Class 3 - X-Wing Project / Lists

Tyler Bounds

April 20, 2018

Introduction to the Project

- This project is designed mainly to teach Scratch's list data type. This project will also cover how to remix a project as well as how to use someone else's project in their own, with proper crediting.
- The first step is to have the students remix the **XWing Kids Coding Club** project. They can either just search the project name from Scratch's main page or search for my studio **tbounds42**.
- Click the project, then **See inside**, and then click **remix**, the orange button in the top-right.
- After they have remixed it, walk them through the high-level details of the project.
  - Stage - Contains the backdrop as well as the scripts for animating the backdrop.
  - X-Wing - This is the player. They can move and shoot currently.
  - Laser - This is the projectile the player fires. It collides with the enemies.
  - Tie Fighter - This is the enemy that spawns at the top, flies straight to the bottom, and dies to the lasers.
  - Gem - This is currently just a sprite with a collection of various gem costumes.
  - UI - This is just a floating sprite meant to hold the text for the scores.

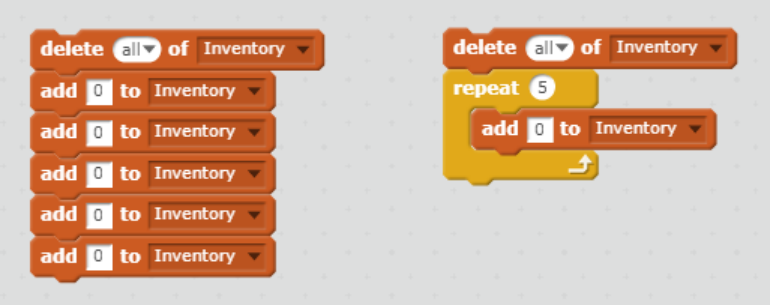
Introduction to Lists

- Lists are exactly what like what they sound like, a list. Think grocery list, or attendance list, or Christmas list. They are containers for data that are stored in sequential order.
- In Scratch, you can find them under the **Data** tab. If you do not currently have a list, all you will see is the option to **Make a List**.
- We are going to create a list now. Make sure the **X-Wing** sprite is selected and create a new list **For all Sprites** called **Inventory.**

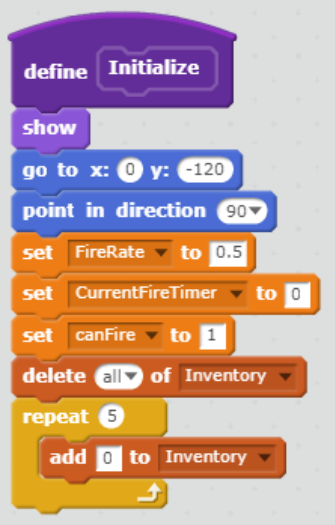
- We can now see the options we have for lists:
  - Add - adds data to the end of a list.
  - Delete - deletes data at a particular spot in the list.
  - Insert - adds data to a particular spot in the list.
  - Replace - rewrite the data at a particular spot in the list.
  - Item - is the value of the list at a particular spot.
  - Length - the number of spots in a list.
  - Contains - returns whether or not a particular value is in the list. True/False

Initializing Player Inventory

- The first thing we are going to do is **initialize the Inventory list** to be the size that we want and empty. This can be done in a couple of ways:
- Show the method on the left first and ask the class if they can come up with a shorter way to write the same thing.



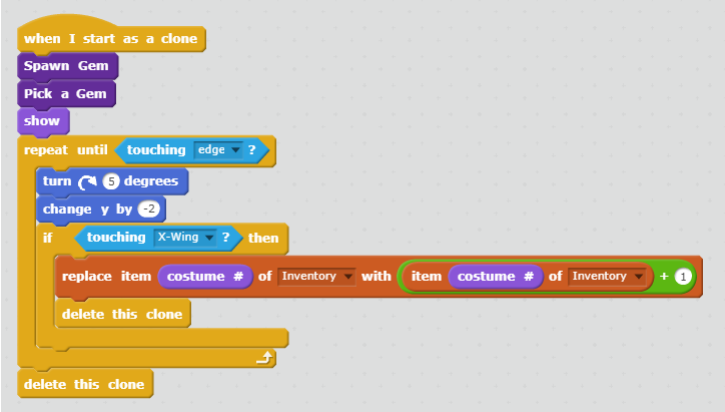
- Both methods work but the method on the right is smaller and more concise. Whichever you want to use is fine. Add your method to the bottom of the **Initialize** block like so:



- Now we need something to put into the inventory. What we will be collecting are gems from defeating Tie Fighters. We will now be working in the **Gem sprite**.

Creating Gems

- I have already provided some skeletal code for us to work off of.
- The **Clone Event** will contain the behavior for the clones. The **Spawn Gem** block will handle where the gem clones are spawned. And the **Pick a Gem** block will handle what kind of gem will be spawned and at what percentage. Continue to work on the **Clone Event** block until it looks like so:

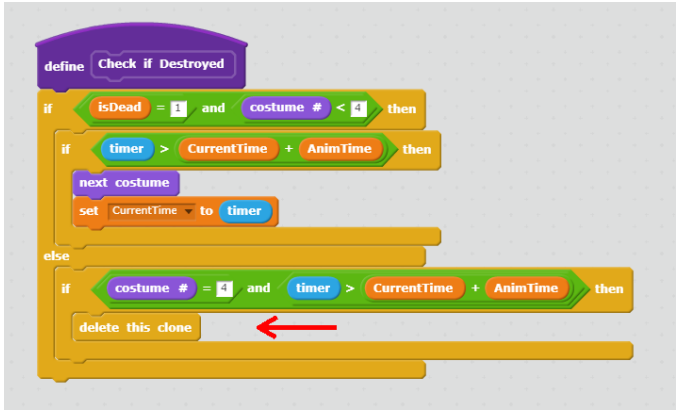


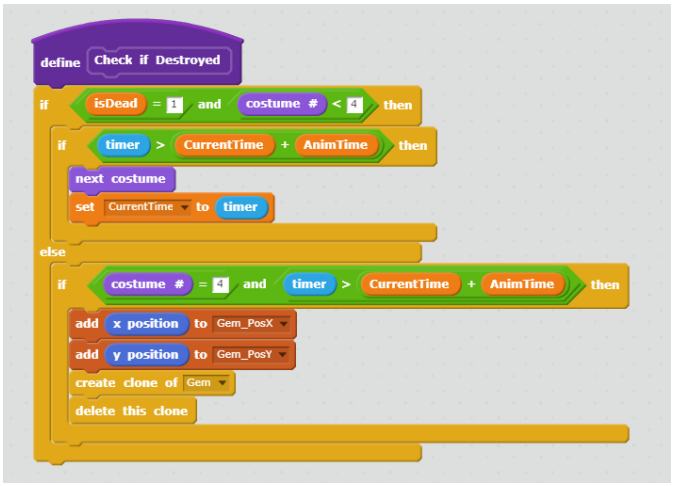
- The way the **replace item** block works is by associating the gem sprite's costume number to an index in the Inventory list. It then fetches that value and adds 1 to it.

Spawning Gems

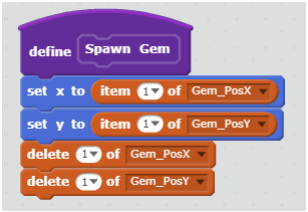
- Ask the class where they think the gems should spawn in and under what condition (when a tie fighter is destroyed).
- Proceed to the **Tie Fighter** sprite. Go to the **Check if Destroyed** block and show the class where the clone is destroyed. We want to spawn a gem right before the clone is destroyed right where the clone is. But how can we do this? We need the clone's coordinates before it is destroyed. We will save these in lists. **Create** two lists **For all Sprites: Gem_PosX** and **Gem_PosY**.

- Add the coordinates of the clone to the appropriate list and spawn a clone of the Gem. Remember, this is the **Tie Fighter** sprite.

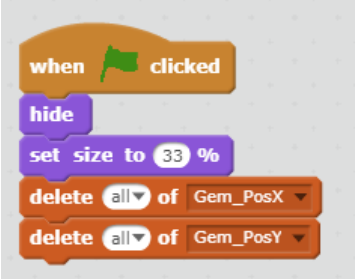




- We have saved the coordinates for the destroyed tie fighter clone. Now we need to use them. Go to the **Gem** sprite. Add the following code to the **Spawn Gem** block:



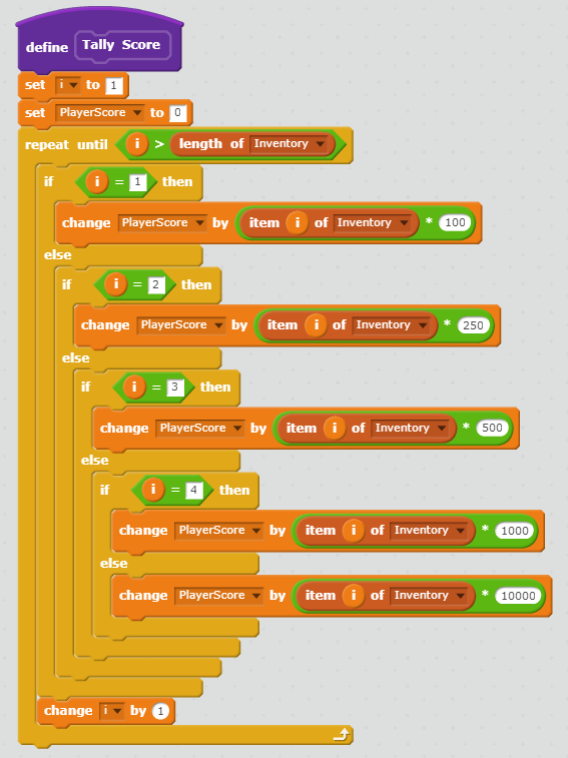
- To finish up the **Gem** sprite, make sure to finish the **Green Flag Event** by clearing the Gem position lists:



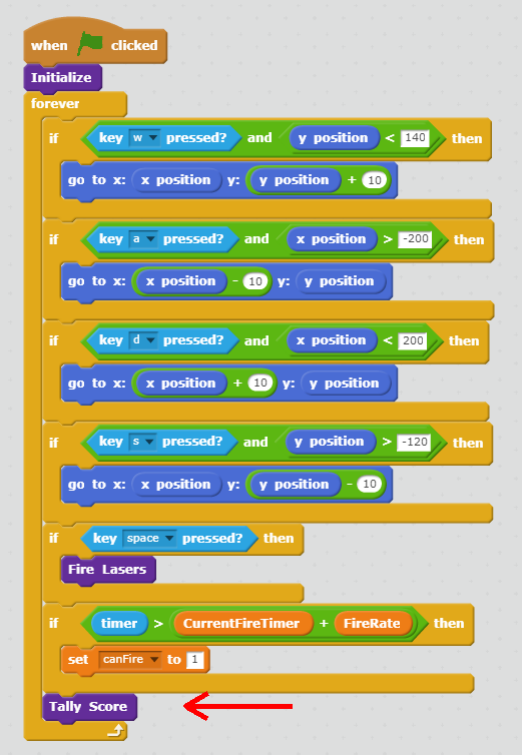
- Test it. Make sure the Inventory list is checked so that it shows up in the game. Collecting gems should increment the values in the list.

Keeping Score

- Now we are collecting a variety of gems and keeping them in our inventory. Let's give our player a score based on the gems they have collected.
- Go to the **X-Wing** sprite and create a global variable **PlayerScore**. Add the following code to **Tally Score**



- and then the **Tally Score** block in the **Green Flag Event** at the bottom but inside the forever loop. Like so:

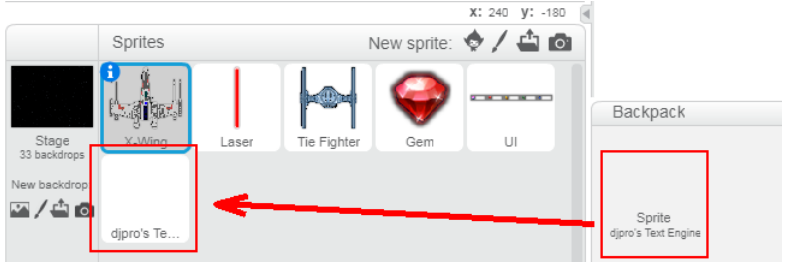


Text Renderer

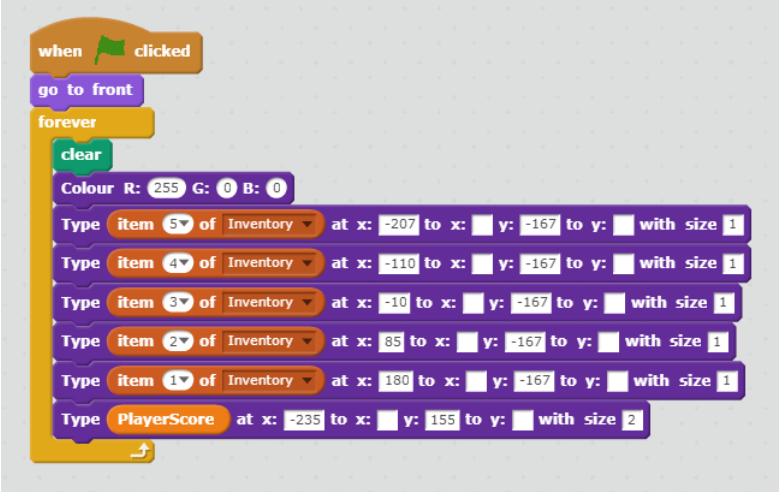
- Using Scratch's built in data tracking isn't glamorous and worse, it blocks a lot of our screen when we are playing. Let's integrate someone else's project into our own to be able to render text to the screen and keep track of our score.

- In a new tab, navigate to Scratch's home page. In the search bar, search for either: **djpro** or **Pen Text Engine** and find djpro's pen text engine. Click **See Inside**, open up your backpack, and drag the first sprite (**djpro's Text Engine**) into your backpack. This may take a bit of time.

- Go back to the tab that has your project, make sure it is saved, and refresh it. Open your backpack up and drag the text renderer from your backpack into your sprite section.



- Find the **Green Flag Event** (it should be to the right) and delete it and create your own. This is what it should look like:



Giving Credit

- Make sure to credit those who helped you out. In the top right of the the project page, click **See Project Page** to add your games information and credits. Make sure to include the controls for your game.

Ideas to Go Further

- We don't really have a game at the moment, we can't die and there's no real goal to play other than to collect gems. Here are some ideas to take the game to the next level"
  - Make it so the player can die.
  - Keep track of waves, maybe add how many waves the player has survived to their score.
  - Make different kinds of enemies or spawn patterns, or make the enemies shoot their own lasers.
  - Add power ups and upgrades.