Fruit Blender

Game Design Document

Ian Kowalski

"Fruit Blender" will be a 2D game that utilizes python and simple GE to show basic game creation skills, and hopefully be fun.

The game will focus around you, an apple, attempting to avoid being running into a blender while colleting other fruit to increase your score. The user can move the apple up, down, left, and right. When you make contact with the fruit, a noise will play, and when you make contact with a blender, a less pleasant noise will play. Fruits and blenders fall from the top of the screen.

In part 2, I would like to be able to have fruit come from all angles of the screen, as well as a score being kept of how many fruits and blenders you've gotten, and a lives system. The Game Class Visual will reflect this, however, these haven't been implemented yet and are subject to change.

The Game Class

It will be the most important class, and is subclassed from simpleGE.Scene

The Game class has visuals that include:

Apple – an instance of Charlie class

Other Fruit – an instance of Coin class

Blender – an instance of Hurt class

Non- visual assets include:

sndCoin – the coin sound effect representing score

sndHurts- the hurt sound effect representing hitting a blender

Game Class Components:

Visuals include a game class as well

Apple (currently Charlie)

Dimensions and size will be determined by the following code:

```
 self.setImage("Charlie.png")
    self.setSize(25,25)
    self.position = (320,400)
    self.moveSpeed = 5
```

event handling is in the process method, which responds to the players directional input

```
def process(self):
            If user presses the left arrow:
                subtract the speed value from the x position
            if user presses the right arrow:
                add the speed value to the x position
            if user presses the up arrow:
```

subtract the speed value from the y position

if user presses the down arrow:

add the speed value to the y position

Other fruit (currently coin):

Coin is a simpleGE subclass sprite

It currently is represented by the following pseudocode, which allows it to fall from the top of the screen, and return to the top of the screen from the bottom of the screen while randomizing its speed

def reset(self):

set the y position to 10

set the coin x position to a random int between 0 and the screen width

set the y movement speed equal to a random interger between the minimum and maximum speed

def checkBounds(self):

if the bottom of the the coin is greater than the screen height:

reset the coin, giving it new random numbers as well

Blender (currently hurts):

The blender behaves identically to the coin, only with fewer on-screen sprites and faster speeds. It is designed to be avoided by the player. It is represented by the following pseudocode:

def __init__ with parameters self and scene

super().__init__(scene) so that we can reuse intialized scenes

set the hurtCharlie image (using self) (its the obstacle charlie should avoid)

set the hurts size to 25,25

set the hurts minimum speed to 5

set the hurts maximum speed to 8

self.reset() call function reset

def reset(self):

set the y position to 10

set the hurts x position to a random int between 0 and the screen width

set the y movement speed equal to a random interger between the minimum and maximum speed

def checkBounds(self):

if the bottom of the the hurt is greater than the screen height:

reset the hurt, giving it new random numbers as well

Main:

This allows the game to run, and is represented by the following code:

```
def main():
    game = Game()
    game.start()
if __name__ == "__main__":
  main()
```

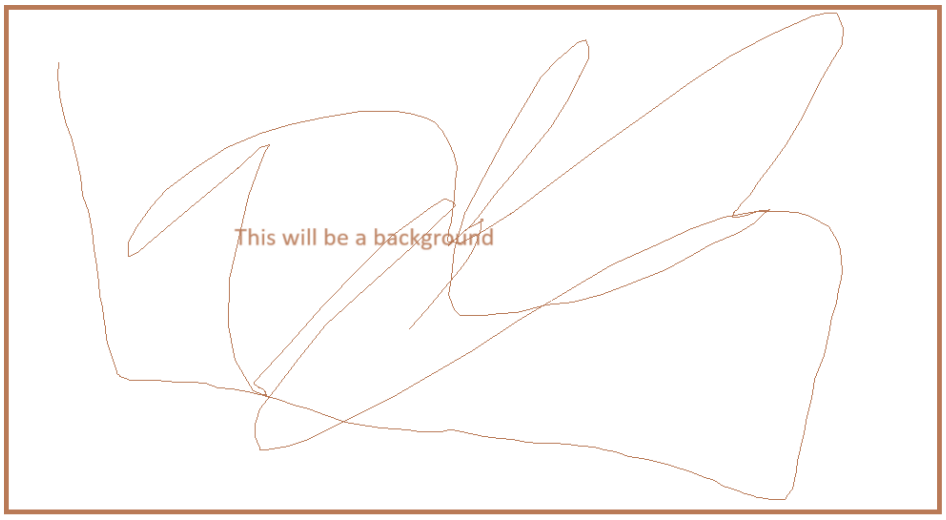
Milestone Plan:

This is the order of my milestone plan. The order will be gameplay, instructions, and then state management. These will each be completed in order, ideally, one after the other.

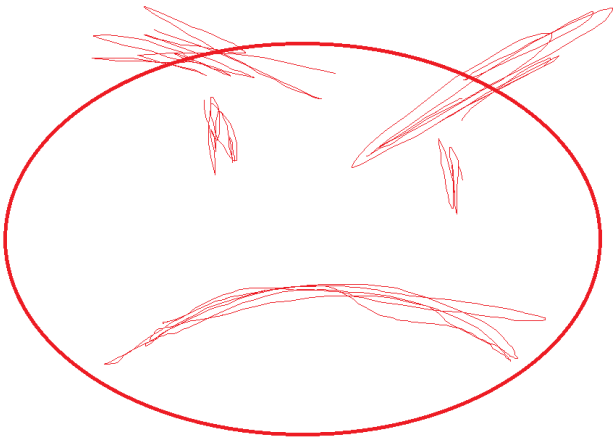
1. Game scene with background image
2. Add basic Charlie sprite
3. Add keyboard motion to Charlie
4. Add single coin with reset, falling and boundary behaviors
5. Add collision effect between charlie and coin, sound effect
6. Modify for multiple (ten) coins including collision behavior
7. Add scorekeeping, timing, and appropriate labels
8. Add instructions class and state transition

Assets: These are my current assets, and are subject to change (they will change)

BackgroundV1 (self created in Paint)



hurtCharlie (self-created in Paint)



Charlie (fair use BSU)

Coin.wav

Custom audio by Andy Harris created with jsfxr (in github)

hurtNoise.wav

custom audio by Ian Kowalski created with jsfxr (in github

Coin.png (self-created in paint)

Game Class Diagram