# ITM 311 - Final Project

# Erick Cabrera

# May 9, 2011

# This program is a tic tac toe game using tkinter for GUI interface

from tkinter import \*

# Graphic ranges

winSize = 600

gridWidth = 2

symWidth = winSize/12

# Symbol size

symSize = 0.5

# X and O color

xColor = 'black'

oColor = 'red'

# draw background

drawColor = 'grey'

gridColor = 'white'

backgroundColor = 'grey'

# define screens

titleScreen = 0

xTurn = 1

oTurn = 2

gameOver = 3

# starting player

firstPlayer = 1

# Size of Box

boxSize = winSize / 3

# defining symbols

none = 0

x = 1

O = 2

class Game(Tk):

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

Tk.\_\_init\_\_(self)

self.canvas = Canvas(

height=winSize,

width=winSize,

bg=backgroundColor)

self.canvas.pack()

self.bind('<x>', self.exit)

self.canvas.bind('<Button-1>', self.click)

self.gamestate = titleScreen

self.title()

self.board = [

[none, none, none],

[none, none, none],

[none, none, none]]

# creates title screen

def title(self):

self.canvas.create\_rectangle(

0, 0,

winSize, winSize,

fill='black',

outline='')

self.canvas.create\_text(

winSize/2,

winSize/2.5,

text='TIC TAC TOE', fill='red',

font=('Roboto', int(-winSize/8), 'bold'))

self.canvas.create\_text(

int(winSize/2),

int(winSize/2),

text='play', fill='white',

font=('Roboto', int(-winSize/20)))

# logic for each click during game

def click(self, event):

x = self.pixelsGrid(event.x)

y = self.pixelsGrid(event.y)

if self.gamestate == titleScreen:

self.newBoard()

self.gamestate = firstPlayer

elif (self.gamestate == xTurn and

self.board[y][x] == none):

self.symLocation(X, x, y)

if self.winner(X):

self.gamestate = gameOver

self.endgame('X Victory')

elif self.draw():

self.gamestate = gameOver

self.endgame('DRAW')

else:

self.gamestate = oTurn

elif (self.gamestate == oTurn and

self.board[y][x] == none):

self.symLocation(O, x, y)

if self.winner(O):

self.gamestate = gameOver

self.endgame('O Victory')

elif self.draw():

self.gamestate = gameOver

self.endgame('DRAW')

else:

self.gamestate = xTurn

elif self.gamestate == gameOver:

self.newBoard()

self.gamestate = firstPlayer

# creates a new board

def newBoard(self):

# delete all objects

self.canvas.delete('all')

# clean board

self.board = [

[none, none, none],

[none, none, none],

[none, none, none]]

# draw grid

for n in range(1, 3):

self.canvas.create\_line(

boxSize\*n, 0,

boxSize\*n, winSize,

width=gridWidth, fill=gridColor)

self.canvas.create\_line(

0, boxSize\*n,

winSize, boxSize\*n,

width=gridWidth, fill=gridColor)

# captures where in grid you want to place symbol

def symLocation(self, player, xGrid, yGrid):

if player == X:

self.createX(xGrid, yGrid)

self.board[yGrid][xGrid] = X

elif player == O:

self.createO(xGrid, yGrid)

self.board[yGrid][xGrid] = O

# draws out the X in the location you clicked

def createX(self, xGrid, yGrid):

x = self.gridPixels(xGrid)

y = self.gridPixels(yGrid)

delta = boxSize/2\*symSize

self.canvas.create\_line(

x-delta, y-delta,

x+delta, y+delta,

width=symWidth, fill=xColor)

self.canvas.create\_line(

x+delta, y-delta,

x-delta, y+delta,

width=symWidth, fill=xColor)

# draws out the O in the location you clicked

def createO(self, xGrid, yGrid):

x = self.gridPixels(xGrid)

y = self.gridPixels(yGrid)

delta = 1.5 \* boxSize / 2 \* symSize

self.canvas.create\_oval(

x-delta, y-delta,

x+delta, y+delta,

fill=oColor, outline="")

self.canvas.create\_oval(

x-delta/3, y-delta/3,

x+delta/3, y+delta/3,

fill=backgroundColor, outline="")

# checks to see if you got 3 symbols either horizontally, vertically, or diagnogally

def winner(self, symbol):

for y in range(3):

if self.board[y] == [symbol, symbol, symbol]:

return True

for x in range(3):

if self.board[0][x] == self.board[1][x] == self.board[2][x] == symbol:

return True

if self.board[0][0] == self.board[1][1] == self.board[2][2] == symbol:

return True

elif self.board[0][2] == self.board[1][1] == self.board[2][0] == symbol:

return True

return False

# if there is no winning pattern and all cells are clicked, game is a draw

def draw(self):

for row in self.board:

if none in row:

return False

return True

# creates end result of game

def endgame(self, result):

# delete all objects

self.canvas.delete('all')

# who wins

if result == 'X Victory':

wintext = 'Victory for X!'

wincolor = xColor

elif result == 'O Victory':

wintext = 'Victory for O!'

wincolor = oColor

elif result == 'DRAW':

wintext = 'Draw'

wincolor = drawColor

self.canvas.create\_rectangle(

0, 0,

winSize, winSize,

fill=wincolor, outline='')

self.canvas.create\_text(

int(winSize/2), int(winSize/2),

text=wintext, fill='white',

font=('Roboto', int(-winSize/7), 'bold'))

self.canvas.create\_text(

int(winSize/2), int(winSize/1.55),

text='play again?', fill='white',

font=('Roboto', int(-winSize/25)))

# grid to pixels to fix viewing

def gridPixels(self, coordinates):

pixCoord = coordinates \* boxSize + boxSize / 2

return pixCoord

# pixels to grid to fix viewing

def pixelsGrid(self, pixCoord):

if pixCoord >= winSize:

pixCoord = winSize - 1

gridCoord = int(pixCoord / boxSize)

return gridCoord

# if you exit game

def exit(self, event):

self.destroy()

# main function to run game

def main():

root = Game()

root.mainloop()

main()