ФИО: Юлдошев Диёрбек Шералиевич
Номер студенческого билета: 245106@edu.fa.ru
Номер лабораторной работы: Лабораторная работа №3
Номер группы:ИД24-2
Дата сдачи:25/11/2025
Название назначенной задачи:Сложная анимация

2- Генератор лабиринтов (166)

Реализуйте алгоритм генерации лабиринтов через поиск в ширину.
1.Реализуйте разные размеры лабиринтов.

2.Реализуйте экспорт получившихся лабиринтов во внешний файл.

1. Цель работы

Основная цель этой лабораторной работы применить ваши навыки работы с графикой и анимацией к более сложным проектам. Каждая задача требует тщательного **проектирования и планирования программы** перед реализацией. Вам потребуется исследовать математические концепции и создавать хорошо структурированный, профессиональный код.

---

2. Порядок выполнения

- **Индивидуальная работа:** Это **индивидуальное задание**. Задачи будут случайным образом распределены преподавателем.
- **Консультации:** На семинарах вы можете попросить преподавателя пояснить требования или математические концепции.
- **Творческий подход:** Каждая задача предоставляет только общую основу. Вы должны принимать дизайнерские решения относительно:
    - Скорости и плавности анимации
    - Размеров элементов и цветовых схем
    - Визуального стиля и пользовательского опыта
    - Преподаватель будет оценивать ваши дизайнерские решения — удачные решения повысят оценку, неудачные, снизят.
- **Исследовательский компонент:** Многие задачи требуют понимания математических концепций. Исследование, необходимая часть успешного выполнения этих проектов.

---

3. Требования к качеству кода

- **Требуется ООП:** Вы должны использовать принципы объектно-ориентированного программирования для организации кода.
- **Качество кода важно:** Ваш код должен соответствовать стилевым соглашениям, содержать комментарии и быть читаемым.
- **Рекомендуется система контроля версий:** Использование Git или подобных систем приветствуется и будет отмечено.
- **Управление настройками:** Используйте конфигурационные файлы (JSON/YAML) для хранения настроек, таких как размер экрана, цвета и скорости.

Пример *config.json*:

```
json
{
  "window_width": 800,
  "window_height": 600,
  "animation_speed": 60,
  "background_color": [0, 0, 0],
  "gravity_strength": 9.8
}
```

---

4. Дополнительные задания (по желанию)

Каждая задача включает пронумерованные дополнительные требования. Вы также можете предложить собственные расширения. Дополнительная функциональность будет оцениваться на основе сложности и качества реализации.

Дополнительные задания часто включают:

- Обработку событий
- Элементы пользовательского интерфейса
- Взаимодействие с мышью и клавиатурой
- Ввод/вывод файлов

Эти навыки важны для будущих заданий.

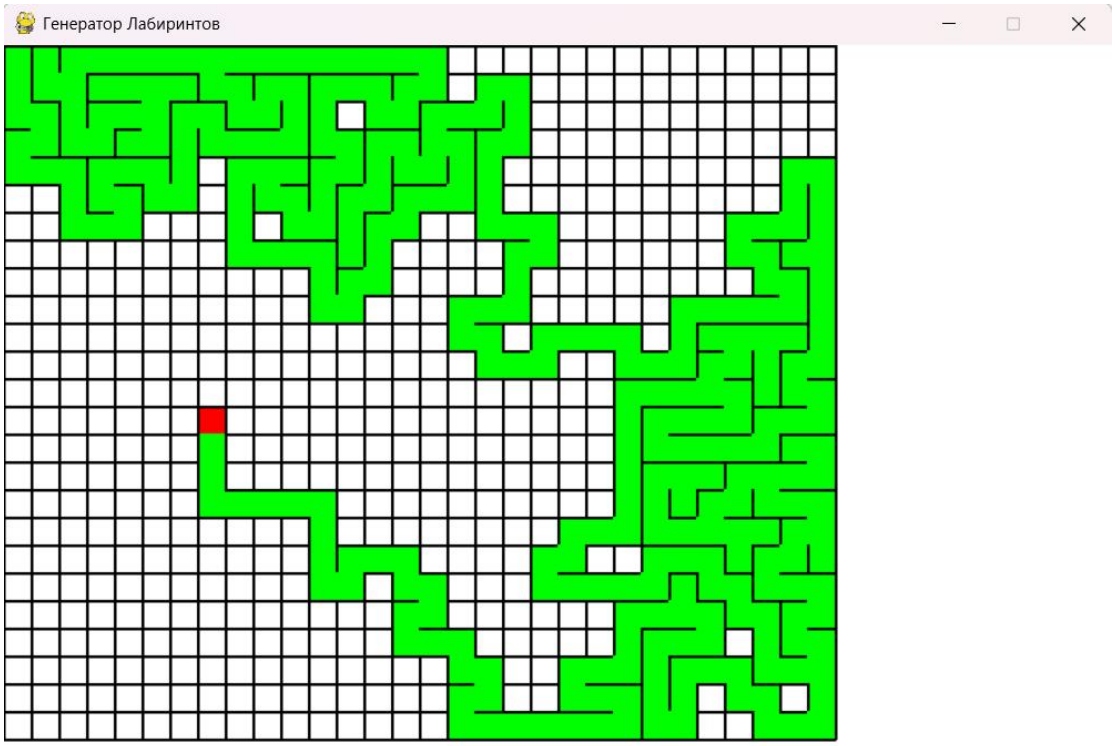
```
 import pygame
import random
import sys
import json

# Загрузка конфигурации из файла
with open('config.json', 'r') as f:
    config = json.load(f)

# Цвета из конфига
BG_COLOR = tuple(config["background_color"])
CURR_COLOR = tuple(config["current_cell_color"])
VISITED_COLOR = tuple(config["visited_cell_color"])
DEFAULT_COLOR = tuple(config["default_cell_color"])
WALL_COLOR = tuple(config["wall_color"])

# Настройки окна и лабиринта
WIDTH, HEIGHT = config["window_width"], config["window_height"]
CELL_SIZE = config["cell_size"]
MAZE_WIDTH, MAZE_HEIGHT = config["maze_width"], config["maze_height"]
FPS = config["animation_speed"]

# Инициализация Pygame
pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Генератор Лабиринтов")
clock = pygame.time.Clock()

class Cell:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

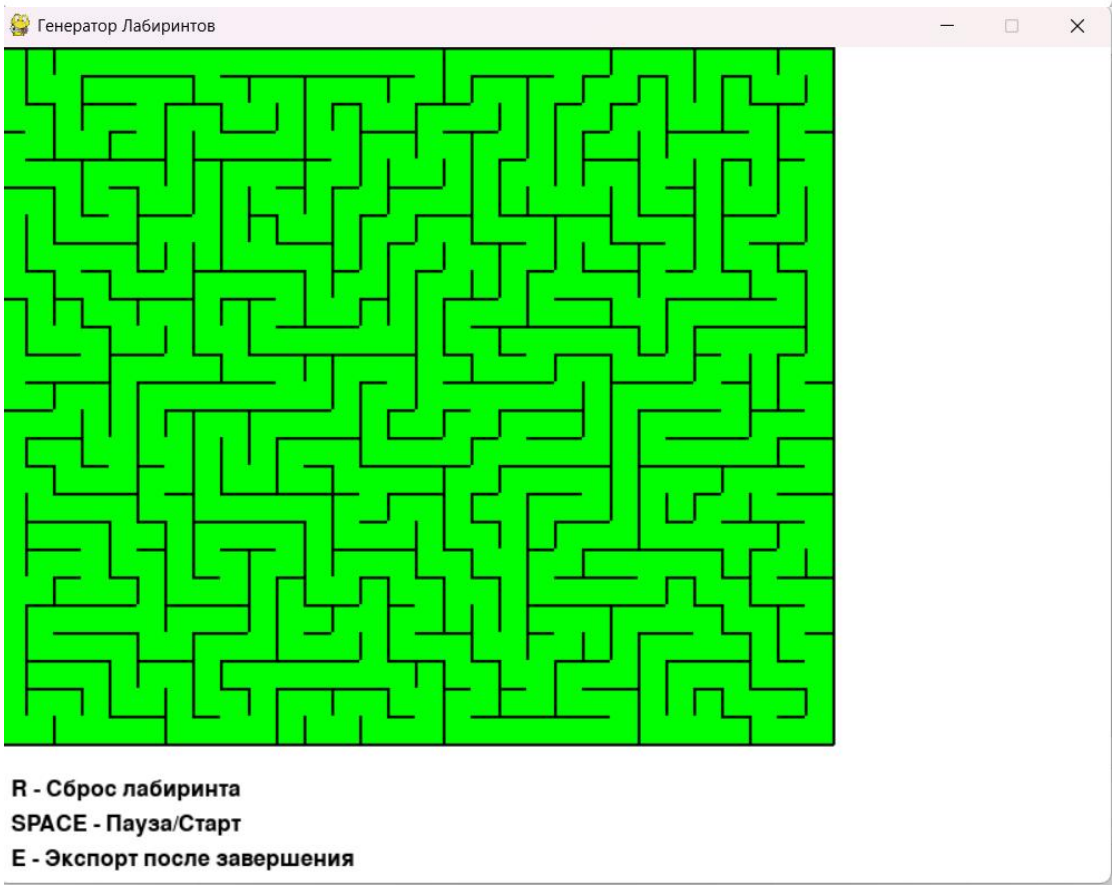
```
        self.walls = {'top': True, 'right': True, 'bottom': True, 'left': True}
        self.visited = False
        self.current = False

    def draw(self):
        x = self.x * CELL_SIZE
        y = self.y * CELL_SIZE

        # Выбираем цвет клетки
        if self.current:
            color = CURR_COLOR
        elif self.visited:
            color = VISITED_COLOR
        else:
            color = DEFAULT_COLOR

        pygame.draw.rect(screen, color, (x, y, CELL_SIZE, CELL_SIZE))

        # Рисуем стенки
        if self.walls['top']:
            pygame.draw.line(screen, WALL_COLOR, (x, y), (x + CELL_SIZE, y), 2)
        if self.walls['right']:
            pygame.draw.line(screen, WALL_COLOR, (x + CELL_SIZE, y), (x + CELL_SIZE, y + CELL_SIZE), 2)
        if self.walls['bottom']:
            pygame.draw.line(screen, WALL_COLOR, (x, y + CELL_SIZE), (x + CELL_SIZE, y + CELL_SIZE), 2)
        if self.walls['left']:
            pygame.draw.line(screen, WALL_COLOR, (x, y), (x, y + CELL_SIZE), 2)

class Maze:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self.grid = [[Cell(x, y) for y in range(height)] for x in range(width)]
        self.stack = []
        self.current_cell = self.grid[0][0]
        self.current_cell.visited = True
        self.current_cell.current = True
        self.generation_complete = False

    def get_unvisited_neighbors(self, cell):
        neighbors = []
        directions = [('left', -1, 0), ('right', 1, 0), ('top', 0, -1), ('bottom', 0, 1)]
        for direction, dx, dy in directions:
            nx, ny = cell.x + dx, cell.y + dy
            if 0 <= nx < self.width and 0 <= ny < self.height:
                neighbor = self.grid[nx][ny]
                if not neighbor.visited:
                    neighbors.append((direction, neighbor))
        return neighbors

    def remove_wall(self, current, next_cell, direction):
        if direction == 'left':
            current.walls['left'] = False
            next_cell.walls['right'] = False
        elif direction == 'right':
            current.walls['right'] = False
            next_cell.walls['left'] = False
```

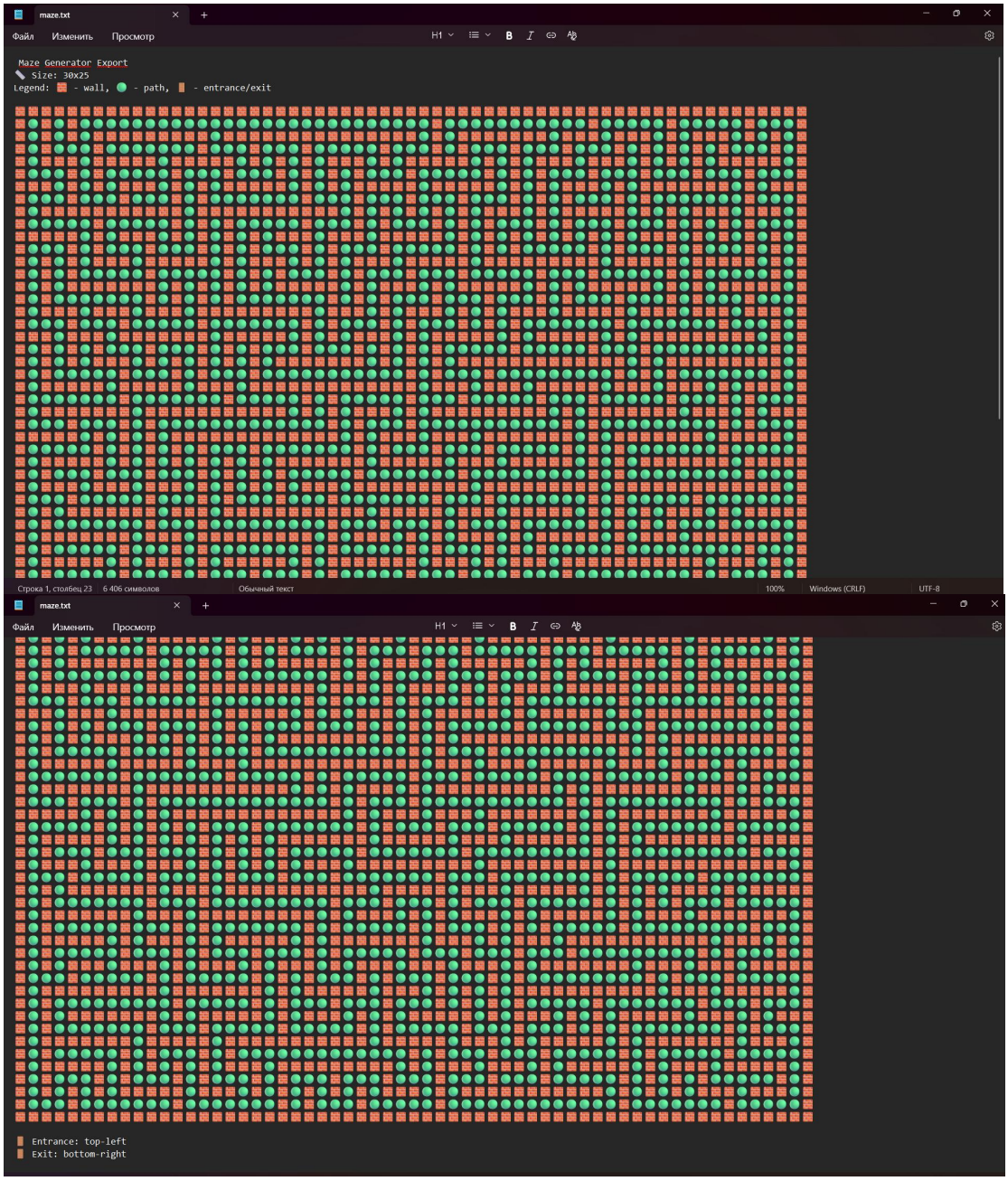
```
        elif direction == 'top':
            current.walls['top'] = False
            next_cell.walls['bottom'] = False
        elif direction == 'bottom':
            current.walls['bottom'] = False
            next_cell.walls['top'] = False

    def generate_step(self):
        if self.generation_complete:
            return

        neighbors = self.get_unvisited_neighbors(self.current_cell)
        if neighbors:
            self.stack.append(self.current_cell)
            direction, next_cell = random.choice(neighbors)
            self.remove_wall(self.current_cell, next_cell, direction)
            self.current_cell.current = False
            next_cell.visited = True
            next_cell.current = True
            self.current_cell = next_cell
        elif self.stack:
            self.current_cell.current = False
            self.current_cell = self.stack.pop()
            self.current_cell.current = True
        else:
            self.current_cell.current = False
            self.generation_complete = True

    def draw(self):
        for row in self.grid:
            for cell in row:
                cell.draw()

    def export_to_file(self, filename="maze_export.txt"):
        with open(filename, 'w', encoding='utf-8') as f:
            f.write(" Maze Generator Export\n")
            f.write(f"   Size: {self.width}x{self.height}\n")
            f.write("Legend:    - wall,    - path,    - entrance/exit\n\n")

        # Верхняя граница
            f.write('  ' * (self.width * 2 + 1) + '\n')

            for y in range(self.height):
            # Вертикальные стенки
                line = '  '
                for x in range(self.width):
                    cell = self.grid[x][y]
                    line += '  ' # проход
                    if cell.walls['right']:
                        line += '  '
                    else:
                        line += '  '
                f.write(line + '\n')

            # Горизонтальные стенки
                line = '  '
                for x in range(self.width):
```

```
                cell = self.grid[x][y]
                if cell.walls['bottom']:
                    line += '    '
                else:
                    line += '    '
            f.write(line + '\n')

        # Отмечаем вход и выход
        f.write("\n   Entrance: top-left")
        f.write("\n   Exit: bottom-right\n")

class Application:
    def __init__(self):
        self.maze = Maze(MAZE_WIDTH, MAZE_HEIGHT)
        self.paused = False

    def run(self):
        running = True
        font = pygame.font.SysFont(None, 24)

        while running:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    running = False

                elif event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_r:
                        # Сброс лабиринта
                        self.maze = Maze(MAZE_WIDTH, MAZE_HEIGHT)
                        self.paused = False
                    elif event.key == pygame.K_SPACE:
                        # Пауза/продолжение генерации
                        self.paused = not self.paused
                    elif event.key == pygame.K_e and self.maze.generation_complete:
                        self.maze.export_to_file("maze.txt")
                        print("Лабиринт экспортирован в maze.txt")

            if not self.paused and not self.maze.generation_complete:
                self.maze.generate_step()

            screen.fill(BG_COLOR)
            self.maze.draw()

            # Отображение инструкций
            screen.blit(font.render("R - Сброс лабиринта", True, WALL_COLOR), (10, HEIGHT - 75))
            screen.blit(font.render("SPACE - Пауза/Старт", True, WALL_COLOR), (10, HEIGHT - 50))
            screen.blit(font.render("E - Экспорт после завершения", True, WALL_COLOR), (10, HEIGHT -
25))

            pygame.display.flip()
            clock.tick(FPS)

        pygame.quit()
        sys.exit()

if __name__ == "__main__":
    app = Application()
```

app.run()


Генератор Лабиринтов
R - Сброс лабиринта
SPACE - Пауза/Старт
E - Экспорт после завершения


Генератор Лабиринтов
R - Сброс лабиринта
SPACE - Пауза/Старт
E - Экспорт после завершения


maze.txt
Файл
Изменить
Просмотр
H1
Maze Generator Export
Size: 30x25
Legend: - wall, - path, - entrance/exit
Строка 1, столбец 23
6 406 символов
Обычный текст
100%
Windows (CRLF)
UTF-8
maze.txt
Файл
Изменить
Просмотр
H1
Entrance: top-left
Exit: bottom-right