Scratch – Space Invaders

Curriculum Connections:
SS 4.2 - Demonstrate an understanding of area of regular and irregular 2-D shapes
SS 5.7 - Identify, create, and analyze single transformations of 2-D shapes (with and without the use of technology).
SS 6.4 - Demonstrate understanding of the first quadrant of the Cartesian plane and ordered pairs with whole number coordinates.

Project Description

This year, one of the largest projects for our E-design campers will be to create their own video games. Using Scratch coding, the campers will follow along in the creation of a basic game. Campers will gain the knowledge required, through this step by step tutorial, to later create their own game.

NOTE: Campers will be doing this tutorial activity as only a base template for their later game.

Big Ideas

Campers will be taking their first steps into computer coding. They will be learning all about the basics of coding, consisting of movements, collisions, and conditional statements. The campers will learn the logic behind coding and gain an understanding of the dynamic world of video game creation.

Safety Considerations

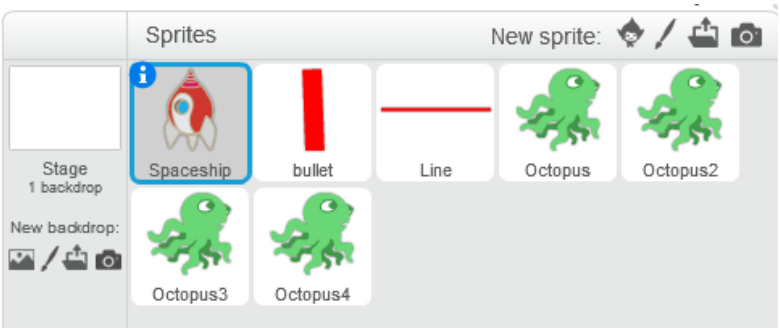
-none

Equipment

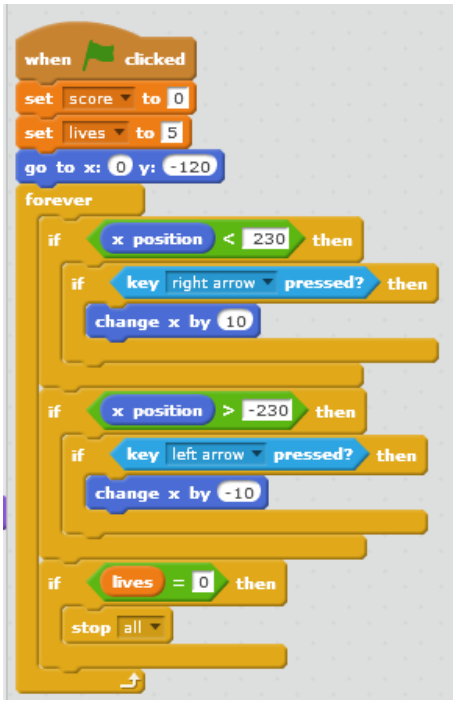
-laptops/computers – any way to access the internet
-https://scratch.mit.edu/ - this website will take you to scratch.

Procedure

1. To get to scratch, follow the link above. Scratch can be touchy, therefore try multiple browsers if it is not working.
   1.1. To begin, it is best to join scratch. Have the campers follow the instructions on the site, they can either use their own email, or use a generated EYES sign in of username-EYESyouth2018 password-summer2018. We will have them sign up to save their work for easier use later.
2. Once signed in, click the "create" tab to get started. If the flash player is restricting access, the camper will have to allow the flash extension.
3. Because we will be creating a general space invaders game, I will be going through a completed code piece by piece.
4. The first step in the game creation process is the creation of our sprites. For this, under the sprites tab, where it says "New sprite" we can choose and sprite from a pre-set list, we can draw our own, we can download one from online, or we could upload a picture file. I've chosen the spaceship, and two lines, one used as a bullet and one used as a border. For enemies, I've chosen an octopus. Initially there should only be one octopus, and because I want all enemies to behave the same, I will duplicate them once all the code is finished.



   Now you'll notice the space ship is the selected. This is when a sprite is selected, it will be the object to which we can apply code.
5. In the space ship code, I have done what is below. Now as we all know, there is no one way for anything, so this is only my way of doing this. Also, all the pieces of code are color coded in their own category, so if campers struggle to find pieces, they are findable by color.



Going top down, the "when 'green flag' clicked" will indicate that the code will only run when the game is started. This is going to occur in sprite, as I do not want to have enemies or other things moving around when the game isn't running. The score and lives we will discuss later as they are created variables.

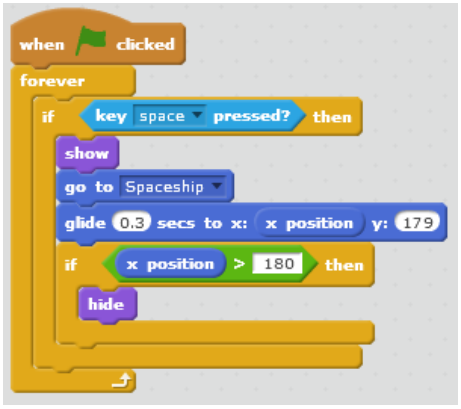
The first go to x = 0, y = -120 will set the position of the space ship near the bottom of the screen, in the middle. This is just to start the game in a good spot.

Next the big thing to discuss is the 'forever loop'. This loop will make it so that if a piece of code needs to act for the duration of the game, it will never stop until the game is over if it is in this forever loop.

Within this loop, there are multiple 'if' loops. These set the condition that if something occurs, some action will be executed. The first loop states that the inner loop will only happen if and only if the rocket's x position is less than the right most side of the screen. So IF the x position is less than the right side of the screen, and also IF the right arrow key is pressed, the rocket will move to the right. Following this logic, the rocket will never fly off the screen, as once it gets to the end of the screen, it will fall out of the if condition. This is reoccurring in the left direction in the next set of loops. Having the condition set on the left side of the screen, the rocket will only move left if the left button is clicked while the rocket is not near the left side of the screen.

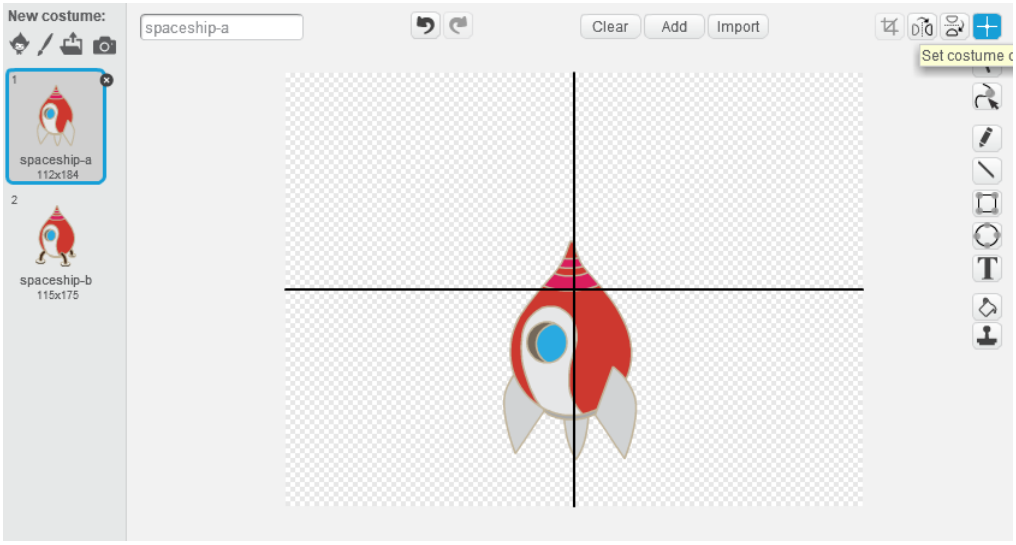
The final if loop will again be discussed later, as it uses the created lives variable.

6. Next, we will discuss the bullet code (remember to click on the bullet sprite to apply code to it):



This code starts again with the flag and a forever loop. As I want to be using the space key to fire, the first if statement is used to check when the space bar is clicked. Once clicked, the show code will ensure that the bullet will be visible as it travels. Once it is visible, the bullet will 'go to' the spaceship. This allows for the bullet to be shot from where we want it to be shot. Although with this, there comes an issue in the collision detection.

To ensure the bullet comes from the where we want it to, we need to check the center of our sprites. Inside the costume sprite, the cross-hair symbol will allow for scratch to check for the exact spot you want to be the center of the collision detection. Make sure every sprite is properly checked.

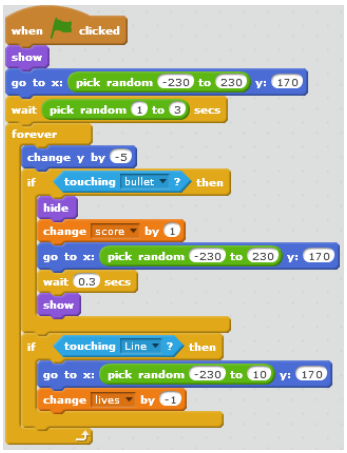


From here, the bullet will glide from the space ship position to the top of the screen in 0.3 seconds. The final thing we need to do is make sure the bullet doesn't stay at the top of the screen, so by creating that final if loop, we will make sure that if the bullet is at the top of the screen, it hides. The hide code will hide the sprite and stop its active collision detections. This hide code is another reason we have the show code at the top.

7. The next piece to look at is the border line:

This one is my personal favorite as it requires no code at all. All this line is used for is to have a collision for when the octopus get all the way down, we can lose lives and they can go back to the top of the screen.

8. The final major piece is the octopus. Like before, at this point there should only be one octopus sprite that will be coded and later duplicated.

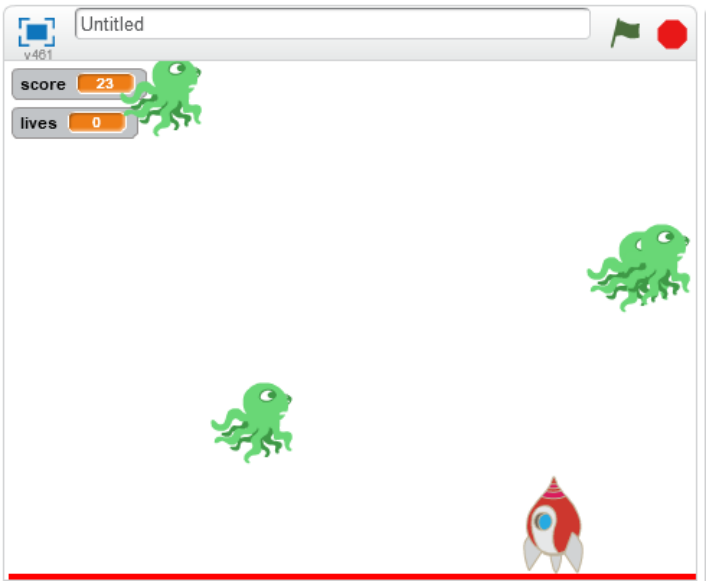


Again, we start with the flag, but then we have a show code. This was previously discussed, and only ensures the octopus are visible when the game starts.
The next part is the go to a random x value and y = 170. The reason for the random x value is so that it looks like the octopus start in an undetermined spot at the start. Now the y value is at the top of the screen because the game of space invaders is aliens coming down from space.
Now I added a wait code so that it gives the players a chance to get ready before the aliens come. Once again, the random was used so it seems more natural as a video game.
Now in the forever loop, we start with a change y block. By having a change by a negative number, the enemy will constantly fall. Now, because bullets are the only way we have to defend ourselves, we use a collision detection if statement, so that if the bullet touches the enemy, the enemy will disappear (hide), go to a new random x position at the top of the screen (go to x, y), have a small delay (wait), and show itself again before going back to falling. The score change will be discussed later.
Now because this is a game, we need to have a way to lose. The final if loop is going to set that. First, this is where that bottom line comes. If the octopus falls past the ship, it will touch the line. If this occurs, it will reset the position to that random x value at the top of the screen, incrementing the lives down.

9. For the lives and score aspect of this game, we will need to create our own variables. This is done in the ‘data’ tab. Under this tab we will create two new variable and apply them to every sprite. Now these variables only have a few code aspects, but the few we will use are the set and the change. Using the set code, I have set the score to 0 and the lives to 5. Because these variables affect the rocket, I put them under its script, but the set up could easily go under any script. Make sure the set-up is placed under a ‘when flag clicked’ and not in a forever loop for this instance.
When it comes to the change code, this will allow for the variable to change based on what happens. I’ve changed the score for every time the bullet collides with the enemy. Placing a change score by 1 under the collision of the bullet and enemy is what creates that winnable game. The

"change lives" is placed under the collision between the octopus and the floor line. This is used so that every time you miss an enemy, you lose a life. The final use of these variables is to end the game. To end a game, use the stop all code, but I've set the condition that if the lives hits 0 only then will the game end.

10. To set up the game, place all the sprites somewhere on the screen. Because of all our set up, there is no need to set things up anywhere specific. Now the final thing is that to make this game difficult, we can duplicate the octopi. To do this, right click the octopus sprite and select the duplicate. This will duplicate both the sprite and code, so don't do this step until you are ready to say the game is done! The final game will look something like this:



Follow up

The skills learned today will be transferable to your own unique game. Therefore, most ideas should be explored to a mastery. Unfortunately, there will always be some unanswered questions in every tutorial but play around with the code and use logic (and ask) to find your answer.

Extensions and Accommodations

In this activity, some cool extensions are trying to add a high score bar or add difficulty transitions. There are many possible extensions with scratch as it is a video game creator. Tips to make a high score bar, create a high score variable and set the high score to the current score only if that score exceeds the existing high score. To make incremental difficulties, you can change the speeds of the octopi either after a set amount of time, or if the player gets to a certain score.

Future...

Would you do this activity again? Did it turn out how you planned? Do you have any recommendations for the next person who chooses to do this activity?

Resources

- The noggin of Benito
- https://scratch.mit.edu/