COMPUTER SCIENCE PROJECT

Mini game portal



DONE BY

JEBEZ OSWALD.K

XI-G

COMPUTER SCIENCE PROJECT

Mini game portal



DONE BY

DOMINIC XAVIER.J

XI-G

COMPUTER SCIENCE PROJECT

Mini game portal



DONE BY

SUGAMAN

XI-G

DESCRIPTION

This project is about games that make user to feel stress-relieved. The user has to give input 1 for Space invaders and 2 for connect four.

Space invaders is a shooting game where the user is a spaceship and has to shoot enemies. The game ends when the enemy attacks the spaceship.

Connect Four is a two-player connection board game, in which the players choose a color and then take turns dropping colored tokens into a seven-column, six-row vertically suspended grid. The one who dropped 4 tokens in a line vertically or any other way will win the match.

Modules used are pygame, sys, random, math. Pygame is module is used to draw screen and user and sys module is used to open files and execute it. Random module is used to create opponents in both the games. Math module is used to check the coordinates.

**SOURCE CODE**

import os

import pygame

import numpy as np

import random

import sys

import math

from pygame import mixer

# initializing the pygame

pygame.init()

# creating the screen

screen = pygame.display.set\_mode((1000,600))

# Title / caption and Icon

pygame.display.set\_caption('CHOOSE YOUR GAME')

icon = pygame.image.load('shuttle.png')

pygame.display.set\_icon(icon)

# background

background = pygame.image.load('choosing.jpg')

executing = True

while executing:

# RGB - Red, Green, Blue

screen.fill((0, 120, 150))

screen.blit(background, (-80,-50))

for event in pygame.event.get():

if event.type == pygame.QUIT:

executing = False

if event.type == pygame.K\_1:

import main

os.system('main.py')

elif event.type == pygame.K\_2:

import connect\_four

os.system('connect\_four.py')

pygame.display.update()

#SPACE INVADERS

# initializing the pygame

pygame.init()

# creating the screen

screen = pygame.display.set\_mode((800, 600))

# background

background = pygame.image.load('sapcebackground.jpg')

# background music

mixer.music.load('melodyloops-freemium-epic-motivation-2m30s.mp3')

mixer.music.play(-1)

# Title / caption and Icon

pygame.display.set\_caption('SPACE INVADERS')

icon = pygame.image.load('shuttle.png')

pygame.display.set\_icon(icon)

# player

playerimg = pygame.image.load('space-invaders (1).png')

playerx = 360

playery = 490

playerx\_change = 0

# enemy

enemyimg = []

enemyx = []

enemyy = []

enemyx\_change = []

enemyy\_change = []

num\_of\_enemies = 6

for i in range(num\_of\_enemies):

enemyimg.append(pygame.image.load('enemy.png'))

enemyx.append(random.randint(0, 736))

enemyy.append(random.randint(0, 100))

enemyx\_change.append(0.7)

enemyy\_change.append(10)

# bullet

# ready - you can't see the bullet on the screen

# Fire - the bullet is currently moving

bulletimg = pygame.image.load('bullet.png')

bulletx = 0

bullety = 480

bulletx\_change = 0.3

bullety\_change = 2

bullet\_state = 'ready'

# score

score\_value = 0

font = pygame.font.Font('freesansbold.ttf',20 )

textX = 10

textY = 10

#Game over text

over\_font = pygame.font.Font('freesansbold.ttf', 70)

def show\_score(x, y):

score = over\_font.render('SCORE :' + str(score\_value), True, (255, 0, 0))

screen.blit(score, (x, y))

def game\_over\_text():

over\_text = font.render('GAME OVER' , True, (255, 0, 0))

screen.blit(over\_text ,(200, 250))

def player(x, y):

screen.blit(playerimg, (x, y))

def enemy(x, y, i):

screen.blit(enemyimg[i], (x, y))

def fire\_bullet(x, y):

global bullet\_state

bullet\_state = 'fire'

screen.blit(bulletimg, (x + 16, y + 10))

def iscollision(enemyx, enemyy, bulletx, bullety):

distance = math.sqrt(math.pow(enemyx - bulletx, 2) + math.pow(enemyy - bullety, 2))

if distance < 27:

return True

else:

return False

# Game loop

running = True

while running:

# RGB - Red, Green, Blue

screen.fill((0, 120, 150))

# Background image

screen.blit(background, (0, 0))

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# if keystroke is pressed check whether it's left or right

if event.type == pygame.KEYDOWN:

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

playerx\_change = -0.8

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

playerx\_change = 0.8

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

if bullet\_state is 'ready':

bullet\_sound = mixer.Sound('GunShotSnglShotEx PE1097508.mp3')

bullet\_sound.play()

# get the current x co-ordinate of sapceship

bulletx = playerx

fire\_bullet(playerx, bullety)

if event.type == pygame.KEYUP:

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

playerx\_change = 0

# checking for boundaries so that it doesn't go out of bound

playerx += playerx\_change

if playerx <= 0:

playerx = 0

if playerx >= 736:

playerx = 736

# enemy movement

for i in range(num\_of\_enemies):

#game over

if enemyy[i] > 440 :

for j in range(num\_of\_enemies):

enemyy[j]=100

game\_over\_text()

break

enemyx[i] += enemyx\_change[i]

if enemyx[i] <= 0:

enemyx\_change[i] = 0.5

enemyy[i] += enemyy\_change[i]

if enemyx[i] >= 736:

enemyx\_change[i] = -0.5

enemyy[i] += enemyy\_change[i]

# collision

collision = iscollision(enemyx[i], enemyy[i], bulletx, bullety)

if collision:

bullety = 480

bullet\_state = 'ready'

score\_value += 1

enemyx[i] = random.randint(0, 736)

enemyy[i] = random.randint(0, 100)

enemy(enemyx[i], enemyy[i], i)

# bullet movement

if bullety <= 0:

bullety = 480

bullet\_state = 'ready'

if bullet\_state is 'fire':

fire\_bullet(bulletx, bullety)

bullety -= bullety\_change

player(playerx, playery)

show\_score(textX, textY)

pygame.display.update()

#CONNECT FOUR

BLUE = (0,0,255)

BLACK = (0,0,0)

RED = (255,0,0)

YELLOW = (255,255,0)

ROW\_COUNT = 6

COLUMN\_COUNT = 7

PLAYER = 0

AI = 1

EMPTY = 0

PLAYER\_PIECE = 1

AI\_PIECE = 2

WINDOW\_LENGTH = 4

def create\_board():

board = np.zeros((ROW\_COUNT,COLUMN\_COUNT))

return board

def drop\_piece(board, row, col, piece):

board[row][col] = piece

def is\_valid\_location(board, col):

return board[ROW\_COUNT-1][col] == 0

def get\_next\_open\_row(board, col):

for r in range(ROW\_COUNT):

if board[r][col] == 0:

return r

def print\_board(board):

print(np.flip(board, 0))

def winning\_move(board, piece):

# Check horizontal locations for win

for c in range(COLUMN\_COUNT-3):

for r in range(ROW\_COUNT):

if board[r][c] == piece and board[r][c+1] == piece and board[r][c+2] == piece and board[r][c+3] == piece:

return True

# Check vertical locations for win

for c in range(COLUMN\_COUNT):

for r in range(ROW\_COUNT-3):

if board[r][c] == piece and board[r+1][c] == piece and board[r+2][c] == piece and board[r+3][c] == piece:

return True

# Check positively sloped diaganols

for c in range(COLUMN\_COUNT-3):

for r in range(ROW\_COUNT-3):

if board[r][c] == piece and board[r+1][c+1] == piece and board[r+2][c+2] == piece and board[r+3][c+3] == piece:

return True

# Check negatively sloped diaganols

for c in range(COLUMN\_COUNT-3):

for r in range(3, ROW\_COUNT):

if board[r][c] == piece and board[r-1][c+1] == piece and board[r-2][c+2] == piece and board[r-3][c+3] == piece:

return True

def evaluate\_window(window, piece):

score = 0

opp\_piece = PLAYER\_PIECE

if piece == PLAYER\_PIECE:

opp\_piece = AI\_PIECE

if window.count(piece) == 4:

score += 100

elif window.count(piece) == 3 and window.count(EMPTY) == 1:

score += 5

elif window.count(piece) == 2 and window.count(EMPTY) == 2:

score += 2

if window.count(opp\_piece) == 3 and window.count(EMPTY) == 1:

score -= 4

return score

def score\_position(board, piece):

score = 0

## Score center column

center\_array = [int(i) for i in list(board[:, COLUMN\_COUNT//2])]

center\_count = center\_array.count(piece)

score += center\_count \* 3

## Score Horizontal

for r in range(ROW\_COUNT):

row\_array = [int(i) for i in list(board[r,:])]

for c in range(COLUMN\_COUNT-3):

window = row\_array[c:c+WINDOW\_LENGTH]

score += evaluate\_window(window, piece)

## Score Vertical

for c in range(COLUMN\_COUNT):

col\_array = [int(i) for i in list(board[:,c])]

for r in range(ROW\_COUNT-3):

window = col\_array[r:r+WINDOW\_LENGTH]

score += evaluate\_window(window, piece)

## Score posiive sloped diagonal

for r in range(ROW\_COUNT-3):

for c in range(COLUMN\_COUNT-3):

window = [board[r+i][c+i] for i in range(WINDOW\_LENGTH)]

score += evaluate\_window(window, piece)

for r in range(ROW\_COUNT-3):

for c in range(COLUMN\_COUNT-3):

window = [board[r+3-i][c+i] for i in range(WINDOW\_LENGTH)]

score += evaluate\_window(window, piece)

return score

def is\_terminal\_node(board):

return winning\_move(board, PLAYER\_PIECE) or winning\_move(board, AI\_PIECE) or len(get\_valid\_locations(board)) == 0

def minimax(board, depth, alpha, beta, maximizingPlayer):

valid\_locations = get\_valid\_locations(board)

is\_terminal = is\_terminal\_node(board)

if depth == 0 or is\_terminal:

if is\_terminal:

if winning\_move(board, AI\_PIECE):

return (None, 100000000000000)

elif winning\_move(board, PLAYER\_PIECE):

return (None, -10000000000000)

else: # Game is over, no more valid moves

return (None, 0)

else: # Depth is zero

return (None, score\_position(board, AI\_PIECE))

if maximizingPlayer:

value = -math.inf

column = random.choice(valid\_locations)

for col in valid\_locations:

row = get\_next\_open\_row(board, col)

b\_copy = board.copy()

drop\_piece(b\_copy, row, col, AI\_PIECE)

new\_score = minimax(b\_copy, depth-1, alpha, beta, False)[1]

if new\_score > value:

value = new\_score

column = col

alpha = max(alpha, value)

if alpha >= beta:

break

return column, value

else: # Minimizing player

value = math.inf

column = random.choice(valid\_locations)

for col in valid\_locations:

row = get\_next\_open\_row(board, col)

b\_copy = board.copy()

drop\_piece(b\_copy, row, col, PLAYER\_PIECE)

new\_score = minimax(b\_copy, depth-1, alpha, beta, True)[1]

if new\_score < value:

value = new\_score

column = col

beta = min(beta, value)

if alpha >= beta:

break

return column, value

def get\_valid\_locations(board):

valid\_locations = []

for col in range(COLUMN\_COUNT):

if is\_valid\_location(board, col):

valid\_locations.append(col)

return valid\_locations

def pick\_best\_move(board, piece):

valid\_locations = get\_valid\_locations(board)

best\_score = -10000

best\_col = random.choice(valid\_locations)

for col in valid\_locations:

row = get\_next\_open\_row(board, col)

temp\_board = board.copy()

drop\_piece(temp\_board, row, col, piece)

score = score\_position(temp\_board, piece)

if score > best\_score:

best\_score = score

best\_col = col

return best\_col

def draw\_board(board):

for c in range(COLUMN\_COUNT):

for r in range(ROW\_COUNT):

pygame.draw.rect(screen, BLUE, (c\*SQUARESIZE, r\*SQUARESIZE+SQUARESIZE, SQUARESIZE, SQUARESIZE))

pygame.draw.circle(screen, BLACK, (int(c\*SQUARESIZE+SQUARESIZE/2), int(r\*SQUARESIZE+SQUARESIZE+SQUARESIZE/2)), RADIUS)

for c in range(COLUMN\_COUNT):

for r in range(ROW\_COUNT):

if board[r][c] == PLAYER\_PIECE:

pygame.draw.circle(screen, RED, (int(c\*SQUARESIZE+SQUARESIZE/2), height-int(r\*SQUARESIZE+SQUARESIZE/2)), RADIUS)

elif board[r][c] == AI\_PIECE:

pygame.draw.circle(screen, YELLOW, (int(c\*SQUARESIZE+SQUARESIZE/2), height-int(r\*SQUARESIZE+SQUARESIZE/2)), RADIUS)

pygame.display.update()

board = create\_board()

print\_board(board)

game\_over = False

pygame.init()

SQUARESIZE = 100

width = COLUMN\_COUNT \* SQUARESIZE

height = (ROW\_COUNT+1) \* SQUARESIZE

size = (width, height)

RADIUS = int(SQUARESIZE/2 - 5)

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

draw\_board(board)

pygame.display.update()

myfont = pygame.font.SysFont("monospace", 75)

turn = random.randint(PLAYER, AI)

while not game\_over:

for event in pygame.event.get():

if event.type == pygame.QUIT:

sys.exit()

if event.type == pygame.MOUSEMOTION:

pygame.draw.rect(screen, BLACK, (0,0, width, SQUARESIZE))

posx = event.pos[0]

if turn == PLAYER:

pygame.draw.circle(screen, RED, (posx, int(SQUARESIZE/2)), RADIUS)

pygame.display.update()

if event.type == pygame.MOUSEBUTTONDOWN:

pygame.draw.rect(screen, BLACK, (0,0, width, SQUARESIZE))

#print(event.pos)

# Ask for Player 1 Input

if turn == PLAYER:

posx = event.pos[0]

col = int(math.floor(posx/SQUARESIZE))

if is\_valid\_location(board, col):

row = get\_next\_open\_row(board, col)

drop\_piece(board, row, col, PLAYER\_PIECE)

if winning\_move(board, PLAYER\_PIECE):

label = myfont.render("Player 1 wins!!", 1, RED)

screen.blit(label, (40,10))

game\_over = True

turn += 1

turn = turn % 2

print\_board(board)

draw\_board(board)

# # Ask for Player 2 Input

if turn == AI and not game\_over:

#col = random.randint(0, COLUMN\_COUNT-1)

#col = pick\_best\_move(board, AI\_PIECE)

col, minimax\_score = minimax(board, 5, -math.inf, math.inf, True)

if is\_valid\_location(board, col):

#pygame.time.wait(500)

row = get\_next\_open\_row(board, col)

drop\_piece(board, row, col, AI\_PIECE)

if winning\_move(board, AI\_PIECE):

label = myfont.render("Player 2 wins!!", 1,YELLOW) screen.blit(label, (40,10))

game\_over = True

print\_board(board)

draw\_board(board)

turn += 1

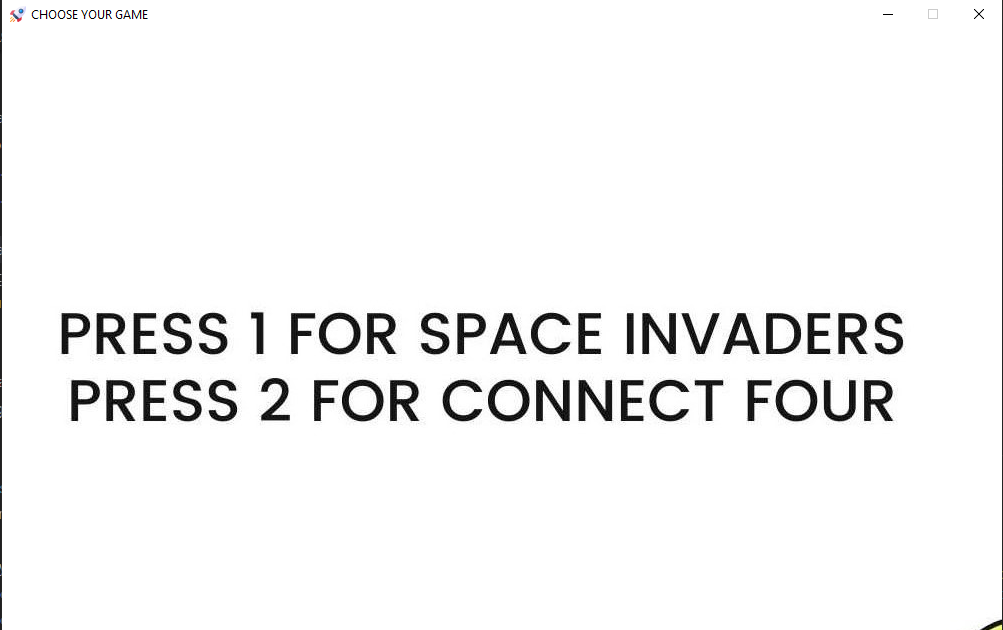
turn = turn % 2

if game\_over:

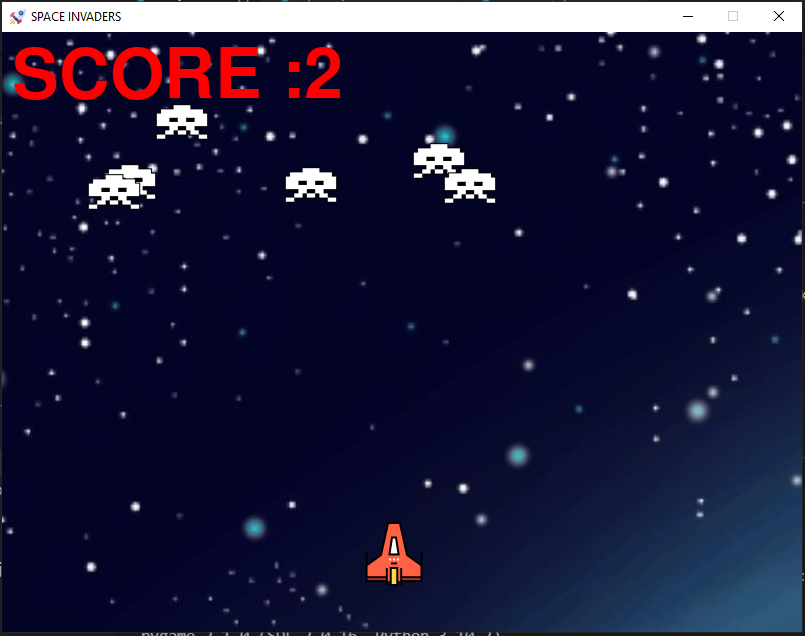
pygame.time.wait(3000)

**OUTPUT**

Window that asks user for choice :



If the user presses 1:



If the user presses 2 :

