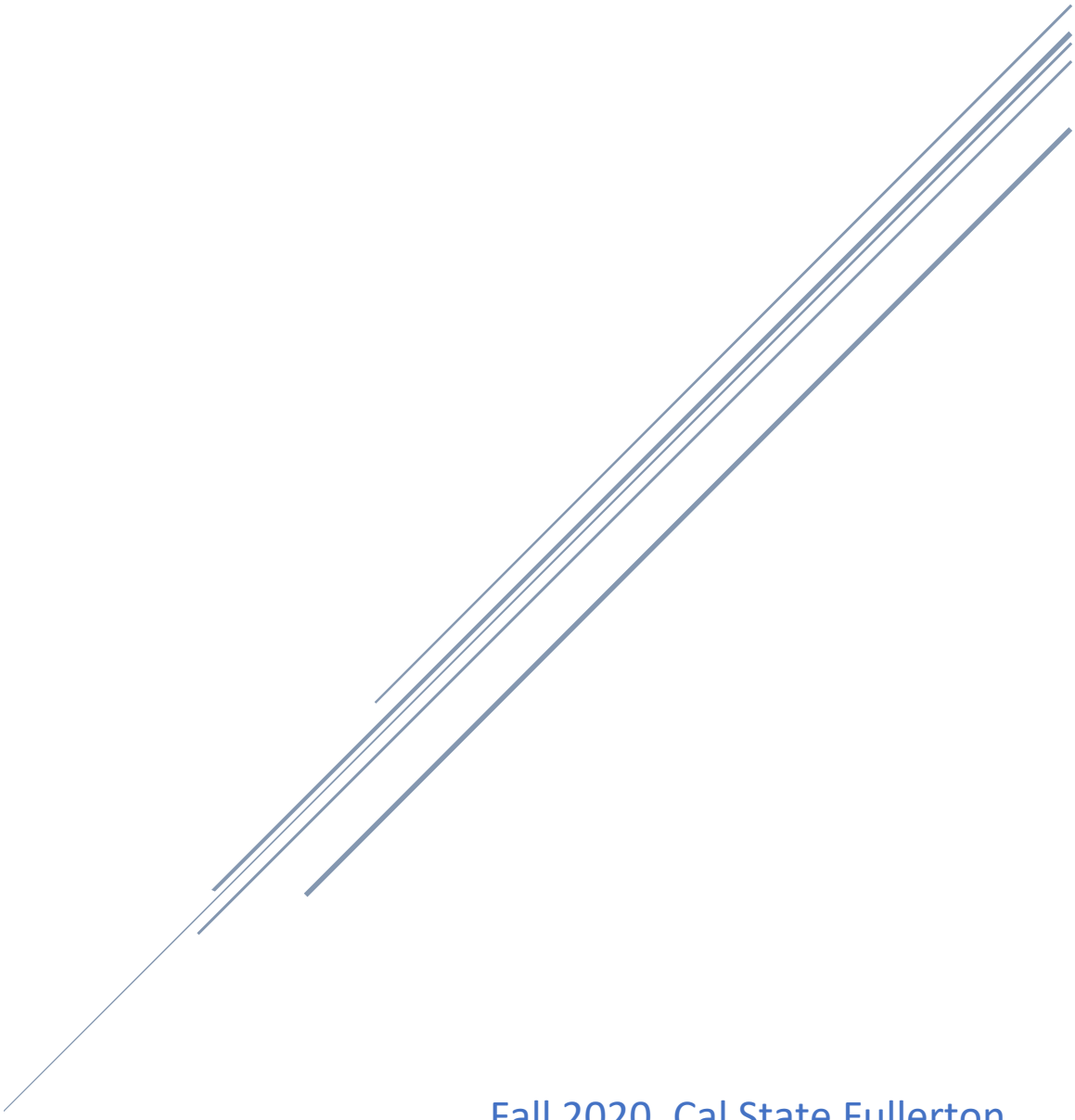
SPACE BUNNIES

Creator: Isabel Silva



Fall 2020, Cal State Fullerton
CPSC 386 Intro to Game Design

Table of Contents

Introduction

Space Bunnies is a shoot 'em up arcade style game that will take players on a laser blasting' journey through space to defeat vicious alien invaders and bring the hostage kits to safety. Space Bunnies is a two-dimensional top-down view shooter game with vertical scrolling background. The game characters consist of a blue spaceship as the player avatar, packman-like alien enemies, baby bunny NPC's, and alien boss enemy. The player avatar is controlled by the player while the enemies are controlled by the computer/game itself.

How to play the game

The player avatar will have the tasks to save all kits that the enemy hold hostage and after the kits are either saved or die the player will need to defeat the boss. Both must be done before the timer runs out. To save the kits the player must use their laser to shoot down the enemy holding the bunny kit. Once the enemy holding the bunny kit dies the bunny kit falls; the player will have to catch/collide with the bunny kit before the bunny kit falls out of display. If the player catch the bunny kit in time the player must take it pass the shield that will pop up once the player has the kit, one the bunny is past the shield the bunny is considered saved. If the bunny reaches the end of display before the player can catch it the bunny is pronounced dead. The more bunnies you save the more points the player gets, the more enemies that are killed the more points the player gets. Killing the boss before the time ends gives a player 1000 points. The goal is to get the most points as possible and be the highest score.

Creator Note

Time frame given for this game was three weeks and languages used are a combination of HTML, CSS, and JavaScript with p5.js/p5.js.sound libraries. One of requirements for this game was to create a bunny NPC of any kind. For this game, the bunny NPC will be the game objects the player must save from the enemies. This is version three of the game, each week had its own version.

Rules

1. Player must wait five seconds for enemies to align themselves to position before the game starts.
2. Player can pause the game anytime the game is running (except in the 5 seconds entry time at the start).
3. Player can turn sound on and off anytime during the game or while in pause or menu screen.
4. Player must save the bunnies and beat the boss within the time allowed.
5. Player must first kill the enemy holding the bunny kit before being able to catch/collide with the bunny.
6. Bunny must be caught by the player before the shield pops up.
7. When player avatar is holding bunny the player speed is reduced by 5.
8. Player must take the bunny caught pass the shield before bunny can be marked as saved and gain bunny saved points.
9. Enemies can not pass shield when shield is up.

10. If bunny kit falls off screen before the player can capture the bunny, the bunny is considered dead and player loses bunny death points.
11. Boss enemy only appears when there are no more bunnies to save.
12. Boss can only attack player when player is in range.
13. Player lasers must hit Boss to lower the Boss's health.
14. Boss's health must be zero for Boss's to be marked as defeated.
15. If Boss or enemy collide with player, the player will lose health points.
16. Player gains 1000-win points to total score only when there are no more bunnies to save AND boss is defeated (Boss health is zero) AND timer is greater than zero.
17. If game timer reaches zero before the bunnies are saved or Boss is defeated the game will be considered as game over. Points acquired at the time will be the last score.
18. Player can not gain more health than the max capacity (no infinity health).

Design

Game States

1. Menu State – player start off at this state to continue to the next state the player must click the start button
2. Setup State – the player waits for the enemies to align themselves
3. Run Game State – the player avatar, game score, bunnies saved, game timer appear on display and game commences this state is true only when setup state completes
4. Pause State – player can access this state while run game state is on, access to this state turns off the previous run game state, the player can restart the game which goes to the second state (setup state) and commences from there or resume the current game turning the run game state back on
5. Game Over State or Game Win State-- player loses when either the timer runs out or the player loses all health, they have access to restart back to the setup state; player wins when they defeat the final boss before the timer runs out player then has the option to continue on to the next level (not in this game version)

Points, Health, Game Duration System

- Enemy kill: player gains 50 points.
- Kit saved; player gains 75 points.
- Kit dies; player loses 100 points.
- Player is hit by enemy/boss player loses 1 health point.
- Boss is hit by laser boss losses 10 health points.
- Player eats carrot, player gains 10 health points, max gain is max player start health.
- Boss eats carrot, boss gains 10 health points.
- Player wins the game player gains 1000 points.
- Player loses the game, player gains/lose zero points.
- Player will have 30 seconds to complete the level in game.

Game Animation and Controls

User control

Controls are is the keyboards left/right/up/down and space keys. These are for the player animations.

The player can move the avatar left/right/up/down based on the keys pressed.

The player can fire lasers by using the space key.

Computer control

Enemies that have kits as hostages are called jailers, they will move left and right only. Enemies without a kit hostage are called sentries, they will move at random up/down/left/right bouncing off display walls until player avatar is near then they will track the player and "attack" by colliding. Once player is out of range the sentries will go back to random movements. if shot down both jailer and sentry enemies will disappear from screen.

Kits move on the left for whichever elements holds it such as the player or enemy jailer. Once free of jailer the kit moves down the screen. The amount saved/total kits will be displayed and updated on the bottom right of game screen.

Boss has the same movements and animation as the sentry enemies, they will also change color when hit to show damage.

Scrolling background animation is a vertical scrolling of a space background.

Shield animation is a sine wave.

Game countdown animation for both start of game when enemies align to position and the countdown for the game itself.

Points gained will be visible on the bottom left of screen and each gained amount will pop up as the player gains them to the right of the player.

Software Architecture Detail

Elements Design

Created an elements base class that have the following local variables:

- Location int variables x, y, used in movement and collision detection
- Int variable speed, used in movement
- Element size, used in collision detection
- Boolean variable hasKit set to false switched to true when element is holding kit
- Int variable health set to max health

Elements base class has the following helper functions

- getDistance(element) takes in an element and returns the distance from that element
- collide(element) takes in an element returns true if collision with that element
- healthBar() displays a health bar below the element

The following classes inherit the base class, each of these classes have there own unique functions that are special to them such as their movement and avatar images (drawing).

- Player class: this will hold the player's avatars information regarding movement, score, lasers, and drawing the player to the screen

- Laser class: creates the lasers that the player holds
- Kit class: are the bunnies and holds the kits information regarding movement
- Enemy class: holds the information regarding the drawing, random movement, type of enemy, and tracking movement
- Boss class: holds the information regarding movement/tracking of player and drawing on screen.

Game Running Design

For the screen display P5.js libraries have a lot of helper functions that helped with creating the game. I used p5.js setup, draw, key-pressed, mouse-clicked, helper functions to automatically run the display screen as well as handle user's inputs.

For the wave function to create the shield I used Daniel Shiffman sine wave rendering.

For change of game states there are Boolean variables that turn on and off depending on the current state, the main game loop (p5.js draw() function) checks for these states using if statements.

Game Demonstration

Game Link

Posted the game online using github as a host the following link will take you to the game: https://isabel2296.github.io/spaceBunniesFall2020/main.html .

Game Images

1. Main game Display
   a. 

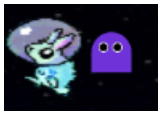
2. Player image
   a. Regular 
   b. Damaged 
   c. With full health 

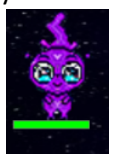
d. Half health 

e. Low health 

3. Enemy & Boss

a. Enemy (sentry) 

b. Enemy with kit (jailer) 

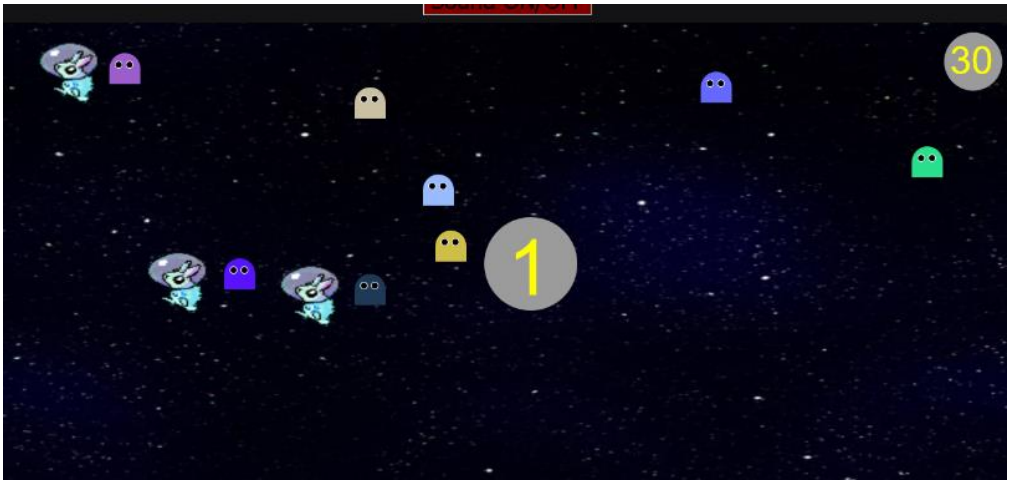
c. Boss 

d. Boss damage 

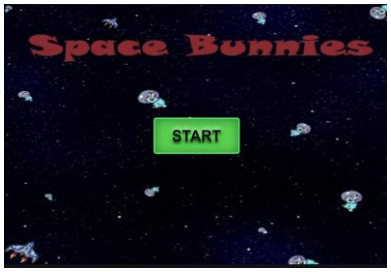
4. Bunny Kit 

5. Carrot 

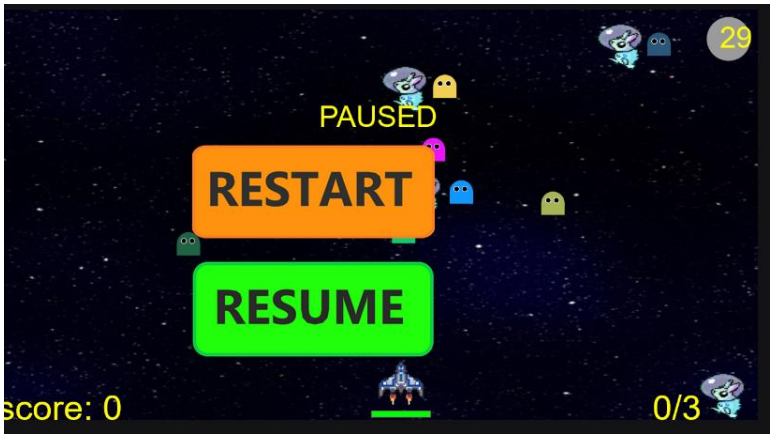
6. Setup Animation Count Down

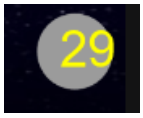


7. Player UI

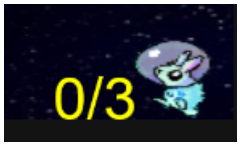
a. Menu 

b. Sound on/off button 

c. Paused 

8. Timer 

9. Score 

10. Kits saved 

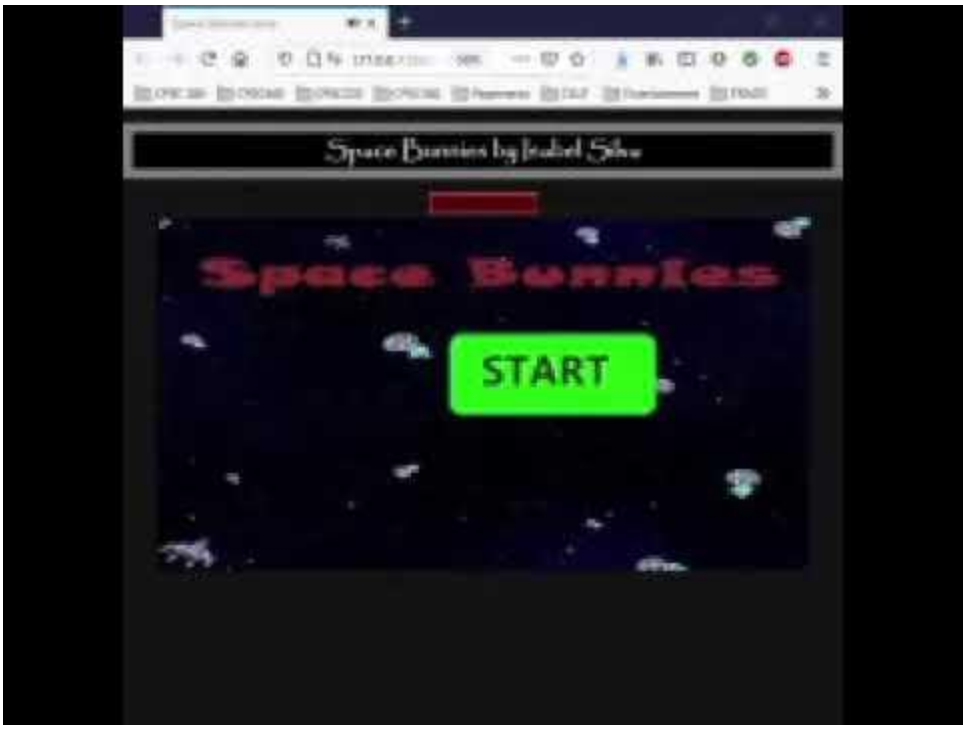
Game Audio

Audio was taken from freesound.org.

Game Video

The following is a video demonstration of full game loss because timer went to 0 before boss was defeated.

https://www.youtube.com/watch?v=AixNF0BbAOo&feature=youtu.be



The following video demonstration is of game win.

https://www.youtube.com/watch?v=H1F_hmXV3FY&feature=youtu.be

Bibliography

Shiffman, D. *Sine Wave*. Retrieved December 09, 2020, from https://p5js.org/examples/math-sine-wave.html