

```

#Imports -----
import time
import pygame
import random

#Default Values -----
winPrint = True
global win
win = False
forceTake = True
global turn
turn = False #red
clock = pygame.time.Clock()
global select
select = False
global coords
coords = ["12black", "14black", "16black", "18black", "21black", "23black", "25black", "27black",
"32black", "34black", "36black", "38black", "61white", "63white", "65white", "67white", "72white",
"74white", "76white", "78white", "81white", "83white", "85white", "87white"]
pygame.init()
gameYes = True
screen = pygame.display.set_mode([1280, 720])
pygame.display.set_caption("CheckersX")
gameIcon = pygame.image.load("icon.ico")
pygame.display.set_icon(gameIcon)
backG = pygame.image.load("background.jpg")
whiteBlank = pygame.image.load("blank.png")
darkBlank = pygame.image.load("dark.png")
counterRed = pygame.image.load("counterRed.png")
counterBlack = pygame.image.load("counterBlack.png")
whiteKing = pygame.image.load("redKing.png")
blackKing = pygame.image.load("blackKing.png")
highlight = pygame.image.load("high.png")
blackGlow = pygame.image.load("blackGlow.png")
whiteGlow = pygame.image.load("whiteGlow.png")
blackKingGlow = pygame.image.load("blackKingGlow.png")
whiteKingGlow = pygame.image.load("redKingGlow.png")

#functions -----
def drawBoard():
    global coords
    switch = False
    x = 320
    y = 40
    l = 0
    for i in range(64):
        if l == 8:
            l = 0
            y = y + 80
            switch = not switch
        row = int(((y - 40) / 80) + 1)
        col = l + 1
        if switch == True:
            if (str(row) + str(col) + "black") in coords: #renders each piece on top of backround
                based on where it finds them in the list
                screen.blit(counterBlack, ((x + (l * 80)), y))
            elif (str(row) + str(col) + "white") in coords:
                screen.blit(counterRed, ((x + (l * 80)), y))
            elif (str(row) + str(col) + "whiteKing") in coords:

```

```

        screen.blit(whiteKing, ((x + (1 * 80)), y))
    elif (str(row) + str(col) + "blackKing") in coords:
        screen.blit(blackKing, ((x + (1 * 80)), y))
    elif (str(row) + str(col) + "high") in coords:
        screen.blit(highlight, ((x + (1 * 80)), y))
    elif (str(row) + str(col) + "take") in coords:
        screen.blit(highlight, ((x + (1 * 80)), y))
    else:
        screen.blit(whiteBlank, ((x + (1 * 80)), y))
    if (str(row) + str(col) + "blackGlow") in coords and forceTake == True:
        screen.blit(blackGlow, ((x + (1 * 80)), y))
    if (str(row) + str(col) + "whiteGlow") in coords and forceTake == True:
        screen.blit(whiteGlow, ((x + (1 * 80)), y))
    if (str(row) + str(col) + "blackKingGlow") in coords and forceTake == True:
        screen.blit(blackKingGlow, ((x + (1 * 80)), y))
    if (str(row) + str(col) + "whiteKingGlow") in coords and forceTake == True:
        screen.blit(whiteKingGlow, ((x + (1 * 80)), y))
    switch = False
elif switch == False:
    screen.blit(darkBlank, ((x + (1 * 80)), y))
    switch = True
l = l + 1

def findMouse(mouseX, mouseY): #finds which square the mouse is over
    for i in range(8):
        if (320 + (i * 80)) <= mouseX <= (320 + (i * 80) + 80):
            col = i + 1
            for b in range(8):
                if (40 + (b * 80)) <= mouseY <= (40 + (b * 80) + 80):
                    row = b + 1
    return (str(row) + str(col))

def checkMove(pieceRow, pieceCol): #checks to see which moves can be made after a user has chosen a
piece
    global coords
    global select
    global turn
    global takuth
    global win
    global canTake
    global takable
    canTake = False
    if ((str(pieceRow) + str(pieceCol)) + "black" in coords) and turn == True: #checks what piece is
in location and if it matches then shows all available moves for it
        clearSelection()
        if forceTake == True:
            takable = checkTake()
        else:
            takable = False
        seeMoves(pieceRow, pieceCol, 1, 1, "black", True)
        seeMoves(pieceRow, pieceCol, 1, -1, "black", True)
        seeMoves(pieceRow, pieceCol, 1, 1, "black", False)
        seeMoves(pieceRow, pieceCol, 1, -1, "black", False)
        return True
    elif ((str(pieceRow) + str(pieceCol)) + "blackKing" in coords) and turn == True:
        clearSelection()
        if forceTake == True:
            takable = checkTake()
        else:

```

```
    takable = False
    seeMoves(pieceRow, pieceCol, 1, 1, "black", True)
    seeMoves(pieceRow, pieceCol, 1, -1, "black", True)
    seeMoves(pieceRow, pieceCol, 1, 1, "black", False)
    seeMoves(pieceRow, pieceCol, 1, -1, "black", False)
    seeMoves(pieceRow, pieceCol, -1, 1, "black", True)
    seeMoves(pieceRow, pieceCol, -1, -1, "black", True)
    seeMoves(pieceRow, pieceCol, -1, 1, "black", False)
    seeMoves(pieceRow, pieceCol, -1, -1, "black", False)
    return True
elif ((str(pieceRow) + str(pieceCol)) + "white" in coords) and turn == False):
    clearSelection()
    if forceTake == True:
        takable = checkTake()
    else:
        takable = False
    seeMoves(pieceRow, pieceCol, -1, 1, "white", True)
    seeMoves(pieceRow, pieceCol, -1, -1, "white", True)
    seeMoves(pieceRow, pieceCol, -1, 1, "white", False)
    seeMoves(pieceRow, pieceCol, -1, -1, "white", False)
    return True
elif ((str(pieceRow) + str(pieceCol)) + "whiteKing" in coords) and turn == False):
    clearSelection()
    if forceTake == True:
        takable = checkTake()
    else:
        takable = False
    seeMoves(pieceRow, pieceCol, -1, 1, "white", True)
    seeMoves(pieceRow, pieceCol, -1, -1, "white", True)
    seeMoves(pieceRow, pieceCol, -1, 1, "white", False)
    seeMoves(pieceRow, pieceCol, -1, -1, "white", False)
    seeMoves(pieceRow, pieceCol, 1, 1, "white", True)
    seeMoves(pieceRow, pieceCol, 1, -1, "white", True)
    seeMoves(pieceRow, pieceCol, 1, 1, "white", False)
    seeMoves(pieceRow, pieceCol, 1, -1, "white", False)
    return True
elif ((str(pieceRow) + str(pieceCol)) + "high" in coords):
    coords.remove(currentPiece)
    if currentPiece[2:] == "white" and pieceRow == 1:
        coords.append(str(pieceRow) + str(pieceCol) + currentPiece[2:] + "King")
    elif currentPiece[2:] == "black" and pieceRow == 8:
        coords.append(str(pieceRow) + str(pieceCol) + currentPiece[2:] + "King")
    else:
        coords.append(str(pieceRow) + str(pieceCol) + currentPiece[2:])
    select = False
    clearSelection()
    turn = not turn
    win = checkWin()
    checkTake()
elif ((str(pieceRow) + str(pieceCol)) + "take" in coords):
    coords.remove(currentPiece)
    print("takuth = " + takuth)
    try:
        coords.remove(takuth)
    except:
        coords.remove(takuth + "King")
    if currentPiece[2:] == "white" and pieceRow == 1:
        coords.append(str(pieceRow) + str(pieceCol) + currentPiece[2:] + "King")
    elif currentPiece[2:] == "black" and pieceRow == 8:
```

```
        coords.append(str(pieceRow) + str(pieceCol) + currentPiece[2:] + "King")
    else:
        coords.append(str(pieceRow) + str(pieceCol) + currentPiece[2:])
    select = False
    clearSelection()
    turn = not turn
    win = checkWin()
    checkTake()
return False

def seeMoves(pieceRow, pieceCol, rChange, cChange, colour, taker):
    global canTake
    global takable
    global takuth
    if colour == "white":
        colour = "black"
    else:
        colour = "white"
    if taker == True: #checks for all the possible moves that a certain piece would be allowed to
        take,
        if (((str(pieceRow + rChange) + (str(pieceCol + cChange)) + colour) in coords) or
            ((str(pieceRow + rChange) + (str(pieceCol + cChange)) + colour + "King") in coords)) and (not
            (str(pieceRow + rChange * 2) + (str(pieceCol + cChange * 2)) + "white") in coords) and (not
            (str(pieceRow + rChange * 2) + (str(pieceCol + cChange * 2)) + "black") in coords) and (not
            (str(pieceRow + rChange * 2) + (str(pieceCol + cChange * 2)) + "whiteKing") in coords) and
            (not (str(pieceRow + rChange * 2) + (str(pieceCol + cChange * 2)) + "blackKing") in coords)):
            coords.append((str(pieceRow + rChange * 2) + str(pieceCol + cChange * 2)) + "take")
            takuth = (str(pieceRow + rChange) + (str(pieceCol + cChange)) + colour)
            edge = checkTake()
            if forceTake == True and edge == True:
                canTake = True
        else:
            if ((not (str(pieceRow + rChange) + (str(pieceCol + cChange)) + "white") in coords) and (not
            (str(pieceRow + rChange) + (str(pieceCol + cChange)) + "black") in coords)) and ((not
            (str(pieceRow + rChange) + (str(pieceCol + cChange)) + "whiteKing") in coords) and (not
            (str(pieceRow + rChange) + (str(pieceCol + cChange)) + "blackKing") in coords)) and canTake
            == False and takable == False:
                coords.append((str(pieceRow + rChange) + str(pieceCol + cChange)) + "high")

def findColour(pieceRow, pieceCol): #function that returns what piece is in the location passed by
parameter
    if ((str(pieceRow) + str(pieceCol)) + "white" in coords) or ((str(pieceRow) + str(pieceCol)) +
    "black" in coords) or ((str(pieceRow) + str(pieceCol)) + "blackKing" in coords) or
    ((str(pieceRow) + str(pieceCol)) + "whiteKing" in coords):
        for i in range(len(coords)):
            if (str(pieceRow) + str(pieceCol)) + "blackKing" in coords[i]:
                return "blackKing"
            elif (str(pieceRow) + str(pieceCol)) + "whiteKing" in coords[i]:
                return "whiteKing"
            elif (str(pieceRow) + str(pieceCol)) + "white" in coords[i]:
                return "white"
            elif (str(pieceRow) + str(pieceCol)) + "black" in coords[i]:
                return "black"

def checkTake():
    found = False
    for i in range(len(coords)): #runs through all possibilities of which piece could be taken in any
direction and flags if it finds one
```

b.local/SAFE/PC Backup/Laptop March/Desktop/ComputerScience/PycharmProjects/checkers v0.52/checkers.py

b.local/SAFE/PC Backup/Laptop March/Desktop/ComputerScience/PycharmProjects/checkers v0.52/checkers.py

```
        coords.append(str(coords[i][0]) + str(coords[i][1] + "whiteKingGlow"))
        found = True
    if found == True:
        return True
    return False

def clearSelection():
    global coords
    for b in range(64):
        for i in range(len(coords)):
            try:
                if "high" in coords[i] or "take" in coords[i] or "Glow" in coords[i]:
                    del coords[i]
            except:
                yes = 1

def checkWin():
    white = False
    black = False
    for i in range(len(coords)):
        if "white" in coords[i] or "whiteKing" in coords[i]:
            black = True
        if "black" in coords[i] or "blackKing" in coords[i]:
            white = True
    if black == False: #returns the winner based on whether there are any pieces of each colour left
        return "black"
    if white == False:
        return "white"
    return False

#Main Program -----
while gameYes:
    #FPS Limiter
    clock.tick(120)

    if win != False and winPrint == True:
        print(win + " has won the game!")
        winPrint = False

    #Game Quit
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            gameYes = False

    #display background
    screen.blit(backG, (0, 0))

    #display game board
    drawBoard()

    #mouse detect click
    mouseX, mouseY = pygame.mouse.get_pos()
    if event.type == pygame.MOUSEBUTTONDOWN and 320 <= mouseX <= 960 and 40 <= mouseY <= 680:
        selection = findMouse(mouseX, mouseY)
        selectionX = int(selection[1])
        selectionY = int(selection[0])
        select = checkMove(selectionY, selectionX)
        colour = findColour(selectionY, selectionX)
        if select == True:
```

```
newSelectionX = int(selection[1])
newSelectionY = int(selection[0])
print(selectionY)
print(selectionX)
print(colour)
currentPiece = (str(selectionY) + str(selectionX) + colour)
checkMove(newSelectionY, newSelectionX)
```

```
#Screen Refresh
```

```
pygame.display.flip()
```

```
#Close Game
```

```
pygame.quit()
```