

Color Fill Game

Вовед

Целта во играта е да се обои мапата целосно со бои, така што нема да постојат две соседни полиња (хоризонтално или вертикално) кои ќе имаат иста боја. Во моментот кога ќе се обои мапата, играта завршува, а играчот ја победува играта.

Правило во играта, како што веќе е напоменато да нема две соседни полиња кои се обоени со иста боја. Има четири бои кои се избираат од палетата наведена во горниот дел од екранот.

Документација за кодот

Во горниот дел од кодот, првите неколку редови се вклучување на библиотеките кои ги овозможуваат функционалностите за pygame.

```
import random
import pygame
import sys
from pygame.locals import *
```

Потоа следува дефинирање на променливи кои ги употребувам во текот на кодирањето на целата игра. Ги дефинирам на почетокот за поголема и појасна прегледност на кодот.

```
FPS = 30
WINDOWWIDTH = 800
WINDOWHEIGHT = 1000
REVEALSPEED = 8
BOXSIZE = 120
BOARDDIMENSIONS = 5
XMARGIN = 100
YMARGIN = 200

ORANGE = (255, 165, 0)
PURPLE = (160, 32, 240)
BLUE = (0, 0, 139)
TURQUOISE = (48, 213, 200)
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
BOARD_COLOR = (240, 240, 240)
```

```

colors = {
    "orange": ORANGE,
    "purple": PURPLE,
    "turquoise": TURQUOISE,
    "blue": BLUE,
}

```

```

def getRandomFont():
    return pygame.font.SysFont("arial", size=30, bold=True)

```

Оваа функција ќе ми враќа соодветен фонт со големина кој ќе го користам за испишување на лабелите на екран.

```


def terminate():
    pygame.quit()
    sys.exit()

```

```

def checkForQuit():
    for event in pygame.event.get(QUIT):
        terminate()
    for event in pygame.event.get(KEYUP):
        if event.key == K_ESCAPE:
            terminate()
        pygame.event.post(event)

```

Две функции кои соодветно проверуваат доколку е посегнато по некаква акција која води до исклучување на играта. Доколку се притисне  копчето за исклучување на прозорот на играта во горниот десен агол или се притисне копчето *Esc*, соодветно ќе се терминира и ќе се исклучи играта.

```

def makeText(text, color1, color2, top, left):
    textSurf = BASICFONT.render(text, True, color1, color2)
    textRect = textSurf.get_rect()
    textRect.topleft = (top, left)
    return textSurf, textRect

```

Оваа функција ја користам за да можам да креирам текст лабела која сакам да ја прикажам на екранот. Се креира подлога и текст кој се наоѓа во подлогата. Таа соодветно е голема колку пораката што е испишана, а почнува од (top, left) пикселот и се протега во насока десно-надолу.

```

def getLeftTopOfTile(tileX, tileY):
    left = XMARGIN + (tileX * BOXSIZE) + (tileX - 1)
    top = YMARGIN + (tileY * BOXSIZE) + (tileY - 1)
    return left, top

def getSpotClicked(board, x, y):
    for tileX in range(len(board)):
        for tileY in range(len(board[0])):
            left, top = getLeftTopOfTile(tileX, tileY)
            tileRect = pygame.Rect(left, top, BOXSIZE, BOXSIZE)
            if tileRect.collidepoint(x, y):
                return tileX, tileY
    return None, None

```

Овие две функции допринесуваат за тоа кое поле е обоено на мапата (доколку е соодветно избрано). Доколку ќе врати координати (None, None) што дава за надознание дека не е соодветно избрано поле од мапата.

```

def drawTile(tilex, tiley, color, adjx=0, adjy=0):
    left, top = getLeftTopOfTile(tilex, tiley)
    pygame.draw.rect(DISPLAYSURF, color, (left + adjx, top + adjy,
BOXSIZE, BOXSIZE))
    pygame.draw.rect(DISPLAYSURF, BLACK, (left + adjx, top + adjy,
BOXSIZE, BOXSIZE), 2)

```

Оваа функција се користи за исцртување на една коцка (поле) на мапата.

```

def drawPalette():
    pygame.draw.circle(DISPLAYSURF, ORANGE, (150, 100), 50, 0)
    pygame.draw.circle(DISPLAYSURF, PURPLE, (300, 100), 50, 0)
    pygame.draw.circle(DISPLAYSURF, TURQUOISE, (450, 100), 50, 0)
    pygame.draw.circle(DISPLAYSURF, BLUE, (600, 100), 50, 0)

    pygame.draw.circle(DISPLAYSURF, BLACK, (150, 100), 50, 3)
    pygame.draw.circle(DISPLAYSURF, BLACK, (300, 100), 50, 3)
    pygame.draw.circle(DISPLAYSURF, BLACK, (450, 100), 50, 3)
    pygame.draw.circle(DISPLAYSURF, BLACK, (600, 100), 50, 3)

```

Оваа функција ја користиме за да ги исцртаме соодветно боите во вид на палета во горниот дел од екранот, од каде со клик на некоја од нив, се селектира соодветна боја и доколку се кликне некое од полињата во мапата, ако е возможно ќе го обои во таква боја.

```

def check_valid_move(board, tileX, tileY, selected_color):
    actions = ((0, -1), (0, +1), (1, 0), (-1, 0))
    for action in actions:
        x = tileX + action[0]
        y = tileY + action[1]
        if 0 <= x < len(board) and 0 <= y < len(board[0]):
            if board[x][y] == selected_color:
                return False
    return True

```

Функција која одредува дали некоја акција е дозволена, односно доколку во некое од соседните полиња на притиснатото поле за боење, нема поле обоено со селектираната боја, ќе дозволи обојување на полето во таа боја, во спротивно нема да дозволи.

```

def drawBoard(board):
    DISPLAYSURF.fill(BOARD_COLOR)
    for tilex in range(len(board)):
        for tiley in range(len(board[0])):
            color = board[tilex][tiley] if board[tilex][tiley] else
WHITE
            drawTile(tilex, tiley, color)

    left, top = getLeftTopOfTile(0, 0)
    width = 600
    height = 600
    pygame.draw.rect(DISPLAYSURF, BLACK, (left - 5, top - 5, width +
11, height + 11), 4)

    DISPLAYSURF.blit(RESET_SURF, RESET_RECT)
    DISPLAYSURF.blit(SOLVE_SURF, SOLVE_RECT)

```

Со оваа функција се исцртува мапата со соодветните метрики и се исцртуваат копчињата за ресетирање на таблата и за решавање на мапата. Тие функционалности се репрезентирани во последните 2 реда од функцијата.

```

def checkIfWin(board):
    for tilex in range(len(board)):
        for tiley in range(len(board[0])):
            if board[tilex][tiley] == "":
                return False
    return True

```

```

def fadeOutBoard(board):
    fade_surface = pygame.Surface((WINDOWWIDTH, WINDOWHEIGHT))
    fade_surface.fill(BOARD_COLOR)
    fade_surface.set_alpha(0)

    alpha = 0
    while alpha <= 255:
        checkForQuit()

        drawBoard(board)
        displayMessage("Congratulations, you have won the game!")

        fade_surface.set_alpha(alpha)
        DISPLAYSURF.blit(fade_surface, (0, 0))

        pygame.display.update()
        FPSLOCK.tick(FPS)
        alpha += 5

    DISPLAYSURF.fill(BOARD_COLOR)
    pygame.display.update()

    pygame.time.wait(1000)

```

Функцијата `checkIfWin` гледа доколку не постои необоено поле на мапата, а во текот на боењето се запазува правилото за боење, тоа значи дека мапата е обоена и прогласува победа на играчот. Додека пак функцијата `fadeOutBoard` се активира како анимација решената мапа полека да ја сними од екранот и да се појави нова мапа за повторно решавање.

```

def displayMessage(message):
    textSurf = BASICFONT.render(message, True, BLUE, BOARD_COLOR)
    textRect = textSurf.get_rect()
    textRect.topleft = (175, 50)
    DISPLAYSURF.blit(textSurf, textRect)

```

Ова е функција која е напишана со цел да нема повторување на ист код, за да се прикажува во неколку наврати некаква порака на екранот.

```

def autoSolve(colors):
    board = [["" for _ in range(BOARDDIMENSIONS)] for _ in
range(BOARDDIMENSIONS)]
    color_list = list(colors)
    random.shuffle(color_list)
    for tileX in range(len(board)):
        for tileY in range(len(board[0])):
            if board[tileX][tileY] == "":
                for color in color_list:
                    if check_valid_move(board, tileX, tileY, color):
                        board[tileX][tileY] = color
                        break
    drawBoard(board)
    displayMessage("This is a representation of a solved board")
    pygame.display.update()
    pygame.time.wait(3000)
    return [["" for _ in range(BOARDDIMENSIONS)] for _ in
range(BOARDDIMENSIONS)]

```

Оваа функција ќе се употреби само доколку се повика акцијата (се притисне копчето Solve) за решавање на мапата. Тука се дава можно решение за како би можела мапата и е доста едноставно навидум, со употреба на само 2 бои.

Главна е функцијата

```

def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, RESET_SURF, RESET_RECT,
NEW_SURF, NEW_RECT, SOLVE_SURF, SOLVE_RECT

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Color Fill Puzzle')

    BASICFONT = getRandomFont()

    RESET_SURF, RESET_RECT = makeText('Reset', WHITE, ORANGE,
WINDOWWIDTH - 180, WINDOWHEIGHT - 50)
    SOLVE_SURF, SOLVE_RECT = makeText('Solve', WHITE, BLUE,
WINDOWWIDTH - 180, WINDOWHEIGHT - 90)

    board = [["" for _ in range(BOARDDIMENSIONS)] for _ in
range(BOARDDIMENSIONS)]

```

```

selected_color = None

while True:
    if checkIfWin(board):
        fadeOutBoard(board)
        board = [["" for _ in range(BOARDDIMENSIONS)] for _ in
range(BOARDDIMENSIONS)]
        selected_color = None
        continue

    checkForQuit()
    drawBoard(board)
    drawPalette()

    for event in pygame.event.get():
        if event.type == MOUSEBUTTONUP:
            mouseX, mouseY = event.pos
            if RESET_RECT.collidepoint(mouseX, mouseY):
                print("Reset button clicked!")
                board = [["" for _ in range(BOARDDIMENSIONS)] for _
in range(BOARDDIMENSIONS)]
                selected_color = None
                continue
            elif SOLVE_RECT.collidepoint(mouseX, mouseY):
                print("Solving board automatically!")
                board = autoSolve(colors.keys())
                selected_color = None
                continue
            else:
                if pygame.Rect(100, 50, 100,
100).collidepoint(mouseX, mouseY):
                    selected_color = ORANGE
                elif pygame.Rect(250, 50, 100,
100).collidepoint(mouseX, mouseY):
                    selected_color = PURPLE
                elif pygame.Rect(400, 50, 100,
100).collidepoint(mouseX, mouseY):
                    selected_color = TURQUOISE
                elif pygame.Rect(550, 50, 100,
100).collidepoint(mouseX, mouseY):
                    selected_color = BLUE

```

```

        tileX, tileY = getSpotClicked(board, mouseX,
mouseY)
        if tileX is not None and tileY is not None and
selected_color and check_valid_move(board, tileX,
tileY,
selected_color):
            board[tileX][tileY] = selected_color

pygame.display.update()
FPSLOCK.tick(FPS)

```

Функцијата *main()* е главната логика на играта **Color Fill Puzzle** и ја контролира целокупната интеракција и извршување на програмата.

Иницијализација на Pygame

Се иницијализираат *Pygame* модулите, поставува часовникот за *FPS* и го креира главниот прозорец со соодветните пропорции. Се поставува фонотот за текстови и се креираат копчињата за **Reset** и **Solve** на прозорецот.

Креирање на празна табла

Се креира матрица *board* со димензии *BOARDDIMENSIONS x BOARDDIMENSIONS* (во нашиот случај е 5x5), каде што сите полиња се празни (означени со ""). Променливата *selected_color* служи за чување на бојата која моментално е избрана од корисникот.

Почеток на играта

Почнува бесконечниот циклус каде ќе се одвива се додека играта трае.

Проверка за победа – доколку целата мапа е пополнета (без празни полиња), се прикажува порака за победа и се ресетира мапа.

Ракување со настани – се обработуваат сите кориснички настани, вклучувајќи кликови со глумчето и тастатура.

1. Ако се кликне на **Reset** копчето, мапата се ресетира, односно сите полиња се повторно непополнети;
2. Ако се кликне на **Solve** копчето, се пополнува мапата со автоматски алгоритам за решавање;
3. Ако корисникот избере боја од палетата и потоа кликне на мапата, бојата се пополнува во селектираната позиција само ако потегот е валиден (без соседни полиња со иста боја);

Прикажување на таблата и палетата – мапата и боите за избор постојано се обновуваат на екранот

Ажурирање на екранот

По секоја интеракција, екранот се ажурира и играта чекори во рамките на дефинираните *FPS*.

Оваа функција соодветно ги оркестрира сите елементи на играта, користејќи ги веќе претходно креираните функции.

Изгледот на мапата:

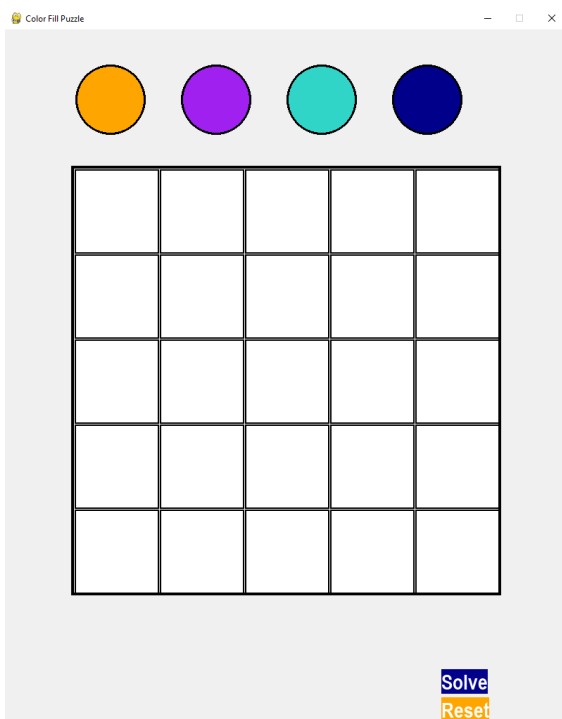


Figure 1 Мапата пред да е решена

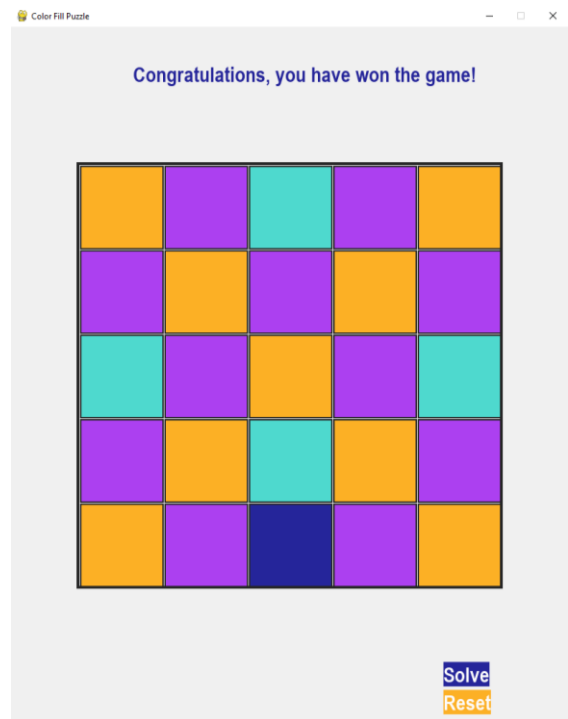


Figure 2 Мапата од кога е решена

Изработил: Леон Асановски 221007