Pygame, Hell Zone

13006107 Introduction to Computers and Programming

Software Engineering Program

Faculty of Engineering, KMITL

By

63011195 Natadthep Kritsanaviparkporn

Table of Contents:

Introduction to the project:

My project is a space shoot'em up game, where the main character is a spaceship, which he has to destroy as many alien ships as possible. Whilst also trying to avoid being hit by them and their laser bullets. There are 3 different types of alien ships, and the player has 3 lives in the beginning to try and survive. However the player has the chance to pick up 4 different items which will help them fight against the alien ships. These items are used to increase the number of guns, increase shooting speed, increase the shield, and give an extra life. It is an endless game, where the player will have to check if they can beat the highest score or not. If they are able to surpass that high score, they're record is kept for the new value that has occured.

Motivation:

In the past I liked to play many different types of games, to the point where I became addicted to it ever since I was a child. I loved games with alot of action and space elements, so when I had the chance to create anything for my project I chose to create a project that was based on a space shooter game against aliens.

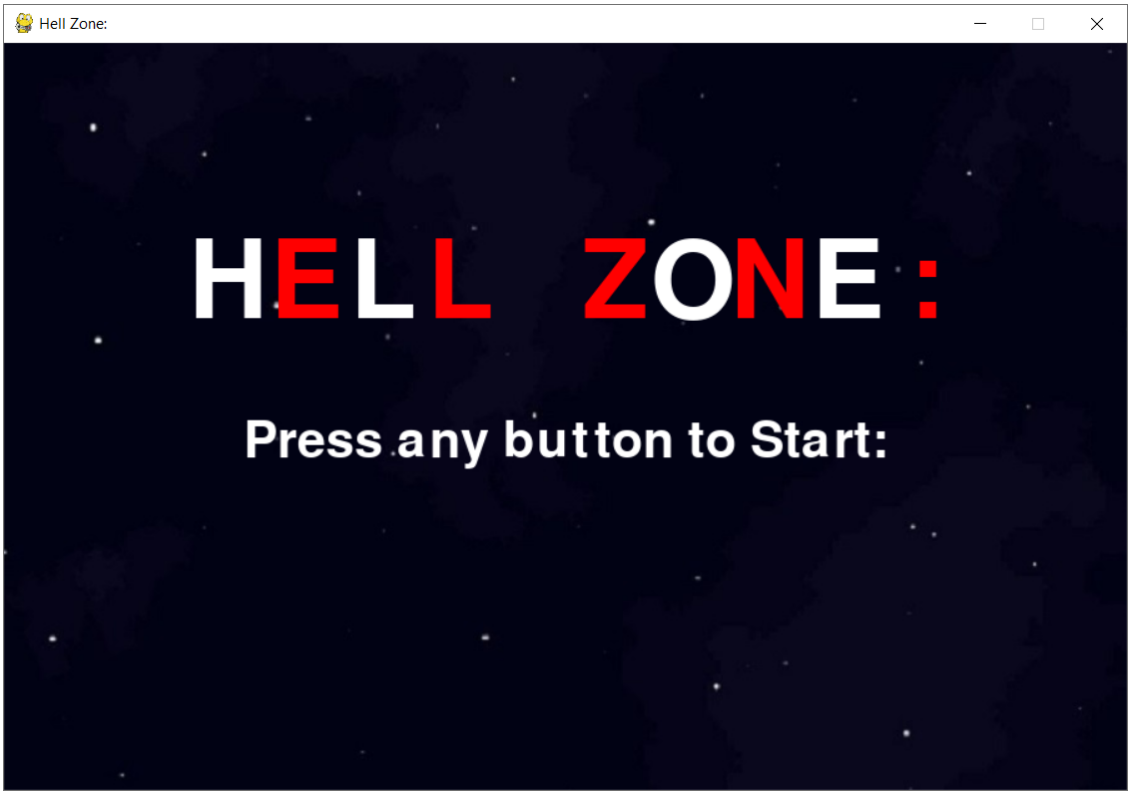
Purpose of the Program & How to Play:

My program is mainly used as a source of relaxation in times of stress, where the user can play the game whenever they want to. As for how to play the game, the user is given a set of instructions for their actions. Where they are able to move the player ship using the arrow keys, shoot laser bullets by holding down the 'x' key, and pause the game by hitting the 'p' key. The player would need to fire these bullets at the enemy with a sound effect occurring, and the bullet will decrease the player's hitbox by 1 point. Depending on the type of enemy that is on the screen, the enemy would either disappear or they would be killed and explode.

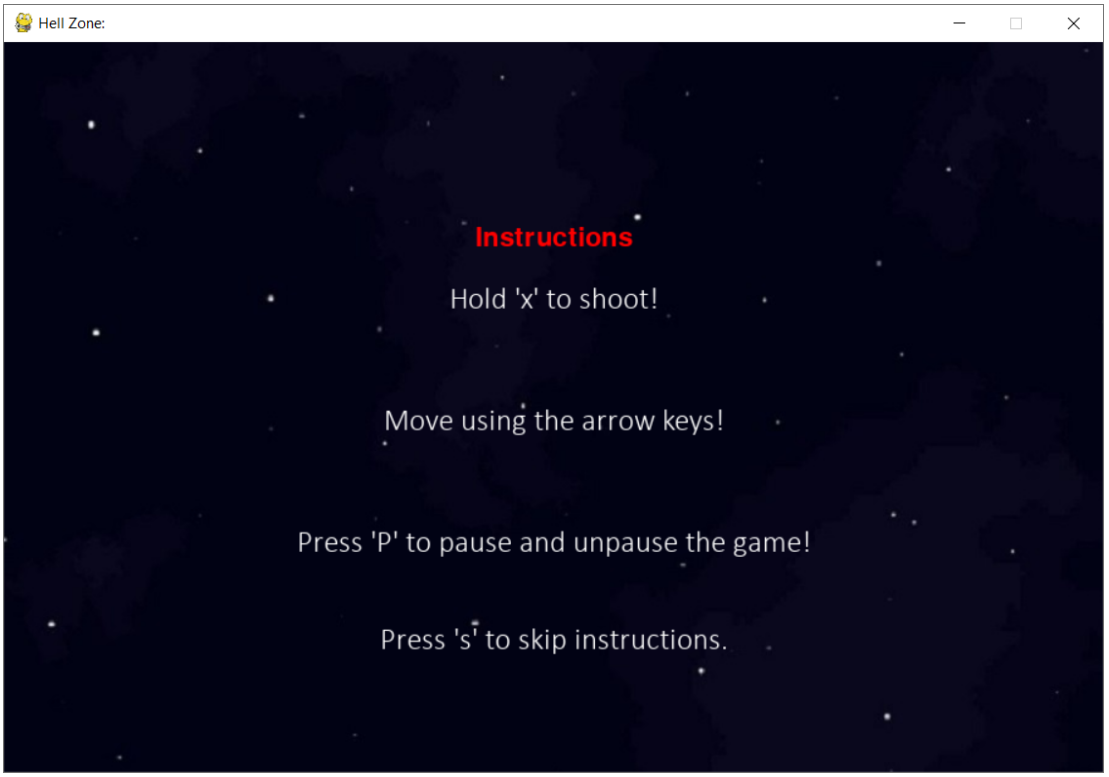
The player also has access to 4 different types of power ups they can pick up, each item has a different effect on the player based on their sprite.To begin,the first power up is one with a wrench and screwdriver crossed between them. Which when picked up by the player, fixes up the shield of the player's ship. Secondly, the next powerup is symbolized by the bullets on its sprite. Where it provides the player with increased amounts of guns to shoot up until the player has 3 guns available. The next power up is one with a lightning bolt symbol on it, when picked up it causes the bullets of the player to be fired at a faster rate. Finally the last powerup is used to increase the number of lives for the player, however the player has to pick up 3 of these powerups to allow for the player's lives to increase.

Screen Captures:

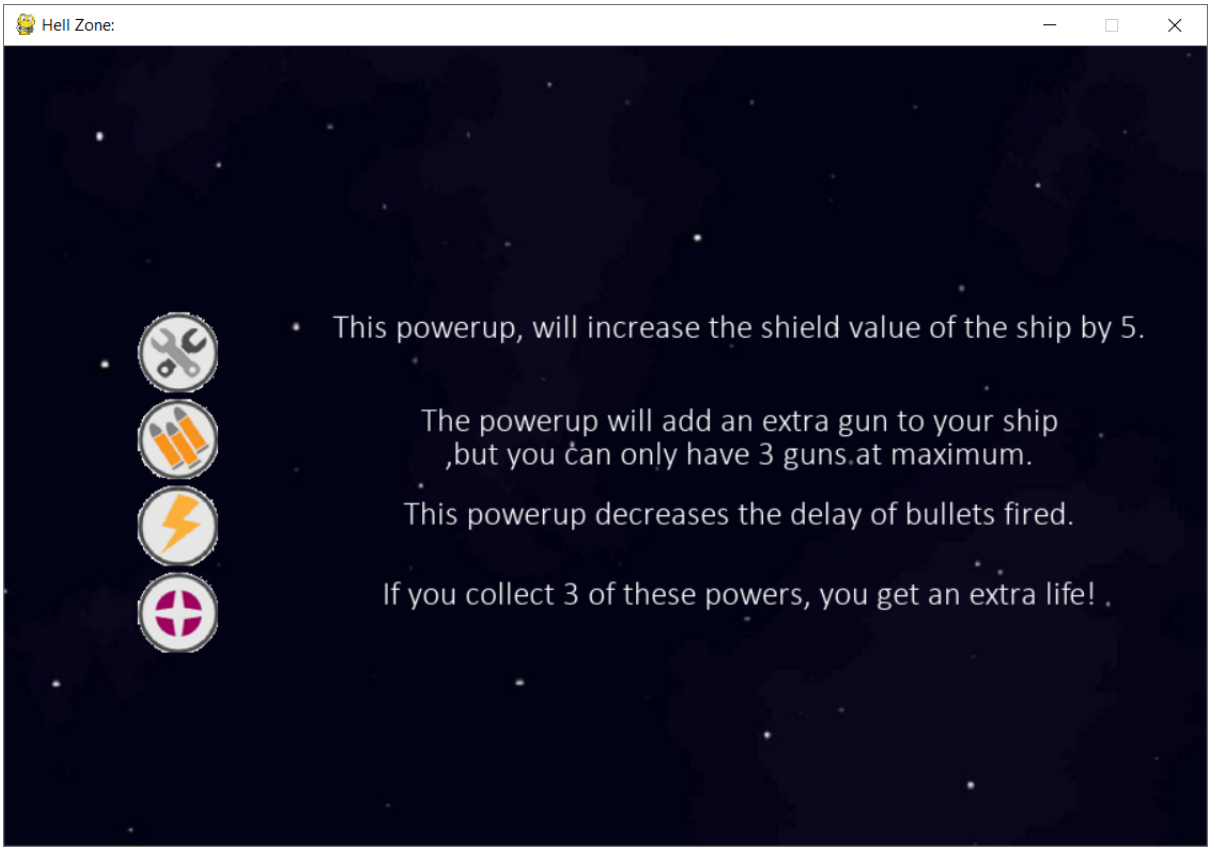
Start Screen:



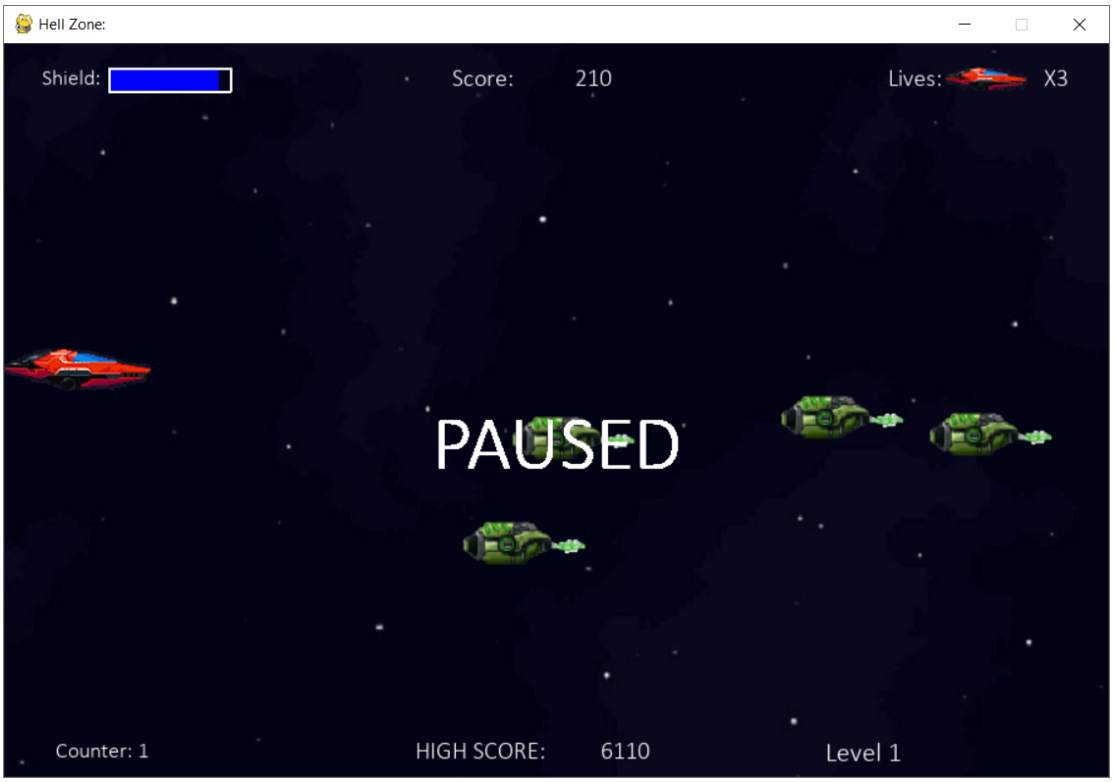
Instructions Page1:



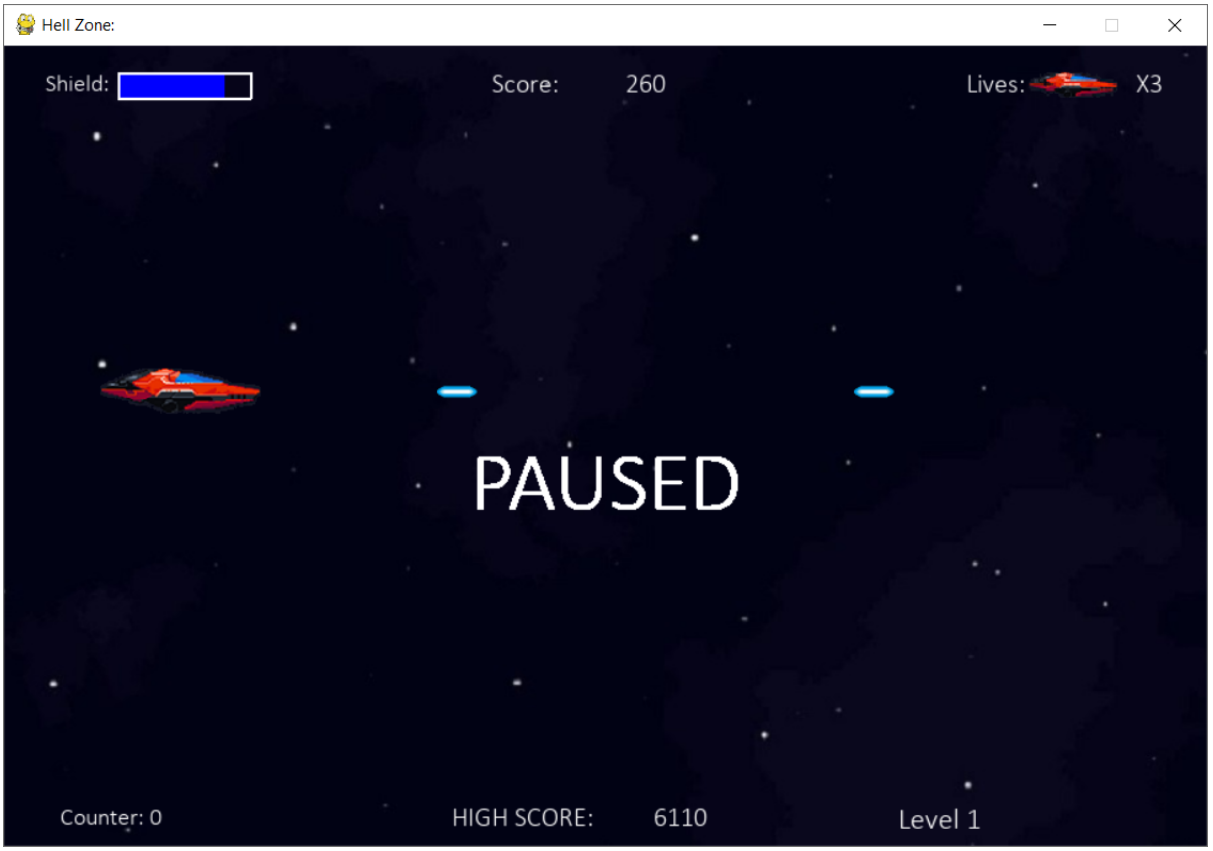
Instructions Page 2:



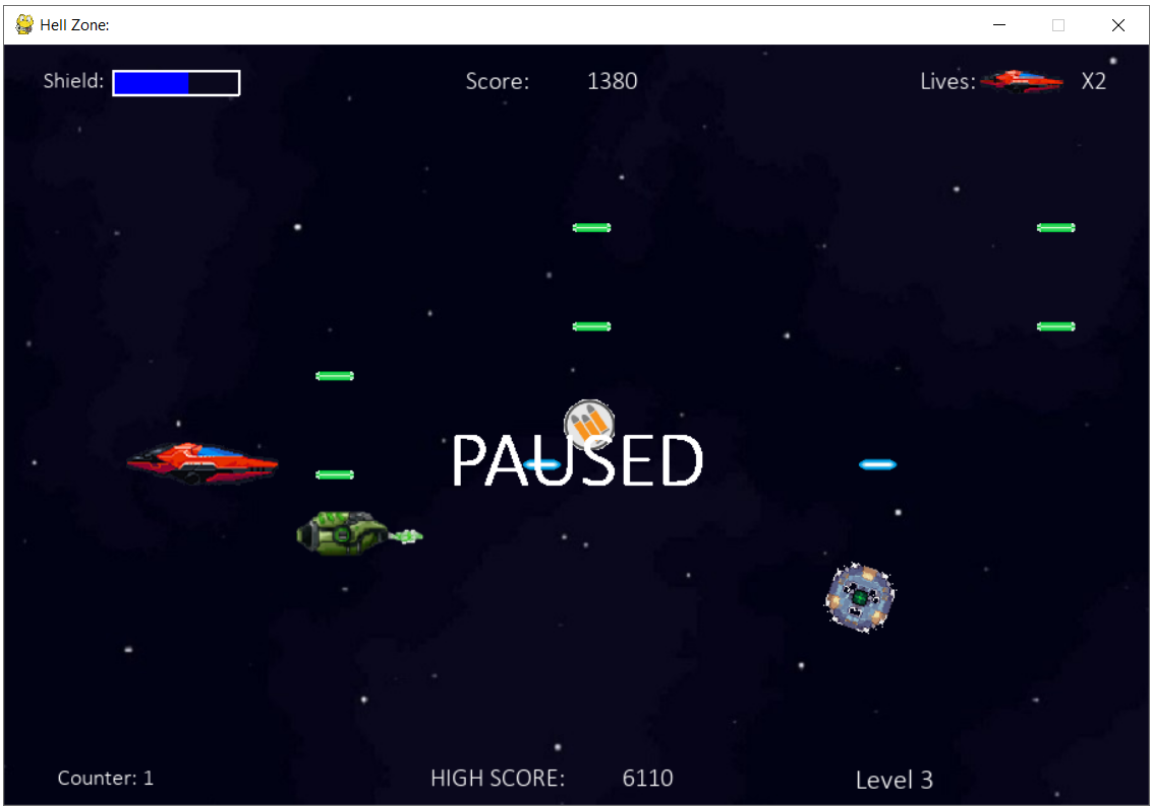
Pause Screen:



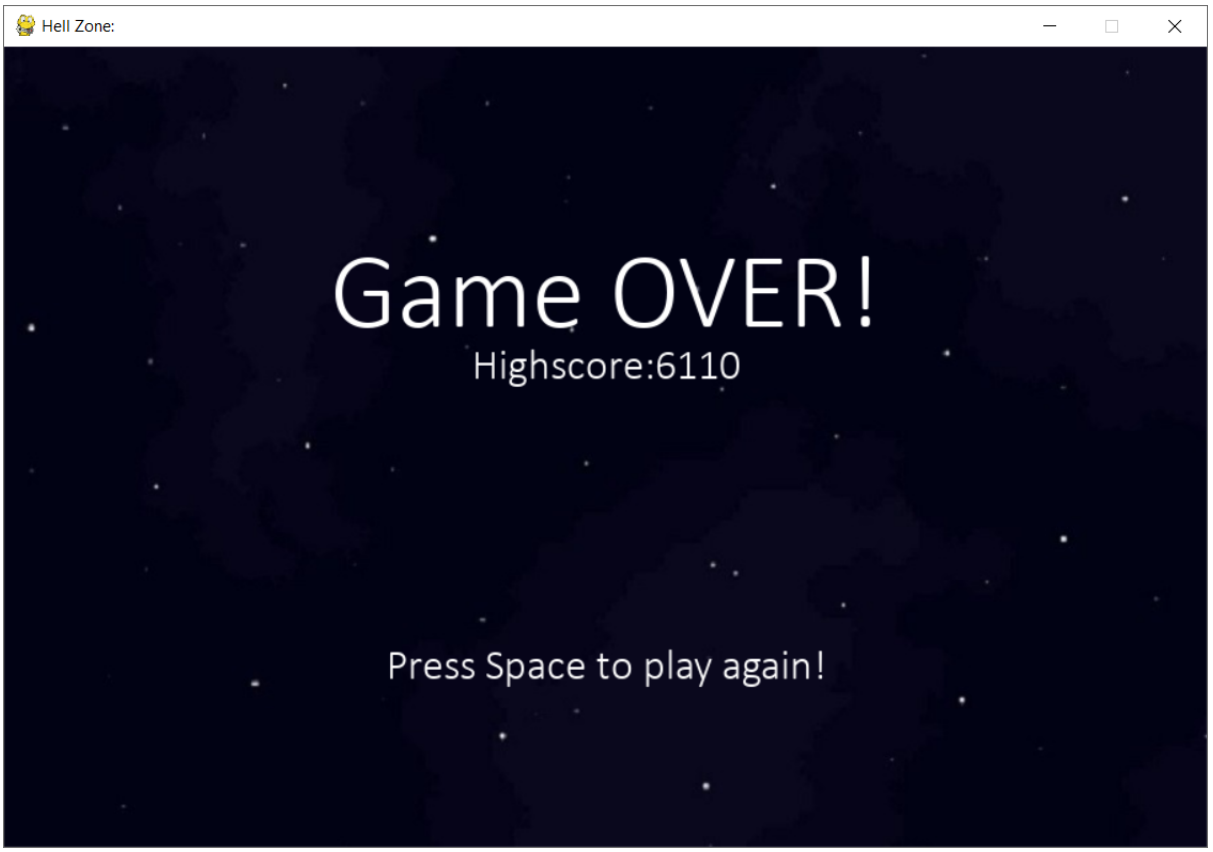
Shooting:



Powerup and Enemy Bullets:



Game Over Screen:



How to use my program:

For my program there needs to be specific graphics required for each one of the player sprites, enemies, lasers, enemy lasers, and background available for the game. As for sound effects, they are also required in a specific folder, however the sound effects also have the music inside of the same folder. The program also needs to have pygame installed inside of the software to allow for the game to be able to be played, preferably the latest version of pygame allows for the game to work.

Source Code:

```
import pygame

import random

import time

import math

from pygame.locals import (

    K_UP,

    K_DOWN,

    K_LEFT,

    K_RIGHT,

    K_ESCAPE,

    K_x,

    K_p,

    K_SPACE,

    KEYDOWN,

    KEYUP,

    QUIT,

    K_s

)



screen_width =900

screen_height = 600

pygame.init()
```

```
pygame.display.set_caption("Hell Zone:")
pygame.display.set_mode((screen_width, screen_height))
run = True

#BackGrounds
level1_background1 = pygame.image.load("level1.jpg")

Background_color = (22,24,39)
first_enemy_background = (223,223,223)

list_of_ships = ["Mainship.png","Downship.png","Upship.png"]

levels = ["level1.jpg"]

GREEN = (0, 255, 0)
RED = (255, 0, 0)
BLACK = (0,0,0)
WHITE = (255,255,255)
BLUE = (0,0,255)
clock = pygame.time.Clock()

#Group of sprites

global all_sprites,mobs,bullets,enemybullets,mobs_type3,powerups
all_sprites = pygame.sprite.Group()
mobs = pygame.sprite.Group()
bullets = pygame.sprite.Group()
enemybullets = pygame.sprite.Group()
mobs_type3 = pygame.sprite.Group()
powerups = pygame.sprite.Group()

#Music from opengameart.org
'https://opengameart.org/content/space-boss-battle-theme'
# Bossmusic creator Page: http://www.matthewpablo.com/services
ship_laser = pygame.mixer.Sound("Sounds\Ship_Laser_Shoot.wav")
```

```
explosions = []
for x in ["Sounds\EnemyExplosion.wav","Sounds\PlayerExplosion.wav"]:
    explosions.append(pygame.mixer.Sound(x))
#Explosion art by Master484 on opengameart.org
explosion_art = {}
explosion_art["Player"] = []
explosion_art["Enemy"] = []
explosion_art["Enemyshooter"] = []
explosion_art["Enemycollision"] = []
for i in range(1,8):
    file = "Explosion_player\Player{}_explosion.png".format(i)
    image = pygame.image.load(file).convert()
    image.set_colorkey((BLACK))
    image_large = pygame.transform.scale(image, (120,34))
    explosion_art["Player"].append(image_large)
    file = "Explosion_enemy\Enemy{}_explosion.png".format(i)
    image = pygame.image.load(file).convert()
    image.set_colorkey(BLACK)
    image_enemy = pygame.transform.scale(image, (50,50))
    explosion_art["Enemy"].append(image_enemy)
    image_shooter = pygame.transform.scale(image, (60,60))
    explosion_art["Enemyshooter"].append(image_shooter)
    image_collision = pygame.transform.scale(image, (20,20))
    explosion_art["Enemycollision"].append(image_collision)


musics = ["Sounds\Interplanetary Odyssey.ogg","Sounds\Vaerionii.wav"]
enemylaser = pygame.mixer.Sound("Sounds\Laser_Shoot_Enemy.wav")
powerup1 = pygame.mixer.Sound("Sounds\Powerup.wav")
powerup2 = pygame.mixer.Sound("Sounds\Powerup2.wav")
powerup3 = pygame.mixer.Sound("Sounds\Powerup3.wav")
powerup4 = pygame.mixer.Sound("Sounds\Powerup4.wav")
powerup5 = pygame.mixer.Sound("Sounds\Powerup5.wav")
levelup = pygame.mixer.Sound("Sounds\Level_up.wav")
```

```
class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.imagecheck = pygame.image.load(list_of_ships[0]).convert_alpha()
        self.image = pygame.transform.scale(self.imagecheck,(120,34))
        self.image.set_colorkey(Background_color)
        self.ship_lives = pygame.image.load("Mainship -
copy.png").convert_alpha()
        self.ship_lives.set_colorkey(Background_color)
        self.rect = self.image.get_rect()
        self.rect.centerx = 100
        self.rect.bottom = 140
        self.speed = 8
        self.bulletdelay = 310
        self.previousbullet = pygame.time.get_ticks()
        self.health = 20
        self.laserdamage = 1
        self.lives = 3
        self.hide_lives = False
        self.hide_time = pygame.time.get_ticks()
        self.extragun = 0
        self.lives_counter = 0
    def update(self):
        if self.hide_lives == True and pygame.time.get_ticks() -
self.hide_time >= 1000:
            self.hide_lives = False
            self.rect.centerx = 100
            self.rect.bottom = 140
        if self.bulletdelay <250:
            self.bulletdelay = 250
        if self.lives_counter == 3:
            powerup5.play()
            self.lives += 1
            self.lives_counter = 0
        keys = pygame.key.get_pressed()
```

```
        if keys[K_LEFT] and self.rect.left >0:
            self.rect.centerx -= self.speed
        if keys[K_RIGHT] and self.rect.right < screen_width:
            self.rect.centerx += self.speed
        self.imagecheck = pygame.image.load(list_of_ships[0]).convert_alpha()
        self.image = pygame.transform.scale(self.imagecheck,(120,34))
        self.image.set_colorkey(Background_color)
        if keys[K_UP] and self.rect.top > 0:
            self.rect.centery -= self.speed
            self.imagecheck =
pygame.image.load(list_of_ships[2]).convert_alpha()
            self.image = pygame.transform.scale(self.imagecheck,(120,34))
            self.image.set_colorkey(Background_color)
        if keys[K_DOWN] and self.rect.bottom < (screen_height):
            self.rect.centery += self.speed
            self.imagecheck =
pygame.image.load(list_of_ships[1]).convert_alpha()
            self.image = pygame.transform.scale(self.imagecheck,(120,36))
            self.image.set_colorkey(Background_color)
        if keys[K_x]:
            self.shoot()
    def shoot(self):
        now = pygame.time.get_ticks()
        if now - self.previousbullet > self.bulletdelay:
            laser =
Bullet(self.rect.right,self.rect.centery,self.laserdamage)
            all_sprites.add(laser)
            bullets.add(laser)
            ship_laser.set_volume(0.5)
            ship_laser.play()
            if self.extragun >0:
                if self.extragun > 2:
                    self.extragun = 2

self.shoot_plus(self.extragun,self.rect.right,self.rect.centery+(34*3/8))
            self.previousbullet = pygame.time.get_ticks()
```

```
    def shoot_plus(self,guns,xcor,ycor):
        if guns > 0 and guns <= 2:
            if ycor < self.rect.top:
                ycor += (34/2)
            laser = Bullet(xcor,ycor,self.laserdamage)
            all_sprites.add(laser)
            bullets.add(laser)
            if ycor > self.rect.bottom:
                self.shoot_plus(guns-1,xcor,ycor+(34 * 1/3))
            else:
                self.shoot_plus(guns-1,xcor,ycor-(34 * 2/5))
        else:
            laser = Bullet(xcor,ycor,self.laserdamage)
            all_sprites.add(laser)
            bullets.add(laser)
    def death_delay(self):
        self.hide_lives = True
        self.hide_time = pygame.time.get_ticks()
        self.rect.center = (-400,1400)



enemymovement = ["Enemy1move.png","Enemy2.png","Enemy3.png"]
class enemy(pygame.sprite.Sprite):
    def __init__(self,enemytype):
        super().__init__()
        self.enemytype = enemytype
        self.image_original =
pygame.image.load(enemymovement[self.enemytype]).convert_alpha()
        self.image_original.set_colorkey(WHITE)
        self.image = self.image_original.copy()
        self.rect = self.image.get_rect()
        self.rect.x = random.randrange(900,1200)
        self.rect.bottom = random.randrange(150,screen_height - 150)
        self.speedx = random.randrange(3,5)
        self.rot = 0
```

```
        self.bulletdelay = 900
        self.starttime = pygame.time.get_ticks()
        if self.enemytype >0 and self.enemytype <3:
            self.speedy = random.randrange(3,7)
            if self.enemytype == 1:
                self.radius = 24
            if self.enemytype == 2:
                self.speedx = random.randrange(1,3)
                self.radius = 30
            self.rot_speed = random.randrange(-8,8)
        '''else:
            self.speedy = 0
            self.radius = 0
            self.rot_speed = 0'''
        if self.enemytype == 2:
            self.hitbox = 3
        else:
            self.hitbox = 1
        self.lastupdate = pygame.time.get_ticks()
    def update(self):
        if self.enemytype == 1:
            self.rotate()
            self.rect.y = self.rect.y - self.speedy
        self.rect.x -= self.speedx
        if self.enemytype >= 2:
            self.shoot()
        if self.rect.x < -100:
            self.kill()
        if (self.rect.top <=0 and self.speedy > 0) or (self.rect.top >=
screen_height):
            self.speedy *= -1
        if self.hitbox <= 0:
            self.kill()
    def rotate(self):
        now = pygame.time.get_ticks()
```

```
        if now - self.lastupdate >60:
            self.last_update = now
            self.rot = (self.rot + self.rot_speed)%360
            newimage = pygame.transform.rotate(self.image_original,self.rot)
            newimage.set_colorkey(WHITE)
            old_center = self.rect.center
            self.image = newimage
            self.rect = self.image.get_rect()
            self.rect.center = old_center
    def shoot(self):
        now = pygame.time.get_ticks()
        if now -self.lastupdate >= self.bulletdelay:
            laserbottom =
EnemyBullets(self.rect.centerx,self.rect.centery+(self.radius *1.3))
            lasertop =
EnemyBullets(self.rect.centerx,self.rect.centery-(self.radius *1.3))
            all_sprites.add(laserbottom)
            all_sprites.add(lasertop)
            enemybullets.add(laserbottom)
            enemybullets.add(lasertop)
            enemylaser.set_volume(0.5)
            enemylaser.play()
            self.lastupdate = pygame.time.get_ticks()
    def die(self):
        self.kill()



class Bullet(pygame.sprite.Sprite):
    def __init__(self, x, y,bulletstrength):
        super().__init__()
        self.bulletstrength = bulletstrength
        self.image = pygame.image.load("bluebullet1.png").convert_alpha()
        self.image.set_colorkey((0,0,0))
        self.rect = self.image.get_rect()
        self.rect.centery = y
```

```
        self.rect.left = x
        self.speedx = 12
    def update(self):
        self.rect.x += self.speedx
        # kill if it moves off the right of the screen
        if self.rect.right > screen_width:
            self.delete()
    def delete(self):
        self.kill()

class EnemyBullets(Bullet):
    def __init__(self, x, y):
        super().__init__(x,y,bulletstrength=1)
        self.bulletstrength = 1
        self.image = pygame.image.load("enemy1bullet.png").convert_alpha()
        self.image.set_colorkey((255,255,255))
        self.rect = self.image.get_rect()
        self.rect.centery = y
        self.rect.right = x
        self.speedx = 6
    #Polymorphism: changes the direction of the bullet.
    def update(self):
        self.rect.x -= self.speedx
        # kill if it moves off the left of the screen
        if self.rect.left < 0:
            self.delete()
    def delete(self):
        self.kill()

class Explosion(pygame.sprite.Sprite):
    def __init__(self,centerx,centery,size):
        super().__init__()
        self.size = size
        self.image = explosion_art[self.size][0]
```

```
        self.rect = self.image.get_rect()
        self.rect.center = (centerx,centery)
        self.imagekey = 0
        self.last_update = pygame.time.get_ticks()
        self.delay = 40
    def update(self):
        current =  pygame.time.get_ticks()
        #Set longer time for player explosion.
        if self.size == "Player":
            self.delay = 70

        if current - self.last_update > self.delay:
            self.last_update = pygame.time.get_ticks()
            self.imagekey += 1
            if self.imagekey == len(explosion_art[self.size]):
                self.kill()
            else:
                center = self.rect.center
                self.image = explosion_art[self.size][self.imagekey]
                self.rect = self.image.get_rect()
                self.rect.center = center
powerup_images = {}
powerup_images["extra_health"] = pygame.image.load("Larger
Pictures\Health.png").convert_alpha()
powerup_images["moreguns"] = pygame.image.load("Larger
Pictures\Moreguns.png").convert_alpha()
powerup_images["gunspeed"] = pygame.image.load("Larger
Pictures\Gunspeed_increase.png").convert_alpha()
powerup_images["extralives"] = pygame.image.load("Larger
Pictures\Extra_lives.png").convert_alpha()
class PowerUp(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.type =
random.choice(["extra_health","moreguns","gunspeed","extralives"])
        self.images = powerup_images[self.type]
```

```
        self.image = pygame.transform.scale(self.images,(40,40))
        self.image.set_colorkey(WHITE)
        self.rect = self.image.get_rect()
        try:
            self.rect.centery = y
            self.rect.left = x
        except TypeError:
            self.rect.center = y
        self.speedx = 5
        self.radius = 20
    def update(self):
        self.rect.x -= self.speedx
        # kill if it moves off the right of the screen
        if self.rect.right > screen_width:
            self.delete()
    def delete(self):
        self.kill()


def draw_text(pyscreen, text, size, x, y):
    font_name = pygame.font.match_font('Calibri')
    font = pygame.font.Font(font_name, size)
    text_surface = font.render(text, True, WHITE)
    text_rect = text_surface.get_rect()
    text_rect.midtop = (x,y)
    pyscreen.blit(text_surface, text_rect)

def draw_title(pyscreen, title, size, x, y,index):
    font_title_name = pygame.font.match_font('Droid Sans')
    font = pygame.font.Font(font_title_name, size)
    if index %2 == 0:
        text_surface = font.render(title, True, WHITE)
    else:
        text_surface = font.render(title, True, RED)
```

```
    text_rect = text_surface.get_rect()
    text_rect.midtop = (x,y)
    pyscreen.blit(text_surface, text_rect)

def draw_lives(pyscreen,x,y,lives,image):
    rect = image.get_rect()
    rect.centerx = x
    rect.top = y
    pyscreen.blit(image, rect)
    lives_shown = "X" + str(lives)
    draw_text(pyscreen,"Lives:",20,x-(rect.width*7/8),y)
    draw_text(pyscreen,lives_shown,20,x+(rect.width*7/8),y)

def draw_healthbar(screen,xpos,ypos,healthvalue):
    if healthvalue <0:
        healthvalue = 0
    BarLen = 100
    BarHeight = 20
    fill = (healthvalue/20) * BarLen
    outline_rect = pygame.Rect(xpos,ypos,BarLen,BarHeight)
    fill_rect = pygame.Rect(xpos,ypos,fill,BarHeight)
    pygame.draw.rect(screen,BLUE,fill_rect)
    pygame.draw.rect(screen,WHITE,outline_rect,2)

def circle_collide_with_rect(rectcenterx,rectcentery,rectwidth,rectheight,
                             circlecenterx, circlecentery,radius):
    rectleft = rectcenterx - (rectwidth/2)
    rectright = rectleft + rectwidth
    recttopy = rectcentery - rectheight/2
    rectbottomy = recttopy + rectheight
    circleleft = circlecenterx-radius
    circletop = circlecentery-radius
    circleright = circlecenterx+radius
    circlebottom = circlecentery+radius
```

```
    if rectright< circleleft or rectleft > circleright or rectbottomy <
circletop or recttopy > circlebottom:
        return False
    for x in (rectleft, rectright):
        for y in (recttopy, rectbottomy):
            # compare distance between circle's center point and each point
of
            # the rectangle with the circle's radius
            if math.hypot(x-circlecenterx, y-circlecentery) <= radius:
                return True  # collision detected
    if rectleft <= circlecenterx <= rectright and recttopy <= circlecentery
<= rectbottomy:
        return True
    return False


def powerups_collide(player, powerups):
    for i in powerups:
        if
circle_collide_with_rect(player.rect.centerx,player.rect.centery,player.rect.
width,player.rect.height,
                                    i.rect.centerx,i.rect.centery,i.radius)==
True:
            if i.type == "extra_health":
                player.health += 5
                powerup1.play()
                if player.health >= 20:
                    player.health = 20
            if i.type == "moreguns":
                player.extragun +=1
                powerup2.play()
            if i.type == "gunspeed":
                player.bulletdelay -= 10
                powerup3.play()
            elif i.type == "extralives":
                player.lives_counter += 1
                powerup4.play()
            i.kill()
```

```
def circular_enemy_detection(player,mobs,bullet,score,maxran,difference):
    hitship = False
    for i in mobs:
        try:
            if i.radius > 0:
            # check to see if the circular enemy collides with the player
                if
circle_collide_with_rect(player.rect.centerx,player.rect.centery,player.rect.
width,player.rect.height,

i.rect.centerx,i.rect.centery,i.radius)== False:
                    for j in bullets:
                        if
circle_collide_with_rect(j.rect.centerx,j.rect.centery,j.rect.width,j.rect.he
ight,

i.rect.centerx,i.rect.centery,i.radius):
                            i.hitbox -= player.laserdamage
                            j.delete()
                            center_of_circlex = 0
                            center_of_circley = 0
                            if i.enemytype >= 2 and i.hitbox <= 0:
                                score += (i.enemytype**2 +1) *10
                                center_of_circlex = i.rect.centerx
                                center_of_circley = i.rect.centery
                            elif i.enemytype != 2:
                                score += (i.enemytype**2 +1) *10
                                center_of_circlex = i.rect.centerx
                                center_of_circley = i.rect.centery
                            explosions[0].play()
                            previoustype = i.enemytype
                            left = i.rect.left
                            ycenter = i.rect.center
                            i.update()
                            if (center_of_circlex != 0 and center_of_circley
!= 0) or (previoustype != 0):
```

```
                                if i.enemytype == 2 and i.hitbox <= 0:
                                    explosion =
Explosion(center_of_circlex,center_of_circley, "Enemyshooter")
                                    all_sprites.add(explosion)
                                    if difference > 1000:
                                        make_powerup(left,ycenter)
                                elif previoustype != 2:
                                    explosion =
Explosion(center_of_circlex,center_of_circley, "Enemyshooter")
                                    all_sprites.add(explosion)
                                    if difference > 1000:
                                        make_powerup(left,ycenter)
                            available_enemies(maxran-1)
                            hitship = False


                else:
                    explosions[1].play()
                    centerx = i.rect.centerx
                    centery = i.rect.centery
                    i.die()
                    explosion = Explosion(centerx,centery, "Enemycollision")
                    all_sprites.add(explosion)
                    hitship = True
                    decrease = 2
            return hitship
        except AttributeError:
            return False


def newenemy(maxran):
    num = random.randint(0,maxran)
    if num >= 2 and len(mobs_type3)<2:
        e = enemy(num)
        all_sprites.add(e)
        mobs_type3.add(e)
        mobs.add(e)
```

```
    elif len(mobs_type3)>=2:
        num = random.randint(0,1)
        e = enemy(num)
        all_sprites.add(e)
        mobs.add(e)
    else:
        e = enemy(num)
        all_sprites.add(e)
        mobs.add(e)

def available_enemies(maxran):
    try:
        newenemy(maxran)
    except IndexError:
        newenemy(2)

def make_powerup(left,ycenter):
    if random.random() >0.9:
        p = PowerUp(left,ycenter)
        all_sprites.add(p)
        powerups.add(p)

global centerx, centery
class game:
    def __init__(self):
        pygame.init()
        pygame.mixer.init()
        self.screen = pygame.display.set_mode([screen_width, screen_height])
        self.playerx = 30
        self.playery = 60
        self.player = Player()
        all_sprites.add(self.player)
        self.score = 0
        self.number_hits = 0
```

```
        self.background = pygame.image.load(levels[0])
        self.background_rect = self.background.get_rect()
        self.starttime = 0
        self.currenttime = 0
        self.enemies_persecond = 25
        self.musicindex = 0
        self.gameover = True
        self.pauseduration = 0
        self.difference = 0
        self.paused = False
        try:
            self.highscorefile = open("highscore.txt", "r+")
            self.high_score = self.highscorefile.read()
            self.high_score = int(self.high_score)
        except:
            self.highscorefile = open("highscore.txt", "w+")
            self.high_score = 0
        self.highscorefile.close()
        self.intro()
    def intro(self):
        pygame.mixer.music.load(musics[self.musicindex])
        pygame.mixer.music.play(loops=-1)
        running = True
        stop = False
        while running:
            self.screen.fill((0,0,0))
            self.screen.blit(self.background,self.background_rect)
            title = "HELL ZONE:"
            for i in range(len(title)):
                draw_title(self.screen, title[i], 128,
(screen_width*4/20)+i*62, screen_height* 1/4,i)
            draw_title(self.screen, "Press any button to Start:", 60,
screen_width/2, screen_height*4/8,0)
            pygame.display.update()
            pygame.display.flip()
```

```
            for event in pygame.event.get():
                if event.type == QUIT:
                    stop = True
                    running = False
                if event.type == KEYUP:
                    running = False
        if running == False and stop == False:
            self.musicindex =1
            self.starttime = pygame.time.get_ticks()
            self.instructions()
            self.instructionspower_ups()
            self.run()
        pygame.quit()
    def instructions(self):
        skip = False
        while((pygame.time.get_ticks()-self.starttime) <5000) and skip ==
False:
            self.screen.fill((0,0,0))
            self.screen.blit(self.background,self.background_rect)
            self.startime = pygame.time.get_ticks()
            draw_title(self.screen, "Instructions", 32, screen_width/2,
screen_height* 1/4,1)
            draw_text(self.screen,"Hold 'x' to shoot!", 25,  screen_width/2,
screen_height/3)
            draw_text(self.screen,"Move using the arrow keys!", 25,
screen_width/2, screen_height*1/2)
            draw_text(self.screen, "Press 'P' to pause and unpause the
game!", 25,  screen_width/2, screen_height*2/3)
            draw_text(self.screen, "Press 's' to skip instructions.", 25,
screen_width/2, screen_height*4/5)
            for event in pygame.event.get():
                if event.type == KEYDOWN:
                    if event.key == K_s:
                        skip = True
            pygame.display.update()
            pygame.display.flip()
    def instructionspower_ups(self):
```

```
        self.starttime = pygame.time.get_ticks()

        skip = False

        while((pygame.time.get_ticks()-self.starttime) <8000) and skip ==
False:

            self.screen.fill((0,0,0))

            self.screen.blit(self.background,self.background_rect)

            v = ["extra_health","moreguns","gunspeed","extralives"]

            text_choices = ["This powerup, will increase the shield value of
the ship by 5.","The powerup will add an extra gun to your ship,but you can
only have 3 guns at maximum."]

            ypos = screen_height/3

            xpos = 550

            draw_text(self.screen,"This powerup, will increase the shield
value of the ship by 5.",25,xpos,ypos)

            draw_text(self.screen,"The powerup will add an extra gun to your
ship",25,xpos,ypos+70)

            draw_text(self.screen,",but you can only have 3 guns at
maximum.",25,xpos,ypos+95)

            draw_text(self.screen,"This powerup decreases the delay of
bullets fired.",25,xpos,ypos+140)

            draw_text(self.screen,"If you collect 3 of these powers, you get
an extra life!",25,xpos,ypos+200)

            for i in range(0,len(v)):

                powerups = powerup_images[v[i]]

                images = pygame.transform.scale(powerups,(60,60))

                images.set_colorkey(WHITE)

                self.screen.blit(images,(100,ypos))

                ypos += 65

            for event in pygame.event.get():

                if event.type == KEYDOWN:

                    if event.key == K_s:

                        skip = True

            pygame.display.update()

            pygame.display.flip()


    def run(self):

        num = 0

        run = True
```

```
        maxran = 1
        self.spawnrate = 1
        self.starttime = pygame.time.get_ticks()
        hitship = False
        hitbullet = False
        began = False
        pygame.mixer.music.load(musics[self.musicindex])
        pygame.mixer.music.play(loops=-1)
        dead = False
        spawn_increase = False
        game_over = False
        self.paused = False
        while run:
            clock.tick(80)
            for event in pygame.event.get():
                if event.type == QUIT:
                    self.highscorefile = open("highscore.txt", "w+")
                    if int(self.high_score) < self.score:
                        self.high_score = str(self.score)
                    self.highscorefile.write(str(self.high_score))
                    self.highscorefile.close()
                    run = False
                if event.type == KEYDOWN:
                    if event.key == K_p:
                        self.paused = True
                        self.pause_screen(self.difference)
            # Check for any circular enemies:
            hitship =
circular_enemy_detection(self.player,mobs,bullets,self.score,maxran,self.diff
erence)
            self.currenttime = pygame.time.get_ticks()
            if game_over:
                self.gameover_screen()
                num = 0
                run = True
```

```
                maxran = 1
                self.spawnrate = 1
                self.starttime = pygame.time.get_ticks()
                hitship = False
                hitbullet = False
                self.difference = 0
                began = False
                try:
                    pygame.mixer.music.load(musics[self.musicindex])
                    pygame.mixer.music.play(loops=-1)
                    self.background = pygame.image.load(levels[0])
                    self.background_rect = self.background.get_rect()
                except:
                    pass
                dead = False
                spawn_increase = False
                self.score = 0
                self.number_hits = 0
                game_over = False
            if self.difference > 800 and began == False:
                for i in range(5):
                    available_enemies(maxran)
                began = True
            hitsbullet = pygame.sprite.groupcollide(mobs,bullets, True, True)
            #check to see if enemy hits bullet
            for hit in hitsbullet:
                hit.update()
                self.score += (hit.enemytype**2 +1) *10
                explosions[0].play()
                explosion = Explosion(hit.rect.centerx,hit.rect.centery,
"Enemy")
                all_sprites.add(explosion)
                make_powerup(hit.rect.left,hit.rect.centery)
                if (self.difference <48090 and self.difference >1200) and
(self.difference%50 == 0):
```

```
                    available_enemies(maxran-1)
                hit = False
            if self.paused:
                if self.currenttime - self.starttime > self.difference:
                    self.starttime += ((self.currenttime - self.starttime) -
self.difference)
                    self.difference = self.currenttime - self.starttime
                    self.paused = False
            else:
                self.difference = self.currenttime - self.starttime
            if self.difference >50090 and not self.paused:
                maxran += 1
                levelup.play()
                self.starttime = self.currenttime
                began = False
            #used to spawn in the enemies.
            if self.difference <48099 and self.difference >1200 and
(self.difference%80 == 0) and len(mobs) < 7:
                for i in range(0,1*self.spawnrate):
                    available_enemies(maxran-1)
            #Checks to see if the player hits a normal enemy.
            hitsplayer = pygame.sprite.spritecollide(self.player, mobs, True)
            for j in hitsplayer:
                if hitsplayer:
                    centerx = j.rect.centerx
                    centery = j.rect.centery
                    explosions[1].play()
                    hitship = True
                    decrease = 2
            #Checks to see if the player hits a bullet.
            bullethitsplayer = pygame.sprite.spritecollide(self.player,
enemybullets, True)
            for j in bullethitsplayer:
                if bullethitsplayer:
                    centerx = j.rect.centerx
                    centery = j.rect.centery
```

```
                    decrease = j.bulletstrength
                    explosions[1].play()
                    hitship = True
            #Checks to see if powerups hits:
            powerups_collide(self.player, powerups)
            if hitship:
                explosion = Explosion(centerx,centery, "Enemycollision")
                all_sprites.add(explosion)
                self.player.health -= decrease
                hitship = False
            if self.player.health <= 0:
                #use to make the explosions.
                if dead == False:
                    death_explosion =
Explosion(self.player.rect.centerx,self.player.rect.centery, "Player")
                    all_sprites.add(death_explosion)
                    self.player.death_delay()
                    self.player.lives -= 1
                    self.player.extragun = 0
                    self.player.health = 20
                    self.player.bulletdellay = 310
                if self.player.lives <= 0:
                    dead = True
            if self.player.lives <= 0 and not death_explosion.alive():
                game_over = True
            if ((maxran+1)%5 == 0):
                spawn_increase = False
            if (maxran %5) == 0 and spawn_increase == False:
                self.spawnrate +=1
                if self.spawnrate >3:
                    self.spawnrate = 3
                spawn_increase = True
            all_sprites.update()
            #Create the moving background
            try:
```

```
                self.screen.fill((0,0,0))
                self.screen.blit(self.background,self.background_rect)
            except:
                pass
            if self.background_rect.right<=900:
                self.background_rect.left = 0
            else:
                self.background_rect.right -= 2
            hitship = False
            hitbullet = False
            all_sprites.draw(self.screen)
            value = str(self.score)
            display = "Score:" + "{:>13}".format(value)
            if self.score > int(self.high_score):
                self.high_score = value
            highscore_display = "HIGH SCORE:" +
"{:>13}".format(self.high_score)
            # Show score:
            try:
                draw_text(self.screen, display,20, 430,20)
                draw_text(self.screen, highscore_display,20, 430,570)
                #shows Health:
                draw_text(self.screen,"Shield:",18,55,20)
                draw_healthbar(self.screen,85,20,self.player.health)
                #Shows Level:
                Level_text = "Level {}".format(maxran)
                draw_text(self.screen, Level_text,22, 700,570)
                Lives_counter = "Counter:
{}".format(self.player.lives_counter)
                draw_text(self.screen, Lives_counter, 18, 80,570)

draw_lives(self.screen,800,20,self.player.lives,self.player.ship_lives)
                pygame.display.update()
                pygame.display.flip()
            except:
                run = False
```

```
        pygame.quit()
    def gameover_screen(self):
        for i in all_sprites:
            i.kill()
        pygame.mixer.music.load(musics[0])
        pygame.mixer.music.play(loops=-1)
        waiting = True
        continue1 = False
        while waiting and (not continue1):
            clock.tick(80)
            self.screen.fill((0,0,0))
            self.screen.blit(self.background,self.background_rect)
            draw_text(self.screen, "Game OVER!", 80, screen_width/2,
screen_height/4)
            draw_text(self.screen, "Press Space to play again!", 32,
screen_width/2, screen_height*3/4)
            draw_text(self.screen, "Highscore:{}".format(self.high_score),
32, screen_width/2, screen_height*3/8)
            for event in pygame.event.get():
                if event.type == QUIT:
                    self.highscorefile = open("highscore.txt", "w+")
                    if int(self.high_score) < self.score:
                        self.high_score = str(self.score)
                    self.highscorefile.write(str(self.high_score))
                    self.highscorefile.close()
                    waiting = False
                if event.type == KEYDOWN:
                    if event.key == K_SPACE:
                        continue1 = True
            try:
                pygame.display.update()
                pygame.display.flip()
            except:
                pass
        if waiting == False:
            pygame.quit()
```

```
        if continue1 == True:
            self.player = Player()
            all_sprites.add(self.player)
    def pause_screen(self,difference):
        run = True
        while run:
            try:
                for event in pygame.event.get():
                    if event.type == KEYDOWN:
                        if event.key == K_p:
                            run = False
                    if event.type == QUIT:
                        self.highscorefile = open("highscore.txt", "w+")
                        if int(self.high_score) < self.score:
                            self.high_score = str(self.score)
                        self.highscorefile.write(str(self.high_score))
                        self.highscorefile.close()
                        pygame.quit()
                    else:
                        pass
                draw_text(self.screen,"PAUSED",64,
screen_width/2,screen_height/2)
                pygame.display.update()
            except:
                pass
Game = game()
```