Introduction

Project Gamezone is a virtual arcade created entirely using Python and making use of the Pygame module, which is a free and open source python programming language library for making multimedia applications, such as games. The project contains two games:

* **Snake It Off**: A new take on the classic game **Snake**, but with two players instead of one. Both players compete to get the apple/food piece to increase their length and try to cut off the path of the other.
* **Stack1e**: A homage to the popular arcade game **Stacker**, developed by LAI Games. In order to win the game, the player must stack 15 rows.

The project is modular, and new games can easily be added to the arcade.

Requirements

The games are operated entirely using a keyboard, although the menu can also be navigated with a mouse.

In order to run the games, Python does not need to be installed on the system, as they have been converted to a Windows executable (.exe) file using the cx\_Freeze module.

**Minimum System Requirements**

* Windows 7/8/10
* 256MB RAM
* 30MB HDD/SSD space
* 1GHz CPU

**Recommended System Requirements**

* Windows 10
* 512MB RAM
* 100MB HDD/SSD space
* 2.5GHz CPU or greater

Menu

**Menu.py**

import sys

sys.path.insert(0, '../../')

import os

import pygame

import pygameMenu

import Stackie

import Snake

color\_black = (0, 0, 0)

color\_white = (255, 255, 255)

color\_background = (17, 165, 240)

color\_title = (29, 98, 194)

window\_size = (800, 800)

surface = None

def snake():

Snake.main()

def stackie():

Stackie.running = True

Stackie.main()

def main\_background():

global surface

surface.fill((20, 20, 20))

def main():

global surface

pygame.init()

surface = pygame.display.set\_mode(window\_size)

pygame.display.set\_caption('Project Gamezone')

pygame.display.set\_icon(pygame.image.load("images\gamezone.png"))

main\_menu = pygameMenu.Menu(surface,

bgfun=main\_background,

color\_selected=color\_white,

font="fonts/Norwester.otf",

font\_color=color\_black,

font\_size=30,

font\_size\_title=40,

menu\_alpha=100,

menu\_color=color\_background,

menu\_color\_title=color\_title,

menu\_height=int(window\_size[1] \* 0.7),

menu\_width=int(window\_size[0] \* 0.8),

# User press ESC button

onclose=pygameMenu.events.EXIT,

option\_shadow=False,

title='Main menu',

window\_height=window\_size[1],

window\_width=window\_size[0]

)

main\_menu.set\_fps(60)

main\_menu.add\_option('Snake', snake)

main\_menu.add\_option('Stack1e', stackie)

main\_menu.add\_option('Quit', pygameMenu.events.EXIT)

while True:

main\_background()

main\_menu.mainloop()

pygame.display.flip()

if \_\_name\_\_ == '\_\_main\_\_':

main()

Stack1e

**Stackie.py**

import pygame

import os

from pygame.locals import \*

from math import sin

import Menu

boardsize = boardwidth, boardheight = 12, 20

screensize = screenwidth, screenheight = 240, 400

tilewidth = screenwidth/boardwidth

tileheight = screenheight/boardheight

tilesize = tilewidth, tileheight

tilecolor = (232, 130, 6)

tilecolor2 = (6, 232, 134)

black = (0,0,0)

levelspd = (80, 80, 75, 75, 70, 70, 65, 65, 60, 55, 50, 45, 40, 35, 30)

maxwidth = (3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3)

colorchangey = 10

winlevel = 15

current\_speed = 50 #in milliseconds

board = []

lose\_tiles = []

current\_direction = 1

current\_x, current\_y, current\_width = 0, boardheight - 1, 3

current\_level = 0

intro = 0

playing = 1

lose = 2

win = 3

game\_state = intro

data\_py = os.path.abspath(os.path.dirname(\_\_file\_\_))

data\_dir = os.path.normpath(os.path.join(data\_py, 'images'))

def filepath(filename):

return os.path.join(data\_dir, filename)

def load\_image(filename):

return pygame.image.load(os.path.join(data\_dir, filename))

bg\_images = (load\_image("intro.png"), load\_image("game.png"), load\_image("lose.png"), load\_image("win.png"))

bg\_images[win].set\_colorkey(black)

bg\_images[lose].set\_colorkey(black)

running = True

pygame.init()

pygame.display.set\_caption('Stack1e')

pygame.display.set\_icon(pygame.image.load("images\stackie.png"))

def main():

pygame.display.set\_caption('Stack1e')

pygame.display.set\_icon(pygame.image.load("images\stackie.png"))

global game\_state, current\_x, current\_y, current\_speed, running, current\_width, current\_level

screen = pygame.display.set\_mode(screensize)

reset\_game()

while(running):

update\_movement()

update\_board\_info()

update\_screen(screen)

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == KEYDOWN:

if event.key == K\_SPACE:

key\_hit()

elif event.key == K\_ESCAPE:

if game\_state == intro:

Menu.main()

running = False

else:

reset\_game()

elif event.key == K\_LCTRL:

current\_x -= 1

if (current\_x < 0): current\_x = 0

current\_width += 1

if (current\_width >= boardwidth): current\_width = boardwidth - 1

pygame.display.quit()

def reset\_game():

global game\_state, current\_x, current\_y, current\_speed, running, current\_width, current\_level, lose\_tiles

clear\_board()

lose\_tiles = []

running = True

game\_state = intro

current\_x = 0

current\_y = boardheight - 1

current\_level = 0

current\_speed = levelspd[current\_level]

current\_width = maxwidth[current\_level]

def key\_hit():

global running, game\_state, current\_x, current\_y, current\_width, current\_speed, current\_level, lose\_tiles

if game\_state == playing:

if current\_y < boardheight - 1:

for x in range(current\_x, current\_x + current\_width):

if board[x][current\_y + 1] == 0:

current\_width -= 1

board[x][current\_y] = 0

lose\_tiles.append((x, current\_y, pygame.time.get\_ticks()))

current\_level += 1

check\_win\_lose()

current\_y -= 1

elif game\_state == intro:

game\_state = playing

elif (game\_state == lose) or (game\_state == win):

reset\_game()

game\_state = intro

else:

running = False

def check\_win\_lose():

global game\_state, current\_width, current\_level, current\_speed, running, tilecolor

if current\_width == 0:

game\_state = lose

elif current\_level == winlevel:

current\_speed = 100

game\_state = win

else:

current\_speed = levelspd[current\_level]

if current\_width > maxwidth[current\_level]:

current\_width = maxwidth[current\_level]

last\_time = 0

def update\_movement():

global game\_state, last\_time, current\_x, current\_y, current\_width, current\_speed, current\_direction

current\_time = pygame.time.get\_ticks()

if (last\_time + current\_speed <= current\_time):

if game\_state == playing:

new\_x = current\_x + current\_direction

if (new\_x < 0) or (new\_x + current\_width > boardwidth):

current\_direction = -current\_direction

current\_x += current\_direction

last\_time = current\_time

def update\_screen(screen):

global game\_state

if game\_state == playing:

draw\_background(screen)

draw\_board(screen)

elif game\_state == intro:

draw\_background(screen)

pass

elif (game\_state == lose) or (game\_state == win):

screen.fill(black)

draw\_board(screen)

draw\_background(screen)

pygame.display.flip()

def draw\_background(screen):

global game\_state

screen.blit(bg\_images[game\_state], (0,0,screenwidth,screenheight), (0,0,screenwidth,screenheight))

def update\_board\_info():

global game\_state

if game\_state == playing:

clear\_row(current\_y)

fill\_current\_row()

def draw\_board(screen):

for x in range(boardwidth):

for y in range(boardheight):

if board[x][y] == 1:

draw\_tile(screen, x, y)

draw\_lose\_tiles(screen)

def draw\_tile(screen, x, y):

xoffset = 0

col = tilecolor

if (y < colorchangey):

col = tilecolor2

if game\_state == win:

xoffset = sin(pygame.time.get\_ticks() \* 0.004 + y \* 0.5) \* (screenwidth / 4)

pygame.draw.rect(screen, col, (x \* tilewidth + xoffset, y \* tileheight, tilewidth, tileheight))

pygame.draw.rect(screen, black, (x \* tilewidth + xoffset, y \* tileheight, tilewidth, tileheight), 2)

#Block falling animation

def draw\_lose\_tiles(screen):

for lt in lose\_tiles:

falltime = (pygame.time.get\_ticks() - lt[2]) \* 0.008 #Time falling

x = lt[0] \* tilewidth

y = lt[1] \* tileheight + falltime \* falltime

col = tilecolor

if (lt[1] < colorchangey):

col = tilecolor2

if (y > screenheight):

lose\_tiles.remove(lt)

else:

pygame.draw.rect(screen, col, (x+2, y+2, tilewidth-3, tileheight-3))

def clear\_board():

global board

board = []

for x in range(boardwidth):

board.append([])

for y in range (boardheight):

board[x].append(0)

def clear\_row(y):

for x in range(boardwidth):

board[x][y] = 0

def fill\_current\_row():

global current\_x, current\_y, current\_width

for x in range(current\_x, current\_x + current\_width):

board[x][current\_y] = 1

if \_\_name\_\_ == '\_\_main\_\_':

main()

Snake It Off

**Game.py**

from Scene import \*

from Entity import \*

import random

from pygame import gfxdraw

class mainGame(Scene):

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

Scene.\_\_init\_\_(self, director)

self.plyronewins = False

self.plyrtwowins = False

self.plyrdraw = False

self.plyronewins\_txt = director.font.render("Player One Wins", True, (255, 255, 255))

self.plyrtwowins\_txt = director.font.render("Player Two Wins", True, (255, 255, 255))

self.plyrdraw\_txt = director.font.render("Draw", True, (255, 255, 255))

self.screen\_rect = director.screen.get\_rect()

self.plyr = pygame.font.Font("fonts\AD.ttf", 20 \* director.scale)

self.txt = self.plyr.render("esc", True, (255, 255, 255))

self.a = director.scale

self.w, self.h = pygame.display.get\_surface().get\_size()

self.dx1 = 10 \* director.scale

self.dy1 = 0

self.dx2 = -10 \* director.scale

self.dy2 = 0

#Snake 1

self.head\_1 = Entity(21 \* director.scale, 1 \* director.scale, 9 \* director.scale, 9 \* director.scale, (11, 27, 176))

self.tail\_1 = []

#Snake 2

self.head\_2 = Entity(371 \* director.scale, 1 \* director.scale, 9 \* director.scale, 9 \* director.scale, (12, 120, 6))

self.tail\_2 = []

self.apple = Entity(191 \* director.scale, 191 \* director.scale, 9 \* director.scale, 9 \* director.scale, (199, 10, 10))

#Starting condition

pygame.mixer.music.load("music\shake.ogg")

pygame.mixer.music.play(-1)

for i in range(1, 3):

self.tail\_1.append(Entity(self.head\_1.x - 10 \* i \* director.scale, self.head\_1.y \* director.scale, 9 \* director.scale, 9 \* director.scale, self.director.p1color))

for i in range(1, 3):

self.tail\_2.append(Entity(self.head\_2.x + 10 \* i \* director.scale, self.head\_2.y \* director.scale, 9 \* director.scale, 9 \* director.scale, self.director.p2color))

def on\_event(self, event):

#Snake 1 movement

if event.type == pygame.KEYDOWN and event.key == pygame.K\_w and self.dy1 == 0:

self.dy1 = -10 \* self.director.scale

self.dx1 = 0

if event.type == pygame.KEYDOWN and event.key == pygame.K\_s and self.dy1 == 0:

self.dy1 = 10 \* self.director.scale

self.dx1 = 0

if event.type == pygame.KEYDOWN and event.key == pygame.K\_a and self.dx1 == 0:

self.dx1 = -10 \* self.director.scale

self.dy1 = 0

if event.type == pygame.KEYDOWN and event.key == pygame.K\_d and self.dx1 == 0:

self.dx1 = 10 \* self.director.scale

self.dy1 = 0

#Snake 2 movement

if event.type == pygame.KEYDOWN and event.key == pygame.K\_UP and self.dy2 == 0:

self.dy2 = -10 \* self.director.scale

self.dx2 = 0

if event.type == pygame.KEYDOWN and event.key == pygame.K\_DOWN and self.dy2 == 0:

self.dy2 = 10 \* self.director.scale

self.dx2 = 0

if event.type == pygame.KEYDOWN and event.key == pygame.K\_LEFT and self.dx2 == 0:

self.dx2 = -10 \* self.director.scale

self.dy2 = 0

if event.type == pygame.KEYDOWN and event.key == pygame.K\_RIGHT and self.dx2 == 0:

self.dx2 = 10 \* self.director.scale

self.dy2 = 0

#Reset game

if event.type == pygame.KEYDOWN and event.key == pygame.K\_r:

pygame.mixer.music.stop()

self.director.change\_scene(mainGame(self.director))

pygame.mixer.music.load("music\shake.ogg")

pygame.mixer.music.play(-1)

#Exit game

if event.type == pygame.KEYDOWN and (event.key == pygame.K\_ESCAPE):

pygame.mixer.music.stop()

self.director.quit()

def on\_update(self):

self.director.fps = 15

if not self.plyronewins and not self.plyrtwowins and not self.plyrdraw:

self.did\_eat()

#Movement: Snake 1

for i in range(len(self.tail\_1) - 1, 0, -1):

self.tail\_1[i].x = self.tail\_1[i - 1].x

self.tail\_1[i].y = self.tail\_1[i - 1].y

self.tail\_1[0].x, self.tail\_1[0].y = (self.head\_1.x, self.head\_1.y)

self.head\_1.x += self.dx1

self.head\_1.y += self.dy1

if self.head\_1.x < 0:

self.head\_1.x = self.w - (9 \* self.a)

if self.head\_1.x > self.w:

self.head\_1.x = self.a

if self.head\_1.y < 0:

self.head\_1.y = self.h - (9 \* self.a)

if self.head\_1.y > self.h:

self.head\_1.y = self.a

#Endgame check

self.is\_collide()

if self.plyrdraw or self.plyronewins or self.plyrtwowins:

return

#Movement: Snake 2

for i in range(len(self.tail\_2) - 1, 0, -1):

self.tail\_2[i].x = self.tail\_2[i - 1].x

self.tail\_2[i].y = self.tail\_2[i - 1].y

self.tail\_2[0].x, self.tail\_2[0].y = (self.head\_2.x, self.head\_2.y)

self.head\_2.x += self.dx2

self.head\_2.y += self.dy2

if self.head\_2.x < 0:

self.head\_2.x = self.w - (9 \* self.a)

if self.head\_2.x > self.w:

self.head\_2.x = self.a

if self.head\_2.y < 0:

self.head\_2.y = self.h - (9 \* self.a)

if self.head\_2.y > self.h:

self.head\_2.y = self.a

#Endgame check

self.is\_collide()

if self.plyrdraw or self.plyronewins or self.plyrtwowins:

return

def on\_draw(self, screen):

#Wipe screen

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

self.draw\_grid(screen)

self.apple.draw(screen)

self.head\_1.draw(screen)

self.head\_2.draw(screen)

self.print\_tails(screen)

#Player 1 wins

if self.plyronewins:

screen.blit(self.plyronewins\_txt, self.plyronewins\_txt.get\_rect(center=self.screen\_rect.center))

self.gameEnd(screen)

#Player 2 wins

if self.plyrtwowins:

screen.blit(self.plyrtwowins\_txt, self.plyronewins\_txt.get\_rect(center=self.screen\_rect.center))

self.gameEnd(screen)

#Draw

if self.plyrdraw:

screen.blit(self.plyrdraw\_txt, self.plyrdraw\_txt.get\_rect(center=self.screen\_rect.center))

self.gameEnd(screen)

def print\_tails(self, screen):

for i in range(len(self.tail\_1)):

pygame.draw.rect(screen, self.tail\_1[i].color, self.tail\_1[i].rect())

for i in range(len(self.tail\_2)):

pygame.draw.rect(screen, self.tail\_2[i].color, self.tail\_2[i].rect())

def draw\_grid(self, screen):

for i in range(40):

pygame.draw.line(screen, (56, 56, 56), (i \* 10 \* self.director.scale, 0), (i \* 10 \* self.director.scale, self.h))

for i in range(40):

pygame.draw.line(screen, (56, 56, 56), (0, i \* 10 \* self.director.scale), (self.w, i \* 10 \* self.director.scale))

def did\_eat(self):

spaceEmpty = True

currX = self.apple.x

currY = self.apple.y

if self.head\_1.x == self.apple.x and self.head\_1.y == self.apple.y:

prevX, prevY = currX, currY

currX = myround(random.randint(0, self.w / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

while prevX == currX:

currX = myround(random.randint(0, self.w / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

currY = myround(random.randint(0, self.h / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

while prevY == currY:

currY = myround(random.randint(0, self.h / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

for i in self.tail\_1:

if currX == i.x and currY == i.y:

spaceEmpty = False

while not spaceEmpty and prevX != currX and prevY != currY:

currX = myround(random.randint(0, self.w / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

currY = myround(random.randint(0, self.h / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

for j in self.tail\_1:

if currX == j.x and currY == j.y:

spaceEmpty = False

break

spaceEmpty = True

self.tail\_1.append(Entity(self.tail\_1[len(self.tail\_1) - 1].x \* self.director.scale, self.tail\_1[len(self.tail\_1) - 1 \* self.director.scale].y, 9 \* self.director.scale, 9 \* self.director.scale, self.director.p1color))

if self.head\_2.x == self.apple.x and self.head\_2.y == self.apple.y:

prevX, prevY = currX, currY

currX = myround(random.randint(0, self.w / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

while prevX == currX:

currX = myround(random.randint(0, self.w / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

currY = myround(random.randint(0, self.h / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

while prevY == currY:

currY = myround(random.randint(0, self.h / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

for i in self.tail\_2:

if currX == i.x and currY == i.y:

spaceEmpty = False

while not spaceEmpty and prevX != currX and prevY != currY:

currX = myround(random.randint(0, self.w / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

currY = myround(random.randint(0, self.h / 10 - 10 \* self.director.scale), 10 \* self.director.scale) \* 10 + 1 \* self.director.scale

for j in self.tail\_2:

if currX == j.x and currY == j.y:

spaceEmpty = False

break

spaceEmpty = True

self.tail\_2.append(Entity(self.tail\_2[len(self.tail\_2) - 1].x \* self.director.scale, self.tail\_2[len(self.tail\_2) - 1 \* self.director.scale].y, 9 \* self.director.scale, 9 \* self.director.scale, self.director.p2color))

#Place new apple

self.apple.x = currX

self.apple.y = currY

def is\_collide(self): #Head-On Collision

if self.head\_1.x == self.head\_2.x and self.head\_1.y == self.head\_2.y:

self.plyrdraw = True

pygame.mixer.music.stop()

return

for i in self.tail\_1:

if self.head\_2.x == i.x and self.head\_2.y == i.y:

self.plyronewins = True

pygame.mixer.music.stop()

return

if self.head\_1.x == i.x and self.head\_1.y == i.y:

self.plyrtwowins = True

pygame.mixer.music.stop()

return

for i in self.tail\_2:

if self.head\_1.x == i.x and self.head\_1.y == i.y:

self.plyrtwowins = True

pygame.mixer.music.stop()

return

if self.head\_2.x == i.x and self.head\_2.y == i.y:

self.plyronewins = True

pygame.mixer.music.stop()

return

def gameEnd(self, screen):

self.txt = self.plyr.render("esc", True, (255, 255, 255))

screen.blit(self.txt, (25 \* self.director.scale, 1 \* self.director.scale))

gfxdraw.filled\_trigon(screen, 5 \* self.director.scale, 19 \* self.director.scale, 20 \* self.director.scale, 26 \* self.director.scale, 20 \* self.director.scale, 12 \* self.director.scale, (232, 107, 35))

gfxdraw.aatrigon(screen, 5 \* self.director.scale, 19 \* self.director.scale, 20 \* self.director.scale, 26 \* self.director.scale, 20 \* self.director.scale, 12 \* self.director.scale, (232, 107, 35))

**Director.py**

import pygame

class Director:

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

self.screen = pygame.display.set\_mode((800, 800))

self.scale = int(pygame.display.get\_surface().get\_width() / 400)

pygame.display.set\_caption("Snake It Off")

pygame.display.set\_icon(pygame.image.load("images\snake.png"))

self.font = pygame.font.Font("fonts\AD.ttf", 30 \* self.scale)

self.scene = None

self.fps = 15

self.quit\_flag = False

self.p1color = (16, 99, 232)

self.p2color = (25, 232, 14)

self.index\_one = 0

self.index\_two = 1

self.clock = pygame.time.Clock()

def loop(self):

while not self.quit\_flag:

time = self.clock.tick(self.fps)

for event in pygame.event.get():

if event.type == pygame.QUIT:

self.quit()

if event.type == pygame.KEYDOWN:

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

self.quit()

self.scene.on\_event(event)

self.scene.on\_update()

self.scene.on\_draw(self.screen)

pygame.display.flip()

def change\_scene(self, scene):

self.scene = scene

def quit(self):

pygame.QUIT

self.quit\_flag = True

**Entity.py**

import pygame

from pygame import Rect

class Entity:

def \_\_init\_\_(self, x, y, width, height, color):

self.x = x

self.y = y

self.width = width

self.height = height

self.color = color

def rect(self):

return Rect(self.x, self.y, self.width, self.height)

def draw(self, screen):

pygame.draw.rect(screen, self.color, self.rect())

def myround(x, base=5):

return int(base \* round(float(x)/base))

**Scene.py**

from pygame import \*

class Scene:

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

self.director = director

def on\_update(self):

raise NotImplementedError("on\_update abstract method must be defined in subclass")

def on\_event(self, event):

raise NotImplementedError("on\_event abstract method must be defined in subclass")

def on\_draw(self, screen):

raise NotImplementedError("on\_Draw abstract method must be defined in subclass")

**Snake.py**

from Director import \*

from Game import \*

def main():

dire = Director()

twop = mainGame(dire)

dire.change\_scene(twop)

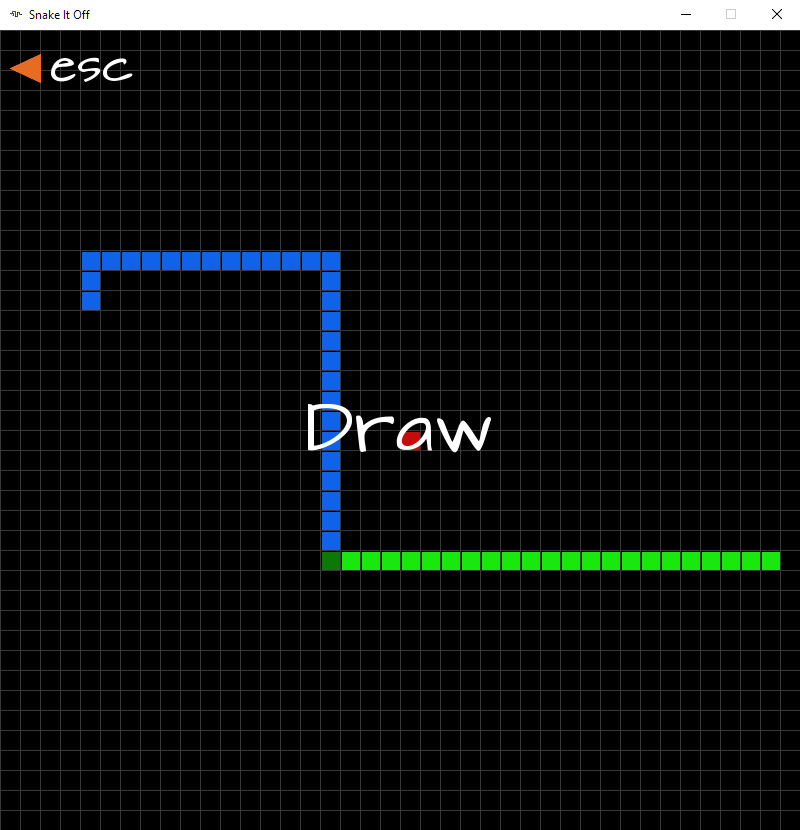
dire.loop()

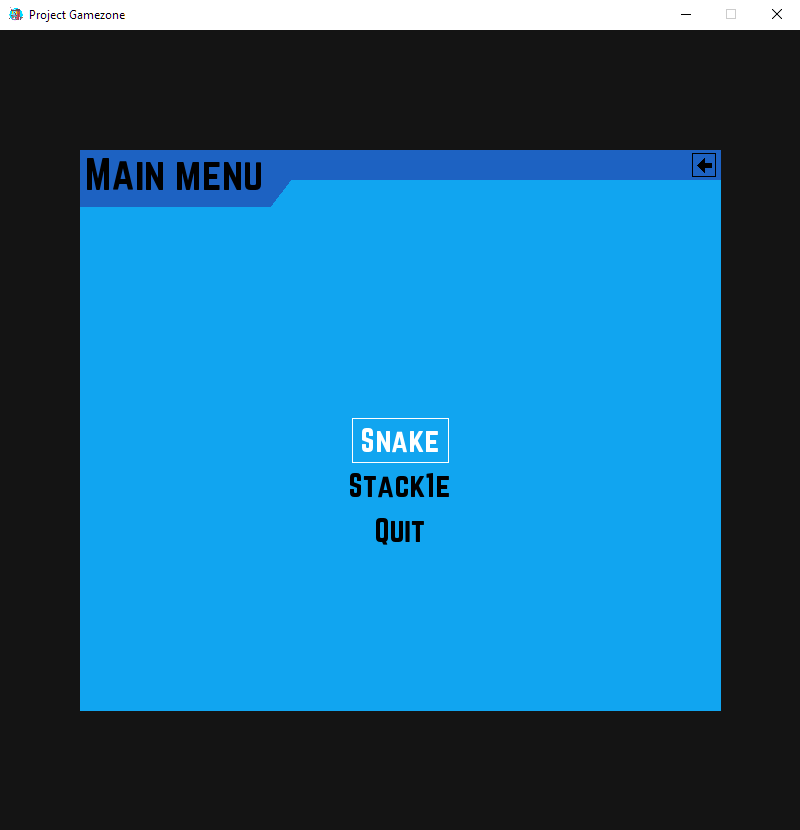
pygame.init()

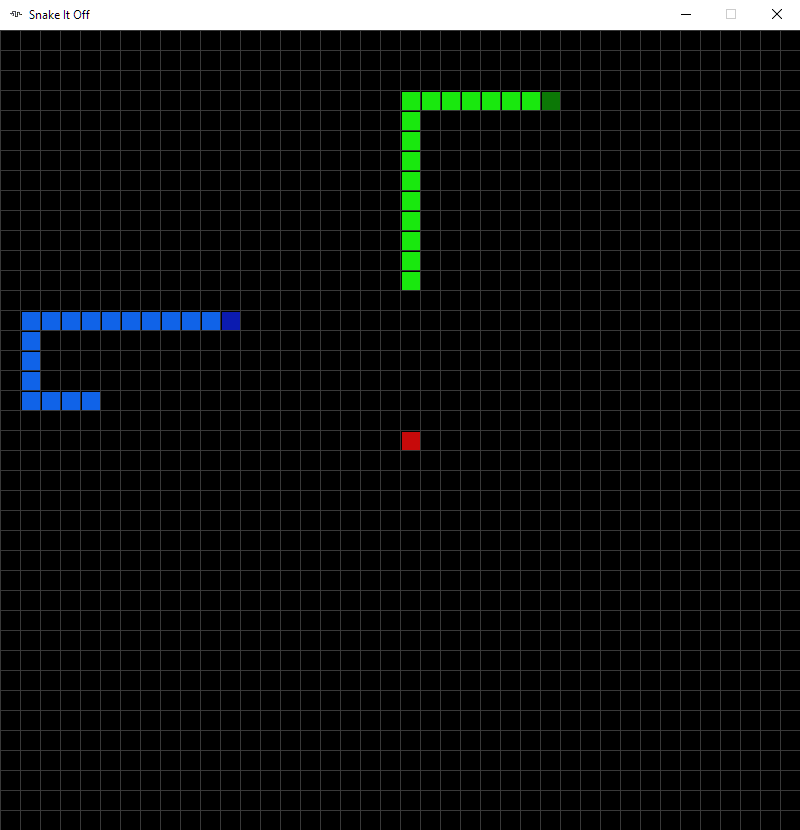
if \_\_name\_\_ == '\_\_main\_\_':

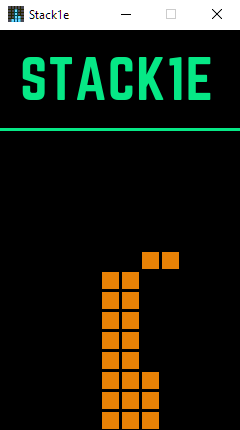
main()

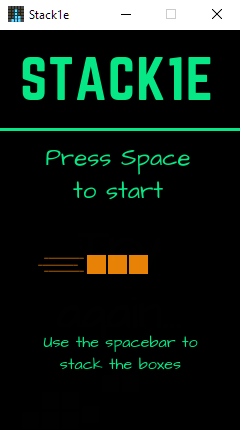
Game Screens

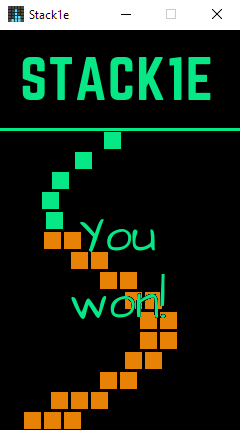
**Menu Screen Snake Draw**

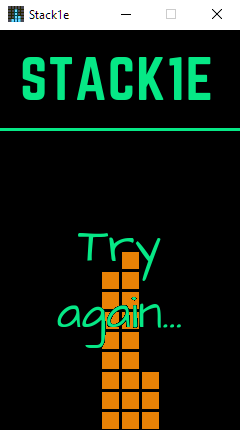


******Snake Gameplay Snake Win**

**Stack1e Title Screen Stack1e Gameplay**

****

**Stack1e Lose Stack1e Win**

****

Conclusion

The games were designed in Python using the Atom IDE, owing to its direct integration with GitHub.

The project is online at <https://github.com/arcanemagic/csproject/tree/master/2020>.

Future Plans

* More arcade games can be added.
* Project Gamezone can be converted into a mobile application for Android and iOS.
* Snake It Off and other future multiplayer games can be played over the network.

Bibliography

* Pygame Docs (<https://www.pygame.org/docs/>)
* cx\_Freeze Docs (<https://cx-freeze.readthedocs.io/en/latest/overview.html>)
* Pygame-Menu Module (<https://github.com/ppizarror/pygame-menu>)
* Stack Overflow (<https://stackoverflow.com/>)
* Reddit r/pygame (<https://reddit.com/r/pygame>)
* Reddit r/learnpython (<https://www.reddit.com/r/learnpython>)