Python Project Proposal

The Chicken Invader

Introduction to Computer and Programming

International College, King Mongkut’s Institute of Technology Ladkrabang

Panupong Viriyasuebpong

ID 59090018

Semester 1, 2016

Contents

Topic Page

Project Proposal 3

Screenshots 5

Source Code 10

1. Project Developer

59090018 - Panupong Viriyasuebpong

2. Project Title

The Chicken Invader

3. Project Description and Functions

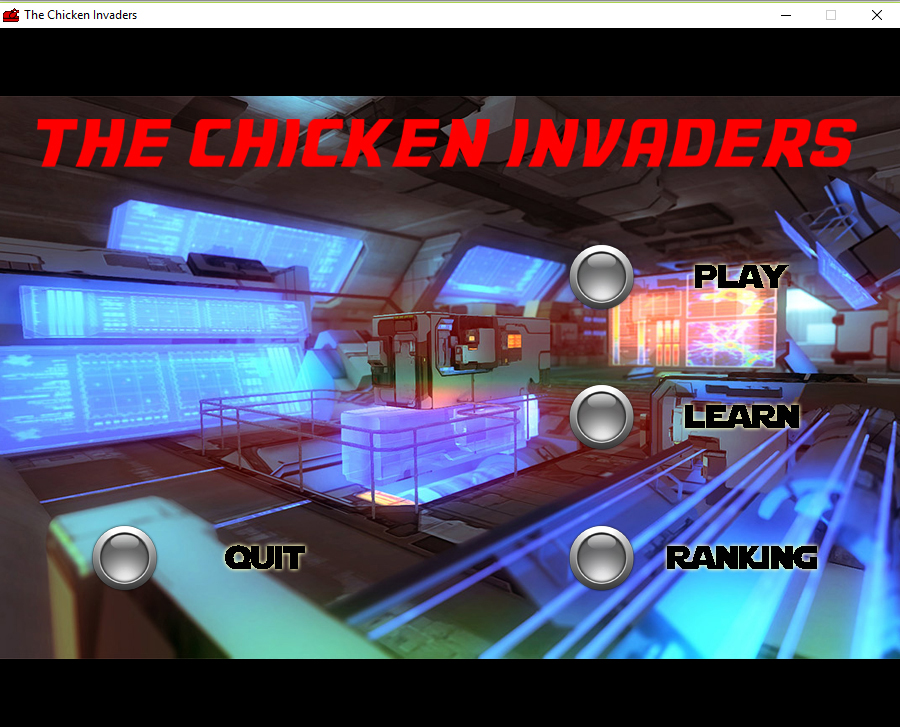
The Chicken Invader original game was released in 1999.  
It’s my inspiration on developing this project. It is an arcade – action - shooting game with impressive graphic interface. Player play as a spaceship controlled my mouse cursor. The objective is to survive through many kind of moving mad chickens and their boss. Provided with weapon and bonus randomly dropped, the player get stronger. After several levels (5 - 10) the survivors get to inscribe their name on the Hall of Fame or high score board.

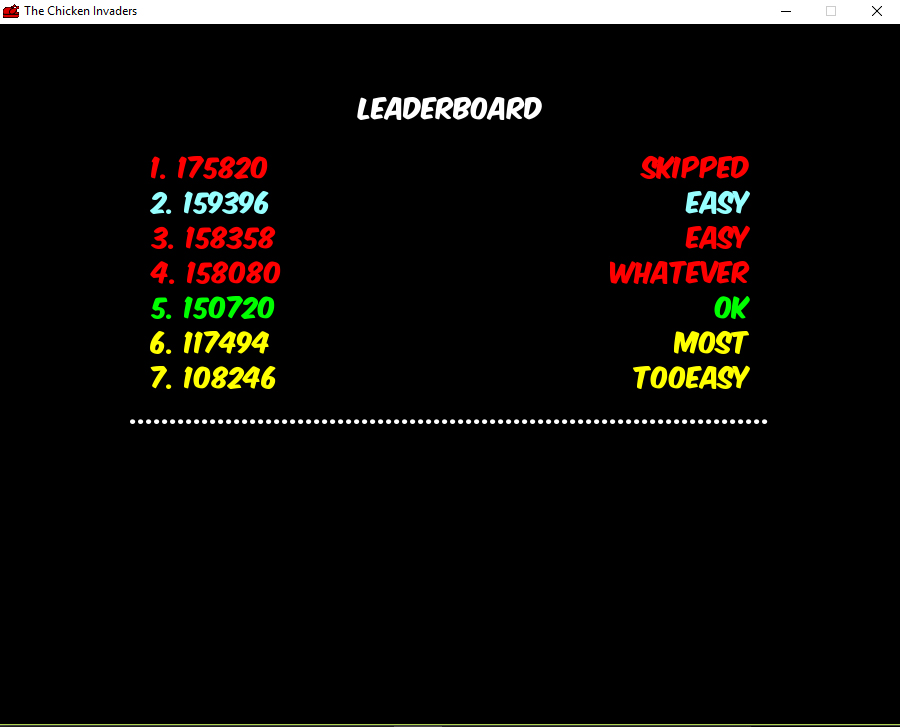
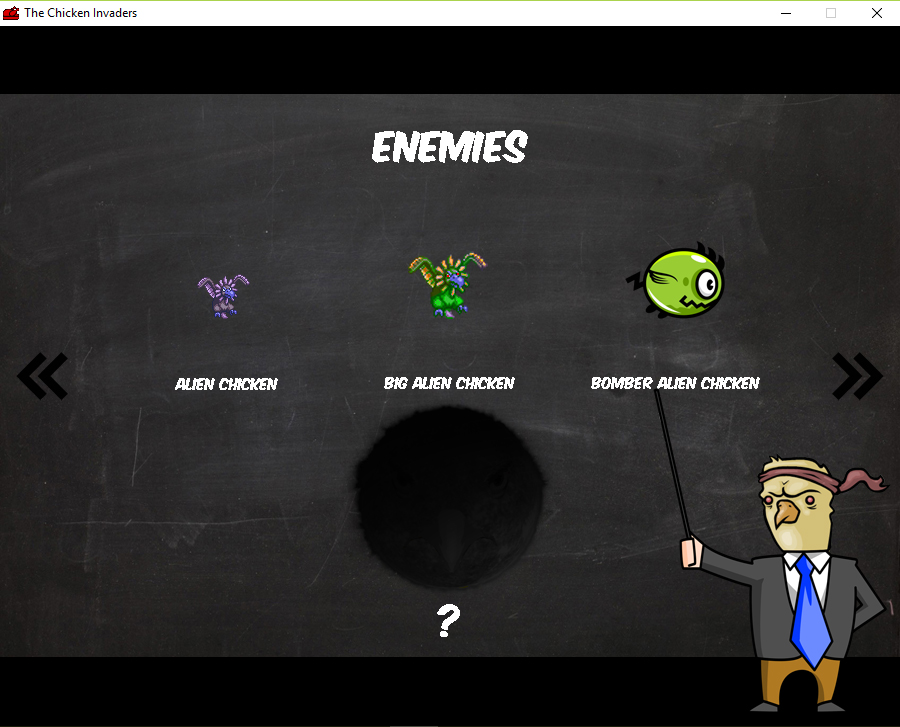
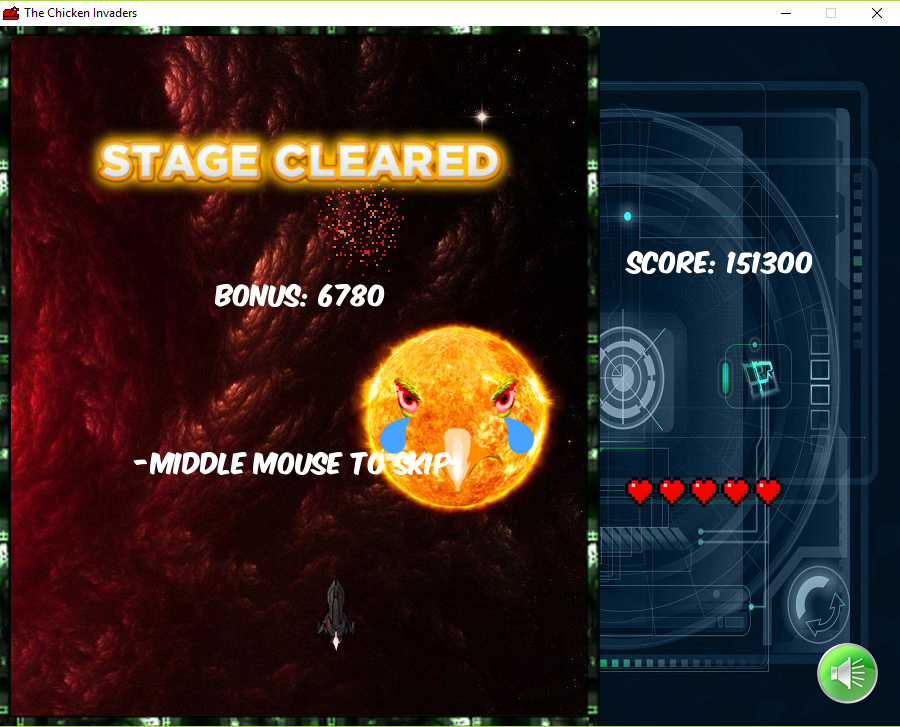
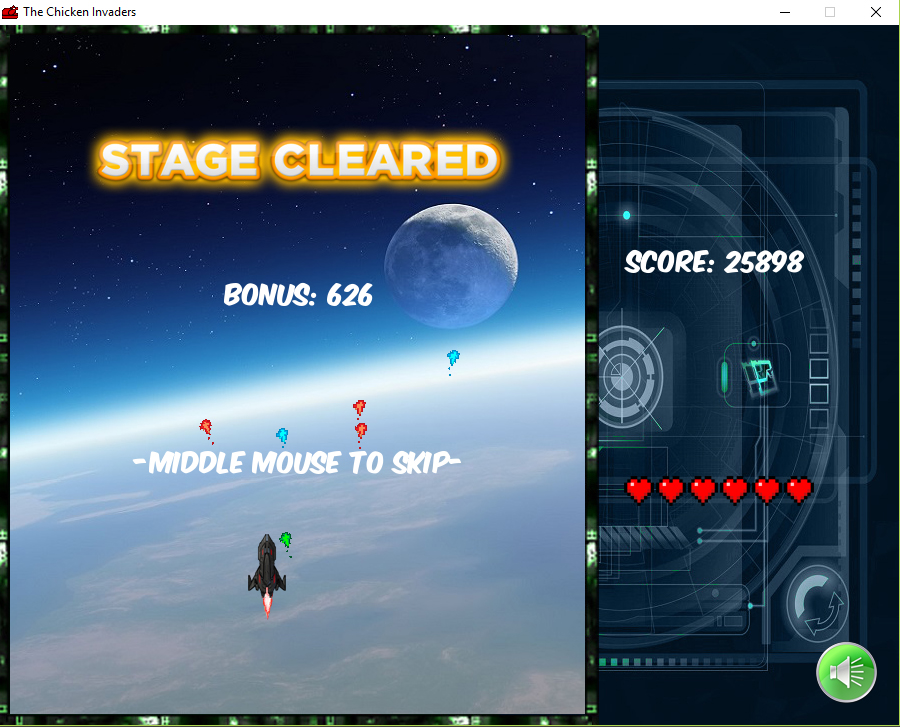
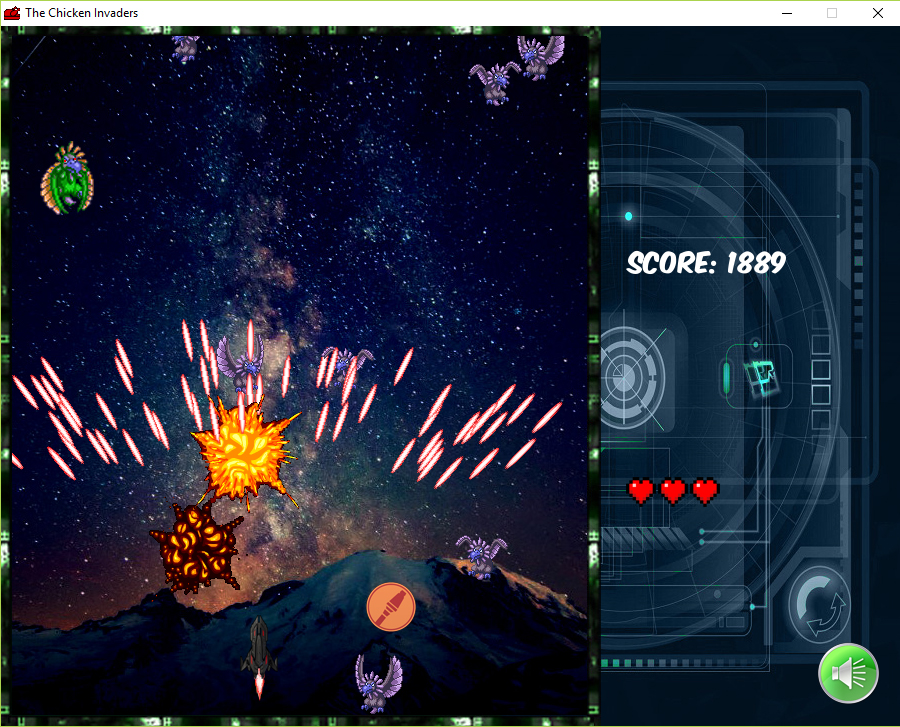
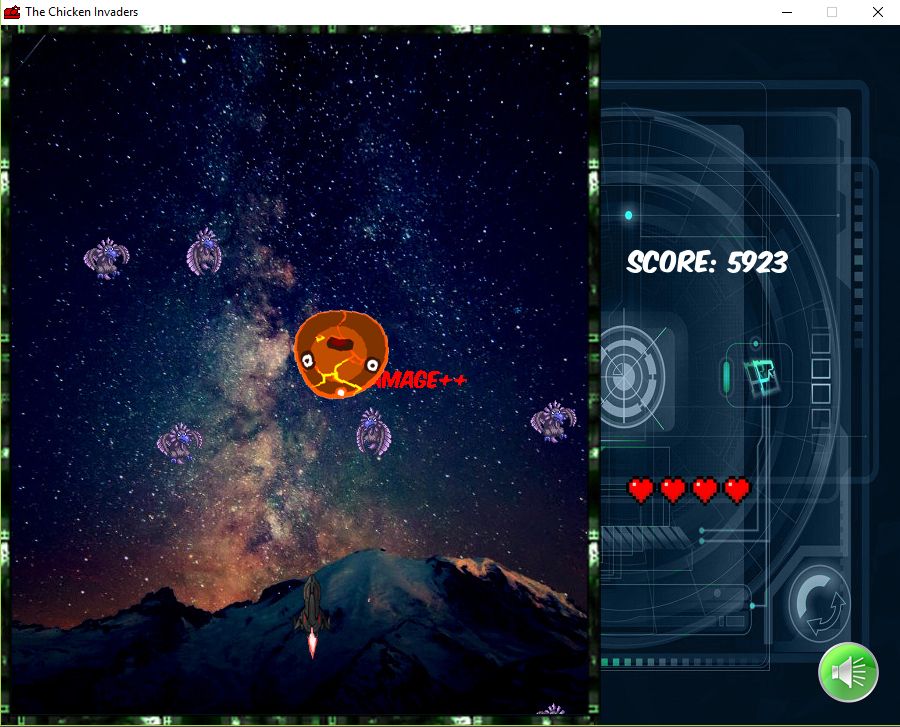
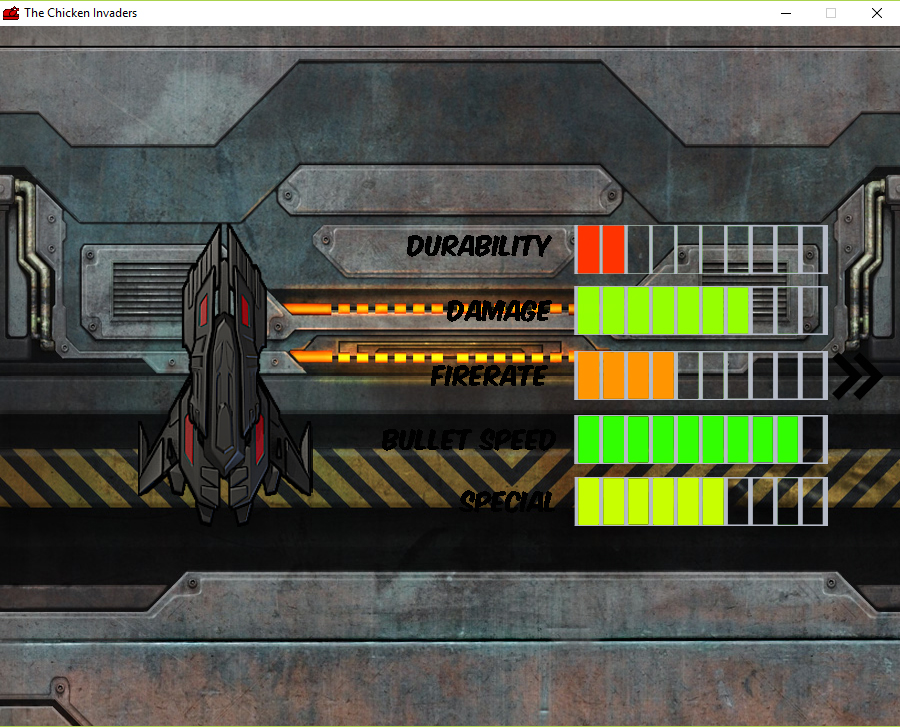
This game will be develop based on concept of OOP, which makes the program more stable and efficient. And also   
this will be a good opportunity for developer to get familiar with  
more object-oriented programming

4. Project Requirements

This project is based on the Pygame module. With ability to creates GUI window and receiving real time inputs.  
Also it comes with built-in classes and functions which is really useful on game developing. Without it all sprites, character and interfaces won’t be able to come to life.

Screenshots





Source code

#Main.py

from TheChickenInvaders import \*

def main():

TheChickenInvaders().main()

main()

#TheChickenInvaders.py

import pygame

from constants import \*

from Menus import \*

class TheChickenInvaders(object):

def \_\_init\_\_(self):

pygame.init()

pygame.display.set\_caption("The Chicken Invaders")

pygame.display.set\_icon(pygame.image.load("Resources\\Generals\\redicon.png"))

pygame.mixer.init()

self.screen = pygame.display.set\_mode([SCREEN\_WIDTH,SCREEN\_HEIGHT])

def main(self):

StartScreen(self.screen).main()

#Menus.py

import pygame

import sys

import random

import GameLoop

import Instruction

import Ranking

from constants import \*

from Buttons import \*

from Doors import \*

class StartScreen(object):

def \_\_init\_\_(self, screen):

self.screen = screen

self.clock = pygame.time.Clock()

self.done = False

pygame.mixer.music.load("Resources\\Musics\\intro.mp3")

pygame.mixer.music.play()

self.startImages = [pygame.image.load("Resources\\Menus\\sun1.jpg"),

pygame.image.load("Resources\\Menus\\sun2.jpg"),

pygame.image.load("Resources\\Menus\\sun3.jpg"),

pygame.image.load("Resources\\Menus\\sun4.jpg")]

def eventsInput(self):

for event in pygame.event.get():

if event.type == pygame.MOUSEBUTTONDOWN:

return True

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

def main(self):

index = 0

counter = 0

done = False

while not done:

done = self.eventsInput()

self.screen.blit(self.startImages[index], (0,(700 - 563)//2))

if counter >= 15:

index = (index + 1) % 4

counter = 0

counter += 1

self.clock.tick(60)

pygame.display.update()

MainMenu(self.screen).main()

class MainMenu(object):

def \_\_init\_\_(self, screen):

pygame.mouse.set\_visible(True)

if pygame.mixer.music.get\_busy():

pygame.mixer.music.fadeout(4000)

self.screen = screen

self.clock = pygame.time.Clock()

self.index = 0

self.counter = 0

self.images = [pygame.image.load("Resources\\Menus\\default.jpg"),

pygame.image.load("Resources\\Menus\\flashing.jpg")]

self.done = False

self.choice = 0

self.alreadyClick = False

self.buttonList = pygame.sprite.Group()

self.playBut = PlayButton(900 \* 0.63, 68.5 + (563 \* 0.2), self.buttonList)

self.learnBut = LearnButton(900 \* 0.63, 68.5 + (563 \* 0.45), self.buttonList)

self.rankingBut = RankingButton(900 \* 0.63, 68.5 + (563 \* 0.7), self.buttonList)

self.quitBut = QuitButton(900 \* 0.1, 68.5 + (563 \* 0.7), self.buttonList)

self.doorList = pygame.sprite.Group()

self.upperDoor = UpperDoor(self.doorList)

self.lowerDoor = LowerDoor(self.doorList)

self.mouseX = 0

self.mouseY = 0

def eventsInput(self):

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

for event in pygame.event.get():

if event.type == pygame.MOUSEBUTTONDOWN:

self.processChoice()

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

def processChoice(self):

if self.choice:

self.alreadyClick = True

self.upperDoor.playSound()

def displayFrame(self):

self.screen.fill(BLACK)

self.screen.blit(self.images[self.index], [0,(SCREEN\_HEIGHT - self.images[self.index].get\_height())//2])

self.buttonList.draw(self.screen)

self.doorList.draw(self.screen)

pygame.display.update()

def runLogic(self):

self.counter += random.randint(0,2)

if self.counter % 50 == 0 or self.counter % 51 == 0 or \

self.counter % 60 == 0 or self.counter % 61 == 0 or \

self.counter % 62 == 0:

self.index = 1

else:

self.index = 0

if not self.alreadyClick:

self.choice = 0

for button in self.buttonList:

if button.pointing:

self.choice = button.getChoice()

self.buttonList.update(self)

if self.alreadyClick:

self.doorList.update(self)

def main(self):

while not self.done:

self.eventsInput()

self.displayFrame()

self.runLogic()

self.clock.tick(60)

if self.choice == "Quit":

pygame.quit()

sys.exit()

elif self.choice == "Play":

CharacterSelect(self.screen).main()

elif self.choice == "Learn":

Instruction.Instruction(self.screen).main()

elif self.choice == "Ranking":

Ranking.Ranking(self.screen).main()

class CharacterSelect(object):

def \_\_init\_\_(self, screen):

self.screen = screen

self.clock = pygame.time.Clock()

self.done = False

self.playerClass = 1

self.moveOn = False

self.allSpriteList = pygame.sprite.Group()

self.lowerDoor = SpecialLowerDoor(self.allSpriteList)

self.upperDoor = SpecialUpperDoor(self.allSpriteList)

self.tempBackground = pygame.image.load("Resources\\Generals\\level1.jpg")

self.mask = pygame.image.load("Resources\\Generals\\background.png")

self.leftArrow = pygame.image.load("Resources\\Generals\\arrow14.png")

self.rightArrow = pygame.transform.flip(self.leftArrow, True, False)

self.stats = [pygame.image.load("Resources\\Stats\\1stat.png"),

pygame.image.load("Resources\\Stats\\2stat.png"),

pygame.image.load("Resources\\Stats\\3stat.png"),

pygame.image.load("Resources\\Stats\\4stat.png")]

self.mouseX = 0

self.mouseY = 0

def logics(self):

if self.moveOn:

self.allSpriteList.update(self)

def eventsInput(self):

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

for event in pygame.event.get():

if event.type == pygame.QUIT:

self.done = True

pygame.quit()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN:

if self.mouseX < 10 + self.leftArrow.get\_width():

if self.playerClass > 1:

self.playerClass -= 1

elif self.mouseX > SCREEN\_WIDTH -self.leftArrow.get\_width() - 10:

if self.playerClass < 4:

self.playerClass += 1

else:

self.processChoice()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

if self.playerClass > 1:

self.playerClass -= 1

elif event.key == pygame.K\_RIGHT:

if self.playerClass < 4:

self.playerClass += 1

def processChoice(self):

self.moveOn = True

self.upperDoor.playSound()

def displayFrame(self):

self.screen.blit(self.tempBackground, [10, 10])

self.screen.blit(self.mask, [0, 0])

self.allSpriteList.draw(self.screen)

if not self.moveOn:

self.screen.blit(self.stats[self.playerClass - 1], [0, 0])

if self.playerClass != 1:

self.screen.blit(self.leftArrow, [10, SCREEN\_HEIGHT//2 - self.leftArrow.get\_height()//2])

if self.playerClass != 4:

self.screen.blit(self.rightArrow, [SCREEN\_WIDTH - self.rightArrow.get\_width() - 10,

SCREEN\_HEIGHT//2 - self.leftArrow.get\_height()//2])

pygame.display.update()

def main(self):

while not self.done:

self.eventsInput()

self.displayFrame()

self.logics()

self.clock.tick(60)

GameLoop.GameLoop(self.screen, self.playerClass).main()

#GameLoop.py

import pygame

import Menus

import shortcuts

from Game import \*

from BossBattle import \*

class GameLoop(object):

def \_\_init\_\_(self, screen, playerClass):

self.screen = screen

self.playerClass = playerClass

self.gameOver = False

self.mainScore = 0

self.mainVolume = True

self.musicPlayed = []

self.mutables = [self.gameOver,

self.mainScore,

self.mainVolume,

self.musicPlayed]

def main(self):

for i in range(1, 6):

shortcuts.GameConstructor().get(i, self.screen, self.playerClass, self.mutables).main()

self.gameOver = self.mutables[0]

if self.gameOver:

Menus.MainMenu(self.screen).main()

if not self.gameOver:

BossBattle(self.screen, self.playerClass, self.mutables).main()

#Game.py

import pygame

import sys

import random

import shortcuts

from constants import \*

from Enemies import \*

from Items import \*

from Hazards import \*

from Projectiles import \*

class Game(object):

def \_\_init\_\_(self, screen,

playerClass, mutables,

chickenPerLine, chickenNum, chickenFrequency,

bigChickNum, bigChickFrequency,

meteorNum, boxNum,

bomberLeft, bomberRight,

picName):

self.screen = screen

self.clock = pygame.time.Clock()

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

self.mutables = mutables

self.gameOver = self.mutables[0]

self.score = self.mutables[1]

self.volume = self.mutables[2]

self.musicPlayed = self.mutables[3]

self.bonusScore = 0

self.win = False

self.pause = False

self.done = False

self.broken = False

self.lost = False

self.chickenPerLine = chickenPerLine

self.chickenNum = chickenNum

self.bigChickNum = bigChickNum

self.meteorNum = meteorNum

self.boxNum = boxNum

self.bomberLeft = bomberLeft

self.bomberRight = bomberRight

self.chickenFrequency = chickenFrequency

self.bigChickFrequency = bigChickFrequency

self.enemyList = pygame.sprite.Group()

self.bulletList = pygame.sprite.Group()

self.boxList = pygame.sprite.Group()

self.itemList = pygame.sprite.Group()

self.hazardList = pygame.sprite.Group()

self.fireworkList = pygame.sprite.Group()

self.allSpriteList = pygame.sprite.Group()

self.picName = picName

self.spaceBG = pygame.image.load(self.picName)

self.HUD = pygame.image.load("Resources\\Generals\\background.png")

self.pausemask = pygame.image.load("Resources\\Generals\\pausing.png")

self.heart = pygame.image.load("Resources\\Generals\\heart.png")

self.clear = pygame.image.load("Resources\\Generals\\clear.png")

self.broke = pygame.image.load("Resources\\Generals\\broken.png")

self.index = 0

self.gameOverPics = [pygame.image.load("Resources\\Generals\\gonone.png"),

pygame.image.load("Resources\\Generals\\goquit.png"),

pygame.image.load("Resources\\Generals\\goretry.png")]

self.volOn = pygame.image.load("Resources\\Generals\\volumeon.png")

self.volOff = pygame.image.load("Resources\\Generals\\volumeoff.png")

self.loseSound = pygame.mixer.Sound("Resources\\Sounds\\lose.wav")

if not pygame.mixer.music.get\_busy():

self.trackNum = random.randint(0, 11)

while self.trackNum in self.musicPlayed:

self.trackNum = random.randint(0, 11)

pygame.mixer.music.load("Resources\\Musics\\track" + str(self.trackNum) + ".mp3")

pygame.mixer.music.play(-1, 0.0)

self.mutables[3] = self.mutables[3] + [self.trackNum]

self.font = pygame.font.Font("Resources\\Generals\\font.ttf", 25)

for i in range(self.chickenNum):

for j in range(self.chickenPerLine):

chick = Chicken((i + 1) \* -self.chickenFrequency, self.enemyList, self.allSpriteList)

for i in range(self.bigChickNum):

bigchick = BigChick((i + 1) \* -self.bigChickFrequency, self.enemyList, self.allSpriteList)

if self.bomberRight > 0:

self.bomber = Bomber(self.bomberLeft, self.bomberRight, self.allSpriteList)

for i in range(self.boxNum):

box = Box((i + 1) \* random.randint(-600,-400), self.boxList, self.allSpriteList)

for i in range(self.meteorNum):

self.meteor = Meteor(self.hazardList, self.allSpriteList)

for i in range(6):

firework = Firework(self.fireworkList)

self.playerClass = playerClass

self.player = shortcuts.PlayerDict().get(self.playerClass, self.allSpriteList)

def processEvents(self):

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN:

if SCREEN\_WIDTH - self.volOn.get\_width() - 20 < self.mouseX < SCREEN\_WIDTH - 20 and \

SCREEN\_HEIGHT - self.volOn.get\_height() - 20 < self.mouseY < SCREEN\_HEIGHT - 20:

self.volume = not self.volume

pygame.mixer.music.set\_volume(float(self.volume))

else:

if self.win:

if event.button == 2:

if pygame.mixer.music.get\_busy():

pygame.mixer.music.stop()

if self.bonusScore > 0:

self.score += self.bonusScore

self.done = True

if self.gameOver:

if self.index == 1:

self.done = True

elif self.index == 2:

self.mutables[2] = self.volume

if self.\_\_class\_\_.\_\_name\_\_ == "Game":

self.mutables[1] -= 3000

self.\_\_init\_\_(self.screen,

self.playerClass,

self.mutables,

self.chickenPerLine,

self.chickenNum,

self.chickenFrequency,

self.bigChickNum,

self.bigChickFrequency,

self.meteorNum,

self.boxNum,

self.bomberLeft,

self.bomberRight,

self.picName)

elif self.\_\_class\_\_.\_\_name\_\_ == "BossBattle":

self.\_\_init\_\_(self.screen, self.playerClass, self.mutables)

elif event.button == 1:

self.player.firing = True

elif event.button == 3:

self.player.special(self)

elif event.type == pygame.MOUSEBUTTONUP:

if event.button == 1:

self.player.firing = False

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_p:

if self.pause:

pygame.mouse.set\_pos([self.player.rect.x + self.player.rect.width//2,

self.player.rect.y + self.player.rect.height//2])

self.pause = not self.pause

def runLogic(self):

if self.player.inBound and not self.pause and not self.gameOver:

pygame.mouse.set\_visible(False)

else:

pygame.mouse.set\_visible(True)

if not self.pause:

if len(self.enemyList) == 0 and not self.gameOver:

self.win = True

elif self.player.health <= 0:

self.gameOver = True

if self.win:

if pygame.mixer.music.get\_busy():

pygame.mixer.music.fadeout(2000)

for thing in self.hazardList:

thing.kill()

for box in self.boxList:

box.kill()

try:

self.bomber.kill()

except AttributeError:

pass

self.fireworkList.update(self)

if self.bonusScore > 0:

self.bonusScore -= 10

self.score += 10

elif self.gameOver:

if 10 < self.mouseX < 300:

self.index = 1

elif 300 < self.mouseX < 590:

self.index = 2

else:

self.index = 0

if not self.lost and self.volume:

pygame.mixer.Sound.play(self.loseSound)

self.lost = True

else:

self.bonusScore = self.score \* self.player.health//100

self.allSpriteList.update(self)

if len(self.fireworkList) == 0 and self.bonusScore < 0:

self.done = True

def displayFrame(self):

self.screen.blit(self.spaceBG, [10,10])

self.allSpriteList.draw(self.screen)

self.fireworkList.draw(self.screen)

self.screen.blit(self.HUD, [0,0])

if self.volume:

currentVol = self.volOn

else:

currentVol = self.volOff

self.screen.blit(currentVol, [SCREEN\_WIDTH - 84,

SCREEN\_HEIGHT - 84])

text = self.font.render("score: " + str(int(self.score)), True, WHITE)

x = 625

y = (SCREEN\_HEIGHT//3 - text.get\_height()//2)

self.screen.blit(text,[x,y])

for i in range(self.player.health):

self.screen.blit(self.heart, (625 + (i \* 32), SCREEN\_HEIGHT\*2//3 - self.heart.get\_height()//2))

tempCol = WHITE

if self.player.haveSpecial:

if random.randint(0,1):

tempCol = RED

textStr = "special ready!"

text = self.font.render(textStr, True, tempCol)

x = 625

y = (SCREEN\_HEIGHT//2 - text.get\_height()//2)

self.screen.blit(text,[x,y])

try:

if not self.boss.mad:

self.screen.blit(self.bossQuote, [self.boss.rect.x + self.boss.rect.width\*2//3,

self.boss.rect.y + self.boss.rect.height\*2//3])

pygame.draw.rect(self.screen, self.boss.color, (50, 20, self.boss.health//40, 20))

except:

pass

if self.win:

if self.bonusScore > 0:

scoreStr = "bonus: " + str(int(self.bonusScore))

else:

scoreStr = "bonus: 0"

text = self.font.render(scoreStr, True, WHITE)

x = (SCREEN\_WIDTH//3 - text.get\_width()//2)

y = (SCREEN\_HEIGHT//2 - text.get\_height()\*5//2)

self.screen.blit(text,[x,y])

text = self.font.render("-middle mouse to skip-", True, WHITE)

x = (SCREEN\_WIDTH//3 - text.get\_width()//2)

y = (SCREEN\_HEIGHT//2 + text.get\_height()\*3//2)

self.screen.blit(text,[x,y])

self.screen.blit(self.clear ,[300 - self.clear.get\_width()//2,100])

elif self.gameOver:

self.screen.blit(self.gameOverPics[self.index], [10, 10])

if self.broken:

self.screen.blit(self.broke, [10, 10])

if self.pause:

self.screen.blit(self.pausemask, [10, 10])

pygame.display.update()

def main(self):

while not self.done:

self.processEvents()

self.runLogic()

self.displayFrame()

self.clock.tick(60)

self.mutables[0] = self.gameOver

self.mutables[1] = int(self.score)

self.mutables[2] = self.volume

#BossBattle.py

import Menus

from Game import \*

from HighScore import \*

class BossBattle(Game):

def \_\_init\_\_(self, screen, playerClass, mutables):

super(BossBattle, self).\_\_init\_\_(screen,

playerClass, mutables,

0, 0, 0,

0, 0,

1, 5,

-100, -50,

"Resources\\Generals\\bossbg.jpg")

self.bonusScore = 10000

self.boss = Boss(self.allSpriteList)

self.bossQuote = pygame.image.load("Resources\\Generals\\quote.png")

def processEvents(self):

super().processEvents()

def runLogic(self):

if self.player.inBound and not self.pause and not self.gameOver:

pygame.mouse.set\_visible(False)

else:

pygame.mouse.set\_visible(True)

if not self.pause:

if not self.gameOver and not self.win:

if self.boss.health <= 0:

self.win = True

elif self.player.health <= 0:

self.gameOver = True

if self.win:

for thing in self.hazardList:

thing.kill()

for box in self.boxList:

box.kill()

self.fireworkList.update(self)

self.bonusScore -= 20

self.score += 20

elif self.gameOver:

if 10 < self.mouseX < 300:

self.index = 1

elif 300 < self.mouseX < 590:

self.index = 2

else:

self.index = 0

if not self.lost and self.volume:

pygame.mixer.Sound.play(self.loseSound)

self.lost = True

self.allSpriteList.update(self)

if len(self.fireworkList) == 0 and self.bonusScore <= 0:

self.done = True

def displayFrame(self):

super().displayFrame()

def main(self):

while not self.done:

self.processEvents()

self.runLogic()

self.displayFrame()

self.clock.tick(60)

if not self.gameOver:

HighScore(self.screen, self.playerClass, self.score, self.volume).main()

else:

Menus.MainMenu(self.screen).main()

#HighScore.py

import pygame

import Menus

from constants import \*

from Doors import \*

from ScoreProcessor import \*

class HighScore(object):

def \_\_init\_\_(self, screen, playerClass, score, volume):

self.screen = screen

self.clock = pygame.time.Clock()

self.score = score

self.playerClass = playerClass

if volume:

winSound = pygame.mixer.Sound("Resources\\Sounds\\win.wav")

pygame.mixer.Sound.play(winSound)

self.name = ""

self.background = pygame.image.load("Resources\\Generals\\party.jpg")

self.font = pygame.font.Font("Resources\\Generals\\font.ttf", 40)

self.done = False

self.alreadyClick = False

self.doorList = pygame.sprite.Group()

self.upperDoor = UpperDoor(self.doorList)

self.lowerDoor = LowerDoor(self.doorList)

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

def eventsInput(self):

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

for event in pygame.event.get():

if event.type == pygame.KEYDOWN:

self.name += chr(event.key)

if event.type == pygame.MOUSEBUTTONDOWN:

self.processChoice()

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

def processChoice(self):

self.alreadyClick = True

self.upperDoor.playSound()

def displayFrame(self):

self.screen.fill(BLACK)

self.screen.blit(self.background, [0, 0])

text = self.font.render("your score: " + str(self.score), True, WHITE)

x = SCREEN\_WIDTH//2 - text.get\_width()//2

y = SCREEN\_HEIGHT\*3//5 + text.get\_height()//2

self.screen.blit(text,[x,y])

text = self.font.render("enter your name: " + self.name, True, WHITE)

x = SCREEN\_WIDTH//2 - text.get\_width()//2

y = SCREEN\_HEIGHT\*4//5 - text.get\_height()//2

self.screen.blit(text,[x,y])

self.doorList.draw(self.screen)

pygame.display.update()

def runLogic(self):

if len(self.name):

lastChar = ord(self.name[-1:])

if lastChar == 8:

self.name = self.name[:-2]

elif lastChar == 13:

self.name = self.name[:-1]

self.processChoice()

elif lastChar not in range(49, 58) and lastChar not in range(65, 91) and lastChar not in range(97, 123):

self.name = self.name[:-1]

if self.alreadyClick:

self.doorList.update(self)

def main(self):

while not self.done:

self.displayFrame()

self.eventsInput()

self.runLogic()

self.clock.tick(60)

ScoreProcessor().writeScore(self.score, self.name.strip(), self.playerClass)

Menus.MainMenu(self.screen).main()

#ScoreProcessor.py

import pickle

from constants import \*

class ScoreProcessor(object):

def \_\_init\_\_ (self):

self.scores = self.getScores()

self.colors = self.getColors()

def getScores(self):

try:

inFile = open("Resources\\Documents\\highscore.dat","rb")

self.scores = pickle.load(inFile)

inFile.close()

except IOError:

return dict()

return self.scores

def getColors(self):

try:

inFile = open("Resources\\Documents\\highscore.dat","rb")

pickle.load(inFile)

self.colors = pickle.load(inFile)

inFile.close()

except IOError:

return dict()

return self.colors

def writeScore(self, score, name, playerClass):

if playerClass == 1:

tempCol = RED

elif playerClass == 2:

tempCol = GREEN

elif playerClass == 3:

tempCol = YELLOW

elif playerClass == 4:

tempCol = GREY

else:

tempCol = WHITE

while score in self.scores:

score += 1

self.scores[score] = name

self.colors[score] = tempCol

outFile = open("Resources\\Documents\\highscore.dat","wb")

pickle.dump(self.scores, outFile)

pickle.dump(self.colors, outFile)

outFile.close()

#Instruction.py

import pygame

import Menus

from Enemies import \*

from Doors import \*

class Instruction(object):

def \_\_init\_\_(self, screen):

self.screen = screen

self.clock = pygame.time.Clock()

self.done = False

self.index = 0

self.moveOn = False

self.chickenList = pygame.sprite.Group()

self.allSpriteList = pygame.sprite.Group()

self.lowerDoor = LowerDoor(self.allSpriteList)

self.upperDoor = UpperDoor(self.allSpriteList)

self.chicken = Chicken(0, self.chickenList)

self.bigChick = BigChick(0, self.chickenList)

self.bomber = Bomber(-1, SCREEN\_WIDTH + 1, self.chickenList)

self.chicken.rect.x = 225 - self.chicken.rect.width//2

self.chicken.rect.y = 68 + 563 \* 0.4 - self.chicken.rect.height

self.bigChick.rect.x = 450 - self.bigChick.rect.width//2

self.bigChick.rect.y = 68 + 563 \* 0.4 - self.bigChick.rect.height

self.bomber.rect.x = 675 - self.bomber.rect.width//2

self.bomber.rect.y = 68 + 563 \* 0.4 - self.bomber.rect.height

for chick in self.chickenList:

chick.moveX = chick.moveY = 0

try:

chick.moveSlow = 0

except AttributeError:

continue

self.leftArrow = pygame.image.load("Resources\\Generals\\arrow14.png")

self.rightArrow = pygame.transform.flip(self.leftArrow, True, False)

self.background = pygame.image.load("Resources\\Generals\\blackboard.jpg")

self.teacher = pygame.image.load("Resources\\Generals\\teacher.png")

self.contents = [pygame.image.load("Resources\\Learns\\int0.png"),

pygame.image.load("Resources\\Learns\\int1.png"),

pygame.image.load("Resources\\Learns\\int2.png")]

self.mouseX = 0

self.mouseY = 0

def logics(self):

if self.moveOn:

self.allSpriteList.update(self)

self.chickenList.update(self)

def eventsInput(self):

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

for event in pygame.event.get():

if event.type == pygame.QUIT:

self.done = True

pygame.quit()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN:

if self.mouseX < 10 + self.leftArrow.get\_width():

if self.index > 0:

self.index -= 1

elif self.mouseX > SCREEN\_WIDTH -self.leftArrow.get\_width() - 10:

if self.index < 2:

self.index += 1

else:

self.processChoice()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

if self.index > 0:

self.index -= 1

elif event.key == pygame.K\_RIGHT:

if self.index < 2:

self.index += 1

def processChoice(self):

self.moveOn = True

self.upperDoor.playSound()

def displayFrame(self):

self.screen.fill(BLACK)

self.screen.blit(self.background, [0, 68])

self.screen.blit(self.contents[self.index], [0, 68])

if self.index == 1:

self.chickenList.draw(self.screen)

self.screen.blit(self.teacher, [SCREEN\_WIDTH - self.teacher.get\_width(),

SCREEN\_HEIGHT - self.teacher.get\_height()])

if self.index != 0:

self.screen.blit(self.leftArrow, [10, SCREEN\_HEIGHT//2 - self.leftArrow.get\_height()//2])

if self.index != 2:

self.screen.blit(self.rightArrow, [SCREEN\_WIDTH - self.rightArrow.get\_width() - 10, SCREEN\_HEIGHT//2 - self.leftArrow.get\_height()//2])

self.allSpriteList.draw(self.screen)

pygame.display.update()

def main(self):

while not self.done:

self.eventsInput()

self.displayFrame()

self.logics()

self.clock.tick(60)

Menus.MainMenu(self.screen).main()

#Ranking.py

import pygame

import sys

import Menus

from ScoreProcessor import \*

from Doors import \*

class Ranking(object):

def \_\_init\_\_(self, screen):

self.screen = screen

self.clock = pygame.time.Clock()

self.index = 0

self.counter = 0

self.font = pygame.font.Font("Resources\\Generals\\font.ttf", 25)

self.done = False

self.alreadyClick = False

self.doorList = pygame.sprite.Group()

self.upperDoor = UpperDoor(self.doorList)

self.lowerDoor = LowerDoor(self.doorList)

self.mouseX = 0

self.mouseY = 0

self.scores = ScoreProcessor().getScores()

self.colors = ScoreProcessor().getColors()

self.headers = sorted(self.scores.keys())[::-1]

def eventsInput(self):

self.mouseX, self.mouseY = pygame.mouse.get\_pos()

for event in pygame.event.get():

if event.type == pygame.MOUSEBUTTONDOWN:

self.processChoice()

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

def processChoice(self):

self.alreadyClick = True

self.upperDoor.playSound()

def displayFrame(self):

self.screen.fill(BLACK)

text = self.font.render("leaderboard", True, WHITE)

x = SCREEN\_WIDTH//2 - text.get\_width()//2

y = 60

self.screen.blit(text, [x, y])

for i in range(15):

try:

text = self.font.render(str(i + 1) + ". " + str(self.headers[i]), True, self.colors[self.headers[i]])

x = 150

y = (i + 4) \* 35 - text.get\_height()//2

self.screen.blit(text, [x, y])

text = self.font.render(self.scores[self.headers[i]], True, self.colors[self.headers[i]])

x = SCREEN\_WIDTH - 150 - text.get\_width()

self.screen.blit(text, [x, y])

except IndexError:

break

text = self.font.render("." \* 80, True, WHITE)

x = SCREEN\_WIDTH//2 - text.get\_width()//2

y = (i + 4) \* 35 - text.get\_height()//2

self.screen.blit(text, [x, y])

self.doorList.draw(self.screen)

pygame.display.update()

def runLogic(self):

if self.alreadyClick:

self.doorList.update(self)

def main(self):

while not self.done:

self.eventsInput()

self.displayFrame()

self.runLogic()

self.clock.tick(60)

Menus.MainMenu(self.screen).main()

#contants.py

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLUE = (0, 0, 255)

YELLOW = (255, 255, 0)

GREY = (150, 255, 255)

SCREEN\_WIDTH = 900

SCREEN\_HEIGHT = 700

#shortcuts.py

import Player

import Items

import Game

class PlayerDict(object):

def \_\_init\_\_(self):

self.players = {1:Player.Ship1,

2:Player.Ship2,

3:Player.Ship3,

4:Player.Ship4}

def get(self, num, \*group):

return self.players.get(num)(\*group)

class ItemDict(object):

def \_\_init\_\_(self):

self.items = {1:Items.PowerUp1,

2:Items.PowerUp2,

3:Items.PowerUp3,

4:Items.PowerUp4,

5:Items.PowerUp5}

def get(self, num, \*group):

return self.items.get(num)(\*group)

class GameConstructor(object):

def \_\_init\_\_(self):

self.chickenPerLine = {1:2, 2:3, 3:4, 4:4, 5:4}

self.chickenNum = {1:15, 2:25, 3:30, 4:40, 5:50}

self.chickenFrequency = {1:300, 2:300, 3:300, 4:300, 5:300}

self.bigChickNum = {1:0 , 2:3, 3:4, 4:7, 5:12}

self.bigChickFrequency = {1:1500, 2:1500, 3:1500, 4:1000, 5:750}

self.meteorNum = {1:0 , 2:1, 3:1, 4:1, 5:1}

self.boxNum = {1:2 , 2:2, 3:2, 4:3, 5:3}

self.bomberLeft = {1:-100, 2:-100, 3:-600, 4:-500, 5:-400}

self.bomberRight = {1:-50, 2:-50, 3:1200, 4:1100, 5:1000}

self.picName = {1:"Resources\\Generals\\level1.jpg",

2:"Resources\\Generals\\level2.jpg",

3:"Resources\\Generals\\level3.jpg",

4:"Resources\\Generals\\level4.jpg",

5:"Resources\\Generals\\level5.jpg"}

def get(self, num, screen, playerClass, mutables):

return Game.Game(screen, playerClass, mutables,

self.chickenPerLine[num], self.chickenNum[num], self.chickenFrequency[num],

self.bigChickNum[num], self.bigChickFrequency[num],

self.meteorNum[num], self.boxNum[num],

self.bomberLeft[num], self.bomberRight[num],

self.picName[num])

#Player.py

import pygame

from constants import \*

from Bullets import \*

from Projectiles import \*

from Deaths import \*

class Player(pygame.sprite.Sprite):

def \_\_init\_\_(self, \*group):

super(Player, self).\_\_init\_\_(\*group)

self.firing = False

self.haveSpecial = False

self.inBound = True

self.scoreRate = 1

self.counter = 0

self.index = 0

self.bulletCounter = 20

self.bulletLimit = 20

def update(self, game):

if game.gameOver:

self.kill()

if self.counter >= 5:

self.index = (self.index + 1) % 2

self.image = self.images[self.index]

self.counter = 0

self.counter += 1

if game.mouseX + self.rect.width//2 < 590 and game.mouseX - self.rect.width//2 > 10 \

and game.mouseY - self.rect.height//2 > 10 and game.mouseY + self.rect.height//2 < 690:

self.rect.x = game.mouseX - self.rect.width//2

self.rect.y = game.mouseY - self.rect.height//2

self.inBound = True

else:

self.inBound = False

if self.firing:

self.bulletCounter += self.fireRate

if self.bulletCounter >= self.bulletLimit:

self.fireBullet(game)

self.bulletCounter = 0

else:

self.bulletCounter = self.bulletLimit

self.getItem(game)

def fireBullet(self, game):

pass

def special(self, game):

pass

def takeDamage(self, game):

playerDeath = PlayerDeath(self, game.volume, 1, game.allSpriteList)

self.health -= 1

def getItem(self, game):

getList = pygame.sprite.spritecollide(self, game.itemList, True)

for item in getList:

item.getPower(game)

class Ship1(Player):

def \_\_init\_\_(self, \*group):

super(Ship1, self).\_\_init\_\_(\*group)

self.health = 3

self.damage = 18

self.fireRate = 2

self.images = [pygame.image.load("Resources\\Sprites\\ship11.png"), pygame.image.load("Resources\\Sprites\\ship12.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.radius = self.rect.width//2

self.piercing = True

def update(self, game):

super().update(game)

def fireBullet(self, game):

bullet = Bullet1(self, game.volume, 1, 0, game.bulletList, game.allSpriteList)

def special(self, game):

if self.haveSpecial:

for i in range(60):

bullet = Bullet1(self, game.volume, 2, random.randint(-45,45), game.bulletList, game.allSpriteList)

bullet.speed = random.randint(25,35)

self.haveSpecial = False

class Ship2(Player):

def \_\_init\_\_(self, \*group):

super(Ship2, self).\_\_init\_\_(\*group)

self.health = 5

self.damage = 10

self.fireRate = 4

self.images = [pygame.image.load("Resources\\Sprites\\ship21.png"),

pygame.image.load("Resources\\Sprites\\ship22.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.radius = self.rect.width//2

self.piercing = False

self.specialCounter = 0

self.branch = 1

def update(self, game):

super().update(game)

if self.branch > 1:

self.specialCounter += 1

if self.specialCounter > 150:

self.branch -= 1

self.specialCounter = 0

def fireBullet(self, game):

for i in range(2):

for j in range(-self.branch + 1, self.branch):

bullet = Bullet2(self, game.volume, not i and not j, (self.rect.x + 7 + (i \* 43)), (j \* 20),

game.bulletList, game.allSpriteList)

def special(self, game):

if self.haveSpecial:

self.branch += 1

self.haveSpecial = False

class Ship3(Player):

def \_\_init\_\_(self, \*group):

super(Ship3, self).\_\_init\_\_(\*group)

self.health = 4

self.damage = 15

self.fireRate = 3

self.images = [pygame.image.load("Resources\\Sprites\\ship31.png"),

pygame.image.load("Resources\\Sprites\\ship32.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.radius = self.rect.width//2

self.piercing = False

def update(self, game):

super().update(game)

def fireBullet(self, game):

bullet = Bullet3(self, game.volume, 1, 0, game.bulletList, game.allSpriteList)

def special(self, game):

if self.haveSpecial:

missile = Missile(self, game.volume, game.allSpriteList)

self.haveSpecial = False

class Ship4(Player):

def \_\_init\_\_(self, \*group):

super(Ship4, self).\_\_init\_\_(\*group)

self.health = 4

self.damage = 25

self.fireRate = 1

self.images = [pygame.image.load("Resources\\Sprites\\ship41.png"),

pygame.image.load("Resources\\Sprites\\ship42.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.radius = self.rect.width//2

self.piercing = True

self.specialCounter = 0

def update(self, game):

if game.gameOver:

self.kill()

self.image = self.images[int(self.firing)]

pos = pygame.mouse.get\_pos()

if game.mouseX + self.rect.width//2 < 590 and game.mouseX - self.rect.width//2 > 10 \

and game.mouseY - self.rect.height//2 > 10 and game.mouseY + self.rect.height//2 < 690:

self.rect.x = game.mouseX - self.rect.width//2

self.rect.y = game.mouseY - self.rect.height//2

self.inBound = True

else:

self.inBound = False

if self.firing:

self.bulletCounter += self.fireRate

if self.bulletCounter >= self.bulletLimit:

self.fireBullet(game)

self.bulletCounter = 0

else:

self.bulletCounter = self.bulletLimit

if self.fireRate >= 100:

self.specialCounter += 1

if self.specialCounter >= 150:

self.fireRate = 2

self.specialCounter = 0

self.getItem(game)

def fireBullet(self, game):

if self.fireRate < 100:

bullet = Bullet4(self, game.volume, 1, 0, game.bulletList, game.allSpriteList)

else:

bullet = Bullet4(self, game.volume, 2, random.randint(-5,5), game.bulletList, game.allSpriteList)

def special(self, game):

if self.haveSpecial:

self.fireRate += 100

self.haveSpecial = False

#Bullets.py

import pygame

import random

import math

class Bullet(pygame.sprite.Sprite):

def \_\_init\_\_(self, bend, piercing, \*group):

super(Bullet, self).\_\_init\_\_(\*group)

self.bend = bend

self.piercing = piercing

def update(self, game):

try:

self.rect.y -= math.cos(self.bend \* math.pi / 180) \* self.speed

self.rect.x += math.tan(self.bend \* math.pi / 180) \* self.speed

except:

self.bend += 1

self.rect.y -= math.cos(self.bend \* math.pi / 180) \* self.speed

self.rect.x += math.tan(self.bend \* math.pi / 180) \* self.speed

if self.rect.y <= -self.rect.height:

self.kill()

class Bullet1(Bullet):

def \_\_init\_\_(self, ship, volume, soundType, bend = 0, \*group):

super(Bullet1, self).\_\_init\_\_(bend, ship.piercing, \*group)

self.speed = 30

self.image = pygame.image.load("Resources\\Sprites\\bullet1.png")

self.image = pygame.transform.rotate(self.image, -self.bend)

self.rect = self.image.get\_rect()

self.rect.x = ship.rect.x + ship.rect.width//2 - self.rect.width//2

self.rect.y = ship.rect.y + 10

if volume:

if soundType == 1:

soundRandomer = random.randint(0,2)

if soundRandomer == 0:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\laser.wav")

elif soundRandomer == 1:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\laser2.wav")

else:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\laser3.wav")

else:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\speciallaser.wav")

pygame.mixer.Sound.play(self.sound)

class Bullet2(Bullet):

def \_\_init\_\_(self, ship, volume, soundType, x, bend = 0, \*group):

super(Bullet2, self).\_\_init\_\_(bend, ship.piercing, \*group)

self.speed = 20

self.image = pygame.image.load("Resources\\Sprites\\bullet2.png")

self.image = pygame.transform.rotate(self.image, -self.bend)

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = ship.rect.y + 25

if volume:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\ak.wav")

if soundType == 1:

pygame.mixer.Sound.play(self.sound)

class Bullet3(Bullet):

def \_\_init\_\_(self, ship, volume, soundType, bend = 0, \*group):

super(Bullet3, self).\_\_init\_\_(bend, ship.piercing, \*group)

self.speed = 20

self.image = pygame.image.load("Resources\\Sprites\\bullet3New.png")

self.image = pygame.transform.rotate(self.image, -self.bend)

self.rect = self.image.get\_rect()

self.rect.x = ship.rect.x + ship.rect.width//2 - self.rect.width//2

self.rect.y = ship.rect.y + 10

if volume:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\pulse.wav")

if soundType:

pygame.mixer.Sound.play(self.sound)

class Bullet4(Bullet):

def \_\_init\_\_(self, ship, volume, soundType, bend = 0, \*group):

super(Bullet4, self).\_\_init\_\_(bend, ship.piercing, \*group)

self.speed = 25

self.image = pygame.image.load("Resources\\Sprites\\bullet4New.png")

self.image = pygame.transform.rotate(self.image, -self.bend)

self.rect = self.image.get\_rect()

self.rect.x = ship.rect.x + ship.rect.width//2 - self.rect.width//2

self.rect.y = ship.rect.y + 1

if volume:

if soundType == 1:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\pulse.wav")

else:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\ak.wav")

pygame.mixer.Sound.play(self.sound)

#Items.py

import pygame

import random

import shortcuts

from constants import \*

class Box(pygame.sprite.Sprite):

def \_\_init\_\_(self, y, \*group):

super(Box, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Sprites\\box.png")

self.rect = self.image.get\_rect()

self.rect.x = random.randint(10, 590 - self.rect.width)

self.rect.y = y

def resetPos(self):

self.rect.x = random.randint(10, 590 - self.rect.width)

self.rect.y = -random.randint(1400,1600)

def update(self, game):

self.rect.y += 4

if self.rect.y > SCREEN\_HEIGHT + 100:

self.resetPos()

if self.checkGotHit(game):

self.createBonus(game)

self.resetPos()

def checkGotHit(self, game):

if pygame.sprite.spritecollide(self, game.bulletList, False):

return True

return False

def createBonus(self, game):

item = shortcuts.ItemDict().get(random.randint(1,5), self, game.itemList, game.allSpriteList)

class Item(pygame.sprite.Sprite):

def \_\_init\_\_(self, \*group):

super(Item, self).\_\_init\_\_(\*group)

self.tag = None

def update(self, game):

self.rect.y += 3

def getPower(self, game):

game.player.scoreRate += 0.25

itemTag = ItemDisplay(self, self.tag, game.volume, game.allSpriteList)

class PowerUp1(Item):

def \_\_init\_\_(self, box, \*group):

super(PowerUp1, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Sprites\\getmissile.png")

self.rect = self.image.get\_rect()

self.rect.x = box.rect.x

self.rect.y = box.rect.y

self.tag = "Resources\\Sprites\\missileready.png"

def getPower(self, game):

super().getPower(game)

game.player.haveSpecial = True

class PowerUp2(Item):

def \_\_init\_\_(self, box, \*group):

super(PowerUp2, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Sprites\\gethealth.png")

self.rect = self.image.get\_rect()

self.rect.x = box.rect.x

self.rect.y = box.rect.y

self.tag = "Resources\\Sprites\\health++.png"

def getPower(self, game):

super().getPower(game)

if game.player.health < 7:

game.player.health += 1

class PowerUp3(Item):

def \_\_init\_\_(self, box, \*group):

super(PowerUp3, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Sprites\\getdamage.png")

self.rect = self.image.get\_rect()

self.rect.x = box.rect.x

self.rect.y = box.rect.y

self.tag = "Resources\\Sprites\\damage++.png"

def getPower(self, game):

super().getPower(game)

game.player.damage += 5

class PowerUp4(Item):

def \_\_init\_\_(self, box, \*group):

super(PowerUp4, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Sprites\\getfirerate.png")

self.rect = self.image.get\_rect()

self.rect.x = box.rect.x

self.rect.y = box.rect.y

self.tag = "Resources\\Sprites\\firerate++.png"

def getPower(self, game):

super().getPower(game)

game.player.fireRate += 1

class PowerUp5(Item):

def \_\_init\_\_(self, box, \*group):

super(PowerUp5, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Sprites\\getpierce.png")

self.rect = self.image.get\_rect()

self.rect.x = box.rect.x

self.rect.y = box.rect.y

self.tag1 = "Resources\\Sprites\\piercingbullet.png"

self.tag2 = "Resources\\Sprites\\damage++.png"

def getPower(self, game):

game.player.scoreRate += 0.25

if not game.player.piercing:

game.player.damage += 1

game.player.piercing = True

itemTag = ItemDisplay(self, self.tag1, game.volume, game.allSpriteList)

else:

game.player.damage += 5

itemTag = ItemDisplay(self, self.tag2, game.volume, game.allSpriteList)

class ItemDisplay(pygame.sprite.Sprite):

def \_\_init\_\_(self, bonus, tag, volume, \*group):

super(ItemDisplay, self).\_\_init\_\_(\*group)

self.counter = 0

self.image = pygame.image.load(tag)

self.rect = self.image.get\_rect()

self.rect.x = bonus.rect.x + bonus.rect.width//2 - self.rect.width//2

self.rect.y = bonus.rect.y

if volume:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\coin.wav")

pygame.mixer.Sound.play(self.sound)

def update(self, game):

self.rect.y -= 3

self.counter += 1

if self.counter >= 30:

self.kill()

#Projectiles.py

import pygame

import random

import math

from constants import \*

from Bullets import \*

class Firework(Bullet):

def \_\_init\_\_(self, \*group):

super(Firework, self).\_\_init\_\_(random.randint(-25,25), False, \*group)

self.speed = random.randint(4,6)

self.color = random.randint(0,2)

self.limit = random.randint(100,200)

self.index = 0

self.images = [pygame.image.load("Resources\\Fireworks\\set"+str(self.color)+ "0.png"),

pygame.image.load("Resources\\Fireworks\\set"+str(self.color)+ "1.png"),

pygame.image.load("Resources\\Fireworks\\set"+str(self.color)+ "2.png"),

pygame.image.load("Resources\\Fireworks\\set"+str(self.color)+ "3.png"),

pygame.image.load("Resources\\Fireworks\\set"+str(self.color)+ "4.png")]

self.image = pygame.transform.rotate(self.images[self.index], -self.bend)

self.rect = self.image.get\_rect()

self.rect.x = 300

self.rect.y = SCREEN\_HEIGHT + random.randint(25,75)

self.runSound = pygame.mixer.Sound("Resources\\Sounds\\fwrun.wav")

self.crackSound = pygame.mixer.Sound("Resources\\Sounds\\fwcrack.wav")

self.ran = False

self.exploded = False

def update(self, game):

if self.rect.y > self.limit:

super().update(game)

if not self.ran and self.rect.y > SCREEN\_HEIGHT:

if game.volume:

pygame.mixer.Sound.play(self.runSound)

self.ran = True

else:

if not self.exploded:

if game.volume:

pygame.mixer.Sound.play(self.crackSound)

self.exploded = True

self.index += 1

self.image = self.images[self.index]

if self.index == 4:

self.kill()

class Missile(Bullet):

def \_\_init\_\_(self, ship, volume, \*group):

super(Missile, self).\_\_init\_\_(0, False, \*group)

self.image = pygame.image.load("Resources\\Sprites\\missile.png")

self.rect = self.image.get\_rect()

self.rect.x = ship.rect.x + ship.rect.width//2 - self.rect.width//2

self.rect.y = ship.rect.y - 10

self.centered = False

self.hitted = False

self.explodeImage = pygame.image.load("Resources\\Sprites\\boom.png")

self.sideMove = -3

if self.rect.x + self.rect.width//2 < 300:

self.sideMove = 3

self.delayHit = 0

if volume:

self.initSound = pygame.mixer.Sound("Resources\\Sounds\\cannon.wav")

pygame.mixer.Sound.play(self.initSound)

self.explodeSound = pygame.mixer.Sound("Resources\\Sounds\\explosion.wav")

def update(self, game):

if 290 < self.rect.x + self.rect.width//2 < 310:

self.centered = True

if not self.hitted:

if not self.centered:

self.rect.x += self.sideMove

else:

self.rect.y -= 8

self.checkHitAnything(game)

else:

if self.delayHit > 60:

self.kill()

self.delayHit += 1

def checkHitAnything(self, game):

if pygame.sprite.spritecollide(self, game.enemyList, False) or \

pygame.sprite.spritecollide(self, game.hazardList, False):

if game.volume:

pygame.mixer.Sound.play(self.explodeSound)

game.broken = True

self.hitted = True

self.rect.x -= self.explodeImage.get\_width()//2 - self.image.get\_width()//2

self.rect.y -= self.explodeImage.get\_height()//2 - self.image.get\_height()//2

self.image = self.explodeImage

for enemy in game.enemyList:

if enemy.rect.y > 0:

game.score += int(enemy.health \* game.player.scoreRate)

enemy.kill(game, True)

try:

if pygame.sprite.collide\_rect(self, game.boss):

if game.volume:

pygame.mixer.Sound.play(self.explodeSound)

self.hitted = True

self.rect.x -= self.explodeImage.get\_width()//2 - self.image.get\_width()//2

self.rect.y -= self.explodeImage.get\_height()//2 - self.image.get\_height()//2

self.image = self.explodeImage

game.boss.health -= game.player.damage \* 50

except AttributeError:

pass

#Enemies.py

import pygame

import random

from constants import \*

from Hazards import \*

from Deaths import \*

class Chicken(pygame.sprite.Sprite):

def \_\_init\_\_(self, y, \*group):

super(Chicken, self).\_\_init\_\_(\*group)

self.moveX = random.randint(-2,2) \* 3

while self.moveX == 0:

self.moveX = random.randint(-2,2) \* 3

self.moveY = 6

self.health = 20

self.slowLine = random.randint(275, 325)

self.moveSlow = 2

self.wingSpeed = 2

self.index = random.randint(1,2)

self.changer = -1

self.counter = 0

self.images = [pygame.image.load("Resources\\Sprites\\chicken1.png"),

pygame.image.load("Resources\\Sprites\\chicken2.png"),

pygame.image.load("Resources\\Sprites\\chicken3.png"),

pygame.image.load("Resources\\Sprites\\chicken4.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = random.randint(11, 589 - self.rect.width)

self.rect.y = y + random.randint(-30,30)

def reset\_pos(self):

self.rect.y = random.randint(-300, - 20)

self.rect.x = random.randint(10, 590 - self.rect.width)

def update(self, game):

if self.moveX < 0:

self.image = pygame.transform.flip(self.images[self.index], True, False)

else:

self.image = self.images[self.index]

if self.counter == 10:

if self.index == 0 or self.index == 3:

self.changer \*= -1

self.index += self.changer

self.counter = 0

self.counter += self.wingSpeed

if self.rect.left < 10 or self.rect.right > 590:

self.moveX \*= -1

self.rect.x += self.moveX

if self.rect.y <= self.slowLine:

self.rect.y += self.moveY

else:

self.rect.y += self.moveSlow

if self.rect.y > SCREEN\_HEIGHT:

self.kill()

if self.health <= 0:

self.kill(game, True)

try:

if not game.gameOver:

self.checkHitPlayer(game)

self.checkGotHit(game)

except AttributeError:

pass

def checkGotHit(self, game):

for bullet in pygame.sprite.spritecollide(self, game.bulletList, False):

self.health -= game.player.damage

game.score += int(game.player.damage \* game.player.scoreRate)

if not bullet.piercing:

bullet.kill()

def checkHitPlayer(self, game):

if pygame.sprite.collide\_rect(self, game.player):

self.kill(game, True)

game.player.takeDamage(game)

def kill(self, game = None, inScreen = False):

super().kill()

if inScreen:

death = Death(self, game.volume, 1, game.allSpriteList)

class BigChick(Chicken):

def \_\_init\_\_(self, y, \*group):

super(BigChick, self).\_\_init\_\_(y, \*group)

self.moveY = 4

self.health = 500

self.slowLine = 150

self.moveSlow = 1

self.wingSpeed = 1

self.images=

[pygame.image.load("Resources\\Sprites\\bigchick1.png"), pygame.image.load("Resources\\Sprites\\bigchick2.png"),

pygame.image.load("Resources\\Sprites\\bigchick3.png"),

pygame.image.load("Resources\\Sprites\\bigchick4.png")]

self.rect = self.images[self.index].get\_rect()

self.rect.x = random.randint(11, 589 - self.rect.width)

self.rect.y = y

class Bomber(Chicken):

def \_\_init\_\_(self, leftBorder, rightBorder, \*group):

super(Bomber, self).\_\_init\_\_(random.randint(50, 150), \*group)

self.leftBorder = leftBorder

self.rightBorder = rightBorder

self.index = 1

self.counter = 0

self.changer = 1

self.moveY = 0

self.moveX = 10

self.images = [pygame.image.load("Resources\\Sprites\\bomber1.png"),

pygame.image.load("Resources\\Sprites\\bomber2.png"),

pygame.image.load("Resources\\Sprites\\bomber3.png"),

pygame.image.load("Resources\\Sprites\\bomber4.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = self.leftBorder

self.rect.y = 100

def update(self, game):

self.rect.x += self.moveX

if self.rect.right > self.rightBorder or self.rect.left < self.leftBorder:

self.moveX \*= -1

self.rect.y = random.randint(50, 150)

if self.moveX < 0:

self.image = pygame.transform.flip(self.images[self.index], True, False)

else:

self.image = self.images[self.index]

if self.counter == 10:

if self.index == 0 or self.index == 3:

self.changer \*= -1

self.index += self.changer

self.counter = 0

self.counter += 1

try:

self.checkGotHit(game)

except AttributeError:

pass

def checkGotHit(self, game):

for bullet in pygame.sprite.spritecollide(self, game.bulletList, True):

self.createBomb(game)

bullet.kill()

def createBomb(self, game):

if 10 < self.rect.x < 590 - self.rect.width:

if len(game.hazardList) - game.meteorNum < 1:

egg = EggBomb(self, game.volume, game.hazardList, game.allSpriteList)

class Boss(pygame.sprite.Sprite):

def \_\_init\_\_(self, \*group):

super(Boss, self).\_\_init\_\_(\*group)

self.health = 20000

self.mad = False

self.index = 0

self.image = pygame.image.load("Resources\\Sprites\\boss.png")

self.cry = pygame.image.load("Resources\\Sprites\\bosscry.png")

self.original = self.image

self.color = [0, 255, 0]

self.rect = self.image.get\_rect()

self.rect.x = 300 - self.rect.width//2

self.rect.y = 31

self.radius = 100

self.moveX = 0

self.moveY = 8

self.madSpeed = 3

self.roundTime = 0

self.firstMove = True

def update(self, game):

if self.health > 0:

if not self.mad:

if self.roundTime > 50:

self.shootFireball(game)

self.mad = True

self.firstMove = True

self.roundTime = 0

self.madSpeed += 1

self.image = self.original

self.moveX = 0

self.moveY = 0

else:

if self.roundTime > 400 and self.rect.y < 100 and 200 < self.rect.x + self.rect.width//2 < 400:

self.mad = False

self.roundTime = 0

self.image = pygame.transform.rotate(self.image, 90)

if self.firstMove:

if random.randint(0,1):

self.moveX = self.madSpeed

else:

self.moveX = -self.madSpeed

self.moveY = -self.madSpeed

self.firstMove = False

if self.rect.x + self.rect.width >= 590 or self.rect.x <= 10:

self.moveX \*= -1

if self.rect.y + self.rect.height >= 690 or self.rect.y <= 10:

self.moveY \*= -1

if not game.gameOver:

self.checkHitPlayer(game)

self.checkHitBullet(game)

self.rect.x += self.moveX

self.rect.y += self.moveY

self.roundTime += 1

else:

self.image = self.cry

self.updateColor()

def shootFireball(self, game):

for i in range(-2, 3):

fireball = Fireball(self, game.volume, i \* 10, game.hazardList, game.allSpriteList)

def checkHitPlayer(self, game):

if pygame.sprite.collide\_circle(self, game.player):

game.player.takeDamage(game)

self.moveX \*= -1

self.moveY \*= -1

def checkHitBullet(self, game):

hitList = pygame.sprite.spritecollide(self, game.bulletList, True)

for bullet in hitList:

if not self.mad:

self.health -= game.player.damage

else:

self.health -= game.player.damage//2

def updateColor(self):

if self.health > 10000:

self.color[0] = (1 - ((self.health - 10000)/10000)) \* 255

else:

self.color[0] = 255

self.color[1] = self.health/10000 \* 255

#Hazards.py

import pygame

import math

import sys

import random

from constants import \*

class Meteor(pygame.sprite.Sprite):

index = 0

def \_\_init\_\_(self, \*group):

super(Meteor, self).\_\_init\_\_(\*group)

self.images = []

self.images.append(pygame.image.load("Resources\\Sprites\\meteor.png"))

for i in range(72):

self.images.append(pygame.transform.rotate(self.images[0], (i + 1)\*5))

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = random.randint(10, 590 - self.rect.width)

self.rect.y = -1000

self.passed = False

self.sound = pygame.mixer.Sound("Resources\\Sounds\\meteorpass.wav")

def resetPos(self):

self.rect.x = random.randint(10, 590 - self.rect.width)

self.rect.y -= 3000

self.passed = False

def update(self, game):

self.index = (self.index + 1) % 72

self.image = self.images[self.index]

self.rect.y += 15

if self.rect.y > SCREEN\_HEIGHT + 100:

self.resetPos()

if not self.passed and self.rect.y > -50:

if game.volume:

self.playSound()

self.passed = True

if not game.gameOver:

self.checkHitPlayer(game)

self.checkHitBullet(game)

def checkHitPlayer(self, game):

if pygame.sprite.collide\_rect(self, game.player):

game.player.takeDamage(game)

self.resetPos()

def checkHitBullet(self, game):

pygame.sprite.spritecollide(self, game.bulletList, True)

def playSound(self):

pygame.mixer.Sound.play(self.sound)

self.passed = True

class EggBomb(Meteor):

def \_\_init\_\_(self, bomber, volume, \*group):

super(EggBomb, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Sprites\\rotten.png")

self.rect = self.image.get\_rect()

self.rect.x = bomber.rect.x + bomber.rect.width//2 - self.rect.width//2

self.rect.y = bomber.rect.y

if volume:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\bounce.wav")

pygame.mixer.Sound.play(self.sound)

def update(self, game):

self.rect.y += 15

if self.rect.y > SCREEN\_HEIGHT:

self.kill()

self.checkHitPlayer(game)

self.checkHitBullet(game)

def checkHitPlayer(self, game):

if pygame.sprite.collide\_rect(self, game.player):

game.player.takeDamage(game)

self.kill()

class Fireball(EggBomb):

def \_\_init\_\_(self, boss, volume, bend, \*group):

super(Fireball, self).\_\_init\_\_(boss, False, \*group)

self.bend = bend

self.image = pygame.image.load("Resources\\Sprites\\fireball.png")

self.image = pygame.transform.rotate(self.image, self.bend)

self.rect = self.image.get\_rect()

self.rect.x = boss.rect.x + boss.rect.width//2 - self.rect.width//2

self.rect.y = boss.rect.y + boss.rect.height//2 - self.rect.height//2

if volume:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\fireball.wav")

pygame.mixer.Sound.play(self.sound)

def update(self, game):

if self.rect.y > SCREEN\_HEIGHT:

self.kill()

self.rect.y += math.cos(self.bend \* math.pi / 180) \* 15

self.rect.x += math.tan(self.bend \* math.pi / 180) \* 15

if not game.gameOver:

self.checkHitPlayer(game)

self.checkHitBullet(game)

#Deaths.py

import pygame

import random

class Death(pygame.sprite.Sprite):

def \_\_init\_\_(self, chick, volume, soundType, \*group):

super(Death, self).\_\_init\_\_(\*group)

self.images = []

for i in range(14):

self.images.append(pygame.image.load("Resources\\Explode\\"+str(i)+".png"))

self.index = 0

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = chick.rect.x + chick.rect.width//2 - self.rect.width//2

self.rect.y = chick.rect.y + chick.rect.height//2 - self.rect.height//2

if volume:

if soundType == 1:

soundRandomer = random.randint(0,2)

if soundRandomer == 0:

self.sound = pygame.mixer.Sound("Resources\\Explode\\deathSound.wav")

elif soundRandomer == 1:

self.sound = pygame.mixer.Sound("Resources\\Explode\\deathsound2.wav")

else:

self.sound = pygame.mixer.Sound("Resources\\Explode\\deathsound3.wav")

pygame.mixer.Sound.play(self.sound)

def update(self, game):

self.index += 1

self.image = self.images[self.index]

if self.index == len(self.images) - 1:

self.kill()

class PlayerDeath(Death):

def \_\_init\_\_(self, player, volume, soundType, \*group):

super(PlayerDeath, self).\_\_init\_\_(player, 0, soundType, \*group)

self.volume = volume

self.images = []

for i in range(20):

self.images.append(pygame.image.load("Resources\\Burst\\" + str(i) + ".png"))

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = player.rect.x + player.rect.width//2 - self.rect.width//2

self.rect.y = player.rect.y + player.rect.height//2 - self.rect.height//2

if self.volume:

self.sound = pygame.mixer.Sound("Resources\\Sounds\\crash.wav")

pygame.mixer.Sound.play(self.sound)

#Buttons.py

import pygame

import sys

class Button(pygame.sprite.Sprite):

def \_\_init\_\_(self, \*group):

super(Button, self).\_\_init\_\_(\*group)

self.index = 0

self.choice = 0

self.pointing = False

def update(self, menu):

try:

if self.rect.right > menu.mouseX > self.rect.left and self.rect.y < menu.mouseY < self.rect.y + self.rect.height:

self.index = 1

self.pointing = True

else:

self.index = 0

self.pointing = False

self.image = self.images[self.index]

except AttributeError:

pass

def getChoice(self):

return self.choice

class PlayButton(Button):

def \_\_init\_\_(self, x, y, \*group):

super(PlayButton, self).\_\_init\_\_(\*group)

self.images = [pygame.image.load("Resources\\Menus\\playoff.png"),

pygame.image.load("Resources\\Menus\\playon.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.choice = "Play"

class LearnButton(Button):

def \_\_init\_\_(self, x, y, \*group):

super(LearnButton, self).\_\_init\_\_(\*group)

self.images = [pygame.image.load("Resources\\Menus\\learnoff.png"),

pygame.image.load("Resources\\Menus\\learnon.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.choice = "Learn"

class RankingButton(Button):

def \_\_init\_\_(self, x, y, \*group):

super(RankingButton, self).\_\_init\_\_(\*group)

self.images = [pygame.image.load("Resources\\Menus\\rankingoff.png"),

pygame.image.load("Resources\\Menus\\rankingon.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.choice = "Ranking"

class QuitButton(Button):

def \_\_init\_\_(self, x, y, \*group):

super(QuitButton, self).\_\_init\_\_(\*group)

self.images = [pygame.image.load("Resources\\Menus\\quitoff.png"),

pygame.image.load("Resources\\Menus\\quiton.png")]

self.image = self.images[self.index]

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.choice = "Quit"

#Doors.py

import pygame

from constants import \*

class UpperDoor(pygame.sprite.Sprite):

def \_\_init\_\_(self, \*group):

super(UpperDoor, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Menus\\upper.png")

self.rect = self.image.get\_rect()

self.rect.x = 0

self.rect.y = -self.rect.height - 10

self.sound = pygame.mixer.Sound("Resources\\Sounds\\doorsound.wav")

def playSound(self):

pygame.mixer.Sound.play(self.sound)

def update(self, menu):

self.rect.y += 20

if self.rect.y + self.rect.height >= menu.lowerDoor.rect.y + 20:

menu.done = True

class LowerDoor(pygame.sprite.Sprite):

def \_\_init\_\_(self, \*group):

super(LowerDoor, self).\_\_init\_\_(\*group)

self.image = pygame.image.load("Resources\\Menus\\lower.png")

self.rect = self.image.get\_rect()

self.rect.x = 0

self.rect.y = SCREEN\_HEIGHT + 10

def playSound(self):

pass

def update(self, menu):

self.rect.y -= 20

class SpecialUpperDoor(UpperDoor):

def \_\_init\_\_(self, \*group):

super(SpecialUpperDoor, self).\_\_init\_\_(\*group)

self.rect.y = 350 - self.rect.height

def update(self, menu):

self.rect.y -= 20

if self.rect.y + self.rect.height < 0 and menu.lowerDoor.rect.y > SCREEN\_HEIGHT:

menu.done = True

class SpecialLowerDoor(LowerDoor):

def \_\_init\_\_(self, \*group):

super(SpecialLowerDoor, self).\_\_init\_\_(\*group)

self.rect.y = 350

def update(self, menu):

self.rect.y += 20