Pseudocode for version 1.0.0

///////////////////////////////////////////////////////////////////////////

Imports pygame, sys, random, math, time

from pygame.locals import \*

//two screen globals

screenWidth = 1280

screenHeight = 780

//displays the game

SCREEN = pygame.display.set\_mode((screenWidth, screenHeight), pygame.RESIZABLE)

//Adds a title to our game window. We may not need but we will see.

pygame.display.set\_caption(“ \*nameplaceholder here\* “)

//this is used for the FPS (we can see if we can handle 60FPS, if not lower it down to 30)

framePerSecond = 30

GAMECLOCK = pygame.time.clock()

//what the player and bots can eat other than each other.

foodBalls = pick a number between 2000 and 5000

//to set a max on bots up at all times.

botMax = pick a number between 20 and 50

//To keep track of map size. We can always grow this bigger in the future

mapSize = pick a number between 2000 and 5000

//This will be used to set the player’s initial radius (recommend setting playerSize and botMinSize to the same value.)

playerSize = pick a number between 10 and 25

//This will be used to set the bot’s initial radius

botMinSize = pick a number between 10 and 25

botMaxSize = pick a number between 150 and 200

//color and text stuff

playerColor = (pick a RGB value)

//this is a black color for the screen background

screenColor = (0, 0, 0)

textColor = (255, 255, 255)

FONT = pygame.font.Font(“ \*pick font\* “, pick size)

GAMEOVERFONT = pygame.font.Font((“ \*pick font\* “, pick size)

//size stuff

WIDTH = screenWidth

HEIGHT = screenHeight

//list

balls = []

bots = []

gameOver = false

// added in version 1.1.1 for a fix to a bug

// mouse position globals

mouse\_x = WIDTH / 2

mouse\_y = HEIGHT / 2

class create():

def \_\_init\_\_(self, x, y, color, size, name):

self.name = name

self.size = size

self.color = color

self.status = random.randint(1,8)

self.xPos = x

self.yPos = y

def botMovement(self):

soRandom = random.randint(1, round(self.size))

if soRandom == 1:

self.status = random.randint(1, 8)

if self.status == 1:

self.yPos += 300 / self.size

elif self.status == 2:

self.xPos += 300 / self.size

elif self.status == 3:

self.xPos += 150 / self.size

self.yPos += 150 / self.size

elif self.status == 4:

self.xPos += 150 / self.size

self.yPos -= 150 / self.size

elif self.status == 5:

self.xPos -= 150 / self.size

self.yPos += 150 / self.size

elif self.status == 6:

self.xPos -= 150 / self.size

self.yPos -= 150 / self.size

elif self.status == 7:

self.yPos -= 300 / self.size

elif self.status == 8:

self.yPos -= 300 / self.size

def collision(self, player):

global balls, bots, gameOver

for ball in balls:

if math.sqrt(((player.xPos – (WIDTH/2) + ball.xPos) \*\* 2 + (player.yPos – (HEIGHT/2) + ball.yPos) \*\* 2)) <= ball.size + player.size and ball.size <= player.size:

balls.remove(ball)

//This sets the growth size after player eats a ball

player.size += 0.25

//This will respawn balls

newBall = create(random.randint(-mapSize, mapSize), random.randint(-mapSize, mapSize), (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)), (pick a size that is smaller than the player and bot mass), “Ball”)

balls.append(newBall)

for bot in bots:

if math.sqrt(((player.xPos – (WIDTH/2) + bot.xPos) \*\* 2 + (player.yPos – (HEIGHT/2) + bot.yPos) \*\* 2)) <= bot.size + player.size and bot.size \* 1.05<= player.size:

botSize = math.pi \* (bot.size \*\* 2)

playerSize = math.pi \*(player.size \*\* 2)

newSize = math.sqrt((botSize + playerSize)/math.pi)

player.size += newSize – player.size

bot.remove(bot)

//This will respawn bots

newBot = create(random.randint(-mapSize, mapSize), random.randint(-mapSize, mapSize), (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)), random.randint(botMinSize, botMaxSize), “bot”)

bots.append(newBot)

elif math.sqrt(((player.xPos – (WIDTH/2) + bot.xPos) \*\* 2 + (player.yPos – (HEIGHT/2) + bot.yPos) \*\* 2)) <= bot.size + player.size and bot.size >= player.size \* 1.05:

botSize = math.pi \* (bot.size \*\* 2)

playerSize = math.pi \*(player.size\*\* 2)

newSize = math.sqrt((botSize + playerSize)/math.pi)

bot.size += newSize – player.size

gameOver = True

else:

for eatBot in bots:

if math.sqrt(((eatBot.xPos – (bot.xPos) \*\* 2 + (eatBot.yPos – bot.yPos) \*\* 2)) <= bot.size + eatBot.size and bot.size >= eatBot.size \* 1.05:

bots.remove(eatbot)

botSize = math.pi \* (bot.size\*\* 2)

eatBotSize = math.pi \*(eatBot.size \*\* 2)

newSize = math.sqrt((botSize + eatBotSize)/math.pi)

bot.size += newSize – bot.size

//This will respawn bots

newBot = create(random.randint(-mapSize, mapSize), random.randint(-mapSize, mapSize), (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)), random.randint(botMinSize, botMaxSize), “bot”)

bots.append(newBot)

for ball in balls:

if math.sqrt(((bot.xPos – ball.xPos) \*\* 2 + (bot.yPos – ball.yPos) \*\* 2)) <= ball.size + bot.size and ball.size <= bot.size:

balls.remove(ball)

//This sets the growth size after bot eats a ball

bot.size += 0.25

//This will respawn balls

newBall = create(random.randint(-mapSize, mapSize), random.randint(-mapSize, mapSize), (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)), (pick a size that is smaller than the player and bot mass), “Ball”)

balls.append(newBall)

//This draws the object of the balls, bots, and player. We will probably add to this in a later //version

def draw(self, surface, x, y):

pygame.draw.circle(surface, self.color, (x, y), int(self.size))

if self.name == “bot” or self.name == “player”:

message = FONT.render(str(round(self.size)), false, textColor)

//This centers the bot and player size in the middle of them.

Screen.bliz(message, (x – 17.5, y – 12.5))

//creates the foodBalls on the screen.

//random.randint(0,255) is give you the values for RGB.

for i in range (foodBalls):

newBall = create(random.randint(-mapSize, mapSize), random.randint(-mapSize, mapSize), (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)), (pick a size that is smaller than the player and bot mass), “Ball”)

balls.append(newBall)

for i in range (botMax):

newBot = create(random.randint(-mapSize, mapSize), random.randint(-mapSize, mapSize), (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255)), random.randint(botMinSize, botMaxSize), “bot”)

bots.append(newBot)

//to create the player.

player = create(0, 0, playerColor, playerSize, “Player”)

//game loop starts here

while true:

for event in pygame.event.get():

//to exit the game when needed. I things may change in our next versions when we added the menu.

if event.type == QUIT:

pygame.quit()

sys.exit()

//This is check Mouse x and y so that we can track the mouse for player movement

if event.type == MOUSEMOTION and gameOver == false:

mouse\_x, mouse\_y = event.pos

else: //This prevents errors

mouse\_x = WIDTH/2

mouse\_y = HEIGHT/2

//For drawing player and handling game over message

If gameOver == True:

message = GAMEOVERFONT.render( \* “Haha loser” or whatever \*, textColor or make another text color for this)

SCREEN.blit(message, ((WIDTH/2) - \*a number that will help center it or place it where we want\*, (HEIGHT/2) - \*same thing\*))

Else:

Player.draw(SCREEN, WIDTH/2, HEIGHT/2)

//keeping track of this will create the illusion that our player is moving when ready everything else is.

Player.xPos += round(-((mouse\_x – (WIDTH/2)) /player.size/2))

Player.yPos += round(-((mouse\_y – (HEIGHT/2)) /player.size/2))

//Keeps check of collisions

if not gameOver:

player.collision(player)

//map bound checks

// 1.2.0 fixed map bounds (instead of using just 5 number)

If player.xPos >= mapSize + (WIDTH/2):

player.xPos = mapSize + (WIDTH/2) -10

Elif player.xPos <= -mapsize + (WIDTH/2):

Player.xPos = -mapSize + (WIDTH/2) + 10

Else:

player.xPos += round(-((mouse\_x – (WIDTH/2))/player.size/2))

If player.yPos >= mapSize + (HEIGHT/2):

player.yPos = mapSize + (HEIGHT/2) - 10

Elif player.yPos <= -mapsize + (HEIGHT/2):

Player.yPos = -mapSize + (HEIGHT/2) + 10

Else:

player.yPos += round(-((mouse\_y – (HEIGHT/2))/player.size/2))

//for drawing the foodballs

for ball in balls:

ball.draw(SCREEN, ball.xPos + player.xPos, ball.yPos + player.yPos)

//for drawing the bots and setting bounds //1.2.0v fixed to bot bounds as well

for bot in bots:

bot.draw(SCREEN, bot.xPos + player.xPos, bot.yPos + player.yPos)

bot.botMovement()

if bot.xPos >= mapSize:

bot.xPos = mapSize - 10

elif bot.xPos <= -mapSize:

bot.xPos = -mapSize + 10

if bot.yPos >= mapSize:

bot.yPos = mapSize - 10

elif bot.yPos <= -mapSize:

bot.yPos = -mapSize + 10

WIDTH, HEIGTH = pygame.display.get\_surface().get\_size()

Screen.fill(screenColor)

Pygame.display.update()

GAMECLOCK.tick(framePerSecond)