**WISE PROJECT**

**Connect Four Dots Game**



**BVRIT HYDERABAD  
COLLEGE OF ENGINEERING FOR WOMEN**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)  
Accredited by NAAC with A Grade  
Bachupally, Hyderabad – 500090

**Department of Computer Science & Engineering  
(Artificial Intelligence & Machine Learning)**

**BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN  
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)  
Accredited by NAAC with A Grade  
Bachupally, Hyderabad – 500090**



**Project Statement**

* Connect Four is a two-player, turn-based game, where each player tries to align four-game coins either horizontally, vertically, or diagonally in the ’7 \* 6’ game board.
* Before starting the game, players decide randomly which of them will be the beginner; moves are made alternatively, one by turn.
* Moves entail placing new pieces on the board; pieces slide

downwards from upper holes, falling to the last row, or piling up on the last piece introduced in the same column. So, in every turn, the introduced piece may be placed at most on seven different squares.

* When the player connects four pieces vertically, horizontally, or diagonally player wins otherwise player loses.

**Problem Explanation**

* The board consists of a grid of dots (6x7 grid).
* Two players take turns. Each player has a unique color of discs (red and yellow).
* On each turn, a player drops one of their discs from the top into one of the columns.
* The disc falls straight down, occupying the lowest available space in the column.
* The first player to form an unbroken line of four of their discs horizontally, vertically, or diagonally wins the gam

**PROJECT CODE**

**Code:**

import cv2

import pygame

import numpy as np

import sys

import math

class Button:

def \_\_init\_\_(self, x, y, image\_path, scale):

self.image = pygame.image.load(image\_path)

self.image = pygame.transform.scale(self.image, (int(self.image.get\_width() \* scale), int(self.image.get\_height() \* scale)))

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

self.rect.topleft = (x, y)

def draw(self, surface):

surface.blit(self.image, self.rect)

def clicked(self, pos):

return self.rect.collidepoint(pos)

def play\_intro\_video(video\_path, screen):

cap = cv2.VideoCapture(video\_path)

frame\_delay = 33

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

frame = np.rot90(frame)

frame = np.flipud(frame)

frame = cv2.resize(frame, (screen.get\_width(), screen.get\_height()), interpolation=cv2.INTER\_LINEAR)

screen.blit(pygame.surfarray.make\_surface(frame), (0, 0))

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

pygame.time.delay(frame\_delay)

cap.release()

def run\_background\_video(video\_path, start\_button, replay\_button, exit\_button):

pygame.init()

pygame.mixer.init()

try:

pygame.mixer.music.load('background.mp3')

pygame.mixer.music.play(-1)

button\_click\_sound = pygame.mixer.Sound('startsound.mp3')

except pygame.error:

print("Error loading sounds")

cap = cv2.VideoCapture(video\_path)

screen\_width, screen\_height = 525, 525

frame\_delay = 0.05

screen = pygame.display.set\_mode((screen\_width, screen\_height))

pygame.display.set\_caption("Connect4Dots")

while True:

ret, frame = cap.read()

if not ret:

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, 0)

ret, frame = cap.read()

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

frame = np.rot90(frame)

frame = np.flipud(frame)

frame = cv2.resize(frame, (screen\_width, screen\_height), interpolation=cv2.INTER\_LINEAR)

frame = pygame.surfarray.make\_surface(frame)

screen.blit(frame, (0, 0))

start\_button.draw(screen)

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

cap.release()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN:

if start\_button.clicked(event.pos):

button\_click\_sound.play()

pygame.time.wait(500)

play\_intro\_video("player12.mp4", screen)

screen.fill(BLACK)

pygame.display.update()

pygame.mixer.music.load('background.mp3')

pygame.mixer.music.play(-1)

board = create\_board()

draw\_board(board, screen)

button\_clicked\_function(screen, replay\_button, exit\_button)

pygame.time.delay(int(frame\_delay \* 1000))

def play\_video(video\_path, screen, replay\_button, exit\_button, win=False):

cap = cv2.VideoCapture(video\_path)

frame\_delay = 33

replay\_clicked = False

if win:

pygame.mixer.music.load('win.mp3')

pygame.mixer.music.play(-1)

try:

button\_click\_sound = pygame.mixer.Sound('startsound.mp3')

except pygame.error:

print("Error loading sounds")

frames = []

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

frame = np.rot90(frame)

frame = np.flipud(frame)

frame = cv2.resize(frame, (screen.get\_width(), screen.get\_height()), interpolation=cv2.INTER\_LINEAR)

frames.append(frame)

cap.release()

while True:

for frame in frames:

screen.blit(pygame.surfarray.make\_surface(frame), (0, 0))

replay\_button.draw(screen)

exit\_button.draw(screen)

pygame.display.update()

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN:

if replay\_button.clicked(event.pos):

button\_click\_sound.play()

pygame.time.wait(500)

replay\_clicked = True

break

elif exit\_button.clicked(event.pos):

button\_click\_sound.play()

pygame.quit()

sys.exit()

if replay\_clicked:

break

pygame.time.delay(frame\_delay)

if replay\_clicked:

screen.fill(BLACK)

pygame.display.update()

run\_background\_video("video1.mp4", start\_button, replay\_button, exit\_button)

def button\_clicked\_function(screen, replay\_button, exit\_button):

board = create\_board()

game\_over = False

turn = 0

SQUARESIZE = 75

RADIUS = int(SQUARESIZE/2 - 5)

width = COLUMN\_COUNT \* SQUARESIZE

try:

drop\_sound = pygame.mixer.Sound('game.mp3')

except pygame.error:

print("Error loading sounds")

while not game\_over:

for event in pygame.event.get():

if event.type == pygame.MOUSEBUTTONDOWN:

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, 1 if turn == 0 else 2)

drop\_sound.play()

if winning\_move(board, 1 if turn == 0 else 2):

play\_video("player11.mp4" if turn == 0 else "player22.mp4", screen, replay\_button, exit\_button, win=True)

game\_over = True

turn += 1

turn %= 2

draw\_board(board, screen)

pygame.display.update()

BLUE = (0, 37, 149)

BLACK = (0, 0, 0)

RED = (220, 20, 60)

YELLOW = (255, 215, 0)

ROW\_COUNT = 6

COLUMN\_COUNT = 7

def create\_board():

return np.zeros((ROW\_COUNT, COLUMN\_COUNT))

def draw\_board(board, screen):

pass

if \_\_name\_\_ == "\_\_main\_\_":

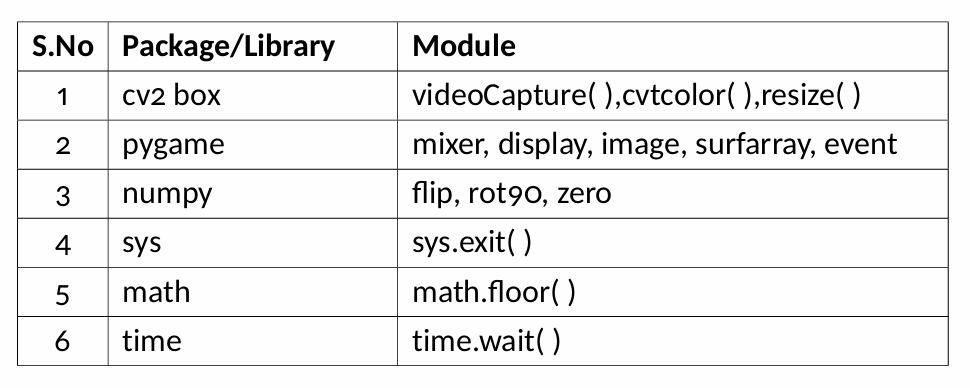
start\_button = Button(210, 275, 'start.jpg', 0.1)

replay\_button = Button(130, 300, 'replay.jpg', 0.1)

exit\_button = Button(280, 300, 'exit.jpg', 0.1)

run\_background\_video("video1.mp4", start\_button, replay\_button, exit\_button)

**Modules and Packages**

****

* cv2box: OpenCV (Open Source Computer Vision Library) is used for real-time computer vision and image processing tasks.
* pygame: It is a set of Python modules designed for writing video games. It includes computer graphics and sound libraries.
* sys: It provides access to some variables used or maintained by the Python interpreter and to functions that interact with the interpreter. math: It provides access to mathematical functions.
* numpy: It is a fundamental package for scientific computing in Python. It provides support for many mathematical functions.

**THANK YOU**