Project 2 - Fruit Ninja

Name: Gage Magar

- **Repo Link:** https://github.com/IGME-202-2241/202-work-GameBuilder101
- **Build Link:** https://igme-202-2241.github.io/202-work-GameBuilder101/Project_02/

Game Overview
Launchable Objects
    Prefab
Object Spawning
Swipe Interaction
User Interface
Game Rules
Scripts
Make It Your Own
Release Notes
    Overview
    Rubric Notes

Game Overview

The game is themed similarly to fruit ninja, where the player is cutting through fruits in an attempt to gain a high score. However, because this project is played on desktop and not mobile, it has been designed to be slightly more challenging and fast-paced. It also has an emphasis on visual effects to make it satisfying.

Launchable Objects

Watermelon

The watermelon acts as the easiest target to slash given its size. It also gets launched at a slightly slower velocity than other fruits. Because of this, it rewards the least points. However, to give the player more of a reason to target it when slashing amongst a bunch of fruits, it gives the player back more health in comparison.

The visual effects and debris left behind by chopped fruits is meant to make the experience feel more chaotic.

This prefab (along with all other launchable objects) functions using the Fruit and SwipeTarget scripts.

Banana

This is a fruit variant which tends to spin faster, has a slightly smaller hitbox, and gives more points than the watermelon due to the reduced size.

Pear

This fruit is smaller than the banana hitbox, gets launched with marginally more speed, and rewards medium stats when hit.

Lemon

This fruit is incredibly small and gets launched at extremely high speeds. However, if the player manages to swipe it, they get rewarded with a large amount of points and a health boost.

Bomb

The bomb is a detrimental launched object, which when hit will reduce the players health and lose them a point. It also leaves a smoke cloud which can briefly obscure fruits behind it, making them more difficult to hit. Hitting a bomb is meant to feel confusing and chaotic.

Watermelon Bomb

This is a version of the bomb which deals even more damage and leaves a larger, longer-lasting smoke cloud. It is also disguised as a watermelon (though with distinguishing features) that can trick the player if they aren't paying attention.

Object Spawning

Object selection is done via a non-uniform weighted randomization system. Each entry in the list of spawnable objects is a struct containing the prefab and its associated weight. The weight of an individual entry divided by the total combined weights of every possible entry represents the chance of that particular entry being selected.

The time between each launchable object being spawned is uniformly random between a minimum and maximum value. To make the game more difficult as time progresses, an additional multiplier is used that gradually grows faster. This multiplier is obtained by evaluating an AnimationCurve given a time range representing how long the curve spans. Once that range of time has passed, the animation curve will always be evaluated at its farthest point (the fastest multiplier).

Positioning is done by generating a random point along a line below the bottom of the screen. The direction where the fruits are launched is determined by randomly generating a point along a smaller line in the center of the screen and then getting a vector from the fruit towards that point.

Swipe Interaction

I used a math-based approach for the swipes as opposed to using colliders. Swiping is handled using two scripts, the SwipeController and SwipeTargets. The SwipeTargets store their circular collision radius and contain methods for detecting whether a line intersects their collision. The SwipeController tracks the current and previous mouse positions, generating a line between the

two and checking for collisions with any SwipeTargets. If a collision check passes and the line is at least as long as the swipe target's radius, an event is called on the target to register the swipe.

However, an additional system needed to be added to prevent an issue where the FPS could get so high that it updates faster than mouse movements are registered by Windows, meaning no movement would be detected in situations where the mouse is being moved as fast as possible. To resolve this, an artificial frame cap is put on the mouse position update detections while it is currently hovering over a launched target.

User Interface

The UI system is using the old canvas-based Unity UI with legacy text systems, as it is the system I am most familiar with. The main game and game over screens are handled in separate scenes, as this is an efficient way to switch between the two game states. In either scene, there is a script attached to the canvas that updates the text components to their relevant values from the GameManager. The health bar is handled by having a fill graphic that stretches to every corner of the bar using anchors. In code, the x value of the anchor maximum is adjusted to match the percentage of health left. A value of 0 makes the bar empty, while a value of 1 makes the bar full.

Game Rules

There are two values for the player to keep track of: points and health. Points are gained by cutting fruits and lost by cutting bombs. Health is lost by cutting bombs, but also gradually decreases over time (the speed of which increases as the game progresses). Health can be regained by cutting fruits. This is to encourage fast-paced gameplay to keep up with the health drain and make it more likely for the player to accidentally hit bombs if they're being careless.

Once the player runs out of health, the game ends and sends them to a gameover screen displaying their score. The objective is to get a high score. Pressing the restart button will allow the player to play again.

Scripts

The game score, health, health drain, and total time passed is managed by the GameManager script placed on an empty game object. A method can be called to set the health, and if the resultant value is 0 it loads the FruitNinjaGameOver scene.

UI for the score/health is handled in the GameManagerUI script. UI for the gameover scene is handled in the GameOverUI script. Both of these are attached to their respective canvas objects.

Launched objects are spawned by the LaunchableSpawner and parented under the spawner's empty game object. The launchable spawner begins running as soon as the scene loads, as the state management is handled by switching scenes.

The SwipeController script handles detection of mouse movement and triggering swipes on the SwipeTargets. The SwipeTarget script, when created, adds itself to the SwipeController's list of targets. SwipeTargets have a UnityEvent that gets called when a swipe is triggered, and this is used to call a Chop() method on the Fruit component.

The Fruit script stores how much score/health is gained or lost from chopping a launched object, as well as multipliers for the linear/angular launch velocities (which is used by the launchable spawner when spawning the fruit). The Fruit script derives from a Choppable script, which handles the visual effects of chopping a fruit. The chop effects are done by disabling the main sprite on the object, enabling sprites for either half, and then de-parenting the half sprites from the main launched GameObject parent.

Make It Your Own

I did a few things to enhance the gameplay experience. When fruits and bombs are cut, they get chopped in half and create particles. This helps with visual satisfaction and also affects gameplay by creating distractions. I also made more than one variant of positive/negative launchable objects to add more variety. Additionally, the difficulty of the game increases over time as objects get launched more often and the health drains faster.

I did not personally have to do research for anything outside of the swipe interaction, as I was already familiar with the Unity systems used to achieve this.

Release Notes

Overview

There are no known scripting errors with the current build. It should run fine in a browser and function regardless of low or high FPS. It does seem to stutter on occasion, although that might be a problem specifically with the mouse trail renderer component.