Project 2 - Fruit Ninja

Name: Gina Piccirilli

- **Repo Link:** https://github.com/IGME-202-2241/202-work-GineBean
- **Build Link:** https://igme-202-2241.github.io/202-work-GineBean/Project_02/
- **Assignment Details:** Project 02 - Fruit Ninja

Game Overview

*Give an overview of the design of your game. This is *not* a detailed technical or design writeup. Instead, give us 2-3 sentences describing the overall theme and main systems of your game. Also mention any unique features you have added to the game.*

The game is a replication of the game “Fruit Ninja”, with different food items as the fruit and an angry-looking fish as the bombs. The player has three lives and gains points by cutting the food and avoiding the fish. When the player holds the left mouse button and swipes across the food or fish being launched across the screen, that game object is cut and disappears, affecting the player’s points and lives according to what object was cut (or missed).

Launchable Objects

Document each type of launchable object prefab you have. Add more subsections if you have more than one prefab.

Fruit

Describe the overall creative and technical design for this prefab.

The Fruit prefab represents all of the spawnable/launchable game objects. It has a Circle Collider 2D and a Rigidbody 2D component so that it has gravity and physics, and can be affected by other colliding game objects. Using scripts, this prefab is instantiated with a randomized spawning location, target location, sprite, and launch force, with the game object's functionality based on the sprite chosen. The spawning frequency of these prefabs is randomized with the range based on how far into the round the player is. The prefab can be destroyed through collision with the cursor object (sword) or with the ScorePlatform object that is below the spawners.

Object Spawning

Detail how you implemented the object spawning system. What sorts of randomness did you use? How does your timing system work?

The prefabs are spawned using the FruitSpawnManager script, and are instantiated randomly at one of four spawn points located below the camera frame. The frequency at which the prefabs are spawned is randomized, with the possible frequencies determined by the length of the round, increasing the frequencies the further into the round the player gets to increase difficulty. When the elapsed time becomes greater than the spawn frequency, a new game object is instantiated. Each newly spawned prefab is also spawned with a randomly chosen sprite of the four possible choices.

Swipe Interaction

Detail how you implemented the swipe interactions. How do you track the mouse position? What sorts of collision/overlap detection are you using for your fruit? etc.

The position of the mouse is tracked using the FollowMouse script, which updates the mouse position and sets the position of the cursor game object (with the sword sprite and a collider) to be the same as the mouse. In the MouseMoving script, the state of the left mouse button in the current frame and previous frame (using Input.GetMouseButton(0)) are used to determine if the left mouse button is being held down. If it is, the sprite renderer of the sword object is turned on and the default cursor is hidden, displaying to the player that it is ready to be used. The script checks to see if the mouse is moving by checking if Input.GetAxis for the mouse's x and y are not equal to zero (which indicates that the mouse is moving). When the mouse is moving and the left mouse button is pressed, the circle collider of the sword cursor is enabled, allowing for it to collide with the "fruit" prefabs. The collisions are detected by using a circle collider on both the prefabs and the sword with the sword collider's IsTrigger set to true, and with scripts determining what happens when the objects collide.

User Interface

Describe the creative and technical design of your UI here. Also, discuss your rationale for this approach.

For the user interface, the TextMeshPro - Text component is attached to the Main Camera, with a font, font-size, and default text of “Score: “ set within the Unity inspector. In the ScoringPlatform script, there is a reference to this TextMeshPro component that is edited using the .text property to display the player’s score, lives, and instructional text, depending on the game’s current state. I used this approach because I have prior experience using it, and it is easy to implement, customize, and to edit within scripts.

Game Rules

Write a summary of the rules of your game, including your scoring system and lifecycle. How do you earn points? How does the game end?

The rules of the game are to try and cut the food items as they fly across the screen but avoid hitting the fish. The items can be cut by holding down the left mouse button and swiping over the game object. Cutting a food item gains the player one point, and missing one takes away three points. The player has three lives, and they lose a life if they cut a fish or if their points hit zero. Once the player runs out of lives, a game over screen pops up and the food stops launching. The player can then press the left mouse button to restart the game.

Scripts

A quick summary of the custom scripts you made. You only need to describe scripts relevant for this project, not all scripts in the repo.

FruitSpawnManager - Keeps track of timing system to reset the game, time out the spawning frequency of game objects, and to allow for the difficulty to increase as a round progresses (by increasing the spawn frequencies that can be randomly chosen from). It also instantiates a new “fruit” game object and chooses a random spawning location and sprite for that game object.

Launcher - When a game object has been created in the spawn manager, the Launch method will be called on that object at the same frequency that the object was spawned. In the Launch method, a random target is chosen which determines where the object will be launched towards. A direction vector is then calculated from the current “fruit”’s spawn to the chosen target. A random force magnitude is chosen, with different possible ranges based on how far the game object has to be launched. The force is then added to the game object, launching it towards the intended target.

ScoringPlatform - The ScoringPlatform script is attached to the ScorePlatform game object in Unity, which is a long rectangle collider beneath the spawners that tracks what objects collide with it (AKA which objects were not cut by the player). The script tracks the player’s score and lives, removing a life if the score falls below zero and setting the score to be zero so it does not go into the negatives. When the player has less than three points there is an instructional message displayed on the screen, and when the player has no lives left the script changes the text to the game over screen. If the player left clicks on this screen, their points, lives, and the timer reset, restarting the game. This script also has an OnTriggerEnter2D method which gets

the sprite renderer component of an object that collides with the platform and checks if it is a fish or food, removing three points if it is a food, then destroying the game object.

FollowMouse - This method gets a reference to the cursor object in game (the sword), gets the mouse's current position, and updates the cursor's position to be the same as the mouse's.

MouseMoving - Tracks the left mouse button and the mouse's movement. If the left mouse button is pressed and was pressed in the last frame, then the default cursor is hidden and the sprite renderer is turned on for the cursor object (sword), therefore displaying when the user is able to cut the "fruit" objects. It also checks for when the mouse is currently in motion, turning on the collider for the sword (allowing it to destroy the "fruit") only when it is moving and the left mouse button is pressed.

Death - This script is on the cursor object and has an OnTriggerEnter2D method that gets a reference to the sprite renderer of the object that collides with the cursor. It then checks the sprite of the game object, and gives the player a point if they cut a food item, or removes three points if they cut a fish. It then destroys the collided-with game object.

Make It Your Own

This section is optional. Remove it if you didn't do anything to go beyond the requirements of the rubric for this project. If you did something unique, describe what it is and what sorts of resources / self- teaching you had to do to accomplish it.

Included different possible targets as well as the different spawning locations to further randomize the launch paths of the game objects. This required some extra adjustment to the force added to the game objects so that they all stayed within the frame with a reasonable speed, yet still randomized.No extra research was needed for me to complete this. Spawn/launch frequency was also changed to be randomly selected within different ranges to allow the game's difficulty to increase over time. Theme changed from fruit to different food items and an angry fish.

Release Notes

Overview

List any errors, lack of error checking, or other information that we need to know here.

There are a few minor features I would like to add or fix in the future. One of these is figuring out how to prevent two launched game objects from colliding with each other. I would also add more to the UI, including a start screen that describes the controls. Another issue is that currently, an object is spawned and launched upon start to avoid null exception errors, but this allows the user little time to react to the first one, potentially losing them a life instantly.

Note: sprites used from/edited on https://game-icons.net/

Rubric Notes

Annotate this copy of the rubric with quick notes about where/how your project meets each criterion. Yes, you are effectively pre-grading yourself. More importantly, it demonstrates to us that you understand how the pieces of your project fit together.

Item	Points	Notes
Launchable Objects	**15**	
At least one “launchable” prefab.	5	The Fruit prefab
Launchables make use of Unity Physics / Rigidbody2D for movement, forces, and gravity.	5	The Fruit prefab has a Rigidbody 2D component that gives it gravity physics, and force is added to the prefab on launch.
Has both beneficial and detrimental objects.	5	Depending on the sprite chosen when a prefab launches, the object will either give or take away points/lives when cut or not cut. Food is beneficial, the fish is detrimental.
Object Spawning	**15**	
Starting spawn positions of objects are randomized.	4	Four different possible spawning locations randomly chosen when an object is spawned. References to the locations are stored in a list.
Each launched object is a randomly selected prefab and/or has a randomly selected sprite.	4	Sprites selected randomly from a list, changing the sprite of the Fruit prefab upon spawning.
Force used to launch objects is randomized for each launch.	3	Range of random forces depending on how far the spawned object needs to launch, and a force is randomly applied from within that range.
A timing system is implemented to control spawning and launching.	4	Randomly generated spawn frequency depending on how long a round has been played, using the elapsed time and frequency to determine when to spawn, with the launch frequency attached to the spawn frequency.
Swipe Interaction	**20**	
Properly tracks the mouse.	6	FollowMouse script getting mouse coordinates and setting to world point.
Short clicks do not count as swipes.	6	The collider of the cursor is disabled unless the mouse is moving and the left mouse button is held across more than one frame.
Properly interacts with objects.	8	Collision components on the “fruit”, cursor, and scoring platform, with the cursor and platform’s

		IsTrigger set to true, destroying the “fruit” on collision.
UI	**10**	
UI properly conveys all relevant information to the player.	10	Displays the player’s score and lives, as well as instructions on what objects to cut and how to cut.
Game Rules	**15**	
Has “beneficial” and “detrimental” launched objects.	5	All food is beneficial to cut and the fish is detrimental.
Has a reasonable score system implemented.	5	When cut, food gives the player one point, and when missed it removes three points. When a fish is cut, the player loses a life. When the player hits zero points, a life is lost. The player has three lives.
Has a lifecycle with a clear start and possible game over states.	5	When the player loses all lives, a game over screen appears and the game is stopped. The player is instructed to left click to restart the game. Currently no initial start screen.
Documentation	**25**	
Launchable object type(s) documented.	5	Yes
Object spawning system documented.	5	Yes
Swipe interaction documented.	5	Yes
UI system documented.	5	Yes
Game rules and scoring system documented.	5	Yes
Total	**100**	

Make It Your Own (Bonus)	**+10**	**Notes**
Made the game unique in a meaningful way that differentiates it from the requirements in the rubric above.	7	Minor changes to the spawning and launching system that provides extra randomization to how the “fruit” behaves, increasing the difficulty, especially as the game progresses. Theme changed from fruit.
README documents these	3	Yes

features/additions.		