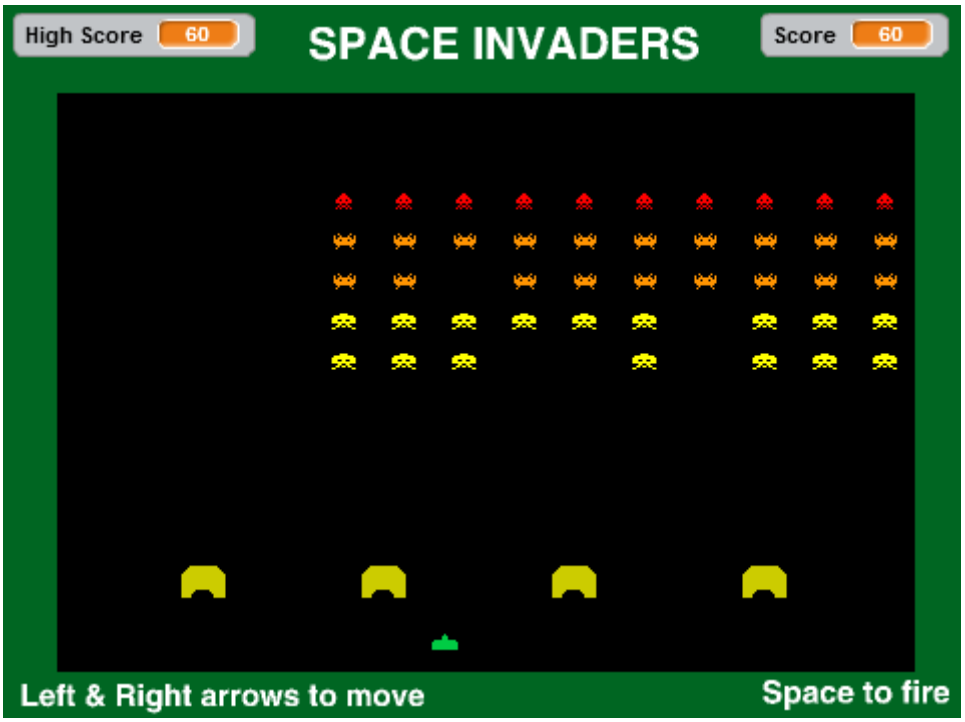
Introducing Scratch

7 – Space Invaders

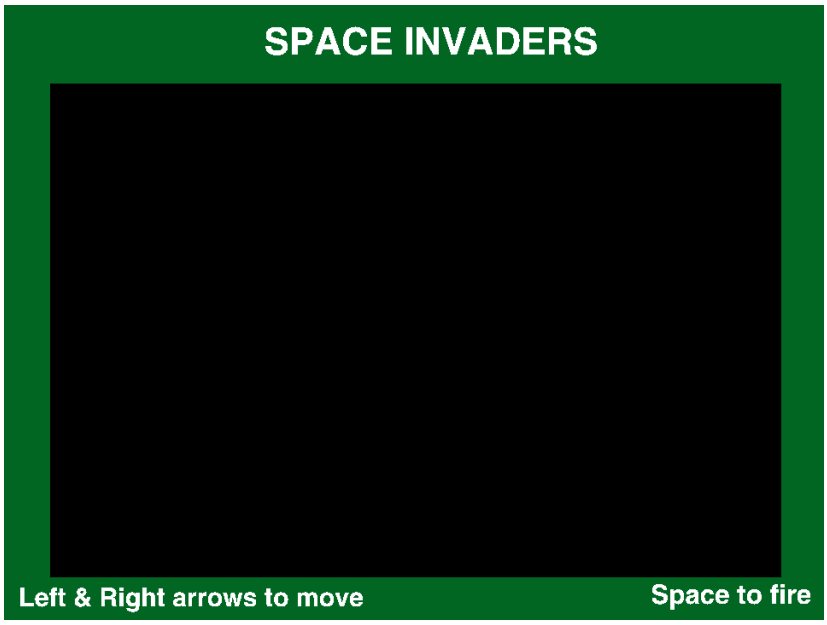
In this exercise we will recreate the classic video game Space Invaders. Although an old game like space invaders might seem quite simple there is actually a lot going on. Plus, there are a lot of sprites. So getting it working like the original will take some work.

To save you some of the work, you can use existing sprites and sounds that are similar to ones from the original game. Your teacher might be able to provide you with a copy of these sprites & sounds or you can download them as a zip file from the website at the bottom of the page.

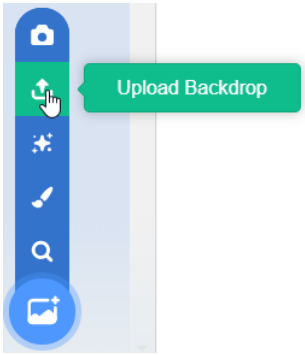


Exercise 1. Create the Background

1. Make a new Scratch file and delete the default cat sprite.
2. Modify the stage Backdrop to look similar to the following example.



3. If you have the collection of provided sprites, you can select your stage and then import the image called **Backdrop**. You could then remove the original blank backdrop, though it wouldn't make a difference to our program if the blank one stayed.

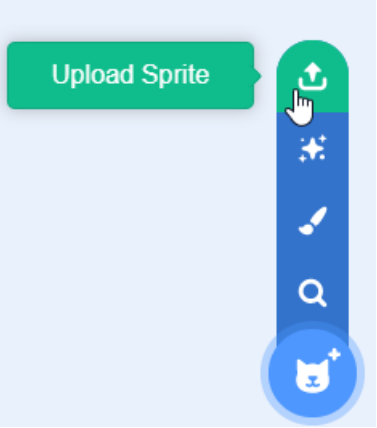


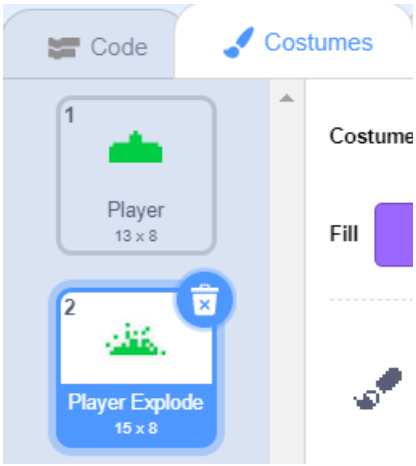
4. Save your project as **Space Invaders**. Remember to save often.

Fun Fact	In the original Space Invaders game, every time you destroyed an enemy ship, the rest of them began moving faster. That wasn't intentional. Because of the limitations of the hardware, the game simply played faster when there were fewer sprites on the screen. When the game's creators realised it got faster as you progressed they didn't mind since they thought it added more challenge. That 'feature' of speeding up wasn't actually in the code though. We won't be adding that 'feature' in our version but for an added challenge, you might be able to work out a way to make your invaders move faster as invaders are destroyed.

Exercise 2. Creating the Player Sprite

1. Create a new sprite and call it **Player**.
2. Paint 2 costumes for the sprite. One for the normal version of the player ship and one for the destroyed version of the player ship. You can import the provided versions of the sprites if they are available (upload sprite to import the **Player** picture and then upload costume to add the **Player Explode** picture as a second costume).





Tip If you are drawing them, zoom in close to make it easier.

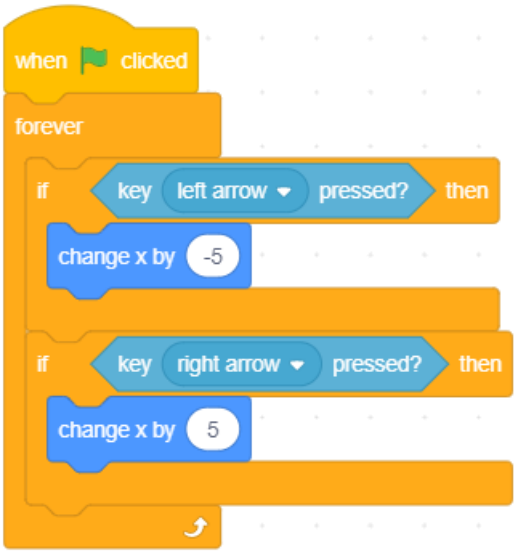
3. If you are left with a blank costume you can remove it.

Remember to make sure the costumes are in the centre of the costume editing area.

Exercise 3. Moving the Player Sprite

The player sprite needs to move when the left and right arrow keys are pressed which will be simple enough.

1. Add the following code blocks and test the movement with your arrow keys when done.



2. Add additional code blocks to set the starting position and costume for the player. Click the green flag to test when done.

Exercise 4. Adding Shields

The shields in the game don't do much other than provide a bit of cover for the player. In the original game, shots from the player or the enemies took pieces out of the shields but we will skip that in this version.

1. Add 4 new sprites for shields as shown below (the image is in the provided sprites).

Tip	If you want to position a sprite exactly, type the coordinates directly in to the sprite options at the top of the sprite area and then press enter.

The coordinates for each shield in the example shown above are:
-140, -120
-50, -120
50, -120
140, -120

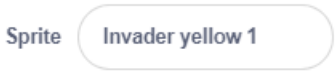
Important	Remember to save regularly. Now would certainly be a good time to save what you have done. There's a lot of work involved in a project like this and you wouldn't want to lose a lot of work if something goes wrong.

Exercise 5. Adding an Invader

Our final game will have a total of 50 invader ships arranged in 5 rows. These invader sprites will have more programming blocks than any other sprites. To make things easier, we will only make one invader to start with. Then when that is working correctly, we will make the rest.

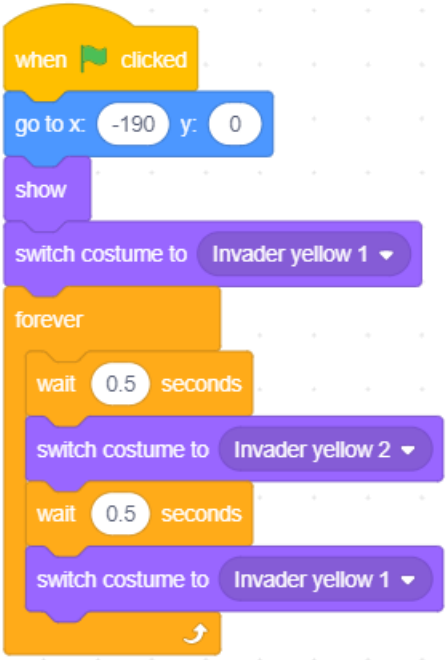
1. Create a new sprite called **Invader Yellow 1**.

There will be 2 rows of yellow invaders with 10 on each row but for now we'll just do one of them.



2. Create / import costumes for the sprite as shown.

3. Add the following code blocks.



Explanation - When the program starts this sprite will:

- Move to the starting position
- Set to show and set to display the first costume. This is because this sprite may have been hidden from being destroyed in a previous game.
- Switch between the two costumes every half a second.

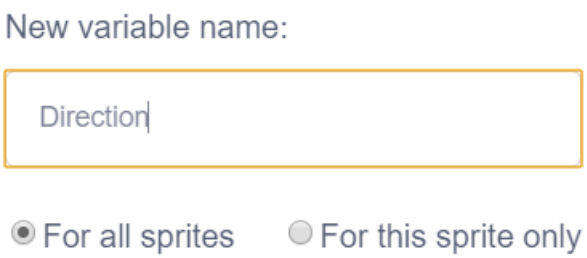
4. Click the start button to test the sprite so far.

Exercise 6. Adding Movement to an Invader

The invader will need to move sideways. When we have several invaders, they will move together in unison. When any of the invaders reaches the right of the screen, they will move downwards slightly (closer to the player). They will also continue moving in the opposite direction.

First we'll simply get it moving.

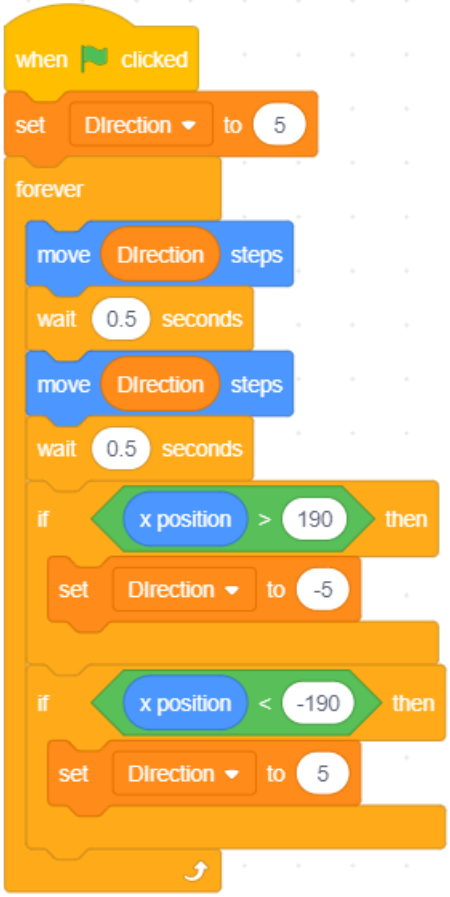
1. Create a variable called Direction.



We will use this variable to determine which way the invaders will move. When the invaders reach the right side of the screen, the direction variable will change to a negative number (so

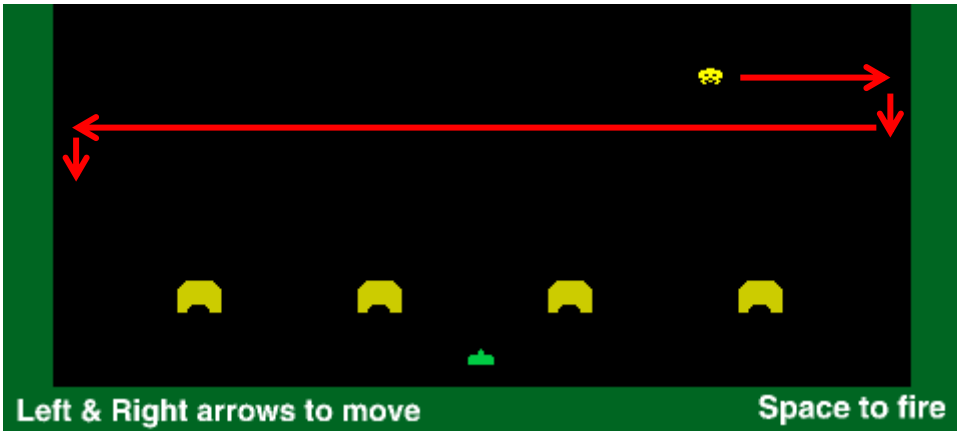
they will move left). When the invaders reach the left side of the screen, the variable will change to a positive number (so they will move right).

2. Add the following code blocks (you can place them next to your existing blocks).



Explanation - When the program starts this sprite will:

- Set the **Direction** variable to 5. This will be used to determine how far the sprite will move with each step.
- A loop begins
- The sprite will move a small amount, wait half a second and then move again. The half second pauses will be in time with the costume changes.
- An **if** block checks to see if the sprite has reached the right side of the stage (x coordinates of 190). If it has, the direction variable will change so that the sprite starts moving the other way (moving a negative x value goes to the left).
- Another **If** block checks to see if the sprite has reached the left edge. If it has, it changes direction to the right.



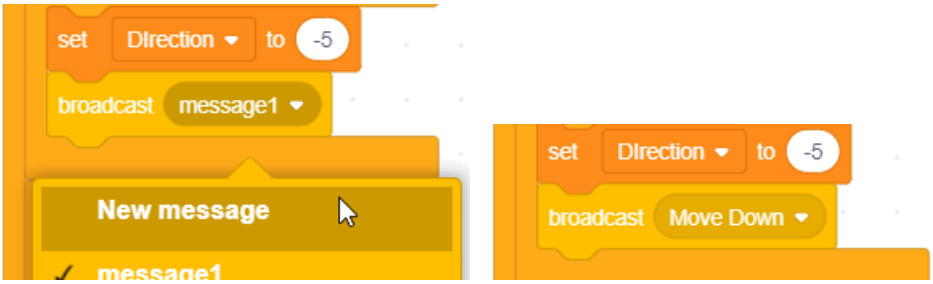
When the invader reaches the edge it changes direction but it also needs to move downward. That way each time they move all the way across the screen they move closer to the player. Eventually if the player isn't careful, they get to the bottom and the game ends.

We already have If blocks that detect when the invader reaches the side of the screen. We could add a block in there that makes it move down. The problem is that when an invader reaches the edge, they all need to move down. Not just the one that reached the edge. So how do we tell them all to move?

The solution is broadcasts. A broadcast allows a sprite to send a message to all the other sprites. In this case, when an invader reaches the edge, it will send a message to all the others that it's time to move down a bit. As invaders sprites receive that message, they will then move down.

3. Add a **Broadcast** block to the first **If** block.

4. Click the arrow at the end of the broadcast block.
5. Select new **message** and enter **Move Down** for the message.



6. **Right-click** the broadcast block and then select **duplicate**.
7. Put the new copy in to the other **If** block.

Now when the invader reaches the side it will send out a message. Remember that all the other invader sprites will have the same code as this one. Now we will add a block that will move the sprite downward whenever it receives a "move down" message from an invader.

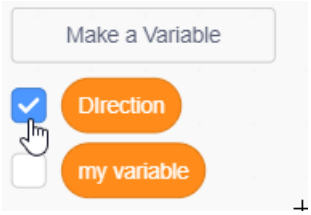
8. Add the following code blocks. Make sure the message is set to “Move Down”.

Now any time a sprite receives the Move Down message, it will move down (negative y value) as well as changing direction.

9. Test the program to check that the sprite correctly changes direction and moves down when it reaches the edge.

Notice that our **Direction** variable is showing on the stage. That can be useful while you’re testing to make sure it works but once you’ve got the movement working, you no longer need it showing.

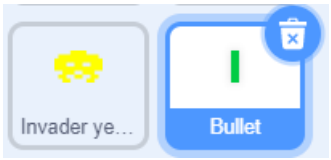
10. Go to the Variables category and turn off the tick next to the Direction variable to hide it from the stage.



Exercise 7. Add the Player Shooting

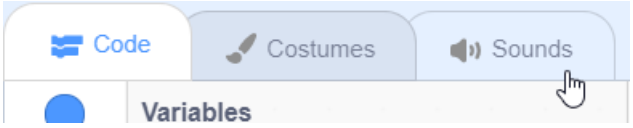
When the player presses the spacebar, the player sprite will shoot. When an invader sprite is shot, it will disappear and add points to the score. When all of the invaders have been destroyed, they will re-appear so that a new level can begin.

1. Create a new sprite called **Bullet**.
2. Draw / Import a small vertical line to act as the bullet.

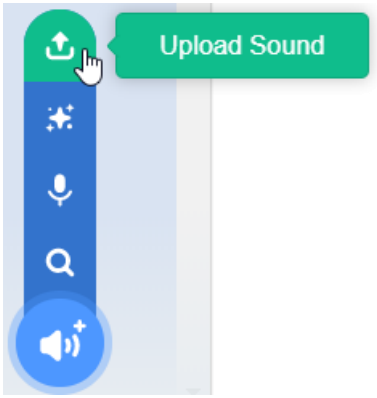


If you have the sound files available, then we’ll import the shoot sound. Otherwise you can skip these steps along with any pink sound blocks.

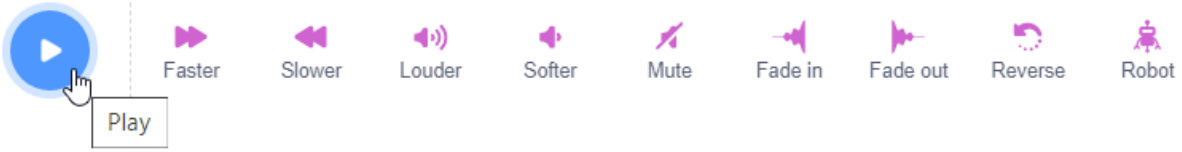
3. With the Bullet sprite still selected click the Sounds tab.



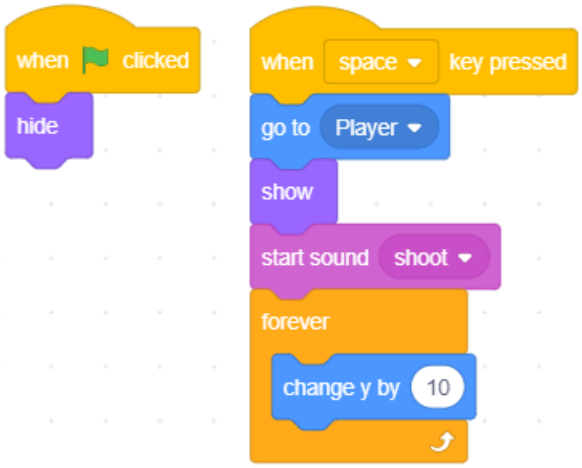
4. Move your mouse over the **Choose a Sound** icon at the bottom and then select **Upload Sound**.



5. Select the sound file called **Shoot**.
6. You can click the play button beneath the imported sound to preview it.



7. On the code tab add the following code blocks.



Explanation – When the player presses the spacebar the following happens

- The bullet will move to the current position of the player sprite
- The hidden bullet will now become visible
- The shoot sound will play
- A loop begins which will make the bullet continually move upward

If you press the spacebar it should work fine. The problem is when the bullet reaches to top it won't stop. We'll add some extra code to make it stop when it reaches the top.

8. Add an **if** block inside the **forever** block so that it looks like the following.

Explanation – The **If** block will check the current Y (vertical) position of the bullet. When it goes past a certain point near the top it will do the following.

- Return to the current position of the player sprite
- Become hidden again
- Stop everything in this script from running so that it no longer moves.

9. Test the program. Now when you press space the bullet will move upwards. This time though, when it reaches the top it will be ready for firing again.

Currently we have those shields that aren't doing anything but we don't want bullets to go through them.

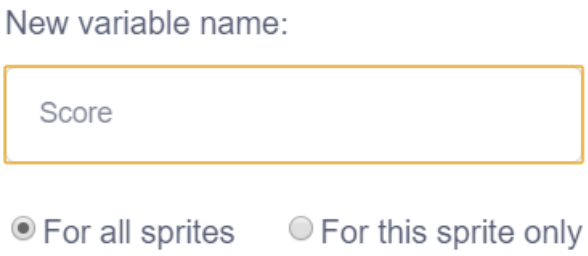
10. Add the following If section to your code so that bullets no longer go through the shields. The if block will activate when the bullet comes in to contact with the colour the shields are made of (click the colour square in the block and then click on one of the shields to select the colour).

Hint	The parts inside the **if** block are the same as the **if** block you just did so you can right-click and duplicate them.

Exercise 8. Destroying Invaders

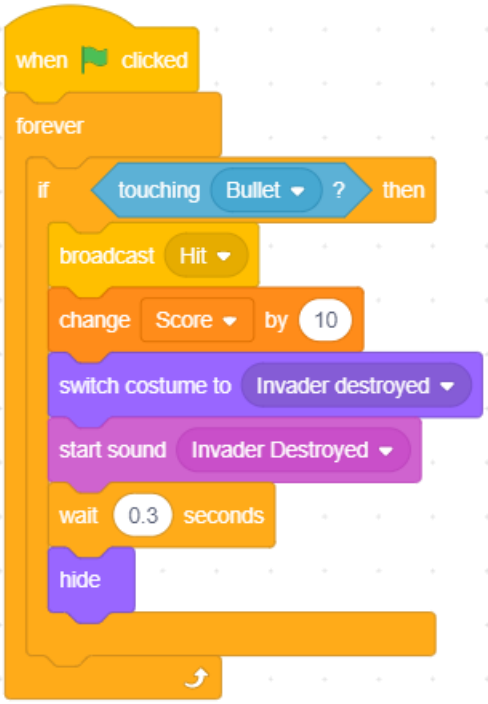
When an invader gets shot a few things need to happen. It will disappear. It will send a message to the bullet sprite so that it can move back to the player sprite. It will also add some points to a **score** variable.

1. Select the **Invader Yellow 1** variable and select the **Variables** category.
2. Add a variable called **Score**.



3. Add / Import a new costume for the invader being destroyed

4. Import the **Invader Destroyed** sound.
5. Add the following code blocks.



Explanation – When the program starts, a loop for this sprite will begin. Within that loop is an **if** block which checks to see if the sprite is touching the bullet sprite. When it is touching the bullet:

- A broadcast named **Hit** will be sent. We will use this in the bullet sprite so that it can return to the player sprite.
- 10 points is added to the score
- The costume changes to a small explosion
- A sound plays
- Slight pause so there is time for the exploded costume to display
- The sprite disappears.

6. Select the **Bullet** sprite and add the following code blocks.

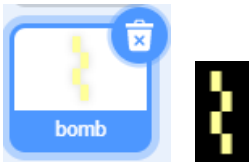
Explanation – When one of the invader sprites get hit the bullet will

- Return to the position of the player sprite
- Become hidden
- Stop other bullet code so that it stops moving until the player shoots again.

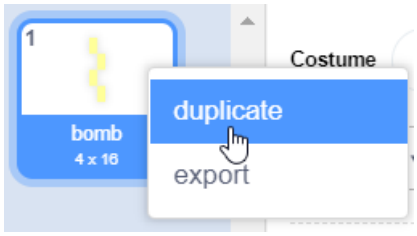
Exercise 9. Bombs Away

A game can end in 2 ways. When the invaders get to close to the bottom or when the player is hit by a falling bomb. In this section we'll add those 2 game end conditions. We'll start by adding the bombs.

1. Create a new sprite called **Bomb**.
2. Draw / import a sprite for the bomb.



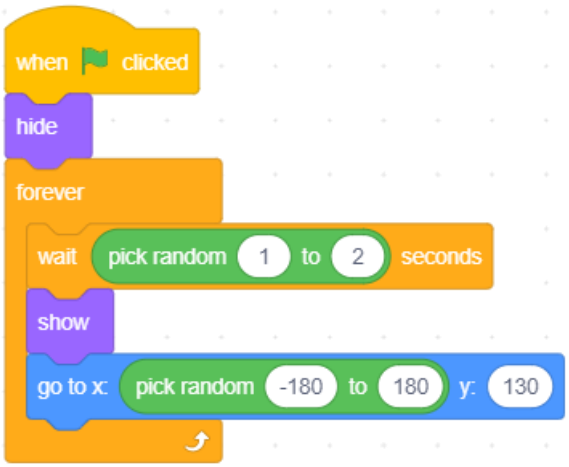
3. In the Costumes tab, right-click the costume to make a duplicate.



4. Click the Flip Horizontal button in the top of the drawing area so that the second costume is reversed.

We will make the bomb appear at the top of the screen in a random position after a random number of seconds. For the purpose of testing we'll make the delay short for now but we'll make it longer once it's working properly.

5. Add the following code blocks.

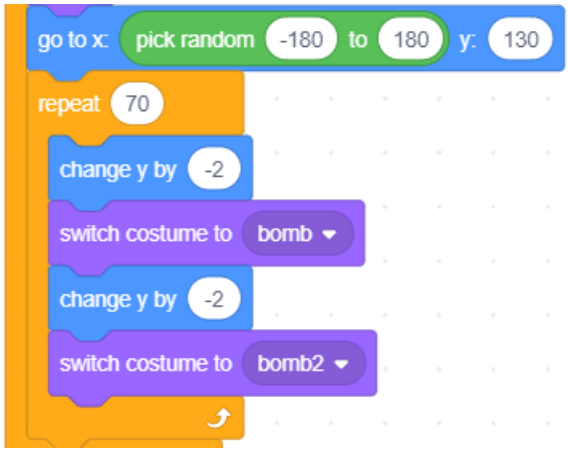


Explanation – When the program begins the bomb will be hidden. A loop then begins.

There will be a pause for 1 or 2 seconds (later we'll increase the duration). Then the bomb will appear in a random location along the top.

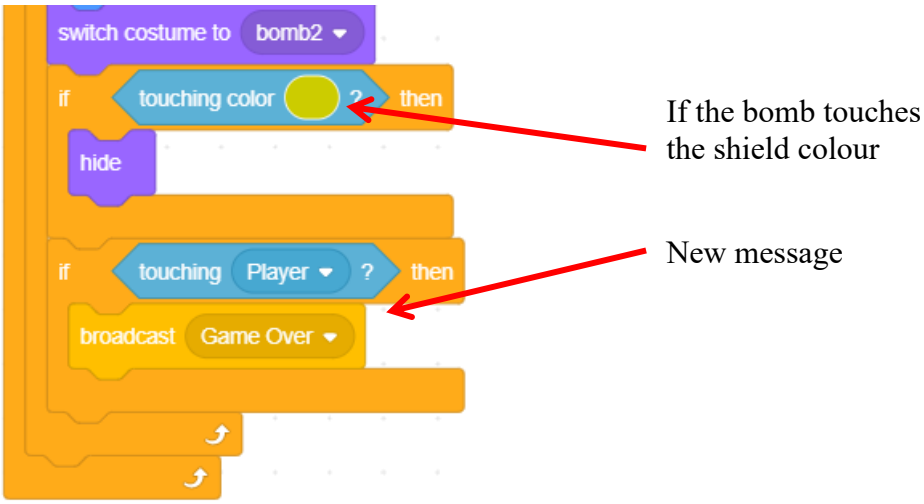
Now we'll add a loop that will make the bomb move downward while swapping between the two costumes. We only want it to keep on looping enough times for it to reach the bottom.

6. Add the following before the end of your existing **forever** block. In this example the loop will repeat 70 times but you may need to adjust the number of repetitions until it's just right.

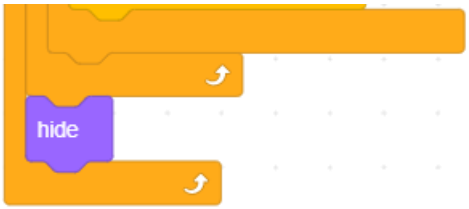


Now we need to add 2 **if** blocks. One of them will make a bomb hide and return to the top of the stage if it hits a shield, since it's not supposed to go through a shield. The other **if** block will send a "Game Over" broadcast when the bomb touches the player sprite.

7. Add the following 2 **If** blocks before the end of your **repeat 70** block.



Lastly, we'll add a hide block at the end of the main loop so that the bomb will disappear when it reaches the bottom. Otherwise it will stay there until it is told to re-appear at the top.

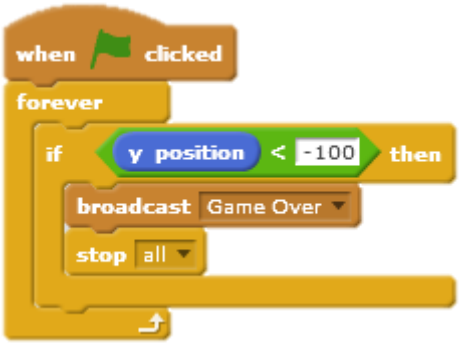


8. Select the **Player** sprite and add the following code blocks. Remember to import the Explosion sound in to the sprite before adding the play sound block.

9. Test the program. You might need to deliberately move under the bomb to see it work.

Our other game over condition is when the invaders get close to the bottom.

10. Select the **Invader Yellow 1** sprite and add the following code blocks.

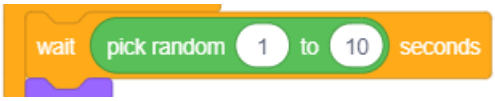


11. Test the game to make sure it ends correctly when the invader sprite reaches the top of the shields.

Tip	If you don't feel like waiting for the invader to move down several rows, you can drag it while the game is running. Simply drag it to a position a bit higher than the top of the shields so you don't have to wait so long.

Now to increase the random interval for the bomb and then make a few duplicates.

12. Select the **Bomb** sprite and change the wait interval amounts



13. **Right-click** on the bomb sprite and select **duplicate**.

14. Make 2 more duplicates (you can add even more if you want it to be extra challenging but 4 should do for now.

Exercise 10. Adding a Win Condition

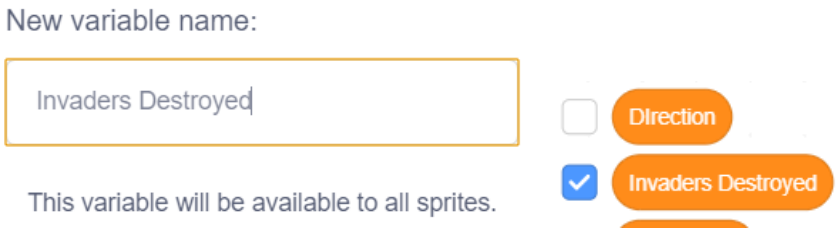
When all of the invaders have been destroyed, we want them all to appear again as a new wave of invaders. So it will be like starting another level with the score continuing. Like most things in programming there is more than one way to do it.

The way we will do it is with a variable. Every time an invader gets hit we will add 1 to the variable. Since we will eventually have 50 invaders, we will set it so that when that variable reaches 50, a broadcast message will cause each of the invaders to re-appear in their original positions.

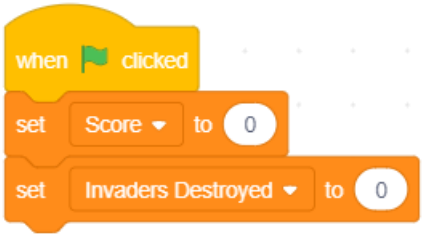
This will be a little tricky to test. We could add all the rest of the invaders now by copying our current one but that would mean that we would then have to add the invader reset code to each of them.

Instead, we'll add the new code now so there's less work, but we won't be able to test it until we add the rest of our 50 invader sprites.

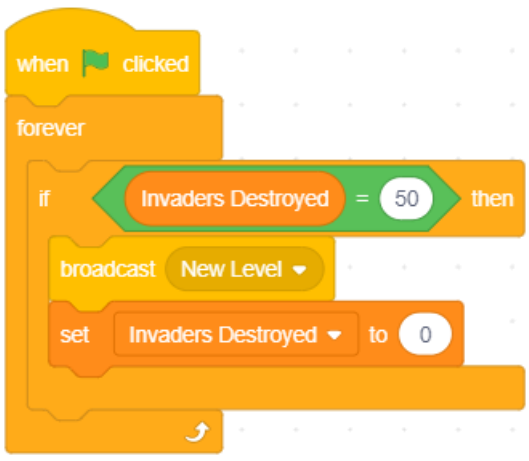
1. Select the **Stage** in the sprites area.
2. Create a new **variable** called "Invaders Destroyed".



3. Add the following code blocks. One that will reset the score to 0 at the start of each game and another that will reset our new variable to 0 at the start of each game.



4. Now add some new code blocks that will check to see if the Invaders Destroyed variable has reached 50. Once it has, a message will be broadcast and the variable will be set back to 0 so a new level can start.

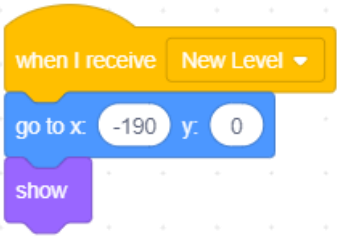


5. Select the **Invader Yellow 1** sprite.

We need to make 2 changes to this sprite. One that will add 1 to the **Invaders Destroyed** variable when this sprite is hit. Another that will reset this sprite to its original position when it receives the **New Level** message.

6. In the "touching bullet" section we'll add 2 new blocks. One to add to the variable and another that moves the sprite further up when it's hit. Otherwise it might keep on moving down and give the player a shock when a sprite they already destroyed reaches the bottom, making them lose. This will move it up enough to put it beyond any other invaders that are still in play.

7. Finally add the following blocks. These will move the sprite to its original position when it receives the **New Level** message.



Exercise 11. Invaders Everywhere!!

Our game has 1 invader. It needs to have 50. We could take that one sprite we have created, duplicate it so that there are 50 and then make individual changes to each one. That means a lot of sprites in our project and it also means a lot of hassle if we need to make changes to the invader code (changing it 50 times).

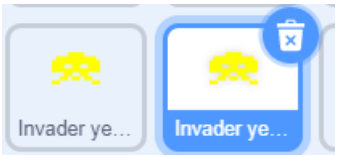
Instead we can clone sprites. That allows us to have several copies of a sprite appear. Each one appearing slightly different. We will have one sprite for each of the 5 rows in our invader formation. Then each of those sprites will be cloned.

We'll start by creating a sprite for each row in the formation.

1. Right-click on your Invader yellow 1 sprite and duplicate it. Change the name of the new sprite to **Invader yellow 2** (it might already be called that anyway).

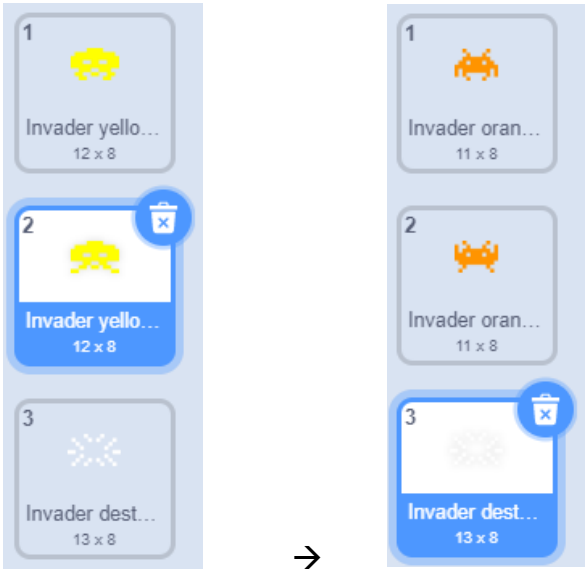
In the code there are 2 places where you have a block that has the coordinates. Change them both as follows.

2. You can change the order of sprites in the sprite area by dragging them. Drag the **Invader yellow 2** sprite so that it is next to the **Invader yellow 1** sprite. It makes sense to have similar sprites together.

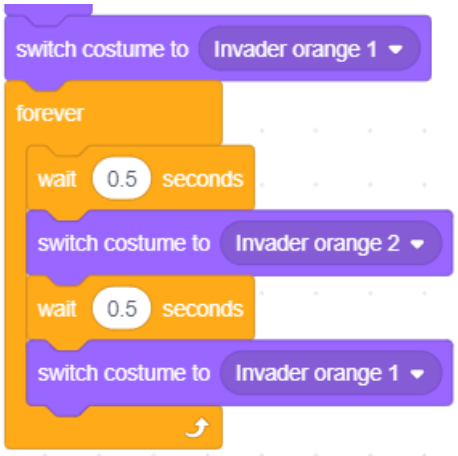


Next we will make the orange invaders. These will require additional changes since they will use different costumes and will award more points when they are destroyed.

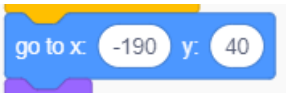
3. Make another copy of the sprite and call it **Invader Orange 1**.
4. Change both of the costumes by drawing / importing orange invader images.



5. In the code section change each switch costume block to refer to the orange costumes.



6. Change the starting and new level coordinates to the following

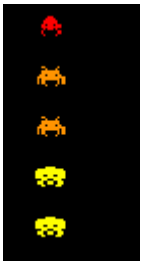


7. The orange ones are worth more points than the yellow ones. In the touching bullet code, change the number of points scored to 20.

8. Make a copy of this sprite called **Invader Orange 2**.
9. Change its starting and new level coordinates to **y: 60**.
10. Make a copy of this sprite and call it **Invader Red 1**.
11. Import or draw suitable costumes for it.

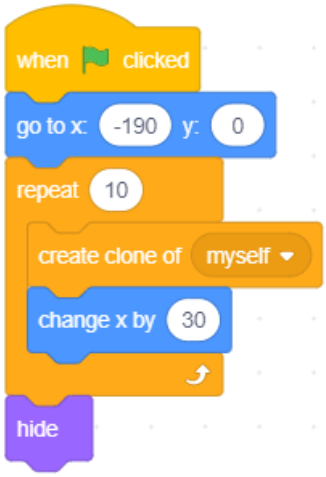
12. In the code section change each switch costume block to refer to the red costumes.
13. Change its starting and new level coordinates to **y: 80**.
14. Change the number or **points** scored to **30**.

Now when you test the game you will have 5 invaders neatly lined up and moving at the same speed.



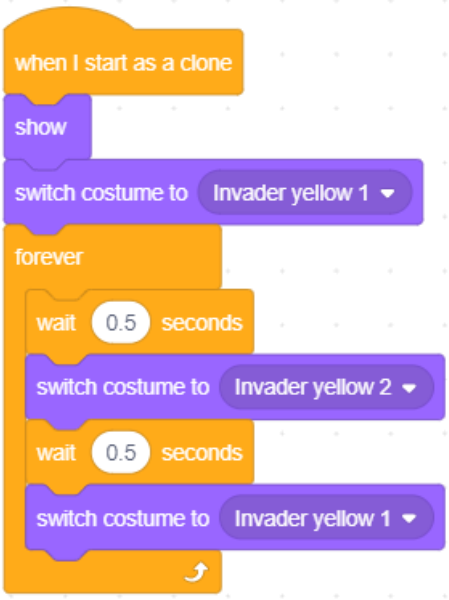
It's time to clone each of these sprites so that there are 10 copies of each.

15. Select the **Invader yellow 1** sprite.
16. Add the following code blocks.



Explanation – When the game starts, this sprite will be hidden and then 10 clones of this sprite will be created. Each one will be 30 pixels to the right of the previous clone.

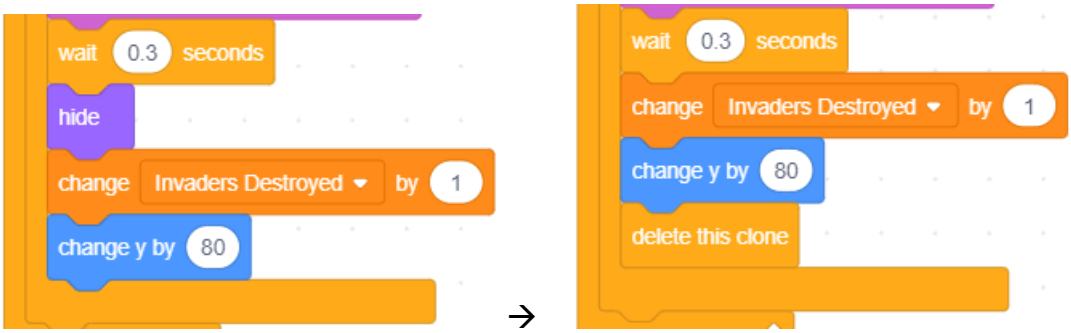
17. In the costume changing code block, we will replace the **when**  **clicked** block with a **when I start as a clone** block. We will also remove the go to block since we have now placed that in the code above and add a show block so that each clone is visible when it is created.



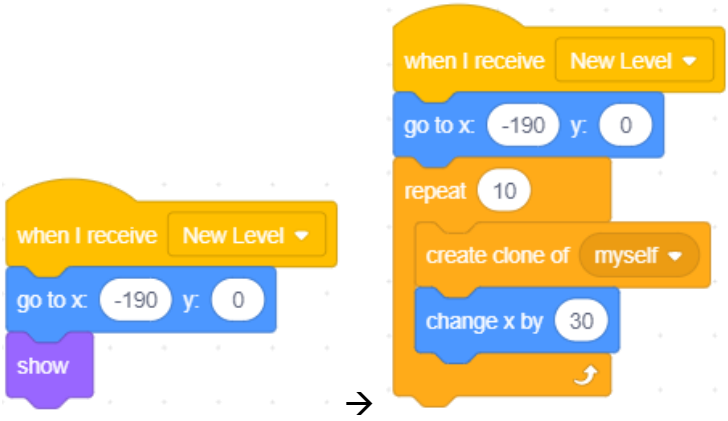
18. Change the rest of the code so that each **when**  **clicked** block is replaced with a **when I start as a clone** block. The only one that should still have a **when**  **clicked** block is the one we created on this page where the clones are created.
19. Make similar changes to each of your other invader sprites so that each one spawns 10 clones.

At the moment the game will work, until it's time for a level change. We need to make sure that each time an invader is hit, it is gone. Then it will appear again when a new level starts. To make that possible we will make a small change to the code so that instead of becoming hidden when an invader is hit, that cloned instance of the sprite will be deleted.

20. In each invader sprite make the following small change.

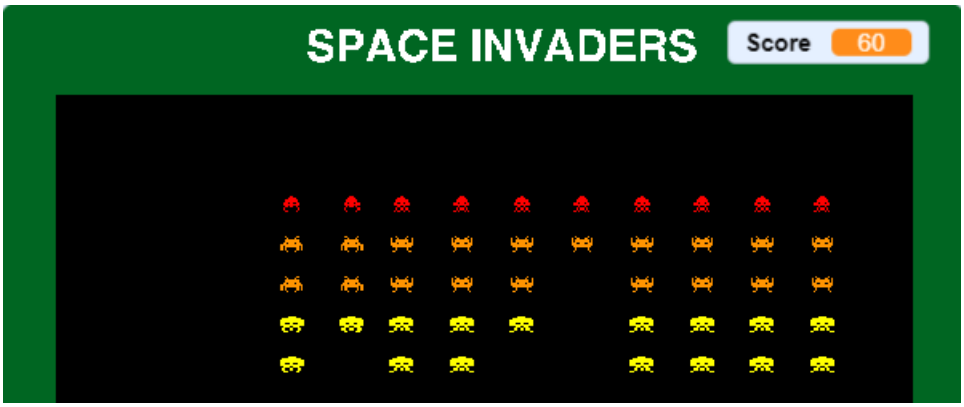


In the new level code, make the following changes (you can copy some of the blocks from your start blocks where the sprite is initially cloned).



Now if you play your game and managed to destroy all 50 invaders before they get you, they will re-appear in their starting positions ready for a new round.

21. Now that we've got the main parts of the game working, hide the **Invaders Destroyed** variable from your stage and move the **Score** variable in to a suitable spot.

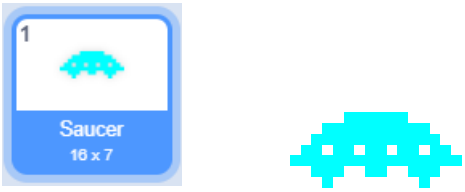


Exercise 12. Finishing Touches

Now to put the final touches on the game. Firstly, we'll add the flying saucer that would occasionally appear across the top during the original game. Then, we'll add a high score variable.

1. Create a new sprite called **Saucer**.

2. Draw / Import a Saucer costume.



3. Import the **Saucer** sound.
4. Add the following code blocks.

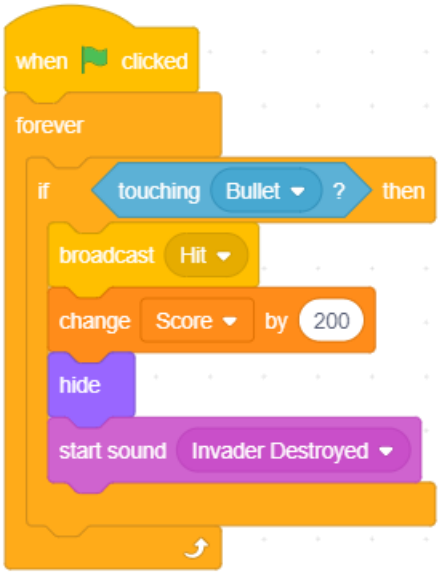
Explanation – When the program begins, the Saucer will be hidden and then a loop will start.

- Waits a between 10 and 30 seconds before appearing in the top left corner of the stage.
- Plays a sound.

- Starts a loop that repeats enough times for it to move to the other side (you might need to adjust the number of repetitions until it's about right).
- While the loop repeats, it will move and flash different colours.

The saucer can be shot just like any of the other invaders, only it awards a cool 200 points if you manage to hit it.

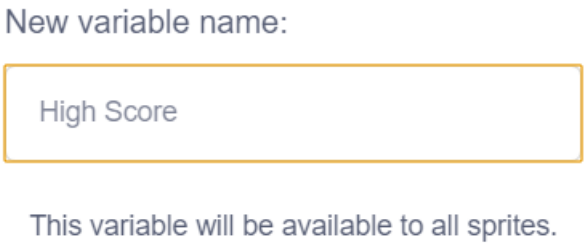
5. Add the following code blocks to the saucer (import the **Invader Destroyed** sound if you want that to play when this sprite is shot).



Exercise 13. High Score

Last of all we'll add a high score variable. When the score becomes greater than the high score, then the high score will increase to match it.

1. Select the **Stage**.
2. Create a **High Score** variable.



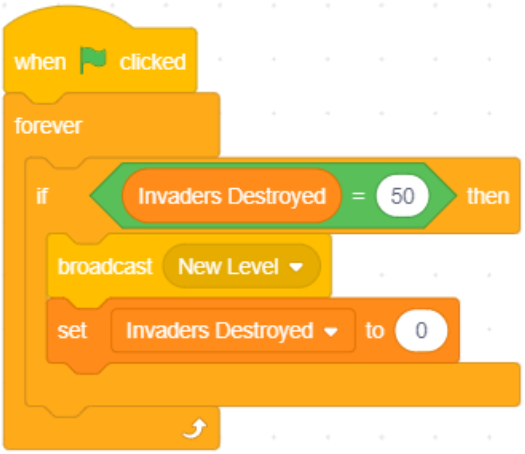
3. Add to the first block of code so that it looks like the following.

4. Move the high score variable to a suitable spot on the stage.

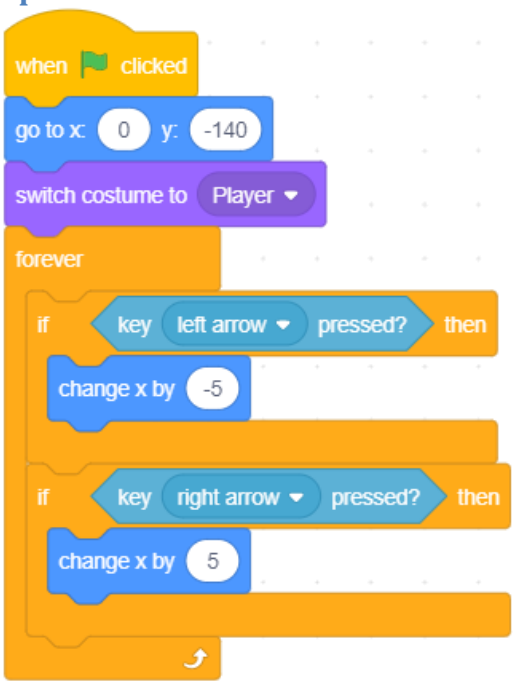
Exercise 14. Code Reference

Just in case you get stuck and want to check what your code blocks should look like, you can refer to the following.

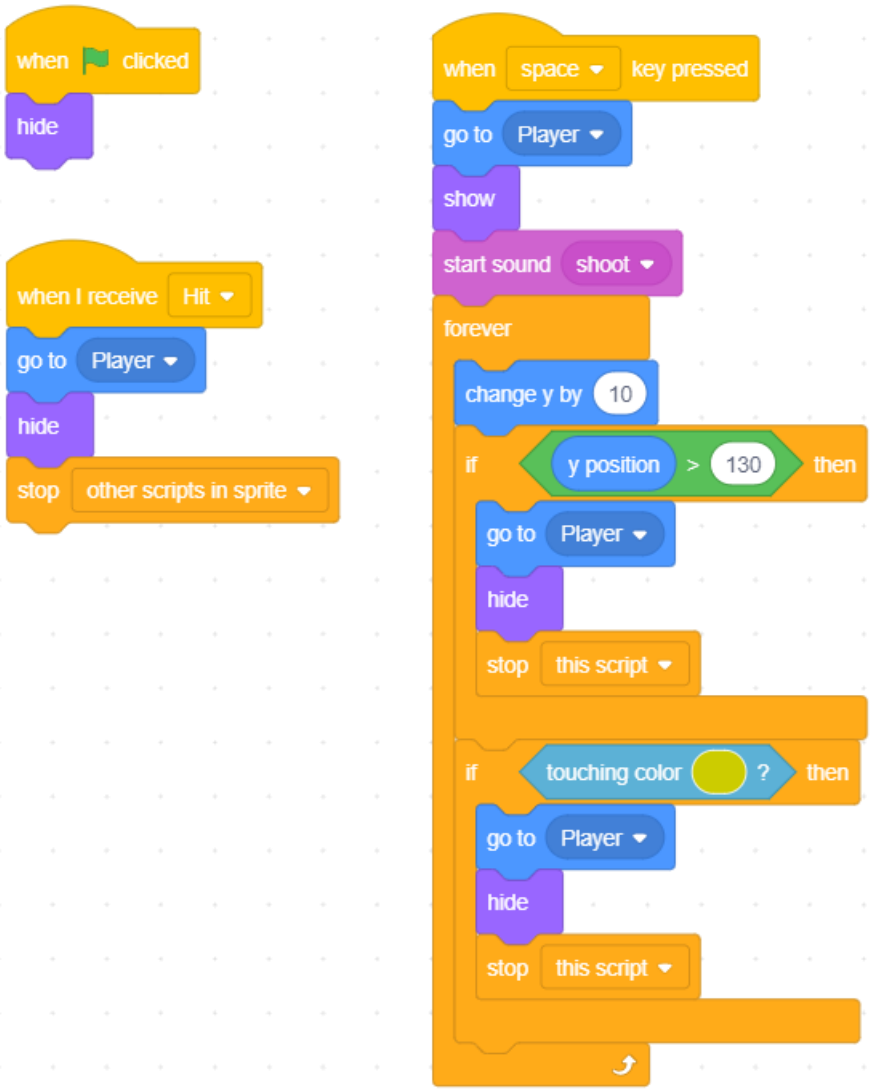
Stage



Player Sprite



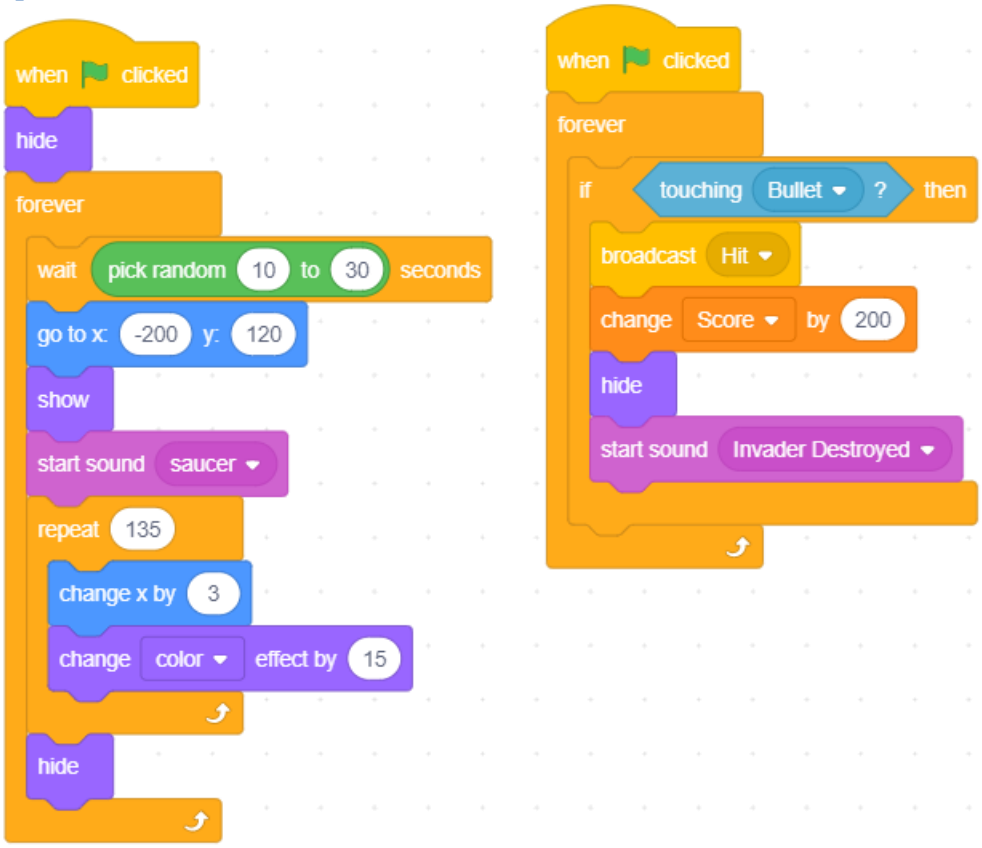
Bullet Sprite



Bomb Sprite


when clicked
hide
forever
wait pick random 1 to 10 seconds
show
go to x: pick random -180 to 180 y: 130
repeat 70
change y by -2
switch costume to bomb
change y by -2
switch costume to bomb2
if touching color ? then
hide
if touching Player ? then
broadcast Game Over
hide

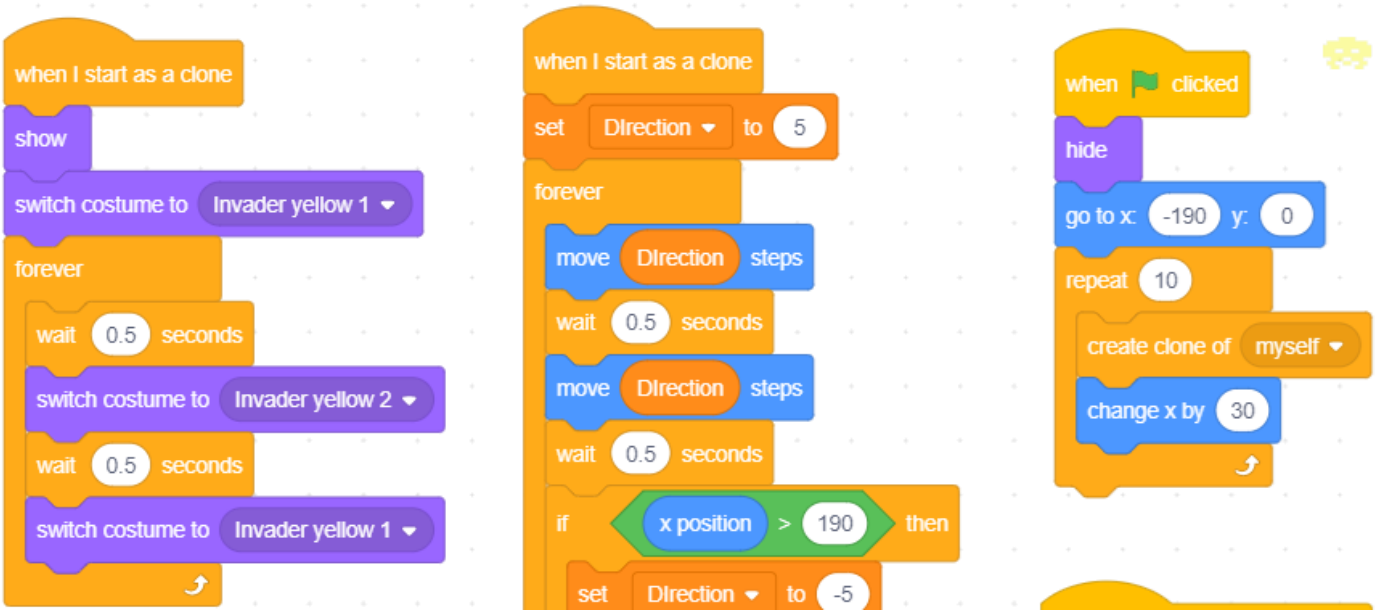
Saucer Sprite

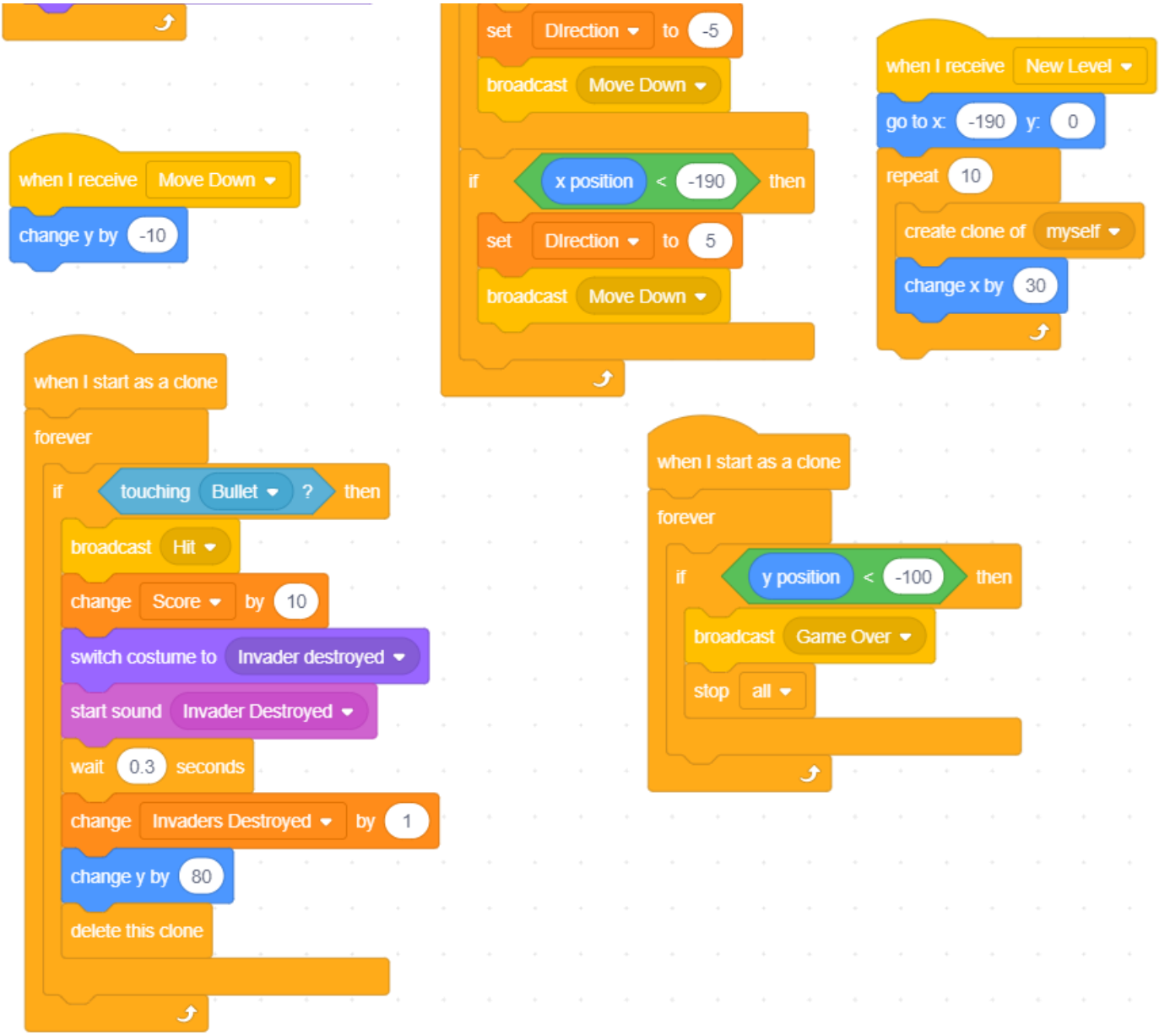


Invader Sprites

This example shows the first yellow one.

Remember that each invader sprite will have different coordinates, scores and costumes.




when I receive Move Down
change y by -10
set Direction to -5
broadcast Move Down
if x position < -190 then
set Direction to 5
broadcast Move Down
when I receive New Level
go to x: -190 y: 0
repeat 10
create clone of myself
change x by 30
when I start as a clone
forever
if touching Bullet ? then
broadcast Hit
change Score by 10
switch costume to Invader destroyed
start sound Invader Destroyed
wait 0.3 seconds
change Invaders Destroyed by 1
change y by 80
delete this clone
when I start as a clone
forever
if y position < -100 then
broadcast Game Over
stop all