

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Тема: Алгоритм A*

Студенты:

Зейферт Александр

Леонов Богдан

Предмет: Практикум по программированию

Группа: ИД24-1

Дата сдачи: 25.11.2025

Название задачи:

Алгоритм A*

Описание задания

Цель лабораторной работы — разработать приложение, визуализирующее работу алгоритмов поиска пути на сетке. Основной алгоритм — A*, при этом необходимо реализовать и другие классические алгоритмы поиска.

Задача включает:

1. Реализацию алгоритма A*.
 2. Подсветку посещённых клеток, текущей клетки и итогового пути.
 3. Реализацию алгоритмов BFS и Dijkstra.
 4. Создание GUI на PyQt6.
 5. Импорт лабиринтов из YAML-файлов.
 6. Управление симуляцией: старт, пауза, шаг, сброс.
 7. Слайдер регулировки скорости.
 8. ООП-архитектуру и модульность проекта.
-

Достигнутая сложность и реализация

Выполненные основные требования

- Полная реализация алгоритма A*.
- Дополнительно — алгоритмы BFS и Dijkstra.
- Пошаговая и покадровая визуализация поиска.

- Графический интерфейс на PyQt6.
- Импорт YAML-лабиринтов.
- Управление симуляцией через GUI.
- Структура кода построена на принципах ООП.

Выполненная дополнительная функциональность

- Генерация случайных лабиринтов.
- Менеджер лабиринтов: обновление списка, удаление.
- Автоматическое обновление отображения.
- Возможность визуализации больших карт.

Оригинальные расширения

- Полностью параметризованная визуализация.
- Чёткое разделение между логикой и интерфейсом.
- Унифицированный механизм импорта YAML с проверкой корректности.

Объяснение проектирования программы

Структура ООП: созданные классы и их обязанности

1. Maze

Хранит структуру лабиринта: сетку, размеры, точки старта и финиша.

2. Алгоритмы (A*, BFS, Dijkstra)

Каждый алгоритм — отдельный модуль. Все они имеют унифицированный интерфейс:

- принимают старт, финиш, сетку;
- возвращают путь и посещённые клетки.

3. MazeView

Рендер сетки и визуализация работы алгоритма.

4. ControlsPanel

Все элементы управления:

- выбор лабиринта,
- запуск алгоритмов,
- генерация,

- управление симуляцией,
- слайдер скорости.

Генерирует сигналы, вызываемые в MainWindow.

6. MainWindow

Главный класс, собирающий интерфейс и управляющий данными через подключённые сигналы.

Основная часть (по требованиям)

Ниже приведено описание выполнения основных требований с кодом.

1. Требование: Реализовать алгоритм A*

Решение:

Алгоритм реализован в astar.py через приоритетную очередь и эвристику Чебышёва.

Фрагмент кода:

```
while open_set:
```

```
    _, current = heapq.heappop(open_set)
```

```
    if current == goal:
```

```
        return reconstruct_path(came_from, current), visited
```

```
for neighbor in neighbors(current):
```

```
    tentative_g = g[current] + 1
```

```
    if neighbor not in g or tentative_g < g[neighbor]:
```

```
        g[neighbor] = tentative_g
```

```
        f[neighbor] = tentative_g + heuristic(neighbor, goal)
```

```
        heapq.heappush(open_set, (f[neighbor], neighbor))
```

2. Требование: Подсветить посещённые и текущие клетки

Решение:

В MazeView при отрисовке проверяется состояние клетки.

Фрагмент кода:

```
if (x, y) in self.visited:

    painter.fillRect(rect, self.colors['visited'])

elif (x, y) == self.current:

    painter.fillRect(rect, self.colors['active'])
```

3. Требование: Импорт YAML-файлов**Фрагмент кода:**

```
file_path, _ = QFileDialog.getOpenFileName(

    self, "Импорт лабиринта", "", "YAML Files (*.yaml)"

)

with open(file_path, 'r') as f:

    data = yaml.safe_load(f)
```

4. Требование: Управление симуляцией**Решение:**

Система сигналов/слотов Qt в ControlsPanel.

Фрагмент:

```
self.btn_start.clicked.connect(self.start_requested)

self.btn_pause.clicked.connect(self.pause_requested)

self.btn_step.clicked.connect(self.step_requested)

self.btn_reset.clicked.connect(self.reset_requested)
```

5. Требование: Скорость анимации через слайдер**Фрагмент:**

```
delay = self.slider_speed.value() / 1000.0

self.speed_changed.emit(delay)
```

Полный исходный код

Исходный код проекта доступен в открытом репозитории:

<https://github.com/lsnob/A->