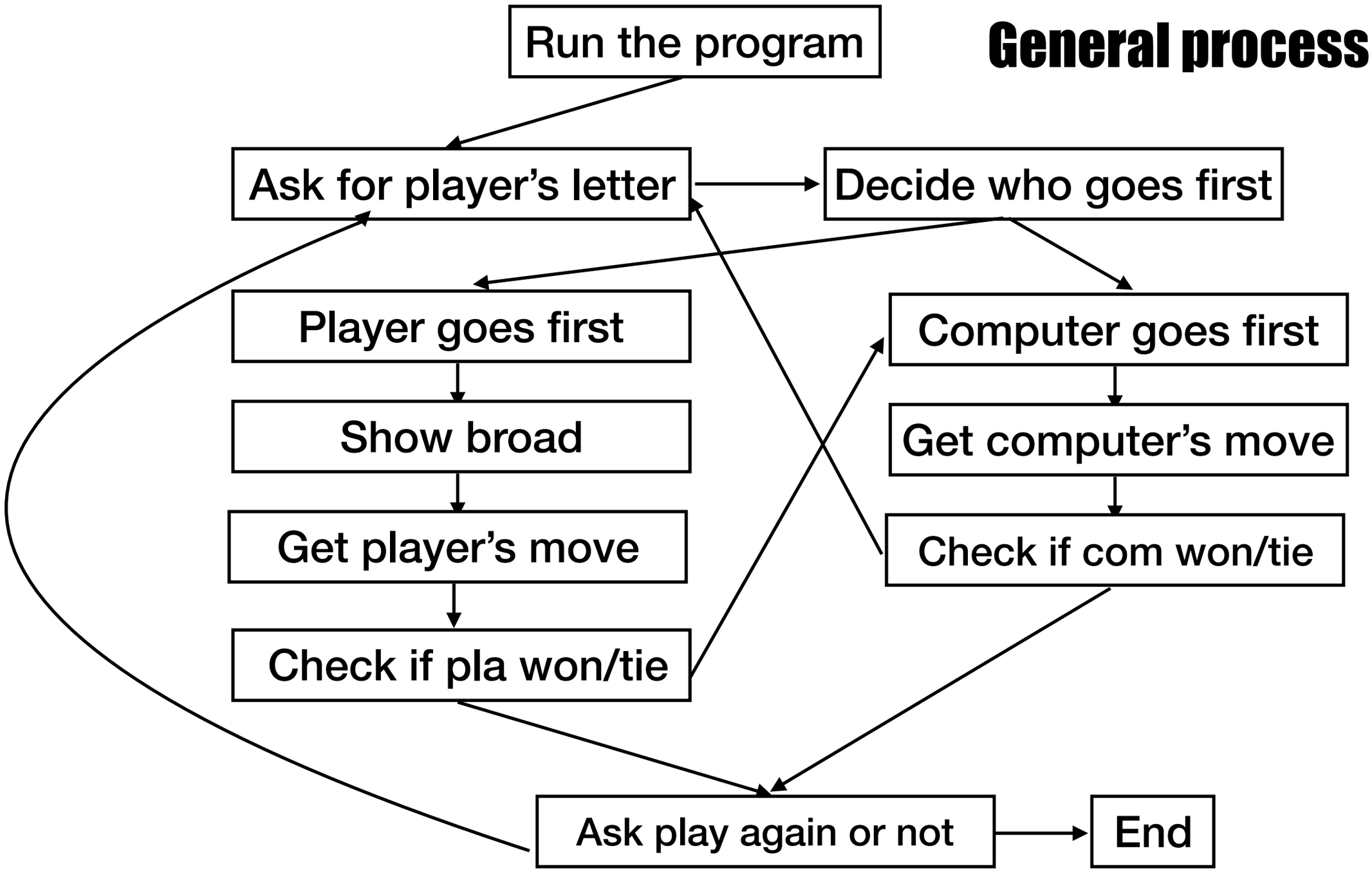
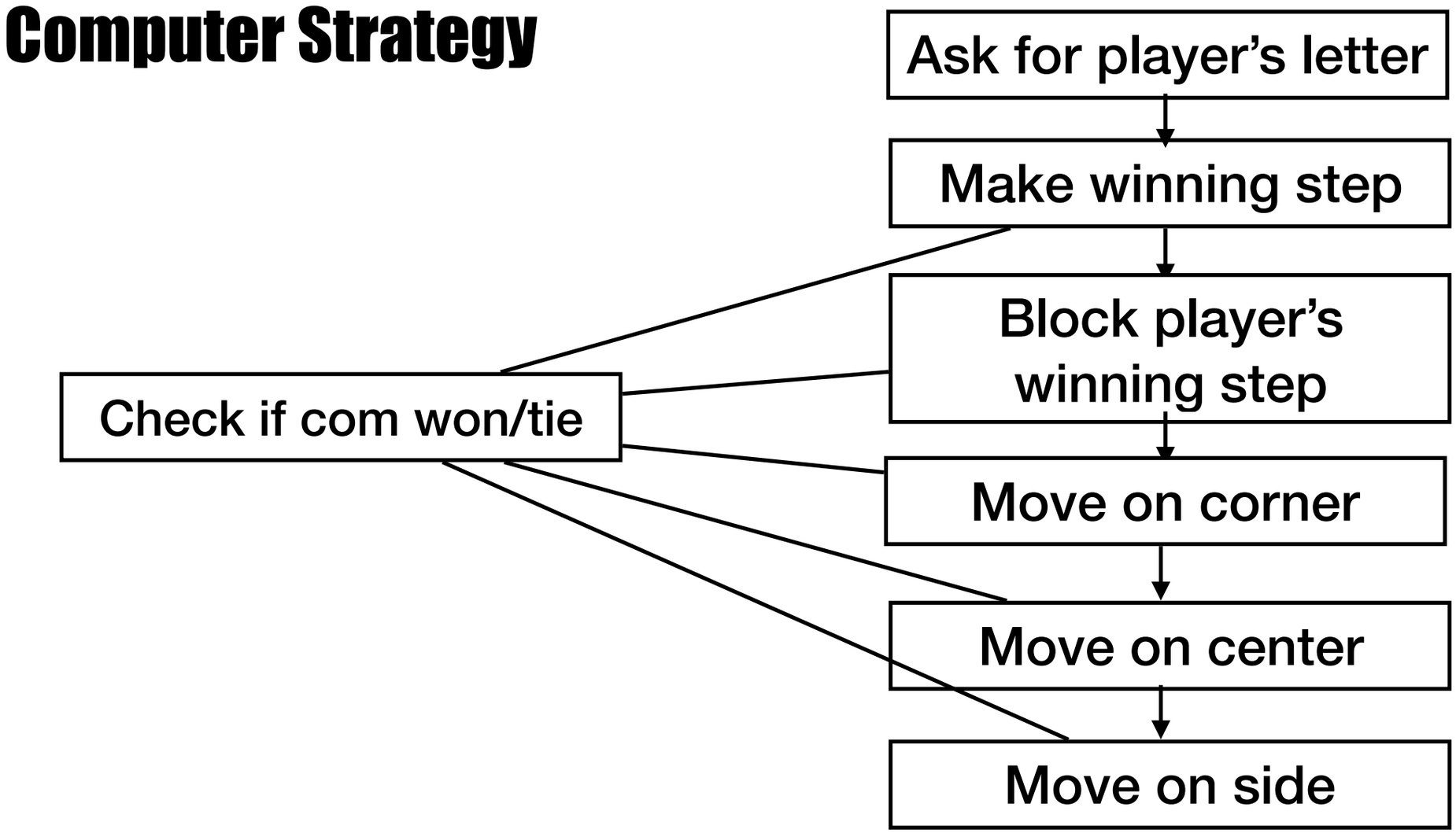
Tic Tac Toe Pygame



Criterion B: Design

Tic Tac Toe Pygame



Ask player’s for letter

```
def inputPlayerLetter():
    # Lets the player type which letter they want to be.
    # Returns a list with the player’s letter as the first item, and the computer's letter as the second.
    letter = ''
    while not(letter == 'X' or letter == 'O'):
        print('Do you want to be X or O?')
        letter = input().upper()

    # the first element in the list is the player's letter, the second is the computer's letter as the second.
    if letter == 'X':
        return ['X','O']
    else:
        return ['O','X']
```

Decide who goes first

```
def whoGoesFirst():
    # Randomly choose the player whoi goes first.
    if random.randint(0,1) == 0:
        return 'computer'
    else:
        return 'player'
```

Player goes first

```
def getPlayerMove(board):
    # Let the player type in their move.
    move = ' '
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board,int(move)):
        print('What is your next move?(1-9)')
        move = input()
return int(move)
```

Show broad

```
def draw_game():
    #导入背景图

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

    #用法：pygame.draw.line(显示,颜色,开始位置,结束位置,宽度)

    pygame.draw.line(screen, black, (160, 0), (160, 480), 5)
    pygame.draw.line(screen, black, (320, 0), (320, 480), 5)
    pygame.draw.line(screen, black, (0, 160), (480, 160), 5)
    pygame.draw.line(screen, black, (0, 320), (480, 320), 5)
    #遍历列表中的元素及他们的下标 row横col竖 row col是下标

    for row, line in enumerate(state):
        for col, val in enumerate(line):
            if val == -1:
                #画x

                upper_left = (col * 160 + 5, row * 160 + 5)
                lower_right = (col * 160 + 155, row * 160 + 155)
                pygame.draw.line(screen, red, upper_left, lower_right, 5)

                upper_right = (col * 160 + 155, row * 160 + 5)
                lower_left = (col * 160 + 5, row * 160 + 155)
                pygame.draw.line(screen, red, upper_right, lower_left, 5)
            elif val == 1:
                #创建一个矩形.在矩形里画圆

                rect = (col * 160 + 5, row * 160 + 5, 150, 150)
                pygame.draw.ellipse(screen, blue, rect, 5)
            else:
                assert val == empty
                continue
    pygame.display.flip()
```

Get player's move

```
def getPlayerMove(board):
    # Let the player type in their move.
    move = ' '
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board,int(move)):
        print('What is your next move?(1-9)')
        move = input()
    return int(move)
```

Check if pla won/tie

```python
def isWinner(bo,le):
    # Given a board and a player's letter, this function returns True if that player has won.
    # We use bo instead of board and le instead of letter so we dont have to type as much.
    return ((bo[7] == le and bo[8] == le and bo[9] == le) or # across the top
    (bo[4] == le and bo[5] == le and bo[6] == le) or # across the middle
    (bo[1] == le and bo[2] == le and bo[3] == le) or # across the bottom
    (bo[7] == le and bo[4] == le and bo[1] == le) or # down the left side
    (bo[8] == le and bo[5] == le and bo[2] == le) or # down the middle
    (bo[9] == le and bo[6] == le and bo[3] == le) or # down the right side
    (bo[7] == le and bo[5] == le and bo[3] == le) or # diagonal
    (bo[9] == le and bo[5] == le and bo[1] == le)) # diagonal
```

```python
def is_won():
        for val in range(3):
            # 检查匹配的行三个图形是否都相同且不等于空


            if state[0][val] == state[1][val] == state[2][val] != empty:
              return state[0][val]

            # 检查匹配的列三个图形是否都相同不等于空


            if state[val][0] == state[val][1] == state[val][2] != empty:
              return state[val][0]

            #判断 \ 中三个图形是否都相同

        if state[0][0] == state[1][1] == state[2][2] != empty:
            return state[1][1]
            #判断 / 中三个图形是否都相同

        if state[0][2] == state[1][1] == state[2][0] != empty:
            return state[1][1]
#初始化棋盘
```

Computer goes first

```
def getComputerMove(board,computerLetter):
    # Given a board and the computer's letter, determine where to move and return that move.
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'
```

Get computer’s move

```
def getComputerMove(board,computerLetter):
    # Given a board and the computer's letter, determine where to move and return that move.
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'
```

Check if com won/tie

```
def isWinner(bo,le):
    # Given a board and a player's letter, this function returns True if that player has won.
    # We use bo instead of board and le instead of letter so we dont have to type as much.
    return ((bo[7] == le and bo[8] == le and bo[9] == le) or # across the top
    (bo[4] == le and bo[5] == le and bo[6] == le) or # across the middle
    (bo[1] == le and bo[2] == le and bo[3] == le) or # across the bottom
    (bo[7] == le and bo[4] == le and bo[1] == le) or # down the left side
    (bo[8] == le and bo[5] == le and bo[2] == le) or # down the middle
    (bo[9] == le and bo[6] == le and bo[3] == le) or # down the right side
    (bo[7] == le and bo[5] == le and bo[3] == le) or # diagonal
    (bo[9] == le and bo[5] == le and bo[1] == le)) # diagonal
```

```
def is_won():
        for val in range(3):
                # 检查匹配的行三个图形是否都相同且不等于空


                if state[0][val] == state[1][val] == state[2][val] != empty:
                  return state[0][val]

                # 检查匹配的列三个图形是否都相同不等于空


                if state[val][0] == state[val][1] == state[val][2] != empty:
                  return state[val][0]

                #判断 \ 中三个图形是否都相同

        if state[0][0] == state[1][1] == state[2][2] != empty:
                return state[1][1]
                #判断 / 中三个图形是否都相同

        if state[0][2] == state[1][1] == state[2][0] != empty:
                return state[1][1]
#初始化棋盘
```

Ask play again or not

```
def playAgain():
    # This function returns True if the player wants to play again, otherwise it returns False.
    print('Do you want to play again?(yes or no)')
    return input().lower().startswith('y')
```

Make winning step

```
# Here is our algorithm for our Tic Tac Toe AI:
# First, check if we can win in the next move
for i in range(1,10):
    copy = getBoardCopy(board)
    if isSpaceFree(copy,i):
        makeMove(copy,computerLetter,i)
        if isWinner(copy,computerLetter):
            return i
```

Block player's
winning step

```
# Check if the player could win on their next move, and block them.
for i in range(1,10):
    copy = getBoardCopy(board)
    if isSpaceFree(copy,i):
        makeMove(copy,playerLetter,i)
        if isWinner(copy,playerLetter):
            return i
```

Move on corner

```
# Try to take one of the corners, if they are free.
move = chooseRandomMoveFromList(board,[1,3,7,9])
if move != None:
    return move
```

Move on center

```
# Try to take the center, if it is free.
if isSpaceFree(board,5):
    return 5
```

Move on side

```
# Move on one of the sides.
return chooseRandomMoveFromList(board,[2,4,6,8])
```