```
#Imports#
import pygame, random

#Initialisation#
def init():
    global gameRun, squareList, lose, win, gameStart
    gameRun, squareList, lose, win, gameStart = True, [], False, False, False
global gameRun, squareList, gameStart, difficulty, clicked
difficulty = 0
init()
clock = pygame.time.Clock()
fps = 30
clicked = False

pygame.init()
screen = pygame.display.set_mode((900, 900))
numFont = pygame.font.Font("numFont.ttf", 35) #import custom font
bombImage = pygame.image.load("bomb.png")
bombImage = pygame.transform.scale(bombImage, (120, 120)).convert_alpha()
flagImage = pygame.transform.scale(pygame.image.load("flag.png"), (44, 44)).convert_alpha()

#Classes#
class Box():
    def __init__(self, posX, posY, listPos, diff):
        self.flagged = False
        self.posX, self.posY = posX, posY
        self.show = False
        self.listPos = listPos
        match diff:
            case 0:
                diff = 12
            case 1:
                diff = 7
            case 2:
                diff = 3
        tempInt = random.randint(1, diff)
        self.isBomb = False
        if tempInt == 1:
            self.isBomb = True
        self.bombCount = 0
        if (self.listPos + 1) % 20 == 0:
            self.isRight = True
        else:
            self.isRight = False
        if self.listPos == 0 or self.listPos % 20 == 0:
            self.isLeft = True
        else:
            self.isLeft = False
        if self.listPos < 20:
            self.isTop = True
        else:
            self.isTop = False
        if self.listPos > 379:
            self.isBottom = True
        else:
            self.isBottom = False
        if self.isBomb == True:
            self.number = 0
        else:
```

```
            self.number = False
    def display(self):
        global win
        colour = (150, 150, 150)
        if self.show:
            colour = (200, 200, 200)
        if win:
            colour = (50, 200, 50)
        pygame.draw.rect(screen, colour, (self.posX, self.posY, 44, 44)) #draws the square
        pygame.draw.rect(screen, (0, 0, 0), (self.posX + 43, self.posY, 1, 44))
        pygame.draw.rect(screen, (0, 0, 0), (self.posX, self.posY + 43, 44, 1))
        if self.show and not self.isBomb and self.bombCount > 0: #checks if player has clicked it and
    isnt a bomb and has bombs around it
            textCol = (182, 34, 136)
            match self.bombCount: #changes text colour based on amount of bombs
                case 1:
                    textCol = (34, 136, 182)
                case 2:
                    textCol = (34, 182, 136)
                case 3:
                    textCol = (255, 60, 10)
                case 4:
                    textCol = (182, 34, 136)
            number = numFont.render(str(self.bombCount), True, textCol)
            numRect = number.get_rect()
            numRect.x, numRect.y = self.posX + 13, self.posY + 5
            screen.blit(number, numRect)
        elif self.isBomb and self.show:
            screen.blit(bombImage, (self.posX -40, self.posY - 43))
        elif self.flagged and not self.show:
            screen.blit(flagImage, (self.posX, self.posY))
    def countBombs(self):
        bombCount = 0
        for i in range(8):
            match i: #checks in all spaces around to see how many bombs there are
                case 0:
                    run = 0
                    if not self.isTop and not self.isLeft:
                        run = -21
                case 1:
                    run = 0
                    if not self.isTop:
                        run = -20
                case 2:
                    run = 0
                    if not self.isTop and not self.isRight:
                        run = -19
                case 3:
                    run = 0
                    if not self.isLeft:
                        run = -1
                case 4:
                    run = 0
                    if not self.isRight:
                        run = 1
                case 5:
                    run = 0
                    if not self.isBottom and not self.isLeft:
                        run = 19
```

```
                case 6:
                    run = 0
                    if not self.isBottom:
                        run = 20
                case 7:
                    run = 0
                    if not self.isRight and not self.isBottom:
                        run = 21
            if squareList[self.listPos + run].isBomb:
                bombCount += 1
        self.bombCount = bombCount
    def revealBlanks(self): #recursive function to reveal all surrounding blank square if user clicks blank square
        for i in range(8):
            match i:
                case 0:
                    run = 0
                    if not self.isTop and not self.isLeft:
                        run = -21
                case 1:
                    run = 0
                    if not self.isTop:
                        run = -20
                case 2:
                    run = 0
                    if not self.isTop and not self.isRight:
                        run = -19
                case 3:
                    run = 0
                    if not self.isLeft:
                        run = -1
                case 4:
                    run = 0
                    if not self.isRight:
                        run = 1
                case 5:
                    run = 0
                    if not self.isBottom and not self.isLeft:
                        run = 19
                case 6:
                    run = 0
                    if not self.isBottom:
                        run = 20
                case 7:
                    run = 0
                    if not self.isRight and not self.isBottom:
                        run = 21
            if squareList[self.listPos + run].isBomb == False and not squareList[self.listPos + run].show:
                squareList[self.listPos + run].clicked() #recursive call
    def clicked(self):
        global lose
        if lose:
            init()
            return
        self.show = True
        if self.isBomb:
            lose = True
            for i in range(len(squareList)):
```

```
                if squareList[i].isBomb:
                    squareList[i].show = True
        if self.bombCount == 0 and not self.isBomb:
            self.revealBlanks() #calls recursive function

#Functions
def drawBackG():
    pygame.draw.rect(screen, (100, 100, 100), (0, 0, 900, 900))

def makeSquares(): #generates all the boxes at start of game
    global squareList, difficulty
    for i in range(20):
        for j in range(20):
            squareList.append(Box(((j*44) + 10), ((i*44) + 10), ((j + 1) + ((i + 1) * 20 - 21)),
            difficulty))
    for i in range(len(squareList)):
        squareList[i].countBombs()

def drawSquares():
    for i in range(len(squareList)):
        squareList[i].display()

def clickLoop(flag):
    mousePosX, mousePosY = pygame.mouse.get_pos()[0], pygame.mouse.get_pos()[1]
    for i in range(len(squareList)): #checks mouse pos and mouse buttons to see which box gets
    clicked
        if mousePosX > squareList[i].posX and mousePosX < squareList[i].posX + 44:
            if mousePosY > squareList[i].posY and mousePosY < squareList[i].posY + 44:
                if flag:
                    squareList[i].flagged = not squareList[i].flagged
                    break
                else:
                    squareList[i].show = True
                    squareList[i].clicked()
                    break

def checkWin():
    global win, lose
    winFlag = True #checks that all the non-bomb square have been opened to see if user has won
    for i in range(len(squareList)):
        if not squareList[i].show and not squareList[i].isBomb:
            winFlag = False
            break
        else:
            winFlag = True
    if winFlag:
        win = True
        lose = True

def button(colour, text, xPos, yPos): #creates buttons on main menu
    global difficulty, gameRun, clicked, gameStart
    if pygame.mouse.get_pos()[0] >= xPos and pygame.mouse.get_pos()[0] < xPos + 200:
        if pygame.mouse.get_pos()[1] >= yPos and pygame.mouse.get_pos()[1] < yPos + 100:
            colour = (colour[0] - 30, colour[1] - 30, colour[2] - 30)
            if clicked:
                clicked = False
                match text: #difficulty cycle button logic
                    case "EASY":
                        difficulty = 1
```

```
                    case "MED":
                        difficulty = 2
                    case "HARD":
                        difficulty = 0
                if text == "PLAY":
                    init()
                    makeSquares()
                    gameStart = True
    pygame.draw.rect(screen, colour, (xPos, yPos, 200, 100))
    wordFont = pygame.font.Font("numFont.ttf", 40)
    number = wordFont.render(text, True, (150, 50, 50))
    numRect = number.get_rect()
    numRect.center = (xPos + 100, yPos + 50)
    screen.blit(number, numRect)

def menuScreen(): #menu screen main loop
    drawBackG()
    wordFont = pygame.font.Font("numFont.ttf", 120)
    number = wordFont.render("BOMB Bastic", True, (150, 50, 50))
    numRect = number.get_rect()
    numRect.x, numRect.y = 90, 20
    screen.blit(number, numRect)
    match difficulty:
        case 0:
            diffText = "EASY"
        case 1:
            diffText = "MED"
        case 2:
            diffText = "HARD"
    button((150, 150, 150), "PLAY", 350, 300)
    button((150, 150, 150), diffText, 350, 420)

#Main Loop#
while gameRun:
    clicked = False
    clock.tick(fps)
    if not gameStart:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                gameRun = False
            if event.type == pygame.MOUSEBUTTONUP:
                if event.button == 1:
                    clicked = True
        menuScreen()
    else:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                gameRun = False
            if event.type == pygame.MOUSEBUTTONUP:
                if event.button == 1:
                    clickLoop(False)
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_f:
                    clickLoop(True)
        drawBackG()
        drawSquares()
        checkWin()
    pygame.display.flip()
```

```
pygame.quit()
print(len(squareList))
```