**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

По лабораторному практикуму

Тема: Поиск пути в лабиринте с использованием алгоритма A\*

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6382 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Вайгачев А.О |
| гр. 6382 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Воропаев А.О. |
| гр. 6382  Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Жук К.А.  Фирсов М.А. |

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент Вайгачев А.О. группы 6382 | | |
| Студент Воропаев А.О. группы 6382 | | |
| Студент Жук К.А. группы 6382  Тема практики: Поиск пути в лабиринте с использованием алгоритма А\* | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.  Алгоритм: поиск пути в лабиринте методом А\*. | | |
| Сроки прохождения практики: 27.06.2018 – 10.07.2018 | | |
| Дата сдачи отчета: 06.07.2018 | | |
| Дата защиты отчета: 06.07.2018 | | |
|  | | |
| Студент гр.6382 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Вайгачев А.О. |
| Студент гр. 6382 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Воропаев А.О. |
| Студент гр. 6382 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Жук К.А. |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Фирсов М.А. |

**Аннотация**

В данной работе рассмотрена программа, которая с помощью алгоритма А\*, ищет минимальный путь от начальной точки до конечной в лабиринте. Программа разработана в среде IntelliJ IDEA. Язык разработанной программы Java. Программа подробно показывает процесс прохождения лабиринта от начальной точки до конечной.

**Summary**

In this work, we consider a program that uses the algorithm A \* to find the minimal path from the initial point to the final one in the labyrinth. The program is developed in the IntelliJ IDEA environment. The language of the development of program is Java. The program details the process of passing the labyrinth from the initial point to the final.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Требования к программе | 6 |
| 2. | План разработки и распределение ролей в бригаде | 6 |
| 2.1. | План разработки | 6 |
| 2.2. | Распределение ролей в бригаде | 6 |
| 3. | Особенности реализации | 6 |
| 3.1. | Описание классов | 6 |
| 3.2. | Алгоритм работы | 8 |
| 3.3  3.4 | Входные/выходные данные  Описание интерфейса | 9  9 |
| 4. | Тестирование | 11 |
| 4.1 | Тестирование графического интерфейса | 11 |
| 4.2 | Тестирование кода алгоритма | 11 |
|  | Заключение | 15 |
|  | Список использованных источников | 15 |
|  | Приложение А. Код программы. | 16 |

**Введение.**

**Цель учебной практики.** Получение основных сведений о языке программирования Java.

Для выполнения указанной цели поставлены следующие задачи:

* 1. Изучение литературы по Java.
  2. Изучение библиотек Java, позволяющих реализовать графический интерфейс.
  3. Написание программы, демонстрирующий алгоритм А\*.

**A\*** — алгоритм поиска, который пошагово просматривает все пути, ведущие от начальной точки в конечную, пока не найдёт минимальный.

В начале работы просматриваются точки, соседние с начальным. На каждом этапе алгоритм оперирует с множеством путей из начальной точки до всех ещё не раскрытых соседей клетки — множеством частных решений, которое размещается в очереди с приоритетом.

Приоритет пути определяется по значению f(x) = g(x) + h(x), где f(x) – предполагаемая стоимость пути от текущей точки до конца, g(x) – стоимость кратчайшего пути от начала до текущей точки, h(x) – эвристическая функция. Если текущее частное решение дает лучше значение g(x), то меняем значение g(x) для текущей клетки и выписываем новый кратчайший путь до точки.

Алгоритм продолжает свою работу до тех пор, пока значение f(x) конечной точки не окажется меньшим, чем любое значение в очереди, либо пока все клетки не будут просмотрены. В итоге получается путь от начальной точки до конечной с наименьшей стоимостью.

1. **Требование к программе**

Данная программа должна корректно работать на всевозможных входных данных и выводить минимальный путь от начальной точки до конечной. В случае, если пути не существует, уведомлять об этом пользователя.

Программа должна подробно демонстрировать найденный путь и наглядно показывать его.

1. **Исходные требования к программе**
   * 1. **Требования к вводу исходных данных**

Доступно 2 способа ввода:

1. Ввод из файла
2. Выбор ключевых точек с помощью мыши.

В случае ввода из файла:

Данные представлены в файлах “level1.dat”,“level2.dat”, и т.д. Из файла считываются 2 числа по которым устанавливается размер поля, затем само поле, где символ “.” означает что сектор свободен, а “w”, что в секторе находится препятствие.

После поля в файле находятся 4 целых числа, первые 2 обозначают координаты начальной точки, остальные – конечной.

В случае выбора ключевых точек с помощью мыши:

Нужно отметить стартовую и конечную точки, а также заблокированные сектора.

Также доступно опция выбора эвристической функции.

Предложенные эвристические функции на основе которых будет выполняться алгоритм А\*:

H(x) = |goal.x – current.x| + |goal.y – current.y

H(x) = sqrt((goal.x – current.x)^2+(goal.y - current.y)^2)

Где goal - конечная точка, а current - начальная.

* + 1. **Требования к визуализации**

На экран выводится поле соответствующего размера. Сектора поля будут раскрашены в различные цвета в зависимости от их состояния (старт – черный, конец – оранжевый, препятствие – синий, активные – зеленый, возможное продолжение пути – желтый, путь – красный, закрытая - серый). В процессе работы сектора будут перекрашиваться. Также есть дополнительное поле, содержащее комментарии.

Список возможных комментариев:

1) Путь проходит через клетку ( координаты клетки)  
2) Путь начался из клетки (координаты клетки)  
3) Добавляем стартовую клетку в OpenSet  
4) Выбираем клетку (координаты клетки) из OpenSet, так как значение функции f(current) для нее минимально  
5) Восстанавливаем путь:  
6) Удаляем клетку ( координаты клетки) из OpenSet и добавляем в ClosedSet  
7) Рассматриваем всех Unclosed соседей для клетки ( координаты клетки )  
8) Рассматриваем клетку ( координаты клетки)  
9) Промежуточное значение функции g для этой точки = (значение функции g)  
10) Так как клетка (координаты клетки) не находится в OpenSet, то устанавливаем, что мы пришли в эту клетку из клетки ( координаты клетки)  
11) Так как значение функции g для клетки (координаты клетки) меньше промежуточного, то устанавливаем, что мы пришли в эту клетку из клетки ( координаты клетки)  
12) Устанавливаем значение функции g от нее равным промежуточному значению. Теперь функция g от клетки (координаты клетки) равна (значение функции g)  
13) Устанавливаем значение функции f для этой клетки, как сумму значений функции g и эвристической функции. Теперь значение функции f от клетки (координаты клетки) равно (значение функции f)  
14) Добавляем клетку (координаты клетки) в OpenSet  
15) Нет пути от клетки (координаты стартовой клетки) до клетки (координаты конечной клетки)

1. Требования к выходным данным

При запуске проекта будет создаваться отдельное окно (рисунок 1). Данная программа будет иметь 2 окна. Окна представляет собой меню, карту и комментарии. Сверху располагаются кнопки, внизу находится окно вывода лабиринта, справа окно вывода комментариев. Будут доступны следующие кнопки:

1)Restart - начать программу сначала.

2)Start - начать выполнение программы.

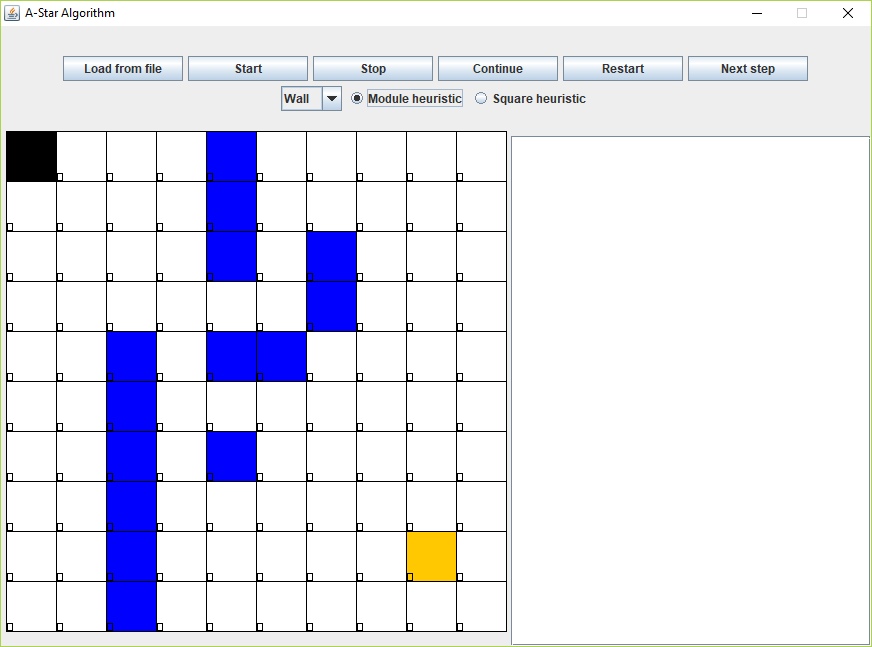
3)Next step - следующий шаг алгоритма (при работе в пошаговом режиме).

4)Stop - остановить программу на текущем шаге.

5)Continue - продолжить работу программы.

6) Load from file – загрузить данные из файла

Также доступен переключатель между эвристическими функциями.



1. **План разработки и распределение ролей в бригаде**
   1. **План разработки:**

02.07.18 Согласована спецификация и план разработки.

04.07.18 Прототип.

06.07.18 Первая версия. Предоставление проекта. Проект пояснительной записки. Будет доступно автоматическое выполнение алгоритма и поле с комментариями.

08.07.18 2-я версия. Будет добавлена возможность пошаговой реализации алгоритма. Приемочные тесты

10.07.18 Конечный проект. Готовая для сдачи версия и пояснительная записка с исправлениями по замечаниям. Будет добавлена возможность выбора эвристической функции и выбора ключевых точек с помощью мыши.

* 1. **Распределение ролей в бригаде:**

**Воропаев А.О.** Реализация алгоритма A\* на языке Java.

**Вайгачев А.О** Разработка пользовательского интерфейса и визуализация.

**Жук К.А** Оформление пояснительной записки и разработка приемочных тестов и тестирование. Разработка структур данных.

1. **Особенности реализации**
   1. **Структуры используемые в программе:**

Классы, используемые в программе, представлены в таблице 1.

|  |  |
| --- | --- |
| Имя | Назначение |
| Model | Содержит поле и методы, изменяющие поле |
| Controller | Вызывает методы для Model и View для обновления состояния |
| Field | Содержит описание поля |
| View | Отвечает за отображение самого поля и его состояния |
| ControlPanel | Содержит кнопки управления программой |
| GameLauncher | Является точкой входа в программу  и инициализирует model view и controller |
| SwingGraphicsAdapter | адаптер, который соединяет библиотеку swing и класс view |

**3.2 Основные методы**

*Class Model.*

* public double recountHeuristic(Point current) *- метод для пересчета эвристической функции.*
* public boolean isActiveFull() — показывает есть ли в клетки OpenSet
* public void setHeuristical(boolean heuristical) — метод для установки способа расчета эвристической функции.
* public Set<Point> getNotUnActiveNeighbours(Point current) — метод

который возвращает все соседние клетки которые не находятся в ClosedSet

* public static Model load () — метод для загрузки пустого поля
* public static Model load (Scanner scanner) — метод, который загружает поле из файла
* public StringBuilder getComments() — метод для вывода комментариев
* public void setFrom(Point current, Point setter) — метод, который устанавливает, что мы пришли в точку current из setter

Point current, Point setter - текущая точка, точка из которой мы пришли в current

* public Point getFrom(Point current) — возвращает двумерный массив по которому устанавливается итоговый путь
* public void clear() — возвращает поле в начальное состояние
* public Point min\_f() — возвращает из OpenSet точку с минимальным значением f
* public double getFunction\_g(Point current) — возвращает значение функции g для точки current
* public double[][] getFunction\_f() — возвращает функцию f для всех точек поля
* public void setFunction\_g (Point current, double setter) —устанавливает функцию g

Point current, double setter – текущая точка, значение

* public double getFunction\_f(Point current) — возвращает функцию f для текущей точки
* public void setFunction\_f(Point current, double setter) — устанавливает функцию f
* public Field getField() — возвращает поле
* public Point getStart() — возвращает стартовую точку
* public void setStart(Point start) — устанавливает координаты для стартовой точки
* public Point getFinish() — возвращает конечную точку
* public void setFinish(Point finish) — устанавливает координаты для конечной точки
* public double[][] getHeuristic() — возвращает массив значений эвристической функции для всех точек
* public void setWall(Point wall) — устанавливает препятствие в точку
* public void setEmpty(Point cords) — устанавливает сектор свободным
* public boolean isWall(Point temp) — проверяет есть в точке стена

*Class Controller*

* private void addCommentToLog(String appender) — добавляет комментарий в лог
* public void viewUpdated() — перерисовывает состояние поля
* private void restoreWay() — восстанавливает путь от конечной точки до стартовой точки
* private void restoreWay(Point current) —восстанавливает путь от текущей точки до стартовой точки
* private synchronized void implementAstar() — осуществляет алгоритм
* private void clearLog() — очистить поле комментариев
* private void updateLog() — обновить поле комментариев
* private void update() — перерисовывает состояние поля
* private void clearModel()— очищает модель
* private void startTimer() — запускает таймер
* public void stopTimer() — останавливает таймер
* public void resume() — продолжить алгоритм
* public void restart() — перезапустить алгоритм
* public synchronized void next() — следующий шаг алгоритма
* public void load() — загрузить из файла
* public void start() — начать алгоритм
* public void mouseClick(MouseEvent e, String boxContents) — событие после клика мыши
* private void setSafeStop() — прекратить алгоритм
* public void changeAlgorithmOnFirst() — смена эвристической функции
* public void changeAlgorithmOnSecond() — смена эвристической функции

*Class Field*

* public int getNumRows() — возвращает количество строк поля
* public int getNumColumns() — возвращает количество столбцов поля
* public boolean isActive(Point current) - проверяет есть ли клетка в OpenSet

Point current – текущая клетка

* public Sector getSector(int x, int y) - возвращает сектор

int x, int y - координаты

* public void setSectorWall(Point current) — установить в сектор препятствие
* public void setSectorActive(Point current) — установить сектор в OpenSet
* public void setSectorUnActive(Point current) - установить сектор в ClosedSet
* public void setSectorRealWay (Point current) - включает сектор в путь
* public void setSectorCurrent (Point current) — рассматривает клетку как возможное продолжение пути
* public void setSectorFree (Point current) — устанавливает сектор пустым
* public static Field load() — загружает поле
* public void clearField() — очищает поле
* public static Field load(Scanner input) — загружает поле из источника input

*Class View*

* public void setLog(JTextArea log) — устанавливает log

JTextArea log – поле для записи комментариев

* public JTextArea getLog() — возвращает log

JTextArea log – поле для записи комментариев

* public void draw(Model model) – приказывает начать рисовать в Model

Model model - модель

* private void updateCells(Field field) - обновляет размеры клетки

Field field - поле

* private void drawField(Model model) — рисует поле
* public void setGraphics(Graphics2D graphics) — устанавливает то, чем и куда рисовать

*Class ControlPanel*

* private JButton createButton(String text) — метод для создания кнопки
* public String getBoxContents() — возвращает содержимое выплывающего меню
* public void var\_1\_ActionListener(ActionListener listener) - метод для считывания действий пользователя по смене эвристической функции
* public void var\_2\_ActionListener(ActionListener listener) - метод для считывания действий пользователя по смене эвристической функции
* public void loadActionListener(ActionListener listener)- добавляет слушателей в load
* public void startActionListener(ActionListener listener) — добавляет слушателей в start
* public void stopActionListener(ActionListener listener) — добавляет слушателей в stop
* public void resumeActionListener(ActionListener listener) - добавляет слушателей в resume
* public void restartActionListener(ActionListener listener) - добавляет слушателей в restart
* public void nextActionListener(ActionListener listener) — добавляет слушателей в next

*Class GameLaunher*

* public GameLauncher(String title) — точка входа в приложение

String title – заголовок Jframe

* private void initListeners() - инициализирует “слушателей”
* private void toolBarConfig(GridBagConstraints c) - задает размещение toolbar GridBagConstraints c в Jframe
* private void canvasConfig(GridBagConstraints c) — задает размещение canvas GridBagConstraints c в Jframe
* private void logConfig(GridBagConstraints c) — задает размещение log

GridBagConstraints c в Jframe

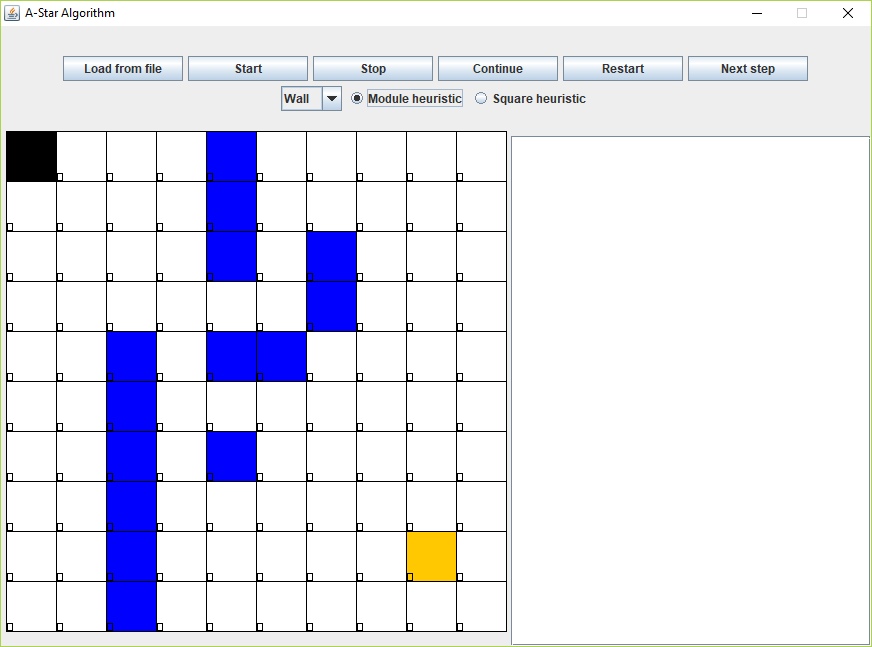
*Class SwingGraphicsAdapter*

* public void drawRect(int x, int y, int width, int height, int rgb) — рисует поле

int x, int y, int width, int height, int rgb – координаты начала рисовки, ширина, высота, цвет

* public void drawText(int x, int y, String text, int rgb) - печатает текст

**3.3. Описание интерфейса:**



Пользовательский интерфейс состоит из:

* Поле лабиринта (пользователь составляет лабиринт вручную или загружает из файла).

Черный квадрат - стартовая точка.

Оранжевый квадрат – конечная точка.

Синий квадрат – препятствие.

При нажатии на квадрат он окрашивается в другой цвет, показывая тем самым стенку в лабиринте. При повторном нажатии стенка убирается.

* Поле комментариев.
* Кнопка Load from file, при нажатии на которую загружается поле из выбранного файла.
* Кнопка Start, при нажатии на которую начинается поиск пути.
* Кнопка Stop, при нажатии на которую происходит остановка поиска пути.
* Кнопка Continue, при нажатии на которую продолжается поиск пути.
* Кнопка Restart, при нажатии на которую поле обновляется полностью.
* Кнопка Next step, при нажатии на которую происходит следующий шаг поиска пути.
* Выбор между двумя эвристическими функциями.
* Выбор чем заполнить клетку, если кликнуть по ней.

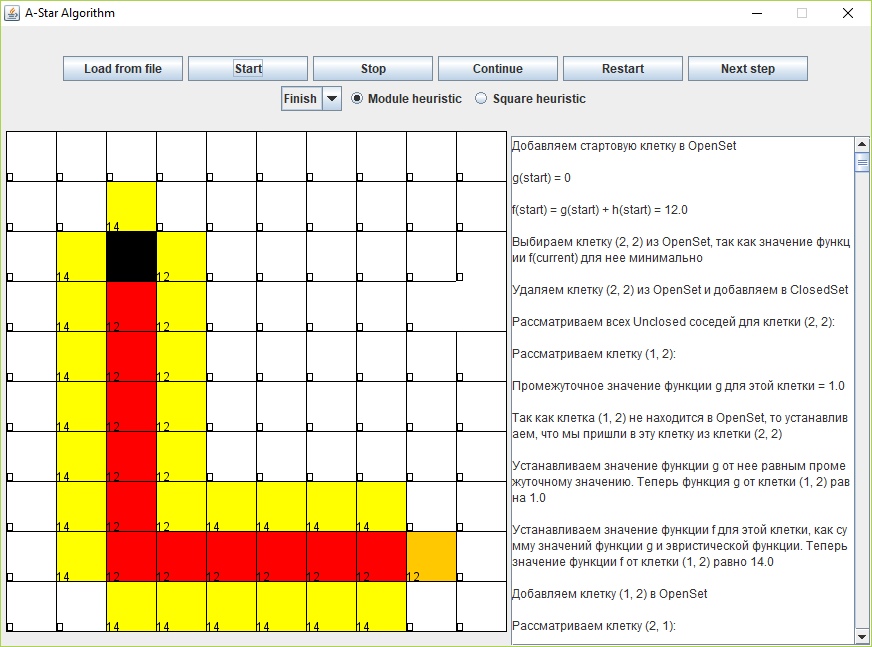
**4. Тестирование**

**4.1. Тестирование графического интерфейса**

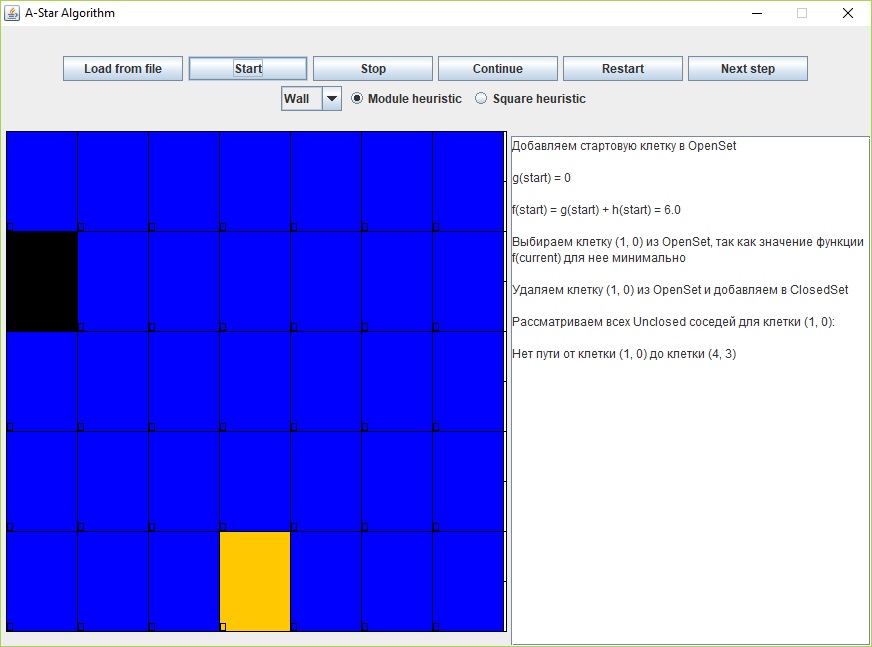
Все кнопки нашего графического интерфейса работают корректно. В ходе тестирования никаких ошибок выявлено не было.

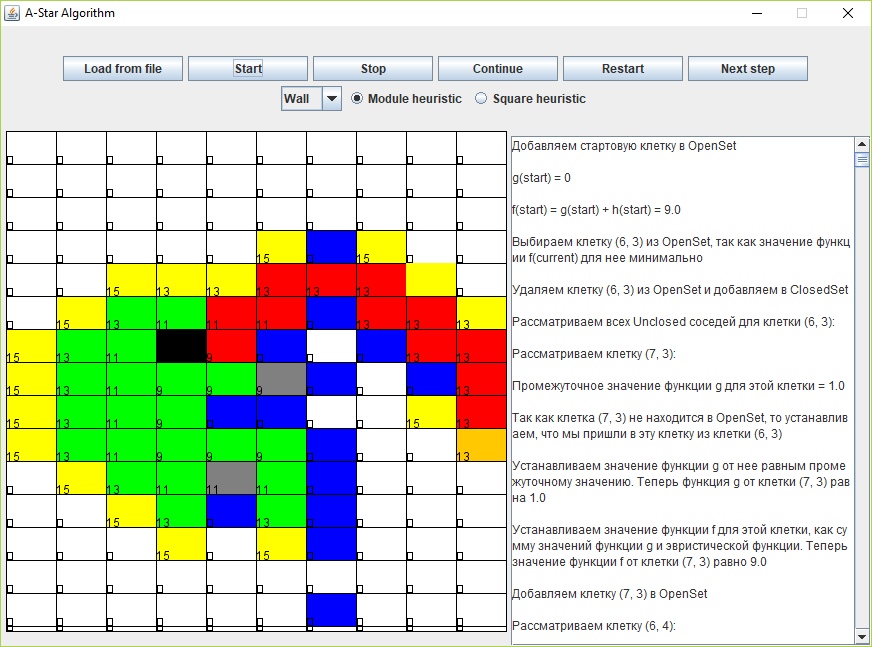
**4.2. Тестирование кода алгоритма**

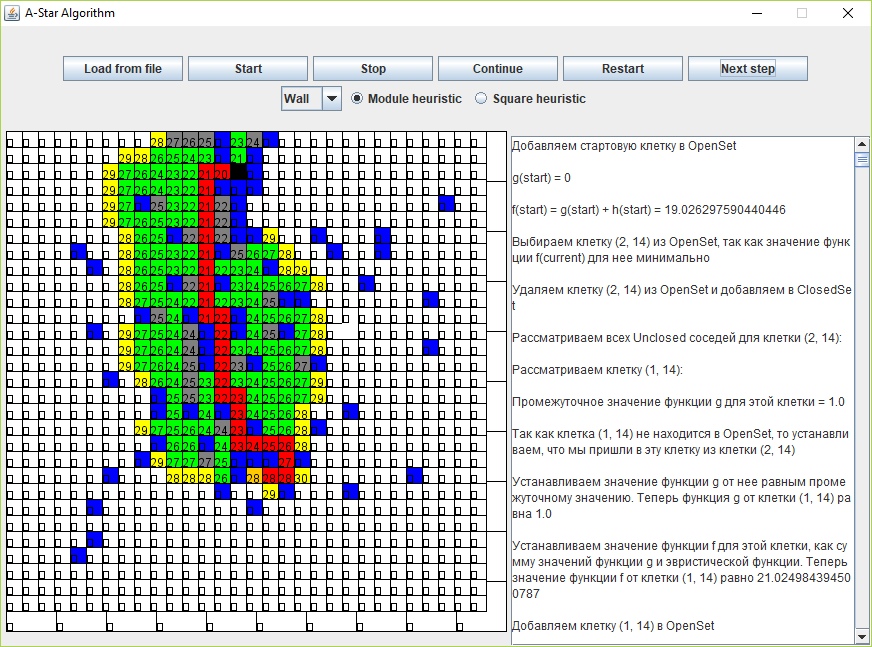
1. Случай, когда на поле нет стенок

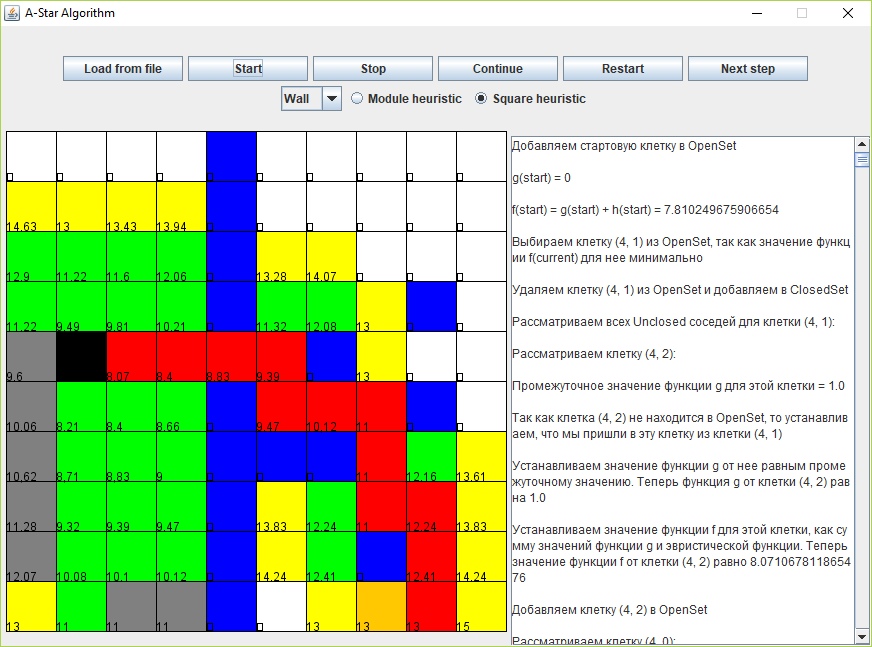


1. Случай, когда на поле расставлены все стенки



1. Случай, когда из лабиринта есть выход (лабиринт составлен пользователем)
2. Случай, когда из лабиринта есть выход ( 30 x 30, лабиринт составлен пользователем)



1. Случай, когда из лабиринта есть выход ( из файла ) 

**Заключение:**

В ходе данной учебной практики были изучены основы программирования на языке Java, пройден интерактивный курс «Java.Базовый курс». После чего была разработана программа, которая с помощью алгоритма А\* находит кратчайший путь от начальной точки до конечной, наглядно демонстрируя его пользователю.

Программа соответствует требованиям спецификации и прошла проверку на всевозможных тестах, никаких ошибок выявлено не было.

**Список использованных источников:**

1. А\*.URL: ru.wikipedia.org/wiki/A\*
2. [Java Базовый курс.URL: stepik.org/course/Java-Базовый-курс-187/syllabus](https://stepik.org/course/Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81-187/syllabus)
3. Статья «Лабиринт».URL: habrahabr.ru/post/262345/

**Приложение А.**

**Код программы.**

**Model.java**

package com.etu.model;

import java.awt.\*;

import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

public class Model {

private Field field;

private Point start;

private Point finish;

private double[][] heuristic;

private double[][] function\_g;

private double[][] function\_f;

private Point[][] from;

private StringBuilder comments;

private boolean heuristical;

private Model(Field field, Point start, Point finish, double[][] heuristic, double[][] function\_g, double[][] function\_f, Point[][] from, StringBuilder comments) {

this.field = field;

this.start = start;

this.finish = finish;

this.heuristic = heuristic;

this.function\_g = function\_g;

this.function\_f = function\_f;

this.from = from;

this.comments = comments;

this.heuristical = false;

}

public double recountHeuristic(Point current)

{

if(!heuristical)

return Math.sqrt(Math.pow((finish.y - current.y), 2) + Math.pow((finish.x - current.x), 2));

else

return (Math.abs(finish.y - current.y) + Math.abs(finish.x - current.x));

}

public boolean isActiveFull() {

boolean flag = false;

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

if(field.getSector(i, j) == Field.Sector.ACTIVE)

flag = true;

}

}

return flag;

}

public void setHeuristical(boolean heuristical) {

this.heuristical = heuristical;

}

public Set<Point> getNotUnActiveNeighbours(Point current)

{

Set<Point> ret = new HashSet<>();

if(((current.x+1) >= 0) && ((current.x+1) < field.getNumRows()))

{

if ((field.getSector(current.x + 1, current.y) == Field.Sector.ACTIVE)

|| (field.getSector(current.x + 1, current.y) == Field.Sector.FREE))

ret.add(new Point(current.x + 1, current.y));

}

if(((current.x-1) >= 0) && ((current.x-1) < field.getNumRows()))

{

if ((field.getSector(current.x - 1, current.y) == Field.Sector.ACTIVE)

|| (field.getSector(current.x - 1, current.y) == Field.Sector.FREE))

ret.add(new Point(current.x - 1, current.y));

}

if(((current.y) >= 0) && ((current.y+1) < field.getNumColumns()))

{

if ((field.getSector(current.x, current.y + 1) == Field.Sector.ACTIVE)

|| (field.getSector(current.x, current.y + 1) == Field.Sector.FREE))

ret.add(new Point(current.x, current.y + 1));

}

if(((current.y-1) >= 0) && ((current.y-1) < field.getNumColumns()))

{

if ((field.getSector(current.x, current.y - 1) == Field.Sector.ACTIVE)

|| (field.getSector(current.x, current.y - 1) == Field.Sector.FREE))

ret.add(new Point(current.x, current.y - 1));

}

return ret;

}

public static Model load()

{

Field field = Field.load();

Point start = new Point(0,0);

double[][] heuristic = new double[10][10];

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

heuristic[i][j] = 0;

}

}

double[][] function\_g = new double[10][10];

double[][] function\_f = new double[10][10];

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

function\_g[i][j] = Double.MAX\_VALUE;

function\_f[i][j] = Double.MAX\_VALUE;

}

}

Point[][] from = new Point[10][10];

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

from[i][j] = new Point(i, j);

}

}

StringBuilder comments = new StringBuilder();

return new Model(field, start, start, heuristic, function\_g, function\_f, from, comments);

}

public static Model load(Scanner scanner)

{

Field field = Field.load(scanner);

Point start = new Point(scanner.nextInt(), scanner.nextInt());

Point finish = new Point(scanner.nextInt(), scanner.nextInt());

double[][] heuristic = new double[field.getNumRows()][field.getNumColumns()];

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

heuristic[i][j] = 0;

}

}

double[][] function\_g = new double[field.getNumRows()][field.getNumColumns()];

double[][] function\_f = new double[field.getNumRows()][field.getNumColumns()];

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

function\_g[i][j] = Double.MAX\_VALUE;

function\_f[i][j] = Double.MAX\_VALUE;

}

}

Point[][] from = new Point[field.getNumRows()][field.getNumColumns()];

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns() ; j++) {

from[i][j] = new Point(i, j);

}

}

StringBuilder comments = new StringBuilder();

return new Model(field, start, finish, heuristic, function\_g, function\_f, from, comments);

}

public StringBuilder getComments()

{

return this.comments;

}

public void setFrom(Point current, Point setter)

{

from[current.x][current.y].move(setter.x, setter.y);

}

public Point getFrom(Point current)

{

return new Point(current.x, current.y);

}

public Point[][] getFrom()

{

return this.from;

}

public void clear()

{

this.field.clearField();

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

function\_g[i][j] = Double.MAX\_VALUE;

function\_f[i][j] = Double.MAX\_VALUE;

}

}

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

from[i][j].move(i, j);

}

comments.delete(0, comments.length());

}

}

public Point min\_f()

{

// System.out.println("----------------------------------");

double min = Double.MAX\_VALUE;

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

Point temp = new Point(i, j);

if(field.isActive(temp) && getFunction\_f(temp) < min)

min = getFunction\_f(temp);

}

}

Point ret = new Point();

for (int i = 0; i < field.getNumRows(); i++) {

for (int j = 0; j < field.getNumColumns(); j++) {

Point temp = new Point(i,j);

if(field.isActive(temp) && getFunction\_f(temp) == min)

ret.move(i, j);

}

}

// System.out.println(getFunction\_f(ret) +" "+ ret);

return ret;

}

// public clear

public double getFunction\_g(Point current) {

return function\_g[current.x][current.y];

}

public double[][] getFunction\_f() { return this.function\_f; }

public void setFunction\_g(Point current, double setter) {

function\_g[current.x][current.y] = setter;

}

public double getFunction\_f(Point current) {

return function\_f[current.x][current.y];

}

public void setFunction\_f(Point current, double setter) {

function\_f[current.x][current.y] = setter;

}

public Field getField() {

return field;

}

public Point getStart() {

return start;

}

public void setStart(Point start) { this.start = start; }

public Point getFinish() {

return finish;

}

public void setFinish(Point finish) {

this.finish = finish;

}

public double[][] getHeuristic() {

return heuristic;

}

public void setWall(Point wall){ field.setSectorWall(wall); }

public void setEmpty(Point cords) { field.setSectorFree(cords);}

public boolean isWall(Point temp){ return field.getSector(temp.x,temp.y) == Field.Sector.WALL;}

}

**Field.java**

package com.etu.model;

import java.awt.\*;

import java.util.Scanner;

public class Field {

private final Sector[][] field;

private Field(Sector[][] field) {

this.field = field;

}

public int getNumRows() {

return field.length;

}

public int getNumColumns() {

return field[0].length;

}

public boolean isActive(Point current)

{

return field[current.x][current.y] == Sector.ACTIVE;

}

public Sector getSector(int x, int y) {

return field[x][y];

}

void setSectorWall(Point current){ field[current.x][current.y] = Sector.WALL; }

public void setSectorActive(Point current)

{

field[current.x][current.y] = Sector.ACTIVE;

}

public void setSectorUnActive(Point current)

{

field[current.x][current.y] = Sector.UNACTIVE;

}

public void setSectorRealWay (Point current) { field[current.x][current.y] = Sector.REALWAY; }

public void setSectorCurrent (Point current) { field[current.x][current.y] = Sector.CURRENT; }

void setSectorFree (Point current) { field[current.x][current.y] = Sector.FREE; }

static Field load()

{

Sector[][] field = new Sector[10][10];

for (int x = 0; x < 10; ++x) {

for (int y = 0; y < 10; ++y) {

field[x][y] = Sector.FREE;

}

}

return new Field(field);

}

void clearField()

{

for (int x = 0; x < getNumRows(); ++x) {

for (int y = 0; y < getNumColumns(); ++y) {

if(field[x][y] == Sector.CURRENT || field[x][y] == Sector.UNACTIVE || field[x][y] == Sector.REALWAY || field[x][y] == Sector.ACTIVE)

field[x][y] = Sector.FREE;

}

}

}

static Field load(Scanner input) {

int numRows = input.nextInt();

int numColumns = input.nextInt();

Sector[][] field = new Sector[numRows][numColumns];

for (int x = 0; x < numRows; ++x) {

for (int y = 0; y < numColumns; ++y) {

field[x][y] = input.next().equals("w") ? Sector.WALL : Sector.FREE;

}

}

return new Field(field);

}

public enum Sector {

FREE,

WALL,

ACTIVE,

UNACTIVE,

CURRENT,

REALWAY

}

}

**Controller.java**

package com.etu.controller;

import com.etu.Swing.GameLauncher;

import com.etu.model.\*;

import com.etu.view.View;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.io.File;

import java.util.Scanner;

import java.util.Set;

public class Controller {

private Model model;

private final View view;

private Thread thr;

private Timer timer;

public Controller(Model model, View view) {

this.model = model;

this.view = view;

this.thr = new Thread();

this.timer = new Timer(0,null);

}

private void addCommentToLog(String appender) {

model.getComments().delete(0, model.getComments().length());

model.getComments().append(appender);

updateLog();

}

/\*

public synchronized void waiter() throws InterruptedException {

thr.wait();

}

public synchronized void notifier()

{

notifyAll();

}

\*/

public void viewUpdated() {

view.draw(model);

}

private void restoreWay() {

Point current = model.getFinish();

// model.getField().setSectorRealWay(model.getStart());

while (!current.equals(model.getStart())) {

model.getField().setSectorRealWay(current);

addCommentToLog("Путь проходит через клетку (" + current.x + ", " + current.y + ")\n");

current = model.getFrom()[current.x][current.y];

}

addCommentToLog("Путь начался из клетки (" + model.getStart().x + ", " + model.getStart().y + ")\n");

}

private void restoreWay(Point current)

{

while (!current.equals(model.getStart())) {

model.getField().setSectorCurrent(current);

current = model.getFrom()[current.x][current.y];

}

}

private synchronized void implementAstar() throws InterruptedException {

double temporary\_g;

for (int i = 0; i < model.getField().getNumRows() ; i++) {

for (int j = 0; j < model.getField().getNumColumns(); j++) {

model.getHeuristic()[i][j] = model.recountHeuristic(new Point(i, j));

}

}

model.getField().setSectorActive(model.getStart());

addCommentToLog("Добавляем стартовую клетку в OpenSet\n\n");

model.setFunction\_g(model.getStart(), 0);

addCommentToLog("g(start) = 0\n\n");

model.setFunction\_f(model.getStart(), model.getFunction\_g((model.getStart())) +

model.getHeuristic()[model.getStart().x][model.getStart().y]);

addCommentToLog("f(start) = g(start) + h(start) = " + model.getFunction\_f(model.getStart()) + "\n\n");

while (model.isActiveFull()) {

Point current = model.min\_f();

addCommentToLog("Выбираем клетку (" + current.x + ", " + current.y + ") из OpenSet, " +

"так как значение функции f(current) для нее минимально\n\n");

wait();

model.getField().setSectorCurrent(current);

wait();

if (current.equals(model.getFinish())) {

addCommentToLog("Восстанавливаем путь: \n\n");

restoreWay();

return;

}

model.getField().setSectorUnActive(current);

addCommentToLog("Удаляем клетку (" + current.x + ", " + current.y + ") из OpenSet и добавляем в ClosedSet\n\n");

Set<Point> neighbours = model.getNotUnActiveNeighbours(current);

// System.out.println(neighbours.size());

addCommentToLog("Рассматриваем всех Unclosed соседей для клетки (" + current.x + ", " + current.y + "): \n\n");

for (Point neighbour : neighbours) {

addCommentToLog("Рассматриваем клетку (" + neighbour.x + ", " + neighbour.y + "):\n\n");

temporary\_g = model.getFunction\_g(current) + 1;

addCommentToLog("Промежуточное значение функции g для этой клетки = " + temporary\_g + "\n\n");

if (!model.getField().isActive(neighbour) || temporary\_g < model.getFunction\_g(neighbour)) {

if (!model.getField().isActive(neighbour))

addCommentToLog("Так как клетка (" + neighbour.x + ", " + neighbour.y + ") не находится в OpenSet, то ");

else if (temporary\_g < model.getFunction\_g(neighbour))

addCommentToLog("Так как значение функции g для клетки (" + neighbour.x + ", " + neighbour.y + ") меньше промежуточного, то ");

model.setFrom(model.getFrom(neighbour), current);

restoreWay(current);

addCommentToLog("устанавливаем, что мы пришли в эту клетку из клетки (" + current.x + ", " + current.y + ")\n\n");

model.setFunction\_g(neighbour, temporary\_g);

addCommentToLog("Устанавливаем значение функции g от нее равным промежуточному значению. " +

"Теперь функция g от клетки (" + neighbour.x + ", " + neighbour.y + ") равна " + model.getFunction\_g(neighbour) + "\n\n");

model.setFunction\_f(neighbour, model.getFunction\_g(neighbour) +

model.getHeuristic()[neighbour.x][neighbour.y]);

addCommentToLog("Устанавливаем значение функции f для этой клетки, как сумму значений функции g и эвристической функции." +

" Теперь значение функции f от клетки (" + neighbour.x + ", " + neighbour.y + ") равно " + model.getFunction\_f(neighbour) + "\n\n");

}

if (!model.getField().isActive(neighbour)) {

model.getField().setSectorActive(neighbour);

addCommentToLog("Добавляем клетку (" + neighbour.x + ", " + neighbour.y + ") в OpenSet\n\n");

}

}

neighbours.clear();

}

addCommentToLog("Нет пути от клетки (" + model.getStart().x + ", " + model.getStart().y + ") до клетки" +

" (" + model.getFinish().x + ", " + model.getFinish().y + ")");

}

private void clearLog(){

view.getLog().setText("");

}

private void updateLog() {

view.getLog().append(model.getComments().toString());

}

private void update() {

viewUpdated();

}

private void clearModel(){

model.clear();

thr.interrupt();

}

private void startTimer() {

timer.start();

update();

}

public void stopTimer() {

timer.stop();

update();

}

public void resume(){

startTimer();

update();

}

public void restart(){

model = Model.load();

clearLog();

stopTimer();

thr.interrupt();

// thr = null;

update();

}

public synchronized void next(){

notifyAll();

update();

}

public void load() {

restart();

// JFileChooser fc = new JFileChooser();

JFileChooser chooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter(

"Data input", "dat", "txt");

File f = new File("./src/com/etu/Swing/data");//"/"+ System.getProperty("user.dir") + "/src/com.etu/controller/Swing/data");

chooser.setCurrentDirectory(f);//new File("/data/"));//"/"+ System.getProperty("user.dir") + "/src/com.etu/controller/Swing/data"));

chooser.setFileFilter(filter);

int returnVal = chooser.showOpenDialog(null);

if (returnVal == JFileChooser.APPROVE\_OPTION) {

Scanner scanner = new Scanner(GameLauncher.class.getResourceAsStream("data/" + chooser.getSelectedFile().getName()));

this.model = Model.load(scanner);

update();

}

}

public void start(){

clearLog();

clearModel();

thr = new Thread(() -> {

try {

implementAstar();

} catch (InterruptedException e) {

e.printStackTrace();

}

});

try {

thr.start();

} catch (IllegalThreadStateException e){

return;

}

timer = new Timer(500, e -> {

next();

viewUpdated();

});

startTimer();

update();

}

public void mouseClick(MouseEvent e, String boxContents){

setSafeStop();

clearModel();

Point cords = view.getCoords(e.getPoint());

switch (boxContents){

case "Start": model.setStart(cords);break;

case "Finish": model.setFinish(cords);break;

case "Wall":

if (model.isWall(cords)) model.setEmpty(cords);

else model.setWall(cords);

break;

}

update();

}

private void setSafeStop(){

if(timer.isRunning()) {

stopTimer();

clearModel();

update();

return;

}

clearModel();

}

public void changeAlgorithmOnFirst() {

setSafeStop();

model.setHeuristical(true);

}

public void changeAlgorithmOnSecond() {

setSafeStop();

model.setHeuristical(false);

}

}

**View.java**

package com.etu.view;

import com.etu.model.Field;

import com.etu.model.Model;

import javax.swing.\*;

import java.awt.\*;

import java.text.DecimalFormat;

public class View {

private final static int CANVAS\_SIZE = 500;

private static int CELL\_SIZE\_X ;

private static int CELL\_SIZE\_Y ;

private JTextArea log;

private Graphics2D graphics;

public void setLog(JTextArea log) {

this.log = log;

}

public JTextArea getLog() {

return log;

}

public void draw(Model model) {

updateCells(model.getField());

drawField(model);

}

private void updateCells(Field field){

CELL\_SIZE\_X = CANVAS\_SIZE / field.getNumRows();

CELL\_SIZE\_Y = CANVAS\_SIZE / field.getNumColumns();

}

private void drawField(Model model) {

Field field = model.getField();

Point pointFinish = model.getFinish();

Point pointStart = model.getStart();

DecimalFormat df = new DecimalFormat("#.##");

double[][] her = model.getFunction\_f();

for (int x = 0; x < field.getNumRows(); x++) {

for (int y = 0; y < field.getNumColumns(); y++) {

Color color = Color.GROUND;

if (pointStart.x == x && pointStart.y == y) color = Color.START;

else {

if (pointFinish.x == x && pointFinish.y == y) color = Color.FINISH;

else {

switch (field.getSector(x, y)){

case ACTIVE: color = Color.ACTIVE; break;

case CURRENT: color = Color.CURRENT; break;

case WALL: color = Color.WALL; break;

case FREE: color = Color.GROUND; break;

case REALWAY: color = Color.WAY;break;

case UNACTIVE: color = Color.UNACTIVE;break;

}

}

}

graphics.drawRect( x \* CELL\_SIZE\_X, y \* CELL\_SIZE\_Y, CELL\_SIZE\_Y, CELL\_SIZE\_X, color.getRGB());

color = Color.INFOCELL;

// inverted coords!!!

graphics.drawText((y)\*CELL\_SIZE\_Y,(x+1)\*CELL\_SIZE\_X,df.format(her[x][y] == Double.MAX\_VALUE ? Double.NaN : her[x][y]),color.getRGB());

}

}

}

public void setGraphics(Graphics2D graphics) {

this.graphics = graphics;

}

private enum Color {

// BONUS\_CHECKED(java.awt.Color.GREEN.getRGB()),

// BONUS\_UNCHECKED(java.awt.Color.ORANGE.getRGB()),

WALL(java.awt.Color.BLUE.getRGB()),//new java.awt.Color(162, 129, 39).getRGB()),

GROUND(java.awt.Color.WHITE.getRGB()),//new java.awt.Color(202, 203, 204).getRGB()),

WAY(java.awt.Color.RED.getRGB()),

ACTIVE(java.awt.Color.YELLOW.getRGB()),

// HEADER(new java.awt.Color(169, 180, 192).getRGB()),

// BORDER(new java.awt.Color(0, 0, 0).getRGB()),

START(java.awt.Color.BLACK.getRGB()),

INFOCELL(java.awt.Color.BLACK.getRGB()),

FINISH(java.awt.Color.ORANGE.getRGB()),

CURRENT(java.awt.Color.GREEN.getRGB()),

UNACTIVE(java.awt.Color.GRAY.getRGB());

private final int rgb;

Color(int rgb) {

this.rgb = rgb;

}

public int getRGB() {

return rgb;

}

}

//case of reverted field i had to revert my coords!!

public Point getCoords(Point mousePoint){

return new Point(mousePoint.y /CELL\_SIZE\_X,mousePoint.x / CELL\_SIZE\_Y);

}

}

**Graphics2D.java**

package com.etu.view;

public interface Graphics2D {

void drawRect(int x, int y, int width, int height, int rgb);

void drawText(int x, int y, String text, int rgb);

// void drawOval(int x, int y, int width, int height, int rgb);

// void showCongratsDialog(); not nesecery

// void showNeedMoreBonusesDialog();

}

**ControlPanel.java**

package com.etu.Swing;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionListener;

public class ControlPanel extends JPanel {

private final JButton load = createButton("Load from file");

private final JButton start = createButton("Start");

private final JButton stop = createButton("Stop");

private final JButton resume = createButton("Continue");

private final JButton restart = createButton("Restart");

private final JButton next = createButton("Next step");

private final JComboBox box = new JComboBox(new String[] {"Start","Finish","Wall"});

private final ButtonGroup group = new ButtonGroup();

private final JRadioButton var1 = new JRadioButton("Module heuristic", false);

private final JRadioButton var2 = new JRadioButton("Square heuristic", true);

public ControlPanel() {

super();

group.add(var1);

group.add(var2);

add(load);

add(start);

add(stop);

add(resume);

add(restart);

add(next);

add(box);

add(var1);

add(var2);

setLayout(new FlowLayout());

}

private JButton createButton(String text){//, int x, int y) {

JButton b = new JButton(text);

//b.setBounds(x, y, 30, 30);

b.setPreferredSize(new Dimension(120,25));

//b.setFocusPainted(false);

//b.setBorder(BorderFactory.createEmptyBorder(1, 0, 0, 0));

return b;

}

public String getBoxContents(){return box.getSelectedItem().toString();}

public void var\_1\_ActionListener(ActionListener listener){var1.addActionListener(listener);}

public void var\_2\_ActionListener(ActionListener listener){var2.addActionListener(listener);}

public void loadActionListener(ActionListener listener){load.addActionListener(listener);}

public void startActionListener(ActionListener listener){start.addActionListener(listener);}

public void stopActionListener(ActionListener listener){stop.addActionListener(listener);}

public void resumeActionListener(ActionListener listener){resume.addActionListener(listener);}

public void restartActionListener(ActionListener listener){restart.addActionListener(listener);}

public void nextActionListener(ActionListener listener){next.addActionListener(listener);}

}

**GameLauncher.java**

package com.etu.Swing;

import com.etu.controller.Controller;

import com.etu.model.Model;

import com.etu.view.View;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class GameLauncher extends JFrame {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new GameLauncher("A-Star Algorithm").setVisible(true));

}

private final JPanel canvas;

private final ControlPanel toolBar;

private final JTextArea textArea;

//private final JScrollPane log;

public GameLauncher(String title) throws HeadlessException {

super(title);

canvas = new JPanel();

canvas.setLayout(new OverlayLayout(canvas));

canvas.setPreferredSize(new Dimension(510, 510));

canvas.setMinimumSize(new Dimension(510,510));

toolBar = new ControlPanel();

toolBar.setPreferredSize(new Dimension(800, 75));

textArea = new JTextArea();

//for(int i = 0;i <= 1000; i++) textArea.append("Test hello world!1111111111");

textArea.setLineWrap(true);

JScrollPane log = new JScrollPane(textArea);

log.setPreferredSize(new Dimension(350, 425));

JPanel rootPanel = new JPanel();

GridBagLayout gbl = new GridBagLayout();

GridBagConstraints c = new GridBagConstraints();

/\*

next 3 command sets tree components on jframe

where c is param

gbl - my GridBagLayot

\*/

toolBarConfig(c);

gbl.setConstraints(toolBar,c);

rootPanel.add(toolBar);

canvasConfig(c);

gbl.setConstraints(canvas,c);

rootPanel.add(canvas,c);

logConfig(c);

gbl.setConstraints(log,c);

rootPanel.add(log,c);

rootPanel.setLayout(gbl);

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setContentPane(rootPanel);

pack();

setResizable(false);

setLocationRelativeTo(null);

initListeners();

}

private void initListeners(){

Model model = Model.load();

View view = new View();

Controller controller = new Controller(model,view);

view.setGraphics(new SwingGraphicsAdapter((Graphics2D) canvas.getGraphics()));

view.setLog(textArea);

toolBar.loadActionListener(e -> controller.load());

toolBar.startActionListener(e -> controller.start());

toolBar.resumeActionListener(e -> controller.resume());

toolBar.restartActionListener(e -> controller.restart());

toolBar.nextActionListener(e -> controller.next());

toolBar.stopActionListener(e -> controller.stopTimer());

canvas.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

controller.mouseClick(e,toolBar.getBoxContents());

}

});

toolBar.var\_1\_ActionListener(e -> controller.changeAlgorithmOnFirst());

toolBar.var\_2\_ActionListener(e -> controller.changeAlgorithmOnSecond());

controller.viewUpdated();

canvas.requestFocus();

/\*

Timer timer = new Timer(1000, e -> {

});

timer.setRepeats(true);

timer.start();

\*/

}

private void toolBarConfig(GridBagConstraints c) {

c.anchor = GridBagConstraints.CENTER;

c.fill = GridBagConstraints.VERTICAL;

c.gridheight = 1;

c.gridwidth = GridBagConstraints.REMAINDER;

c.gridx = GridBagConstraints.RELATIVE;

c.gridy = GridBagConstraints.RELATIVE;

c.insets = new Insets(25, 0, 0, 0);

c.ipadx = 0;

c.ipady = 0;

c.weightx = 1;

c.weighty = 1;

}

private void canvasConfig(GridBagConstraints c) {

c.anchor = GridBagConstraints.CENTER;

c.fill = GridBagConstraints.BOTH;

c.gridheight = 3;

c.gridwidth = 3;

c.gridx = GridBagConstraints.RELATIVE;

c.gridy = GridBagConstraints.RELATIVE;

c.insets = new Insets(0, 0, 0, 0);

c.ipadx = 0;

c.ipady = 0;

c.weightx = 0.0;

c.weighty = 0.0;

}

private void logConfig(GridBagConstraints c) {

c.anchor = GridBagConstraints.CENTER;

c.fill = GridBagConstraints.BOTH;

c.gridheight = GridBagConstraints.REMAINDER;

c.gridwidth = GridBagConstraints.REMAINDER;

c.gridx = GridBagConstraints.RELATIVE;

c.gridy = GridBagConstraints.RELATIVE;

c.insets = new Insets(0, 0, 0, 0);

c.ipadx = 0;

c.ipady = 0;

c.weightx = 0.0;

c.weighty = 0.0;

}

}

**SwingGraphicsAdapter.java**

package com.etu.Swing;

import com.etu.view.Graphics2D;

import java.awt.\*;

public class SwingGraphicsAdapter implements Graphics2D {

private final java.awt.Graphics2D graphics;

SwingGraphicsAdapter(java.awt.Graphics2D graphics) {

this.graphics = graphics;

graphics.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

}

@Override

@SuppressWarnings("SuspiciousNameCombination")

public void drawRect(int x, int y, int width, int height, int rgb) {

graphics.setColor(new Color(rgb));

graphics.fillRect(y, x, width, height);

graphics.setColor(Color.BLACK);

graphics.drawRect(y,x,width,height);

}

@Override

public void drawText(int x, int y, String text, int rgb) {

char[] symbols = text.toCharArray();

graphics.setColor(new Color(rgb));

graphics.drawChars(symbols, 0, symbols.length, x, y);

}

}