Typing Ninja

Andrew Tao, Genie Choi, Joy Cho, Katie Lee
https://github.com/klee120/426finalproj

Abstract

Typing Ninja is a two-dimensional game that involves a Ninja player sprite in the center of the screen and multiple fruit sprites moving toward the Ninja. Each fruit has a corresponding word displayed on top and to clear the fruit, the user must type the word before the fruit reaches the Ninja. If the fruit is successfully cleared, points are added, but if the fruit hits the Ninja, a life is lost. This game is broken down into multiple scenes that represent the start of the game, main gameplay (split by three levels), and when the game is over. Two different object classes were used with one being the fruits with the corresponding text and the other one for the Ninja object. These varying elements were successfully combined to create a working game with three levels of difficulty and a win/lose state.

Introduction

Our main goal was to create a typing game that would clear fruit obstacles as they traveled towards the player/Ninja sprite. Our other goals included giving the player a specified number of lives/hits before the game ends, as well as including a score counter that increases based on the number of fruits sliced. The player must clear three levels of difficulty, with each level having longer/more difficult words to type than the previous level. The game is completed when the score reaches 250 points in the third level. This game is geared towards those learning how to type and also people who enjoy typing games. To tailor to this class, we also included Computer Graphics specific words that can be seen in levels 2 and 3. We pulled resources from previous works such as Princetomon and Cave Climber, which are both Javascript-based 2D games. These works were particularly useful to us as we were figuring out how we would represent two-dimensional sprites in a three-dimensional framework (Three.js), and they were also good references for the general code structure they use to organize the components of their games. To approach our goals for this game, we tried a divide and conquer approach, in which we split up the functionality and object classes between each of our team members. We believe this approach worked well in our particular circumstance since we had different strengths we could use to implement our individual tasks, as well as the fact that our game was modular and could be implemented in components and combined at the end, rather than done all at once.

Methodology

To execute our approach, we had to implement a multitude of components, including object classes and game scenes.

Camera

One challenge we faced right away was that we were aiming to create a 2D game using a 3D framework. In investigating other 2D games created in the past (Cave Climber and Princetémon), we found that using an Orthographic Camera facing the Z = 0 XY-plane worked quite well. We also ended up using the Z axis for many useful tricks regarding layer priority; for example, whenever one of the fruits was completed by the player, we added the slash sprite over it at Z = 1 to ensure that the slash was rendered completely on top of the fruit.

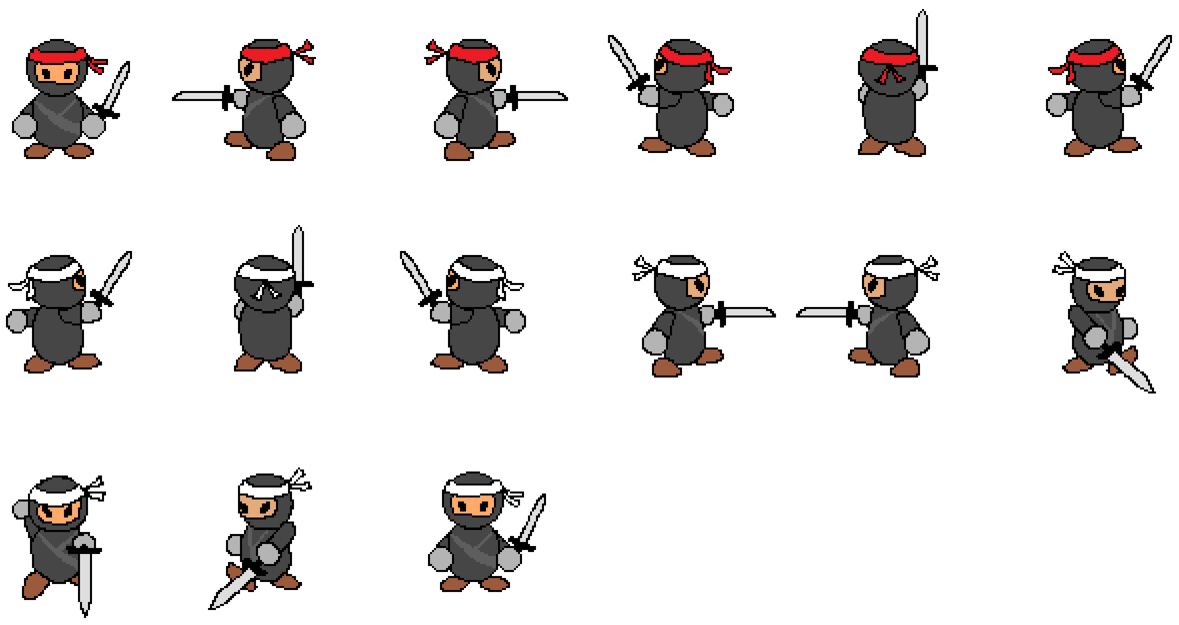
Event Handling

Our game consisted of multiple scenes; Start, Game, End, and Death. Each scene required unique event handlers. The start screen listens for the keydown of the 'Space' key to start the game. The game listens for the keydown of valid alphabetic letters. The end and death screen listen for the keydown of the 'Enter' key to restart the game.

Prominent (in Stage) Objects

Ninja

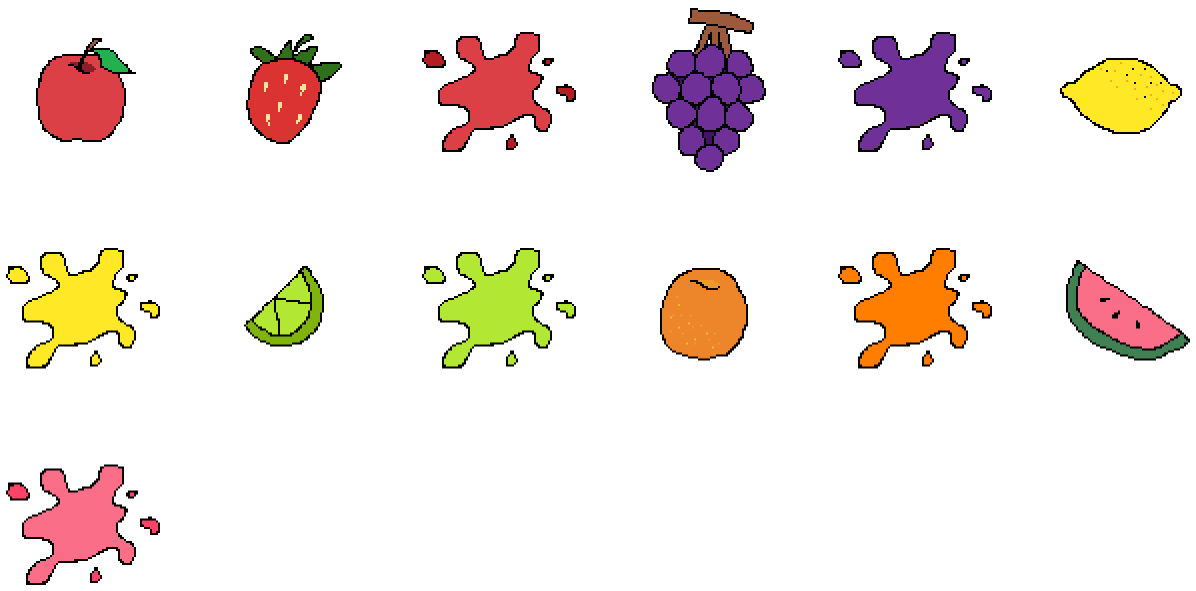
For the Ninja class, we implemented an object class that would take the current level of the game state into the constructor and change the color of the Ninja's headband sprite based on the level. The Ninja class also contains multiple methods as well, including changePosition(), which takes the Vector3 position of a fruit and calculates the angle (0-360) between that fruit and the Ninja's position. Then, based on the angle, we changed the current sprite of the Ninja to one of eight directional sprites. This gave the impression that the Ninja was facing "towards" the fruit when slicing it.



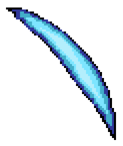
Ninja Sprites (White for Level 1, Red for Level 2, Black for Level 3)

Fruit

For the Fruit class, we implemented a Group object that would progressively move towards the ninja, tracking the progress of the player in typing the word the fruit owned. Every fruit had a corresponding sprite, starting position, speed, and word. The first step to create a fruit was to generate a random word from the list of words for the given stage. This is done by generating a random index into the list using Math.random() and then checking if the starting letter of the corresponding word was the same as one of the words already on the screen. We wanted to avoid generating words that start with the same letter to make sure the player could always know which word they would start typing, so this process is iterated until a word with an unique starting letter is outputted (with a reasonable cutoff to ensure no infinite loops occurred). The next step was to randomly populate the fruit on the border of the screen. This was done by generating a Math.random() to see which side of the screen we were on (top, down, left right), with greater probability on the left and right sides because those sides had greater travel distance to the center of the screen. Once the side was decided, we set the respective coordinate value and randomized the other value so that the fruit would populate from a random position on the given side. After the word and location of the fruit had been decided, we also randomly determined its speed from a predetermined range of speeds, and then added it to the list of fruits tracked by the current stage. Finally, when the stage is done with a fruit, we change the fruit's sprite to a "splat" sprite for 1 second before removing it from the screen. If the fruit was completed as a result of the player successfully typing the word in time we add a "slash" icon on top of the fruit, but not, we just add the splat sprite onto the player's ninja.



Fruit and Splatters

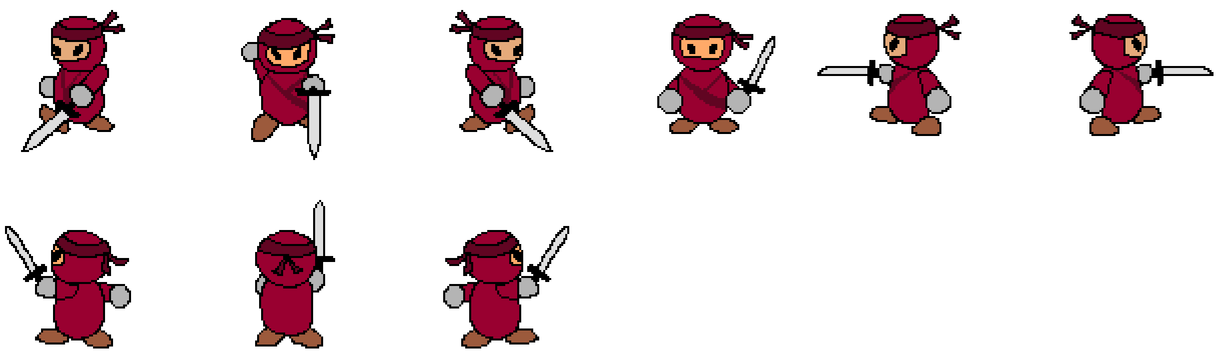


Slash Sprite

Helper Ninja

The Helper class is something that we created which extends the Fruit class. This class represents a helper Ninja with special abilities. It has its own sprite set and when its word is completed, it clears all of the fruit on the screen. The condition for a helper Ninja appearing is that there is a 50% chance one appears as the new fruit added whenever there's already 3 active fruits. The goal of this was to help bail out the player when they're getting overwhelmed by fruits, but not in a way that is so reliable that the player can just slack off waiting for the helper ninja.

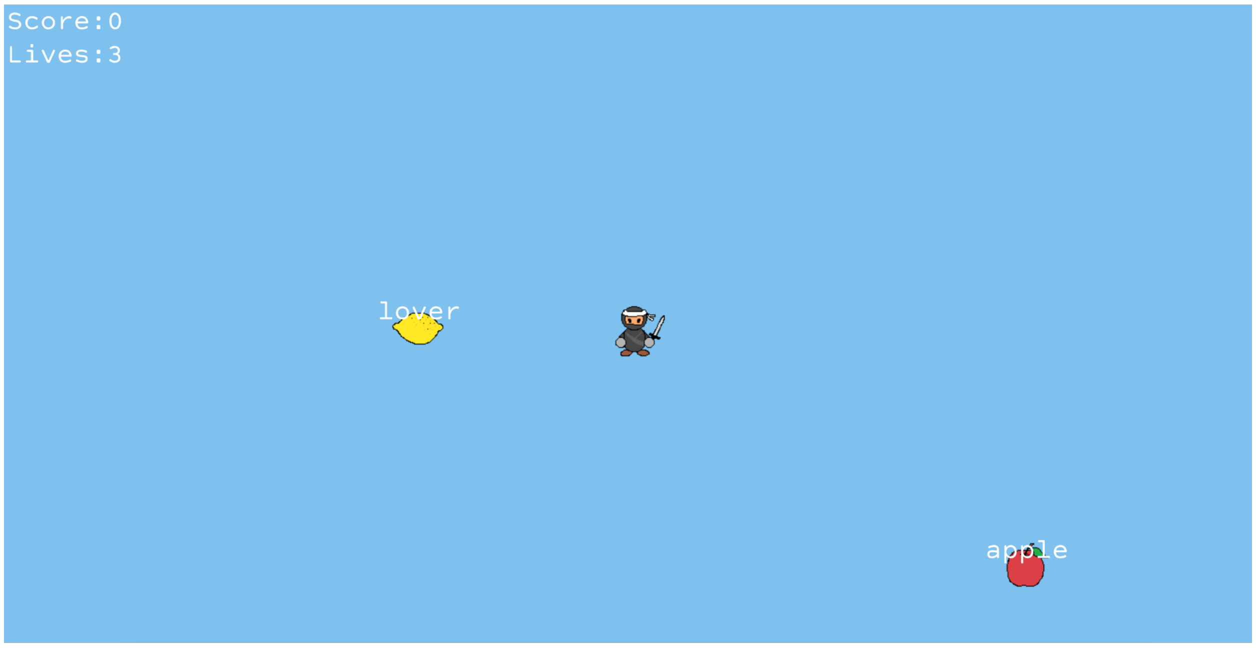
The helper Ninja also has a few small differences from normal fruit behavior; for one, we thought it would be very cruel for it to seem like the player ninja is attempting to slash its friend when the word is completed. As a result, when the helper Ninja's word is completed, the player Ninja does not turn towards the helper Ninja, and the slash sprite is not added. Lastly, the helper Ninja cannot hurt the player Ninja, and instead just disappears without impacting points or lives in the case of a collision.



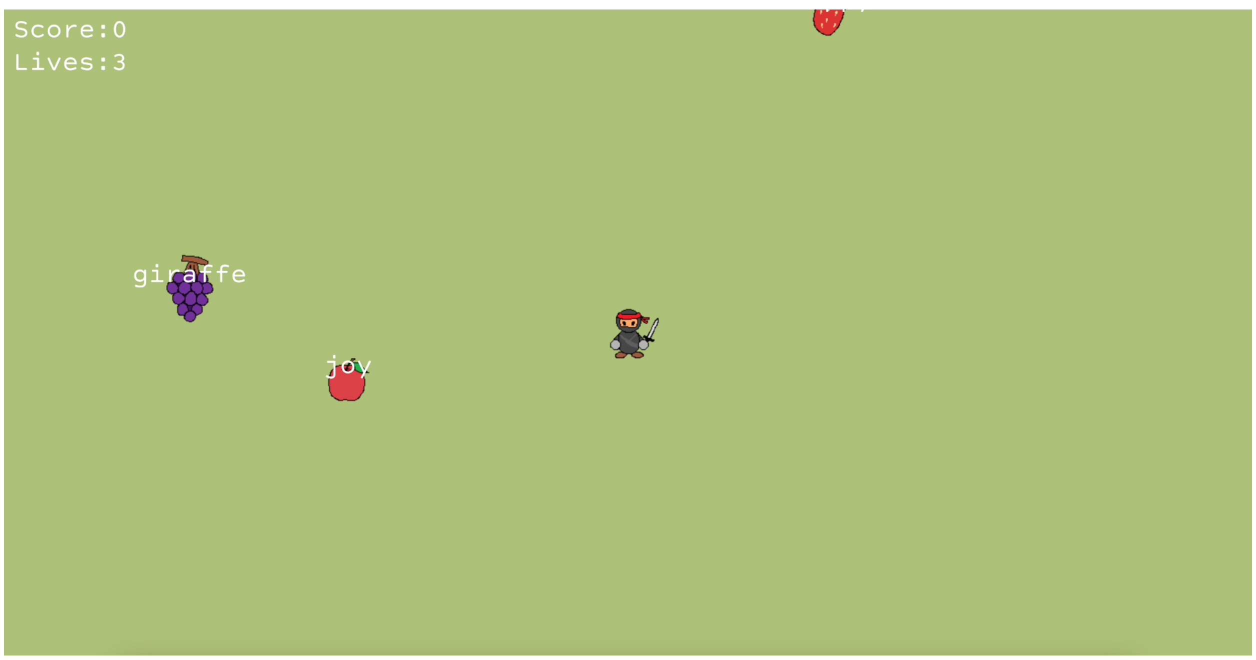
Helper Ninja Sprites

In-Stage State Tracking

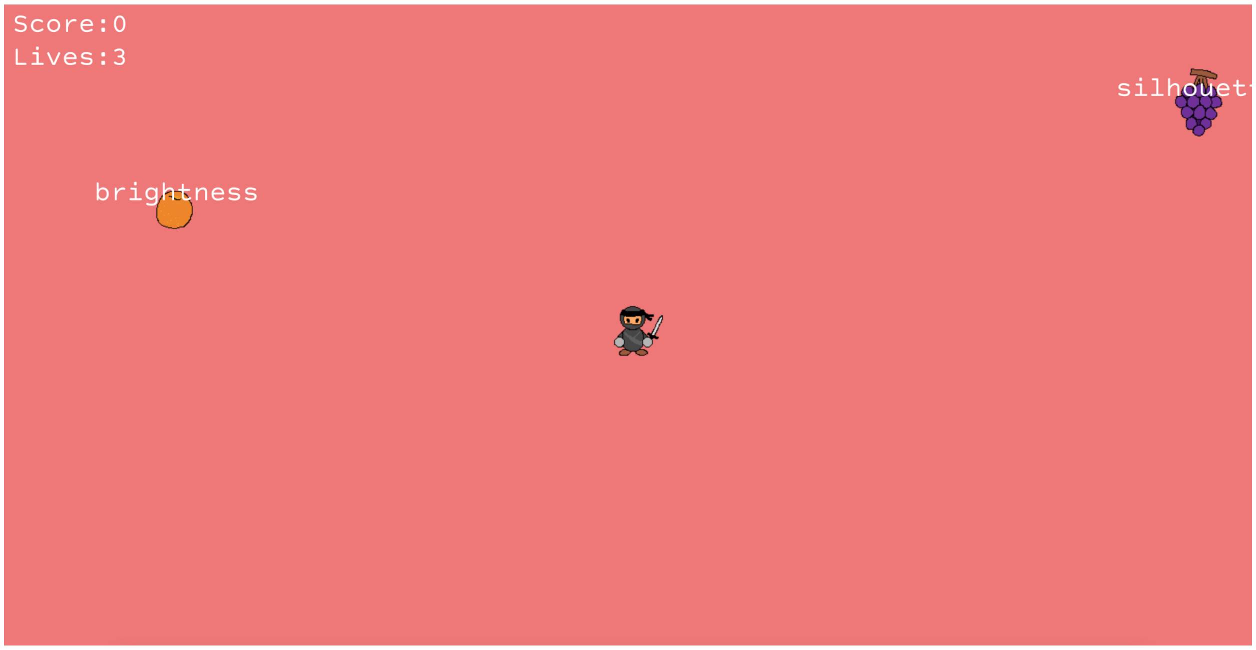
As time progressed within a stage, we moved each active fruit towards the Ninja in the center of the screen. If the player did not clear a fruit before it collided with the Ninja, we would reduce the number of lives the player has, and if the player ran out of lives, we would transition scenes to the Death scene. When a fruit is completed, we give the player a number of points equal to the length of the word that the fruit owned.



Level 1 Screen



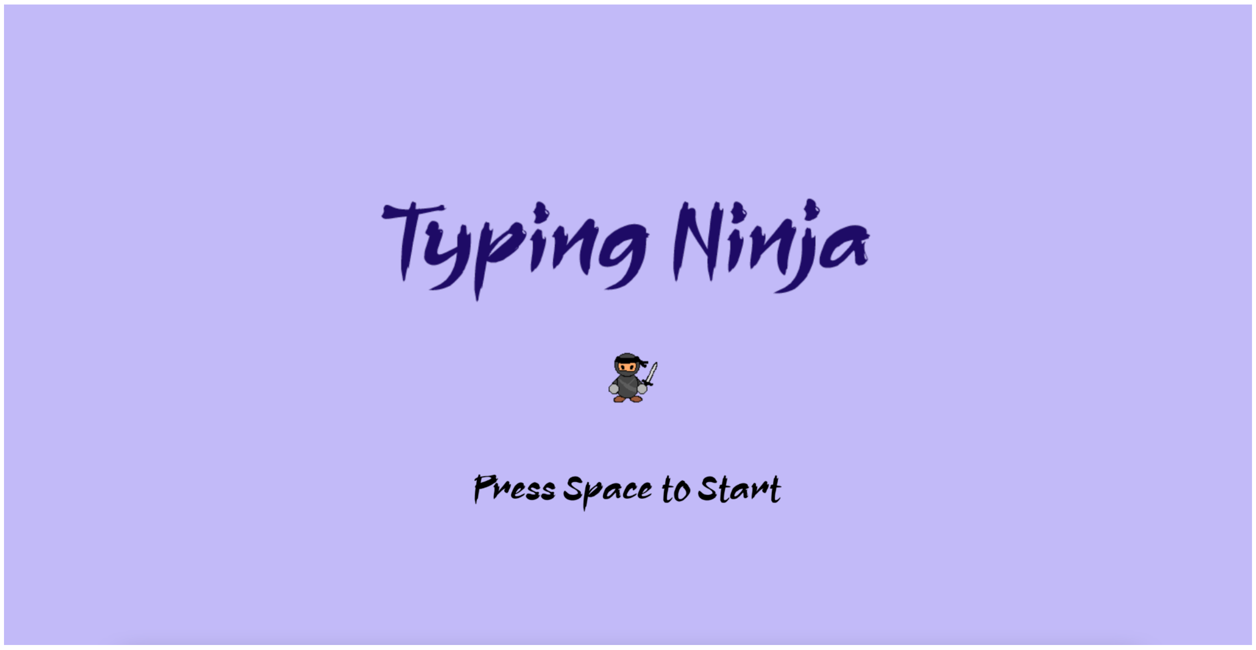
Level 2 Screen



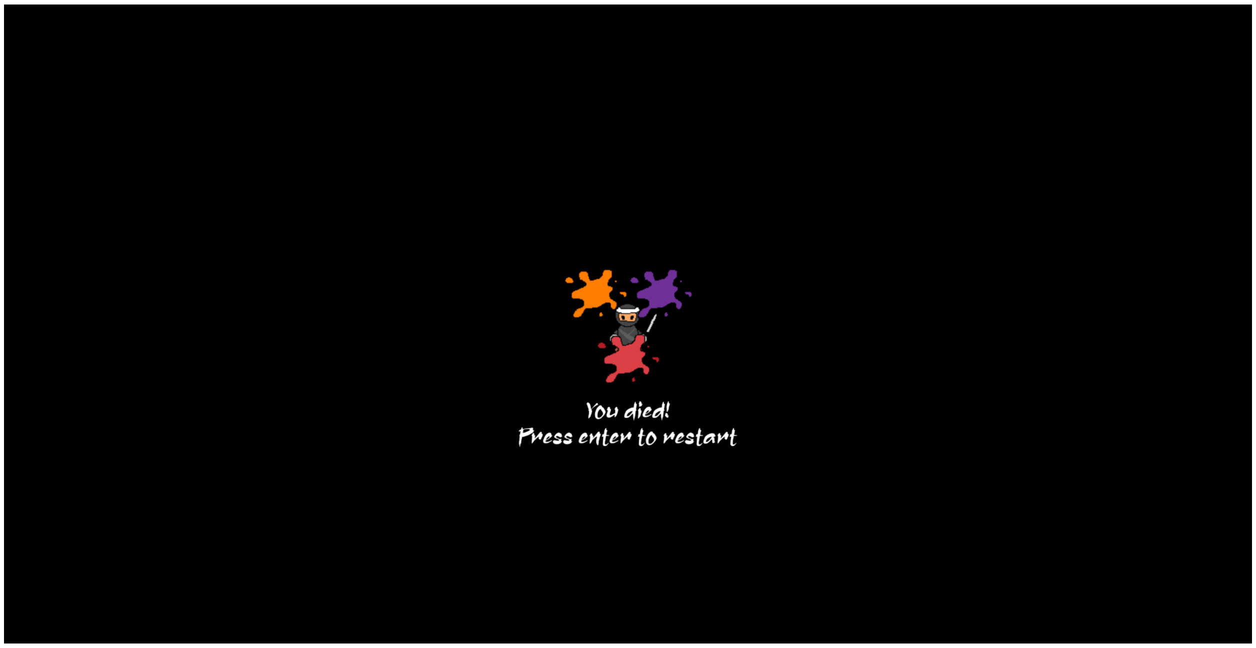
Level 3 Screen

Non-Stage Scenes

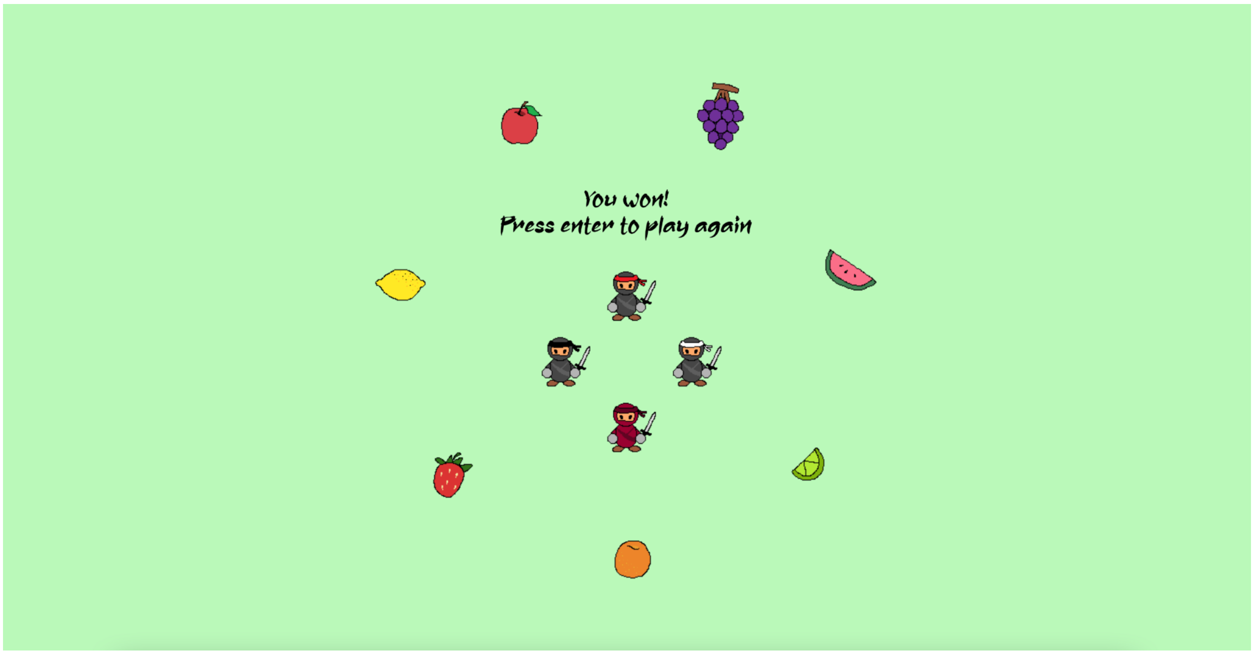
For all scenes without play, we didn't need nearly as much complex logic for individual objects. The start scene, death scene, and ending scene are all composed of sprite objects, and are just constructed once in the singleton SceneManager object. The only notable implementation detail of interest is that the end scene uses the time in order to cycle the fruits in a circle around the ninjas in the center.



Start Screen


You died!
Press enter to restart

Death Screen


You won!
Press enter to play again

Win Screen

Miscellaneous

Audio

The audio we used was all from *freesound.org*, all licensed under Creative Commons 0. The *AudioManager* class was created with heavy inspiration from the class of the same name from Cave Climber. However, a few differences between the context of our audio usage and the cave climber usage meant that the class ended up being very distinct from the Cave Climber one. To begin, we actually have three different tracks of "background" music; one for the death screen, one for the win screen, and one for the rest of the game. We ran into a few issues with stopping and swapping out background music, and so ended up changing a few things about how sounds were played. Additionally, since we only had two other sound effects (the sword noise and the fruit splat), we didn't use an id system like Cave Climber, and instead opted to just have two different instance methods of the *AudioManager* singleton.

Results

Overall, we exceeded our expectations for our game, meeting all of our goals from our MVP and adding extra content we had planned in the case we completed our goals, such as additional sound effects and adding a score counter, as well as a start and ending screen. We measured our success by the replayability and overall enjoyment of our game, which we found was apparent throughout the game's build by the amount of times we found ourselves replaying the game as well as feedback from our peers. We were also able to implement various levels of increasing difficulty, measured by the length of the typed words required to remove the fruit and the speed and frequency at which the fruit moved toward the player and appeared on the screen. Overall, we successfully created a game that was engaging and easy to play multiple times.

Discussion and Conclusions

Something that took us a very surprisingly large amount of time was displaying text over each fruit. In the beginning, we struggled a lot just trying to get the text to show up and move with the fruit. Then, we needed to make sure that the text color would be different for letters the player already typed. Although all of the examples we saw of text used the load() method of the Three.js FontLoader class to create a TextGeometry class, this did not seem to support changing the material type within one message. As a result, we ended up using the parse() method of the FontLoader class instead, which returns an array of Shape objects to use in the ShapeGeometry class, allowing us to specify a different material for each shape.

Besides figuring out how to display the text for the fruit, we had very few major difficulties in figuring out the general skeleton of our game. This allowed us to work on polishing the game much further than our MVP, such as animating our End screen, adding different background music for End and Death, and adding more functionality such as the Helper Ninja power-up.

Although we're very happy with our final product, there are a few minor details for ideas we had that we simply did not have enough time to incorporate. To start, we thought it would be interesting to add animation to the sprites (such as the fruit exploding when sliced or the Ninja sprite moving between positions), allowing the game's transitions between sprites to have a more smooth look. We would also like to add more fruit and powerup variants to the game, such as a Heart power-up to increase the number of lives the player has, or a Freeze power-up which freezes all of the fruits on the screen. It would also be interesting to add an endless mode with infinitely spawning fruit, where the player aims to earn as many points as possible before losing all of their lives.

From this project, we learned how to represent two-dimensional objects in a three-dimensional space, which was one of the main considerations we had to take into account in the construction of our game. We also learned how to organize sections of our game into multiple scenes with their own objects.

Contributions

Andrew - Investigated previous projects (predominantly Cave Climber and Princetémon) to figure out how to set up things such as the general game structure, camera (so that the game is 2D in a 3D framework), and things such as displaying sprites and text. Also spent a lot of time working on the "manager" classes, including the update loops and all of the audio playing. Also worked on fruit generation and typing tracking, and created the helper ninja object.

Katie - Investigated functionalities of other typing games and helped lay out the structure. Helped create the Fruit object class, dealing with the generation of random words, managing to only display unique starting letter words, and then working to display the text. Set up the score and lives update functionality as well as helped build the scenes.

Joy - Helped create the Fruit object class and dealt with random fruit location and speed generation to move towards the ninja, as well as dealing with ninja collisions in the Game scene. Implemented scene transition logic in SceneManager, setting up the starting, game, and ending scenes and adding appropriate event handlers to each.

Genie - Drew all pixel sprites for each of the ninja positions for each level, fruit whole/splatter sprites, and banner sprites. Helped import sprites into the game scene files. Created a Ninja object class and defined functionality for calculating the Ninja's sprite position based on the calculated angle between the fruit currently being sliced by the player and the Ninja's position.

Works Cited

Our code skeleton was the course provided course skeleton, made by Reilly Bova ’20.

Projects referenced:

- Cave Climber - https://github.com/Derndeff/cave-climber/tree/main
- Princetémon - https://github.com/stephanieyen/princetemon
- Typer: https://github.com/berkerol/typer

Audio:

- Sword: https://freesound.org/people/Merrick079/sounds/568170/
- Death Music: https://freesound.org/people/nomiqbomi/sounds/579734/
- Win Music: https://freesound.org/people/josefpres/sounds/691946/
- Normal Background Music: https://freesound.org/people/josefpres/sounds/653716/
- Splat (fruit hitting ninja): https://freesound.org/people/yottasounds/sounds/232135/

Fonts:

- Assassin Ninja font: https://www.fontspace.com/a-assassin-ninja-font-f51018

Other sources

- Real Favicon Generator - https://realfavicongenerator.net
- Three.js documentation: https://threejs.org/
- Pixilart - https://www.pixilart.com/draw#