Assignment 5

Process Work:

My idea for the godot game is to make a suika game while making sure to use godots built in physics system.

Suika game / watermelon game is a japanese puzzle video game where the game involves players trying to build a high score by dropping fruits into a container without it overfilling. (Wikipedia)

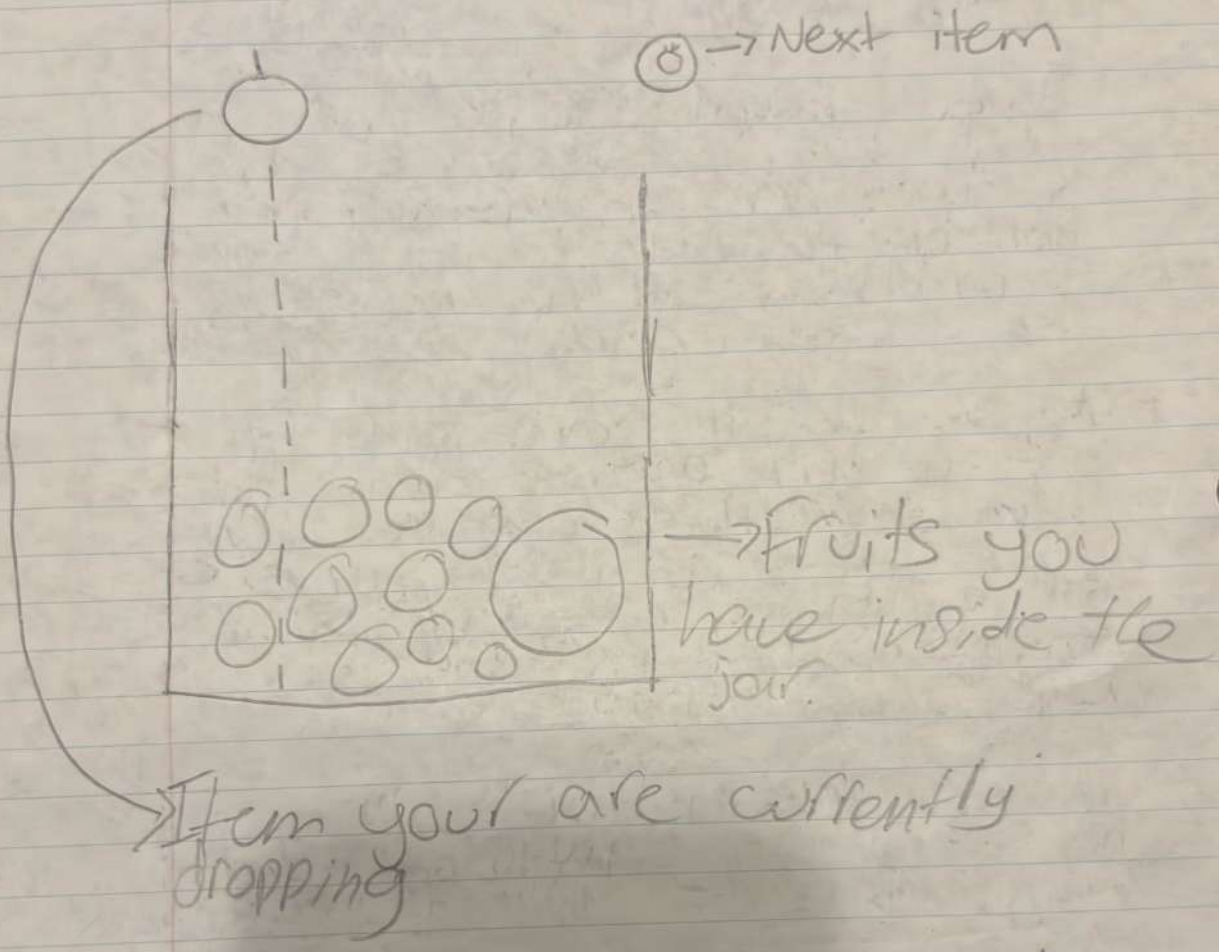
When 2 fruits merge together that are the same they become a bigger fruit. For example 2 blueberries = 1 strawberry 2 strawberries = 1 peach so on and so forth

The game goes like this

Smalles to biggest

Cherry, Strawberry, grapes, dekopon, persimmon, apple, nashi, Pear, Peach, melon, titular watermelon

My plan to do this game idea on godot is to first make all the shapes circles.



Above is a very broad idea of how I would like the final game to actually look.

To add my own idea into this game I want to make it where the fruit you are

about to put in the basket is constantly moving, and you must press the spacebar at the right time if you would like for it to go into a specific spot.

I have never used godot before so I will be needing to do some studying prior to me starting to work on the code, to make sure I understand all the necessary features the engine has to offer.

My plan for the game is to try to get the main program done and have everything actually working first. Then see if I have some spare time on my hands to implement sprites/assets to make my game look more fun to play.

The class on tuesday will hopefully help me get setup with godot and the main features I will be using for it.

Wednesday November 27th 2024

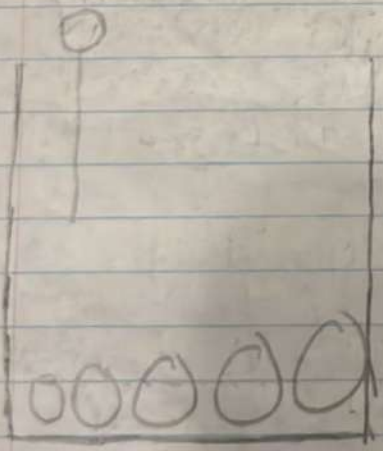
After the run down of yesterdays class I have a somewhat idea of what needs to implemented inside of the game. It seems like there will

be a lot less code actually needed for the game. However the code shown in class looks quite a bit different from what we have used in the past.

Thursday November 28th

My first step for this game project is to just make the ball moving back and forth while being able to drop the same kind of ball that can keep growing to the maximum size.

Example:



I was able to get the main ball to move back and fourth above the jar by having a timer where the ball will switch directions after the time you set the

I was also able to make each fruit spawn and they all seem to work great. I used a color picker to find similar colors to the actual game.

Sunday December 1st 2024

After ALOT of trouble shooting I was able to finally make it when 2 of the same object collide (right now I just have cherrys turning into strawberrys) they will turn into a bigger object.

It was quite simple to get the cherry to turn into a strawberry but it would make two instead of one. Making it just turn into one was a completely different story.

I had to look up a lot of different function godot has built in because the -on_body_entered fuction is triggered for both fruits involved with the collision.

Each fruit is attempting to replace itself and the other which makes two strawberrys.

Call deferred makes sure that the deletion and creation process happens

after the collision processing which can issues when both objects would try to delete each other at the same time

IsQueuedforDeletion(): Is there to check before creating a new fruit to check to see if any of the enemies are already in the queue for deletion. I believe, Sure, to prevent creation of a new fruit so two don't spawn.

I spent pretty much the whole afternoon troubleshooting and trying to figure out how everything works but hopefully this is a great baseline for implementing this stuff to the other fruits!

Tuesday December 3rd

I was able to get collision to work with every fruit and so now when any of the same fruit collide with each other they merge into the next higher fruit. (Except Watermelon)

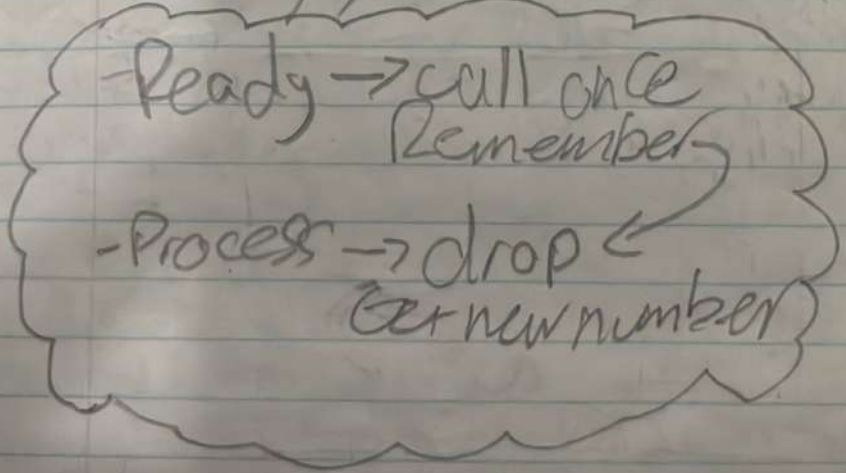
I was also able to make the drops randomly generate from either a cherry,

a Strawberry, a grape, a dekopon, or a persimmon. By using the RandomNumberGenerated.

To have done this I made a function calling each different fruit. I then made a function called RandomCalledFunction which had an Action[] Statement to store the functions of the fruit. Which they at the end used the RandomNumberGenerator to randomize the function called. Then put that inside of update where "drop" is.

My plan next is to try and figure out how you can find what fruit you are currently holding aswell as the next fruit you will drop. Allowing for more Strategic.

Here is a template Soph gave me for trying to implement this.

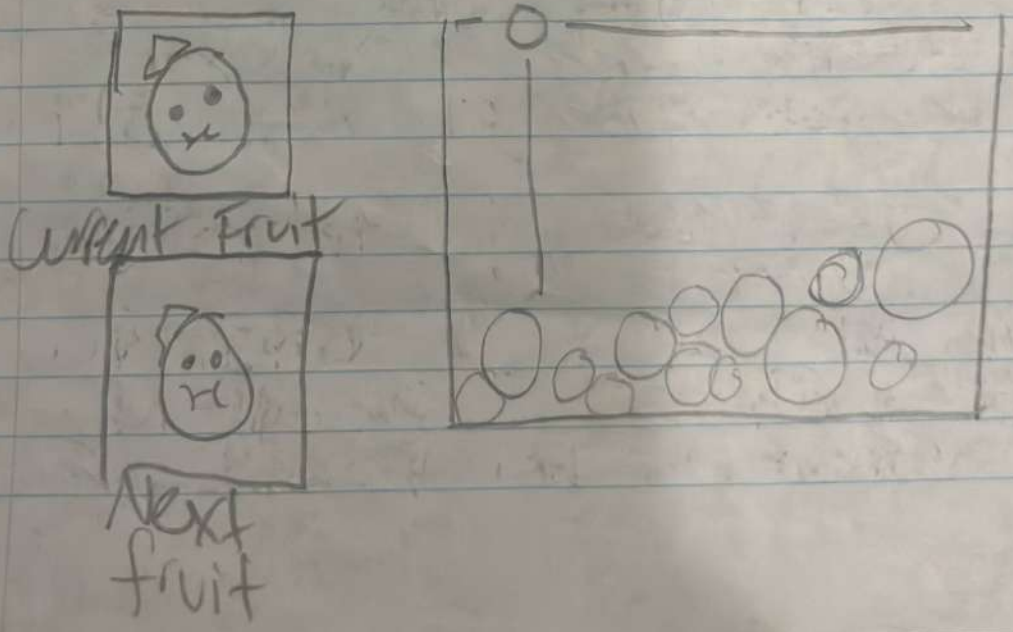


Thursday December 5th

Over tuesday night I spent a couple hours figuring everything out for the current and next fruit. I was able to achieve this

My first initial idea was to have sprites for the droppable fruits then have a detector for knowing which fruit is being dropped. Basically having a visibility check to turn off all the fruits except the ones you are currently dropping and the next one.

for this to work I created a 2D node called UI which had 2 textureRects with photos of the fruits for the current and next drop



In the UI script I did lots of research and I figured out how to get them displayed and all set to invisible unless it was called (bool)

I also changed some things in player, I didn't like how so much of my code was inside of just the player, so I made a script for the Fruit Parent -> changed to Fruit Manager

Inside of the fruitManager is where all of the fruits are created as well as the random number generated for the current and next fruits, which gets called to the UI.

One Issue that I came across that stumped my quite a bit is that the fruits would be displaying everything correctly however the fruits were not spawning on player but rather at (0,0)

This happened because the position was not set properly as the positions were not set as moveable. vector2. To try fixing this I changed the functions to instead not having anything I would

call a Vector2 Ex:

public void SpawnGrape(Vector2 position)

However! This caused issues with the random fruit generator as the Action[] does not allow for there to be Vectors being used.

To bypass this what I did was make the action an action Vector2.

Action<Vector2>[]

↳ This was able to fix the issue of the fruits spawning at (0,0).

Another thing I did tuesday night was also instead of having the balls just a basic color, I deleted the ellipse and imported Suika sprites relative to the collision detector of the object. I did this via Photoshop then each individually made pngs for each fruit.

One other thing I was able to implement was having a one second timer between each

drop to prevent spamming. I might tweak this feature in the future.

My next idea for the project is to have a game over screen if a fruit touches a certain point. This will probably be the last thing I implement for this project. However I do plan to work on this more over the break. Not for marks just because I am enjoying the coding.

Friday December 6th

So I decided instead of adding the game over detector, instead I decided to implement the score instead. The way the score works is that when you merge 2 of the same fruit, the score will update.

Scores merge points:

Cherry: 1 Dekopon: 10 Pear: 28

Strawberry: 3 Persimmon: 15 Peach: 36

Grape: 6 Apple: 21 Pineapple: 45

melon: 55

Implementing the score feature wasn't too difficult to figure out, however I had some issues with trying to get the score called from the different scenes, initially I tried looking to see if I could just use [Export] like (and the bag) however I could not find a way to pick Nodes from other scenes. So what I did was do it from the FruitManager. This worked.

However I've noticed some issues where the score gets doubled or halved? I'm not sure the reason. But I am happy with what I was able to do in the time I had.

Here is a final game look: