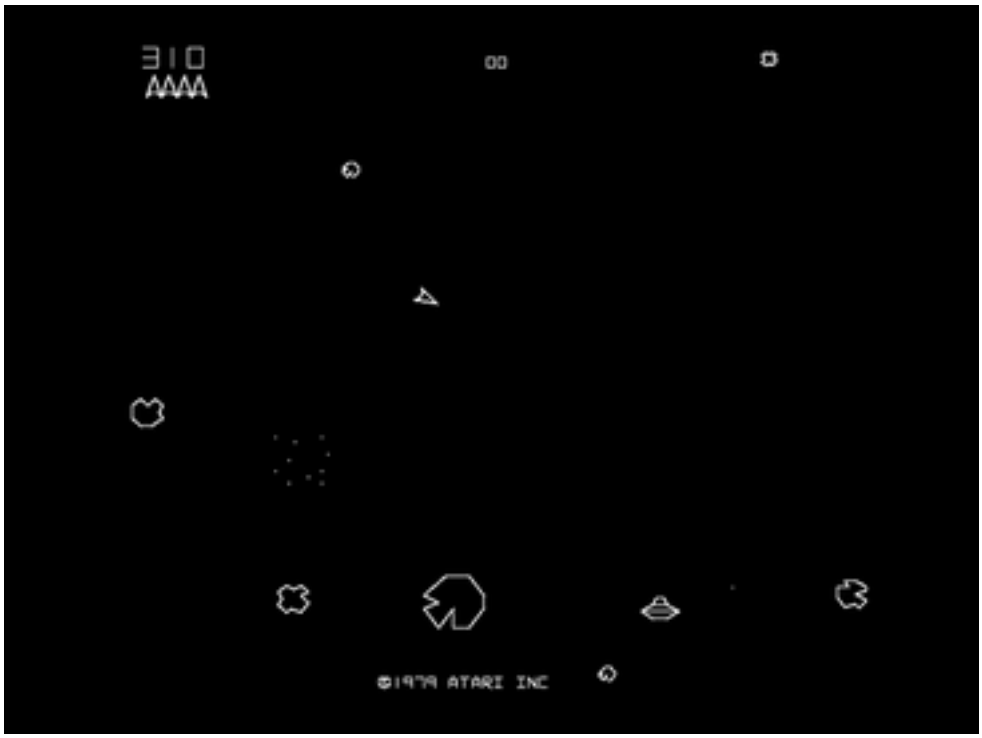
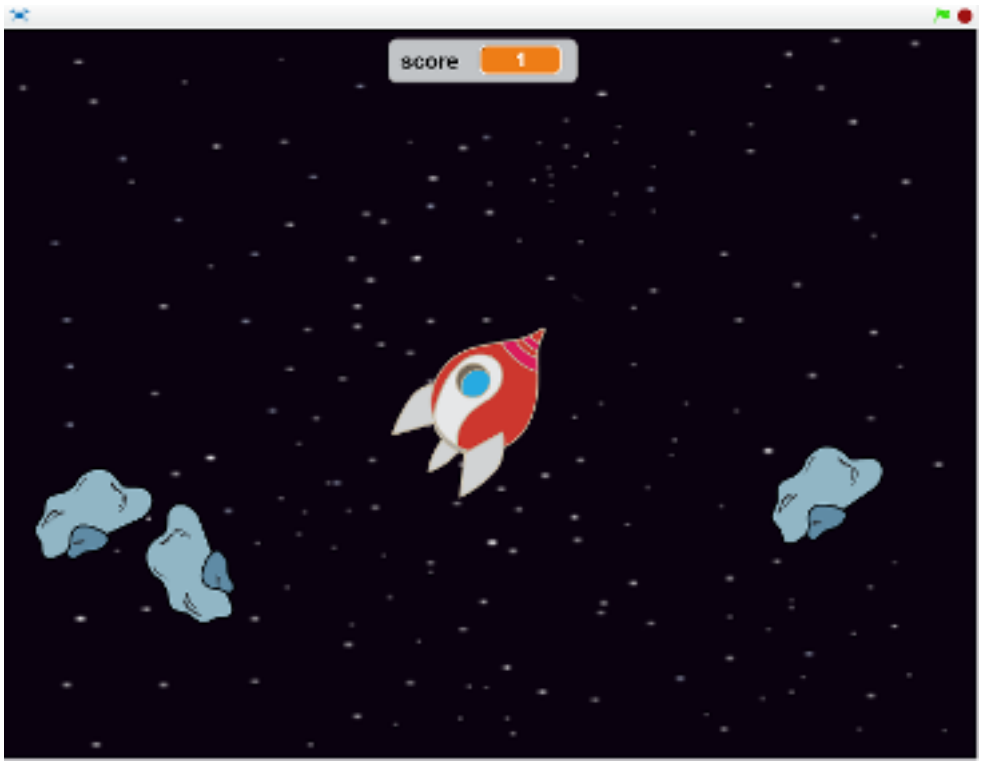
Asteroids



This project is based on the 1979 Atari arcade space shooter game **Asteroids**.

In the original game, the player controls a spaceship in an asteroid field which is periodically traversed by flying saucers.

The object of the game is to shoot and destroy asteroids and saucers while not colliding with either or being hit by the saucers' counter-fire.



Our version includes only the spaceship and the asteroids, but has much better graphics than the original – and can be made using resources already in Scratch!

The player controls a spaceship which moves around deep space dodging the asteroids coming from all directions.

Each asteroid destroyed earns a point.

The game ends when an asteroid collides with the spaceship.

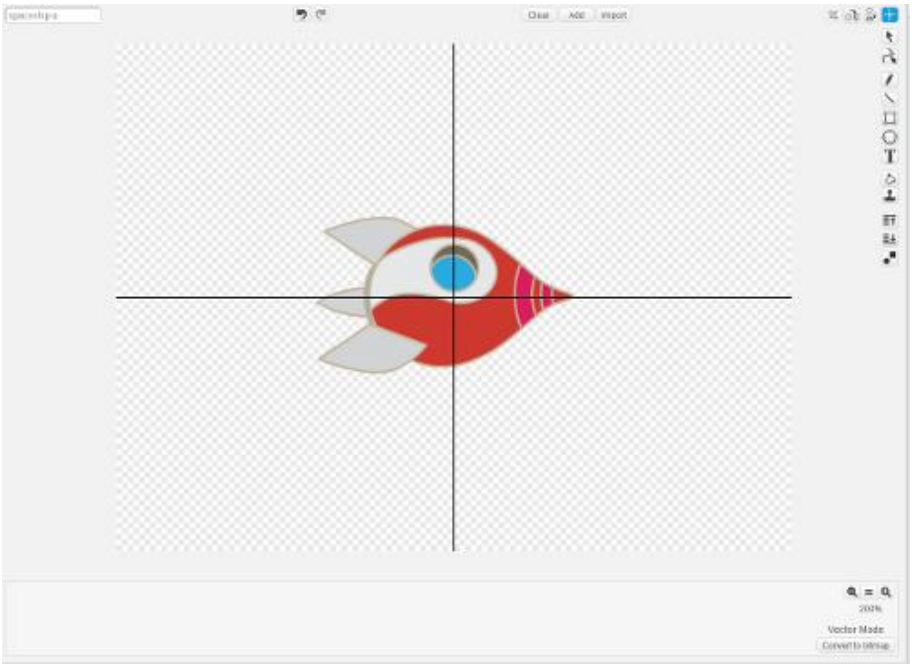
The following **parameters** will be used to change how the game behaves:

- How quickly the spacecraft speeds up when the rocket engine is fired (***thrust***)
- How quickly it slows down when the rocket engine is off (***friction***)
- How far the laser gun will reach (***max_laser***)
- How many asteroids can be visible at the same time (***max_rocks***)
- How close to the spaceship new asteroids can appear (***min_distance***)

Step 1 - Make the Initial Spaceship

Start Scratch and set the backdrop to ***stars***, delete the cat sprite ***Sprite1***, and add the ***Spaceship*** sprite from the library.

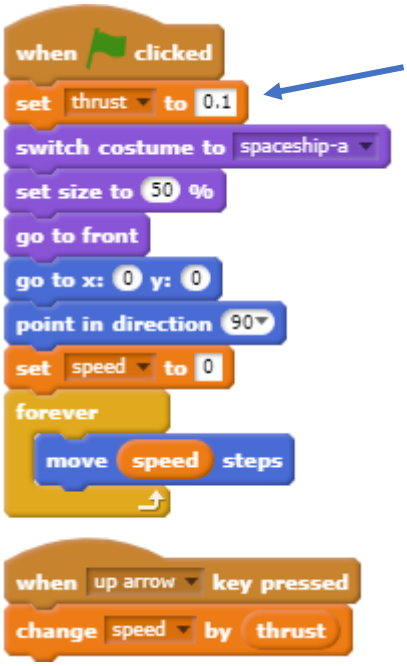
Rotate the ***spaceship-a*** costume to be pointing to the right (90 degrees) and set the centre of the spaceship to line up with the nose and the middle of the window.



Step 2 - Make the Spaceship Move

Create two variables ***thrust*** and ***speed***. The variable *thrust* will control how quickly the spaceship will accelerate (speed up) and *speed* will keep track of how fast it is currently moving. Hide these two variables, and all others you create from now on except when I say.

Add this code to the Scripts of Spaceship.



I have put ***thrust*** at the top as this is one of the **parameters** I mentioned earlier which will control how the game works. It is a good idea to keep these at the top so you can find them easily.

Start the game running by pressing the green flag and the spaceship will start in the middle of the stage pointing to the right.

Press the ***up arrow*** key and the spaceship will start moving to the right. The longer you hold the key down, the faster it will go.

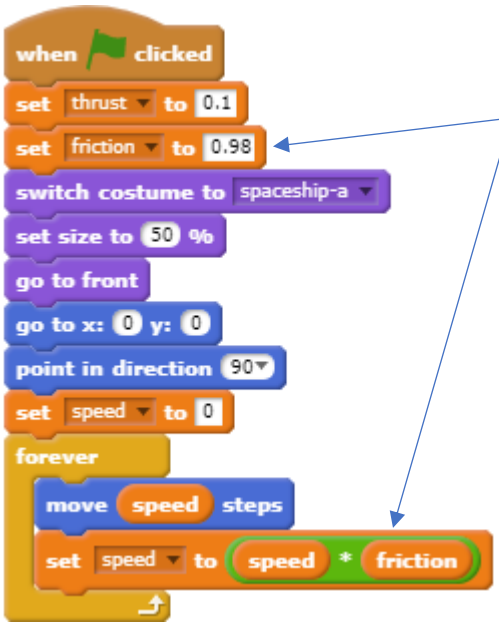
The spaceship will stop when it hits the right-hand edge because Scratch cannot draw off the stage.

Press the red button to stop the game, which you will need to do whenever you finish testing from now on.

Save your project before continuing.

Step 3 - Add Friction

At the moment we have no way of slowing down or stopping the spaceship. We could add the *down arrow* key to slow it down, but a more interesting idea is to pretend that a spaceship travelling through space will slow down by itself. If you know your physics, you will know that this is not the case in real life but it makes the game more interesting!



Create a new variable ***friction***.

Change the code slightly by adding a **set** block at the top and another in the *forever* loop as shown.

As before, I have put the **parameters** at the top to say how quickly the spacecraft will accelerate (*thrust*) and how quickly it will slow down (*friction*).

Start the game again.

Press the ***up arrow*** key and the spaceship will start moving as before but when you release it, the spaceship will gently slow down.

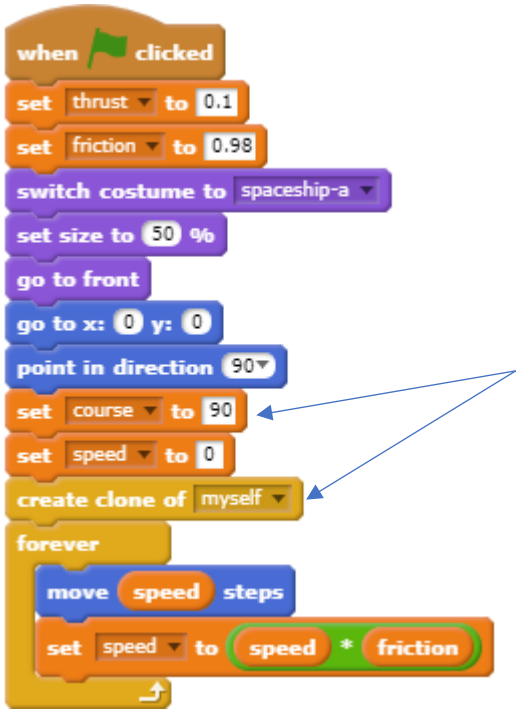
Save your project before continuing.

Step 4 - Add Steering

The spaceship moves but only in one direction, so we will need to add a way of steering it. Add these two sequences to the scripts of Spaceship.

Run the game and we can steer the spaceship using the *left arrow* and *right arrow* keys - but like a car, not like a spaceship!

Changing where the spaceship is pointing should **not** make it change direction - unless the rocket engine is being fired using the *up arrow*. But Scratch will always move a sprite in the direction it is pointing…



We can get around this by having two spaceships, where the second one does nothing except copy the movements of the first spaceship and provide the costume which you can see.

This is easy to do by using the existing spaceship plus a **clone** of the same spaceship.

Create a new variable ***course***, which will keep track of which direction the spaceship is pointing.

Modify the code of *Spaceship* slightly as shown. You need to add a **set** block for ***course*** and a **create clone** block.

You also need to change the movement controls ***up arrow***, ***left arrow*** and ***right arrow*** as shown.

Run the game and fly the spaceship around. The **clone** spaceship stays in the centre of the stage but turns with the spaceship, which moves as before.

When you press the ***left arrow*** or ***right arrow*** keys, nothing seems to happen - until you press the ***up arrow*** key to fire the rocket engine. Now the spaceship goes in the new direction.

Save your project before continuing.

Step 5 - Add Code for the Clone Spaceship

Add this code to the Scripts of Spaceship.

It tells the **clone** spaceship to point whichever way the ***course*** variable says then jump to wherever the Spaceship currently is. This means it is copying what the Spaceship is doing.

Start the game and drive the spaceship around again. Now you will see that the two spaceships stay on top of the each other. When you turn using the ***left arrow*** and ***right arrow*** keys, you can see one of these changes direction but the other carries on in the same direction - until you fire the rocket engine using the ***up arrow***. The spaceship which turns is the **clone**.

Change the script of Spaceship slightly so that we can't see it by adding a **hide** block.

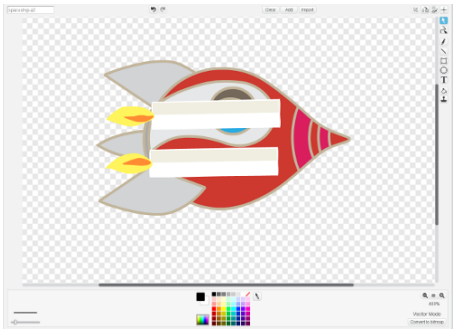
Start the game again and drive the spaceship around. When the spaceship is moving and you are not using the rocket engine, you can rotate the spaceship and it does not change direction - until you use the rocket engine using the up arrow.

This is much closer to how a real spaceship behaves.

Note: When you press the red button to stop the game, the spaceship will disappear.

Save your project before continuing.

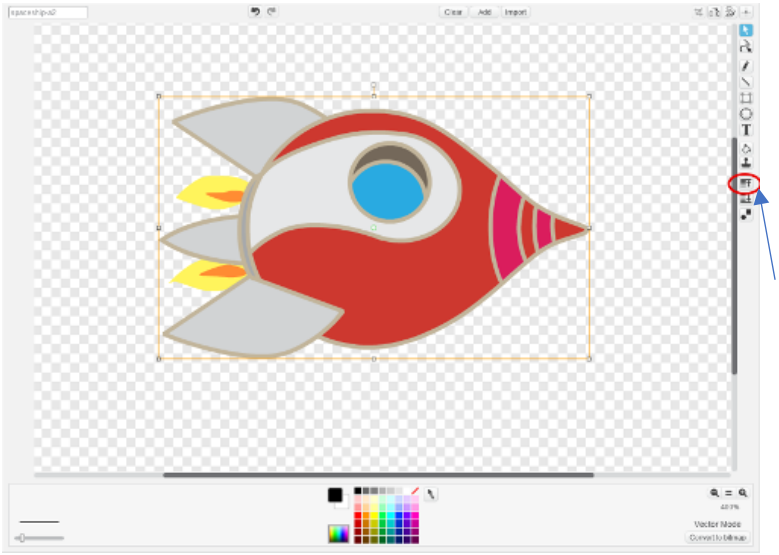
Step 6 - Add Costumes to Show the Rocket Engine Being Used



Duplicate the costume ***spaceship-a***. The new costume will be called ***spaceship-a2***.

Click **Add** at the top of the screen and select the two costumes ***candle1-a*** and ***candle1-b*** from the library (two candles).

Rotate the two candles and position them so that the flames are at the back of the spaceship, something like my one here. They only need to be roughly right at the moment.

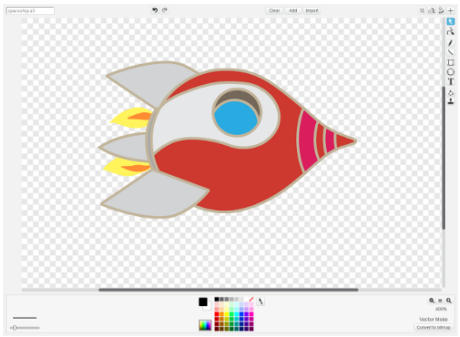


In vector mode, each part of the image is on a different layer – one on top of the other. We need to move the spaceship itself to the top layer.

Click the spaceship to select it so that there is an orange rectangle showing around it.

Press the **Forward a layer** button (circled in red) on the right-hand side **twice** - and the two candles will now be below the spaceship with only the flames showing.

Adjust the two candles so that the flames look how you like them - the candlestick parts remain hidden underneath the spaceship.



Duplicate this costume to create the new costume ***spaceship-a3***.

Swap the two candles around to make the flames slightly different between the two costumes.

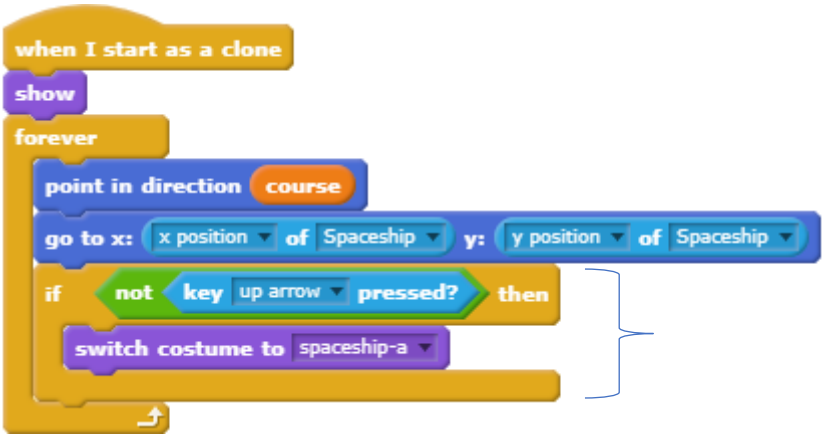
The idea is that by switching from one costume to the other we can get the feeling of a rocket engine burning.

Save your project before continuing.

Step 7 - Add Code to Show the New Costumes

Now we can **add** this code to the Spaceship scripts to use the two new costumes when the ***up arrow*** key is pressed. Be careful to add these blocks, do **not** change the existing ***up arrow*** block.

Run the game and press the ***up arrow*** key to fire the rocket engine. Hey that looks good – but… when you release the *up arrow* key, the rocket engine still looks like it is firing ☹.



Luckily it is easy to check if the ***up arrow*** key is not being pressed in the clone's forever loop as shown.

Run the game again and move around the screen.

This should now look pretty good!

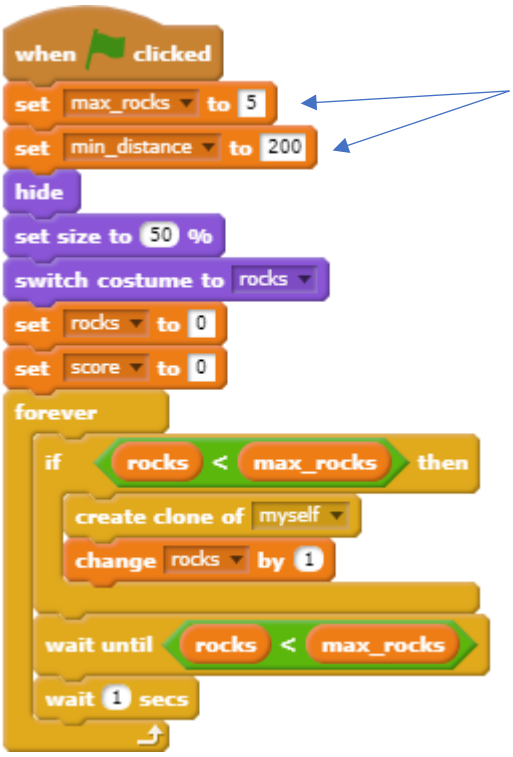
Save your project before continuing.

Step 8 - Add Asteroids

Add the sprite “Rocks” from the library, which will be used as our asteroid hurtling through space. To have more than one asteroid at a time, we will make use of clones again and use a lot of random numbers.

Create the following variables:

- ***max_rocks*** - a **parameter** setting how many rocks can be visible at a time.
- ***min_distance*** - a **parameter** setting how close to the spaceship a new rock can start.
- ***rocks*** - how many rocks are currently showing on the stage?
- ***score*** - to show the current score. Make this visible at the top of the stage (hide all the others).



Add this code to the scripts of *Rocks*.

I have put the two **parameters** at the top so you can find them easily to change how the game behaves.

Rock clones are created provided the number of rocks on the stage is less than the maximum number allowed.

Once the maximum number of rocks is showing, then the code waits until another rock can be added when they have finished moving, hit the edge of the stage, or been destroyed by the spaceship.

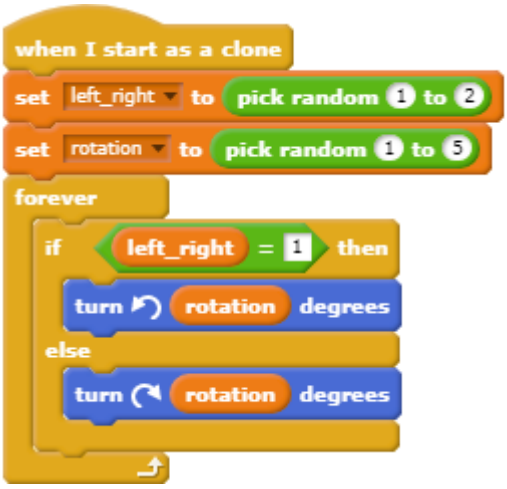
Now add code to define what happens when a clone is created.

The new asteroid starts at a random position further away from the spaceship than the minimum distance we have set.

It then glides to a random position where it is deleted.

Test your code and you should find the asteroids appearing randomly and moving across the stage then disappearing. As each one disappears, another one is added. But… this doesn't look very realistic!

Save your project before continuing.



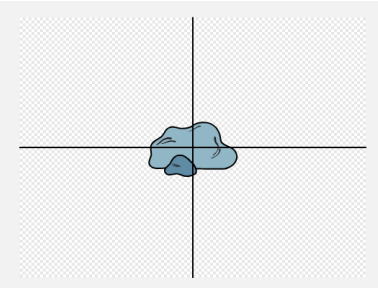
They would be more interesting if they rotated while moving across the stage. As they are gliding, this is very easy to do.

Define two variables which must be **for this sprite only**:

- ***left_right*** which says which way the rock rotates
- ***rotation*** which controls the speed of rotation

Now **add** the code shown – do **not** change the existing "when I start as a clone" which you have already made.

Run the game again and now the rocks should look more interesting.



If the rocks rotate in a strange way, this is probably because the centre of the costume is not in the right place. Use the crosshairs to put it roughly in the middle of the rock and try again.

Save your project before continuing.

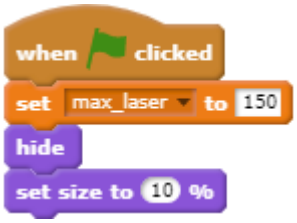
Step 9 – Add Laser Weapon

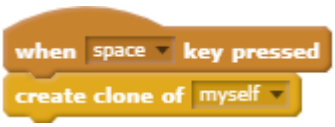
For the weapon system, we will need something which leaves a trail as it is fired. The answer is to clone a simple shape, which means we can have as many of them as we like on the stage at the same time.

Add a new sprite ***Ball*** from the library, which will be used to show the laser being fired.

Create a new variable ***max_laser*** to define the range of the laser weapon.

Add this code to the scripts of ***Ball*** to set max_laser to an appropriate value, hide the ***Ball*** sprite (as we only want to see the clones) and set the Ball to a useful size for the weapon.



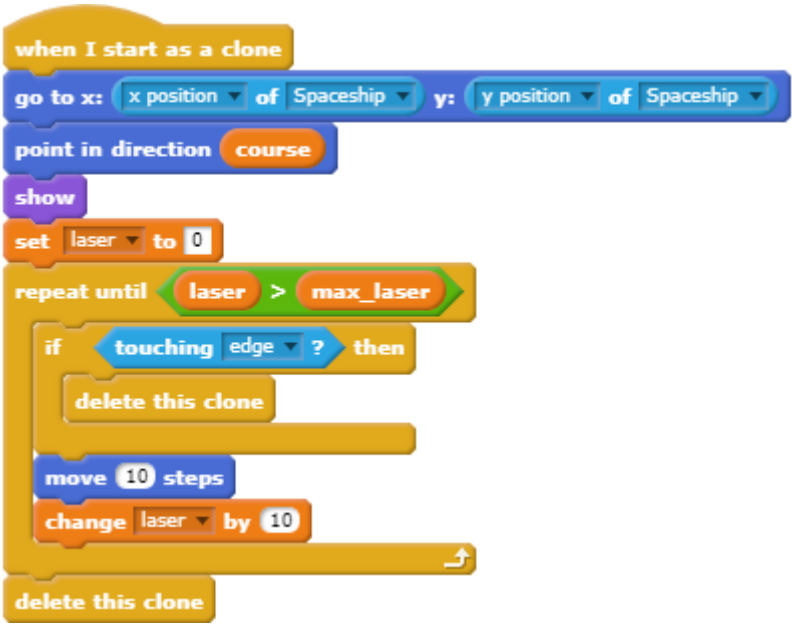


Now we need to decide which key to press to fire the laser weapon and add the necessary code for this. I suggest the space key and all we need to do is create a clone of the ball every time the key is pressed.

Create a new variable ***laser***.

Now add the code shown to say what happens when a clone is created.

- Start at the centre of the spaceship
- Point in the direction the spaceship is facing
- Move the cloned ball as far as allowed by the ***max_laser*** parameter
- Delete the clone if it reaches the edge of the stage before it gets there.

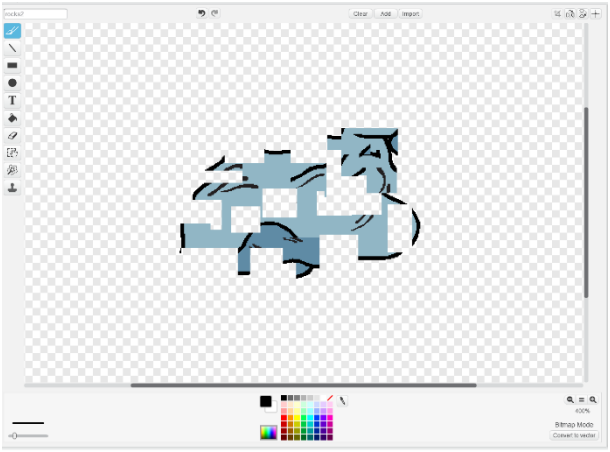


If you run the game, you should be able to see the laser weapon working in the direction the spaceship is pointing every time you press the ***space*** key.

Save your project before continuing.

Step 10 – Explode a Rock if the Laser Weapon Hits It

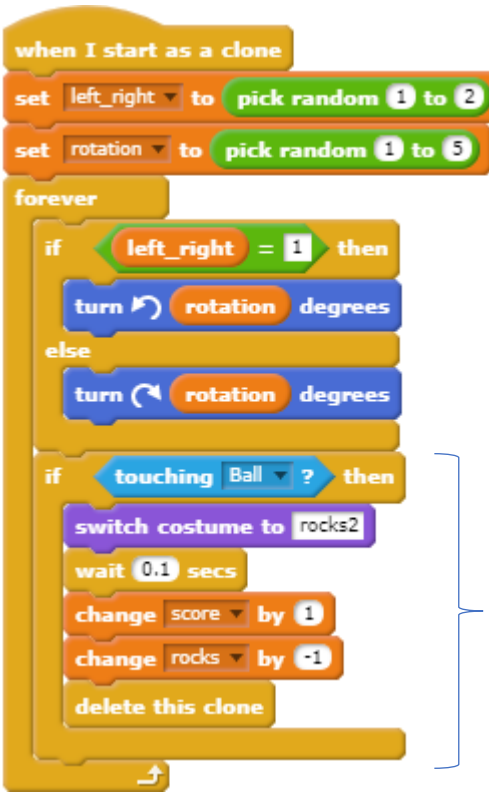
We want an asteroid to disappear if we hit it with the laser. The trouble is that the laser doesn't know which one it has hit because they are clones. Therefore we need to add code to the Rock sprite to detect when a clone has been hit.



First add a second costume to the ***Rocks*** sprite which will be used when the rock is hit by the laser.

In my version, I first duplicated the existing costume (to make ***rocks2***), used the ***Convert to bitmap*** button to change the image to **Bitmap Mode**, then used the ***Select*** tool to split the rock apart.

You can do the same or something different, but keep it simple and name the new costume ***rocks2*** so my instructions make sense.



Now we can **modify** the code by adding what to do when a rock is hit by the laser.

If touching the Ball which is our laser weapon, then:

- Change to the new costume
- Wait a fraction of a second
- Increase our score
- Reduce the number of rocks
- Delete this rock

Run the game and now you should be able to shoot at the rocks and make them explode using your second costume.

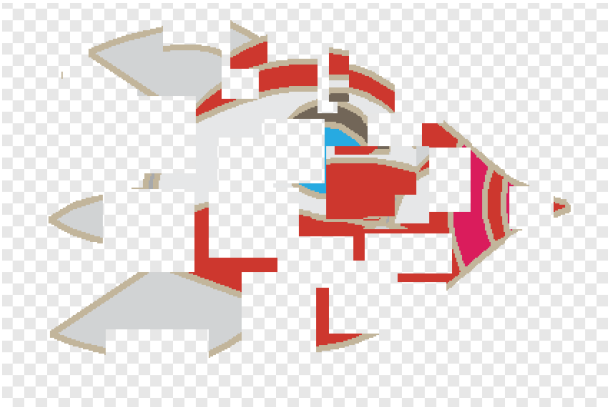
Don't forget that the laser only has a certain range, so you need to get the spaceship close enough to destroy a rock.

Each rock you destroy earns a point.

Save your project before continuing.

Step 11 – Make the Spaceship Explode if a Rock Hits It

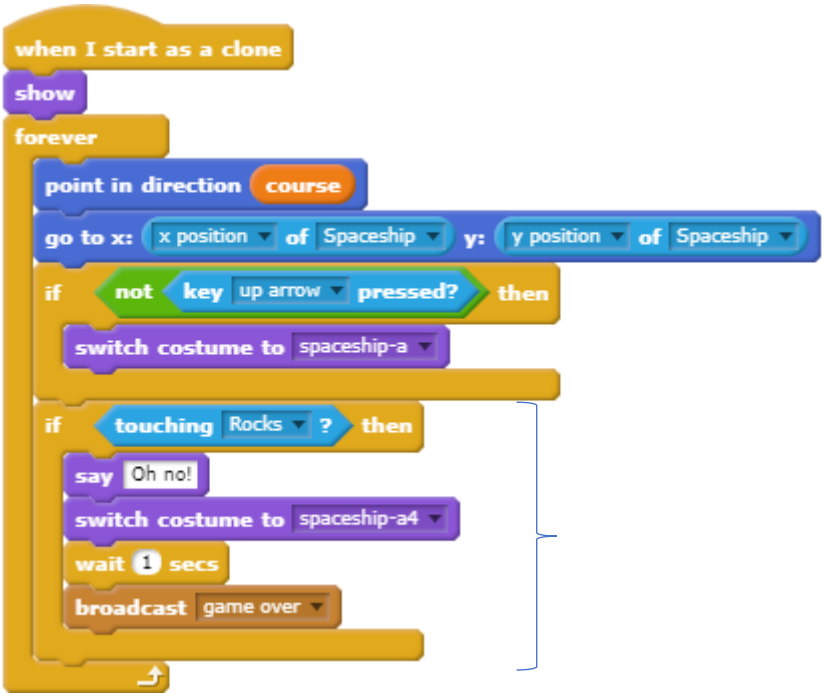
In this case, we can easily get the cloned spaceship to detect when it has hit a rock.



First make a copy of the Spaceship's costume ***spaceship-a*** which will create a new costume ***spaceship-a4***.

Modify ***spaceship-a4*** to give the impression of the spaceship exploding, like you did for the rock. Don't forget to convert it to bitmap first.

Here is my one.



Modify the code for cloned Spaceship by adding the blocks shown.

If you touch an asteroid then you

- Say a message such as “Oh no!”

- Switch to the crashed costume

- Wait long enough to see the new costume

- Broadcast a message to say the game is over.

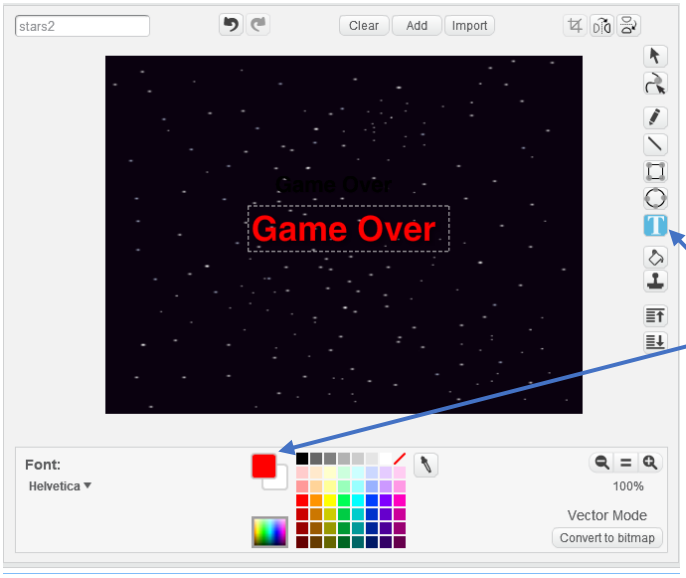
Test the game and allow an asteroid to hit the spaceship.

When this happens, you should see the “Oh no!” message and the changed costume but the game will carry on. That is only because we haven’t put in the code to say what happens when the ***game over*** message is broadcast.

Save your project before continuing.

Step 12 – Show a Game Over Message

There are many ways this can be done, but I will go for having a second Backdrop on the Stage.



Duplicate the ***stars*** backdrop which will create a new backdrop ***stars2***.

On ***stars2***, click the ***Convert to vector*** button at the bottom of the editing area. This will allow us to add text which we can mess around with later.

Select the Text tool and an appropriate colour, for example red, and write “Game Over”.

Adjust the size and position of the text however you wish, but big and central is probably best.

Here is my version.

Now we need to add two small bits of code to the Scripts of the Stage.

First what to do when the ***game over*** message is received which is to switch to the second backdrop with the “Game Over” message on it, then stop the game.

The second part is just to make sure that the ***stars*** backdrop is shown when the game starts.

Run the game again and let one of the asteroids collide with the spaceship. The explosion should occur as before but now the “**Game Over**” message will appear and the game will stop.

Save your project before continuing.

Step 13 – Make Your Own Improvements

You should now have a working and perfectly playable game. However, there are several things you can do to make it even better. Here are some suggestions:

- Try out different variations of the game's **parameters** to see which ones work best for you:
  - How quickly the spacecraft speeds up when the rocket engine is fired (***thrust***)
  - How quickly it slows down when the rocket engine is off (***friction***)
  - How far the laser gun will reach (***max_laser***)
  - How many asteroids can be visible at the same time (***max_rocks***)
  - How close to the spaceship new asteroids can appear (***min_distance***)
- Add appropriate sounds to missile firing, missing hitting an asteroid, and game over.
- The *Ball* sprite has five costumes of different colours – use these to make the laser weapon more interesting.
- Modify the game so that the spaceship has more than one life, with the number of lives being set as a new game **parameter**.
- Make the spaceship run on automatic control so that it moves and destroys asteroids by itself.
- Complete the original game by adding a flying saucer which can randomly appear and fly across the stage shooting at the spaceship.
- Add multiple levels which make the game more difficult by changing the parameters every time a certain score has been reached.