Last Line of Defense

# Group members

Joppe Janssens: r0902342

# Description

Last Line of Defense is a simple sidescroller arcade space shooter based on the classic Space Invaders. Aliens are invading our solar system have destroyed many of Earth’s defenses. You are the last bastion of Humanity. Dodge, upgrade, survive and fight to repel the aliens at all costs!

# Code

I’ve found Tech With Tim on youtube to give very clear explanations on pygame and thoroughly followed his guide to get a core idea of how to get things working. After that I expanded upon the game on my own but I did not copy anything directly from the web. I also made the soundtrack myself.

References:

Assets: <https://itch.io/>

<https://www.geeksforgeeks.org/building-space-invaders-using-pygame-python/>

[Pygame Tutorial - Creating Space Invaders](https://www.youtube.com/watch?v=Q-__8Xw9KTM)

[](https://www.youtube.com/watch?v=Q-__8Xw9KTM)

import pygame

import time

import os

import random

from pygame import mixer

pygame.init()

pygame.font.init()

# Window initialisation

window\_WIDTH, window\_HEIGHT = 1024, 1024

WINDOW = pygame.display.set\_mode((window\_WIDTH, window\_HEIGHT))

pygame.display.set\_caption("Last Line of Defense")

# Load player ship asset image

PLAYER\_SHIP = pygame.image.load(

os.path.join("assets/player", "shipFullHealth.png"))

PLAYER\_SHIP\_SLIGHTDAMAGE = pygame.image.load(

os.path.join("assets/player", "shipSlightDamage.png"))

PLAYER\_SHIP\_DAMAGED = pygame.image.load(

os.path.join("assets/player", "shipDamaged.png"))

PLAYER\_SHIP\_VERYDAMAGED = pygame.image.load(

os.path.join("assets/player", "shipVeryDamaged.png"))

ENGINE\_SHIP = pygame.image.load(os.path.join("assets/player", "shipEngine.png"))

# Set game icon

pygame.display.set\_icon(PLAYER\_SHIP)

# Load and scale background images for levels and main menu

BACKGROUNDMAINMENU = pygame.transform.scale(pygame.image.load(os.path.join(

"assets", "MainMenuBackground.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL1 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_01-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL2 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_02-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL3 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_03-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL4 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_04-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL5 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_05-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL6 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_06-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL7 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_07-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL8 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Blue Nebula", "Blue\_Nebula\_08-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL9 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_01-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL10 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_02-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL11 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_03-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL12 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_04-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL13 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_05-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL14 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_06-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL15 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_07-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL16 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Green Nebula", "Green\_Nebula\_08-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL17 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_01-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL18 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_02-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL19 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_03-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL20 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_04-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL21 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_05-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL22 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_06-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL23 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_07-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

BACKGROUNDL24 = pygame.transform.scale(pygame.image.load(os.path.join(

"assets/Purple Nebula", "Purple\_Nebula\_08-1024x1024.png")), (window\_WIDTH, window\_HEIGHT))

# Background Sound

mixer.music.load(os.path.join("audio", "background.wav"))

mixer.music.play(-1)

enemyKilledSound = mixer.Sound(os.path.join(

"audio", "enemyKilled.wav"))

bulletSound = mixer.Sound(os.path.join(

"audio", "bullet.wav"))

lifeLostSound = mixer.Sound(os.path.join(

"audio", "lifeLost.wav"))

gameOverSound = mixer.Sound(os.path.join(

"audio", "gameOver.wav"))

powerUpSound = mixer.Sound(os.path.join(

"audio", "powerUp.wav"))

winSound = mixer.Sound(os.path.join(

"audio", "win.wav"))

# Load alien ship asset images

ALIEN\_GREEN1 = pygame.image.load(os.path.join(

"assets/aliens", "aliengreen1.png"))

ALIEN\_RED1 = pygame.image.load(os.path.join(

"assets/aliens", "alienred1.png"))

ALIEN\_RED2 = pygame.image.load(os.path.join(

"assets/aliens", "alienred2.png"))

ALIEN\_CYAN1 = pygame.image.load(os.path.join(

"assets/aliens", "aliencyan1.png"))

ALIEN\_YELLOW1 = pygame.image.load(os.path.join(

"assets/aliens", "alienyellow1.png"))

ALIEN\_BOSS = pygame.image.load(os.path.join(

"assets/aliens", "alienBoss.png"))

ALIEN\_BLUE1 = pygame.image.load(os.path.join(

"assets/aliens", "alienblue1.png"))

ALIEN\_PURPLE1 = pygame.image.load(os.path.join(

"assets/aliens", "alienpurple1.png"))

ALIEN\_PURPLE2 = pygame.image.load(os.path.join(

"assets/aliens", "alienpurple2.png"))

# Load laser asset images

GREEN\_LASER = pygame.image.load(

os.path.join("assets/lasers", "greenlaser.png"))

RED\_LASER = pygame.image.load(os.path.join("assets/lasers", "redlaser.png"))

BLUE\_LASER = pygame.image.load(os.path.join("assets/lasers", "bluelaser.png"))

YELLOW\_LASER = pygame.image.load(os.path.join("assets/lasers", "yellowlaser.png"))

PURPLE\_LASER = pygame.image.load(os.path.join("assets/lasers", "purplelaser.png"))

PURPLE\_BULLET = pygame.image.load(os.path.join("assets/lasers", "alienbullet.png"))

PLAYER\_LASER = pygame.image.load(

os.path.join("assets/lasers", "bullet.png"))

# Power ups images

HEALTH\_POWER = pygame.image.load(os.path.join("assets/powerups", "healthPower.png"))

DAMAGE\_POWER = pygame.image.load(os.path.join("assets/powerups", "damagePower.png"))

COOLDOWN\_POWER = pygame.image.load(os.path.join("assets/powerups", "cooldownPower.png"))

# Super class ship for both player and enemy

class Ship:

COOLDOWN = 30

def \_\_init\_\_(self, x, y, health):

self.x = x

self.y = y

self.health = health

self.shipImage = None

self.laserImage = None

self.engineImage = None

self.laserCooldown = 0

self.lasers = []

self.lives = 3

self.hit = 0

# Draw player and enemy ships

def draw(self, window):

window.blit(self.shipImage, (self.x, self.y))

for laser in self.lasers:

laser.draw(WINDOW)

def shoot(self):

if self.laserCooldown == 0:

laser = Laser(self.x, self.y, self.laserImage)

self.lasers.append(laser)

self.laserCooldown = 1

def cooldown(self):

if self.laserCooldown >= self.COOLDOWN:

self.laserCooldown = 0

elif self.laserCooldown > 0:

self.laserCooldown += 1

# Checks if player is hit by laser and moves the laser

def move\_lasers(self, velocity, object):

self.cooldown()

for laser in self.lasers:

laser.move(velocity)

if laser.off\_screen(window\_HEIGHT):

self.lasers.remove(laser)

elif laser.collision(object):

object.health -= 25

if(object.health < 100 and object.health >= 75):

object.shipImage = PLAYER\_SHIP\_SLIGHTDAMAGE

elif(object.health < 75 and object.health >= 50):

object.shipImage = PLAYER\_SHIP\_DAMAGED

elif(object.health < 50 and object.health > 0):

object.shipImage = PLAYER\_SHIP\_VERYDAMAGED

else:

object.lives -= 1

lifeLostSound.play()

object.health = 100

object.shipImage = PLAYER\_SHIP

# To hit player only once with the same laser

self.lasers.remove(laser)

def getWidth(self):

return self.shipImage.get\_width()

def getHeight(self):

return self.shipImage.get\_height()

# Player ship

class Player(Ship):

DEPLETIOND = 25

DEPLETIONC = 50

def \_\_init\_\_(self, x, y, health=100):

super().\_\_init\_\_(x, y, health)

self.shipImage = PLAYER\_SHIP

self.laserImage = PLAYER\_LASER

self.engineImage = ENGINE\_SHIP

# Mask for collision detection

self.mask = pygame.mask.from\_surface(self.shipImage)

self.kills = 0

self.score = 0

self.doubleLaser = False

self.halfCooldown = False

self.untilNextLevel = 4

self.idleEngineSprites = []

self.moveEngineSprites = []

self.maxHealth = health

self.is\_moving = False

self.idleEngineSprites.append(pygame.image.load(os.path.join("assets/player", "idle1.png")))

self.idleEngineSprites.append(pygame.image.load(os.path.join("assets/player", "idle2.png")))

self.idleEngineSprites.append(pygame.image.load(os.path.join("assets/player", "idle3.png")))

self.moveEngineSprites.append(pygame.image.load(os.path.join("assets/player", "moving1.png")))

self.moveEngineSprites.append(pygame.image.load(os.path.join("assets/player", "moving2.png")))

self.moveEngineSprites.append(pygame.image.load(os.path.join("assets/player", "moving3.png")))

self.currentengineSprite = 0

self.engineTrailSprite = self.idleEngineSprites[self.currentengineSprite]

def move\_lasers(self, velocity, objects):

self.cooldown()

for laser in self.lasers:

laser.move(velocity)

if laser.off\_screen(window\_HEIGHT):

self.lasers.remove(laser)

else:

for object in objects:

# Remove enemy ship and laser from game if it is hit by the player

if laser.collision(object):

object.health -= 50

object.hit+=1

if(object.health <= 0):

self.kills += 1

enemyKilledSound.play()

self.untilNextLevel -= 1

if(object.maxHealth == 100):

self.score += object.maxHealth

elif(object.maxHealth == 500):

self.score += object.maxHealth

else:

self.score += object.maxHealth

objects.remove(object)

if laser in self.lasers:

self.lasers.remove(laser)

def draw(self, windowBar):

super().draw(windowBar)

windowBar.blit(self.engineImage, (self.x+26, self.y+50))

windowBar.blit(self.engineTrailSprite, (self.x+25, self.y+53))

self.update()

self.healthbar(windowBar)

def moving(self):

self.is\_moving = True

self.engineTrailSprite = self.moveEngineSprites[self.currentengineSprite]

def idle(self):

self.is\_moving = False

self.engineTrailSprite = self.idleEngineSprites[self.currentengineSprite]

def healthbar(self, windowBar):

# Red part of health bar(missing health)

pygame.draw.rect(windowBar, (255, 0, 0), (self.x, self.y +

self.shipImage.get\_height() + 10, self.shipImage.get\_width(), 10))

# Green part of healthbar

pygame.draw.rect(windowBar, (0, 255, 0), (self.x, self.y + self.shipImage.get\_height() +

10, self.shipImage.get\_width() \* (self.health/self.maxHealth), 10))

def update(self):

if self.is\_moving == True:

self.currentengineSprite += 1

if self.currentengineSprite >= len(self.moveEngineSprites):

self.currentengineSprite = 0

self.engineTrailSprite = self.moveEngineSprites[self.currentengineSprite]

elif self.is\_moving == False:

self.currentengineSprite += 1

if self.currentengineSprite >= len(self.idleEngineSprites):

self.currentengineSprite = 0

self.engineTrailSprite = self.idleEngineSprites[self.currentengineSprite]

def shoot(self):

if self.doubleLaser == False:

if self.laserCooldown == 0:

laser = Laser(self.x+25, self.y, self.laserImage)

self.lasers.append(laser)

self.laserCooldown = 1

bulletSound.play()

if(self.halfCooldown == True):

self.COOLDOWN = 10

self.DEPLETIONC-=1

if(self.DEPLETIONC<=0):

self.halfCooldown = False

self.COOLDOWN = 30

self.DEPLETIONC = 50

else:

if self.laserCooldown == 0:

laser1 = Laser(self.x+50, self.y, self.laserImage)

laser2 = Laser(self.x, self.y, self.laserImage)

self.lasers.append(laser1)

self.lasers.append(laser2)

self.laserCooldown = 1

bulletSound.play()

bulletSound.play()

self.DEPLETIOND -= 1

if(self.DEPLETIOND <= 0):

self.doubleLaser = False

self.DEPLETIOND = 25

if(self.halfCooldown == True):

self.COOLDOWN = 10

self.DEPLETIONC-=1

if(self.DEPLETIONC<=0):

self.halfCooldown = False

self.COOLDOWN = 30

self.DEPLETIONC = 50

# Enemy ships

class Enemy(Ship):

shipColor\_Map = {

"red1": (ALIEN\_RED1, RED\_LASER),

"red2": (ALIEN\_RED2, RED\_LASER),

"green1": (ALIEN\_GREEN1, GREEN\_LASER),

"blue1": (ALIEN\_BLUE1, BLUE\_LASER),

"cyan1": (ALIEN\_CYAN1, BLUE\_LASER),

"yellow1": (ALIEN\_YELLOW1, YELLOW\_LASER),

"purple1": (ALIEN\_PURPLE1, PURPLE\_BULLET),

"purple2": (ALIEN\_PURPLE2, PURPLE\_LASER),

"boss": (ALIEN\_BOSS, RED\_LASER)

}

def \_\_init\_\_(self, x, y, color, health=50):

super().\_\_init\_\_(x, y, health)

self.shipImage, self.laserImage = self.shipColor\_Map[color]

# Mask for collision detection

self.mask = pygame.mask.from\_surface(self.shipImage)

self.maxHealth = health

def draw(self, windowBar):

super().draw(windowBar)

if(self.hit > 0 ):

self.healthbar(windowBar)

def moveDown(self, velocity):

self.y += velocity

def shoot(self):

if self.laserCooldown == 0:

laser = Laser(self.x+23, self.y+50, self.laserImage)

self.lasers.append(laser)

self.laserCooldown = 1

def healthbar(self, windowBar):

# Red part of health bar(missing health)

pygame.draw.rect(windowBar, (255, 0, 0), (self.x, self.y +

self.shipImage.get\_height() + 10, self.shipImage.get\_width(), 10))

# Green part of healthbar

pygame.draw.rect(windowBar, (0, 255, 0), (self.x, self.y + self.shipImage.get\_height() +

10, self.shipImage.get\_width() \* (self.health/self.maxHealth), 10))

# Laser projectiles

class Laser:

def \_\_init\_\_(self, x, y, laserImage):

self.x = x

self.y = y

self.laserImage = laserImage

self.mask = pygame.mask.from\_surface(self.laserImage)

def draw(self, window):

window.blit(self.laserImage, (self.x, self.y))

def move(self, velocity):

self.y += velocity

def off\_screen(self, height):

return not (self.y <= height and self.y >= 0)

def collision(self, obj):

return collide(self, obj)

# Power ups

class PowerUp:

powerUpType = {

"heal": HEALTH\_POWER,

"damage": DAMAGE\_POWER,

"cooldown": COOLDOWN\_POWER

}

def \_\_init\_\_(self, x, y, type):

self.x = x

self.y = y

self.type = type

self.powerImage = self.powerUpType[type]

self.mask = pygame.mask.from\_surface(self.powerImage)

def draw(self, window):

window.blit(self.powerImage, (self.x, self.y))

def moveDown(self, velocity):

self.y += velocity

def off\_screen(self, height):

return not (self.y <= height and self.y >= 0)

def collision(self, player):

return collide(self, player) != None

def activate(self, player):

if(self.collision(player) != None):

powerUpSound.play()

if(self.powerUpType[self.type] == HEALTH\_POWER):

player.health = player.maxHealth

player.shipImage = PLAYER\_SHIP

elif(self.powerUpType[self.type] == DAMAGE\_POWER):

player.doubleLaser = True

else:

player.halfCooldown = True

def getHeight(self):

return self.powerImage.get\_height()

def collide(object1, object2):

# calculate offset to know distance between objects

offset\_x = object2.x - object1.x

offset\_y = object2.y - object1.y

# if the masks of both objects overlap

return object1.mask.overlap(object2.mask, (offset\_x, offset\_y)) != None

def main():

run = True

lost = False

lostTime = 0

winTime = 0

win = False

gameOverFont = pygame.font.Font(

os.path.join("fonts", "Reboot Crush.ttf"), 15)

untilNextLevelFont = pygame.font.Font(

os.path.join("fonts", "Reboot Crush.ttf"), 10)

framesPerSecond = 60

level = 0

player\_velocity = 5

enemy\_velocity = 0.8

laser\_velocity\_enemy = 2

laser\_velocity\_player = -5

player = Player(450, 700)

enemies = []

powers = []

enemyPerWave = 4

levels = {

"1": BACKGROUNDL1,

"2": BACKGROUNDL2,

"3": BACKGROUNDL3,

"4": BACKGROUNDL4,

"5": BACKGROUNDL5,

"6": BACKGROUNDL6,

"7": BACKGROUNDL7,

"8": BACKGROUNDL8,

"9": BACKGROUNDL9,

"10": BACKGROUNDL10,

"11": BACKGROUNDL11,

"12": BACKGROUNDL12,

"13": BACKGROUNDL13,

"14": BACKGROUNDL14,

"15": BACKGROUNDL15,

"16": BACKGROUNDL16,

"17": BACKGROUNDL17,

"18": BACKGROUNDL18,

"19": BACKGROUNDL19,

"20": BACKGROUNDL20,

"21": BACKGROUNDL21,

"22": BACKGROUNDL22,

"23": BACKGROUNDL23,

"24": BACKGROUNDL24

}

currentLevelBackground = BACKGROUNDMAINMENU

clock = pygame.time.Clock()

# Checks for changes

def redraw\_window():

WINDOW.blit(currentLevelBackground, (0,0))

# Render text

lives\_text = gameOverFont.render(f"Lives: {player.lives}", 1, (0, 255, 0))

level\_text = gameOverFont.render(f"Level: {level}", 1, (255, 255, 255))

kills\_text = gameOverFont.render(

f"Total kills: {player.kills}", 1, 'orange')

untilNextLevel\_text = untilNextLevelFont.render(

f"Until next level: {player.untilNextLevel}", 1, 'red')

WINDOW.blit(lives\_text, (window\_WIDTH -

lives\_text.get\_width() - 20, 920))

WINDOW.blit(level\_text, (window\_WIDTH -

level\_text.get\_width() - 20, 950))

WINDOW.blit(kills\_text, (740, 855))

WINDOW.blit(untilNextLevel\_text, (775, 890))

score\_text = gameOverFont.render(f"Score: {player.score}", 1, (255, 255, 255))

for enemy in enemies:

enemy.draw(WINDOW)

for power in powers:

power.draw(WINDOW)

player.draw(WINDOW)

if lost:

game\_over\_text = gameOverFont.render("GAME OVER", 1, (255, 0, 0))

WINDOW.blit(game\_over\_text, (window\_WIDTH/2 -

game\_over\_text.get\_width()/2, 350))

WINDOW.blit(score\_text, (window\_WIDTH/2 -

score\_text.get\_width()/2, 400))

if win:

win\_text = gameOverFont.render("YOU WIN!", 1, (0, 255, 0))

WINDOW.blit(win\_text, (window\_WIDTH/2 -

win\_text.get\_width()/2, 350))

WINDOW.blit(score\_text, (window\_WIDTH/2 -

score\_text.get\_width()/2, 400))

pygame.display.update()

# Provides consistent frames on devices

while run:

clock.tick(framesPerSecond)

redraw\_window()

# Game over

if player.lives <= 0:

lost = True

lostTime += 1

gameOverSound.play()

# Game over message time

if lost:

if lostTime > framesPerSecond \* 5:

run = False

else:

continue

if win:

winTime+=1

if winTime > framesPerSecond \* 3:

run = False

else:

if(event.type == pygame.MOUSEBUTTONDOWN):

main\_menu()

# Start next wave of enemies after all have been killed

if len(enemies) == 0:

if(level==24):

win = True

winTime += 1

winSound.play()

else:

level += 1

if(level%5==0):

player.lives += 1

enemy\_velocity \*=1.02

laser\_velocity\_enemy \*=1.01

player\_velocity \*=1.02

laser\_velocity\_player \*=1.02

player.COOLDOWN\*= 0.98

for item in levels.keys():

if(level == int(item)):

currentLevelBackground = levels[item]

enemyPerWave += 2

for i in range(int(level/3)):

power = PowerUp(random.randrange(50, window\_WIDTH-100),

random.randrange(int((level\*-200)-1000), -100), random.choice(["heal", "damage", "cooldown"]))

powers.append(power)

player.untilNextLevel = enemyPerWave

levelUpSound = mixer.Sound(os.path.join(

"audio", "levelUp.wav"))

levelUpSound.play()

# Spawn enemies

if(level < 3):

for i in range(enemyPerWave):

enemytype = random.choice(["red1", "yellow1", "green1"])

enemy = Enemy(random.randrange(50, window\_WIDTH-100),

random.randrange(-1000, -100), enemytype)

enemies.append(enemy)

elif(level >= 3 and level <5):

for i in range(enemyPerWave):

enemytype = random.choice(["red1", "yellow1", "blue1", "green1"])

enemy = Enemy(random.randrange(50, window\_WIDTH-100),

random.randrange(-1200, -100), enemytype)

enemies.append(enemy)

elif(level >= 5 and level <8):

for i in range(enemyPerWave):

enemytype = random.choice(["red1", "cyan1", "yellow1", "purple1" , "blue1", "green1"])

enemy = Enemy(random.randrange(50, window\_WIDTH-100),

random.randrange(-1400, -100), enemytype)

enemies.append(enemy)

elif(level >= 8 and level <10):

for i in range(enemyPerWave):

enemytype = random.choice(["red1", "red2", "yellow1", "purple1", "purple2" , "blue1", "green1"])

enemy = Enemy(random.randrange(50, window\_WIDTH-100),

random.randrange(-1600, -100), enemytype)

if(enemytype == "purple2" or enemytype == "red2"):

enemy.health = 100

enemy.maxHealth = 100

enemies.append(enemy)

elif(level >= 10):

boss = Enemy(random.randrange(50, window\_WIDTH-100),

random.randrange(-1800, -100), "boss")

boss.health = 500

boss.maxHealth = 500

enemies.append(boss)

for i in range(enemyPerWave-1):

enemytype = random.choice(["red1", "red2", "yellow1", "purple1", "purple2" , "blue1", "green1"])

enemy = Enemy(random.randrange(50, window\_WIDTH-100),

random.randrange(enemyPerWave\*-85, -100), enemytype)

if(enemytype == "purple2" or enemytype == "red2"):

enemy.health = 100

enemy.maxHealth = 100

enemies.append(enemy)

# Check for input

for event in pygame.event.get():

# Turn game off

if event.type == pygame.QUIT:

run = False

# Creates dictionary of pressed keys

keys = pygame.key.get\_pressed()

if keys[pygame.K\_a] and player.x - player\_velocity > 0: # Move left

player.x -= player\_velocity

player.idle()

if keys[pygame.K\_d] and player.x + player\_velocity + player.getWidth() < window\_WIDTH: # Move right

player.x += player\_velocity

player.idle()

if keys[pygame.K\_w] and player.y - player\_velocity > 0: # Move up

player.y -= player\_velocity

player.moving()

else:

player.idle()

if keys[pygame.K\_s] and player.y + player\_velocity + player.getHeight() + 20 < window\_HEIGHT: # Move down

player.y += player\_velocity

player.idle()

if keys[pygame.K\_SPACE]:

player.shoot()

WINDOW.blit(player.laserImage, (player.x+25, player.y))

# Enemy behaviour

for enemy in enemies[:]:

enemy.moveDown(enemy\_velocity)

enemy.move\_lasers(laser\_velocity\_enemy, player)

# Enemies shoot randomly

if random.randrange(0, 5\*framesPerSecond) == 1:

enemy.shoot()

if collide(enemy, player):

player.health -= 25

enemy.health -= 50

enemy.hit+=1

if(enemy.health <= 0):

enemyKilledSound.play()

player.kills += 1

player.untilNextLevel -= 1

if(enemy.maxHealth == 100):

player.score += enemy.maxHealth

elif(enemy.maxHealth == 500):

player.score += enemy.maxHealth

else:

player.score += enemy.maxHealth

enemies.remove(enemy)

if(player.health < 100 and player.health >= 75):

player.shipImage = PLAYER\_SHIP\_SLIGHTDAMAGE

elif(player.health < 75 and player.health >= 50):

player.shipImage = PLAYER\_SHIP\_DAMAGED

elif(player.health < 50 and player.health > 0):

player.shipImage = PLAYER\_SHIP\_VERYDAMAGED

else:

player.lives -= 1

lifeLostSound.play()

player.health = 100

player.shipImage = PLAYER\_SHIP

# If the enemies reach the bottom, lose a life

elif enemy.y + enemy.getHeight() > window\_HEIGHT:

player.lives -= 1

lifeLostSound.play()

enemies.remove(enemy)

player.untilNextLevel -= 1

for power in powers[:]:

power.moveDown(enemy\_velocity)

if collide(power, player):

power.activate(player)

powers.remove(power)

elif power.y + power.getHeight() > window\_HEIGHT+40:

powers.remove(power)

player.move\_lasers(laser\_velocity\_player, enemies)

def main\_menu():

titleFont = pygame.font.Font(os.path.join("fonts", "THE SOLSTICE.ttf"), 25)

run = True

while run:

WINDOW.blit(BACKGROUNDMAINMENU, (0, 0))

title\_text = titleFont.render("Last Line of Defense", 1, 'yellow')

instruction\_text = titleFont.render("Click to play...", 1, 'white')

WINDOW.blit(title\_text, (window\_WIDTH/2 -

title\_text.get\_width()/2, 120))

WINDOW.blit(instruction\_text, (window\_WIDTH/2 -

instruction\_text.get\_width()/2, 450))

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

if event.type == pygame.MOUSEBUTTONDOWN:

main()

pygame.quit()

main\_menu()