## ****Оформление магистерских диссертаций, дипломных, бакалаврских и курсовых работ****

При написании курсовых, дипломных, бакалаврских работ и магистерских диссертаций следует соблюдать требования следующих документов:

* Методические указания «Структура и оформление бакалаврской, дипломной и курсовой работ», 2021 г. (сост. М.Б. Астапов, Ж.О. Карапетян, О.А.Бондаренко, В.В. Бондаренко).

Ссылка:

<https://www.kubsu.ru/sites/default/files/insert/page/uchebno-metodicheskie_ukazaniya_struktura_i_oformlenie_magisterskoy_dissertacii_bakalavrskoy_diplomnoy_i_kursovoy_rabot.pdf>

Изображение выглядит как текст

Автоматически созданное описание

* ГОСТ Р 7.0.100 – 2018 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;

Ссылка:

<https://docs.cntd.ru/document/1200161674>

* ГОСТ 7.32 – 2017 «Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и  правила оформления»;

Ссылка:

<https://docs.cntd.ru/document/1200157208>

**Требования к выполнению курсовых и ВКР**

**(бакалавриат)**

**Курсовая №1 (осень 3 курс)**

1. Титульный лист оформляется согласно п.2.1 методического пособия (с.7).
2. Текст курсовой работы (ПЗ-пояснительная записка) в объеме 20-25 стр, шрифт Times New Roman, 14 pt, полтора интервала. Текст следует печатать соблюдая следующие размеры полей: левое - 30мм, правое - 15 мм, верхнее и нижнее – 20мм.
3. ПЗ содержит: реферат, содержание, введение, разделы работы, приложения с фрагментами кода программы.
4. Реферат оформляется согласно ГОСТ 7.32 – 2017 (п.5.3.2.2, п. 6.12).
5. В содержании номера подразделов приводят после абзацного отступа, равного двум знакам, относительно номеров разделов. Номера пунктов приводят после абзацного отступа, равного двум знакам относительно номеров подразделов.
6. Каждый структурный элемент и каждый раздел основной части отчета начинают с новой страницы.
7. Презентация для доклада на защите (не более 12 слайдов), длительность доклада – 5 минут.
8. Прохождение нормоконтроля у нормоконтролера.

**Курсовая №2 (весна 3 курс)**

1. Титульный лист оформляется согласно п.2.1 методического пособия (с.7).
2. Текст курсовой работы в объеме 25-30 стр, шрифт Times New Roman, 14 pt, полтора интервала. Текст следует печатать соблюдая следующие размеры полей: левое - 30мм, правое - 15 мм, верхнее и нижнее – 20мм.
3. ПЗ содержит реферат, содержание, введение, разделы работы, приложения с фрагментами кода программы.
4. Реферат оформляется согласно ГОСТ 7.32 – 2017 (п.5.3.2.2, п. 6.12).
5. В содержании номера подразделов приводят после абзацного отступа, равного двум знакам, относительно номеров разделов. Номера пунктов приводят после абзацного отступа, равного двум знакам относительно номеров подразделов.
6. Каждый структурный элемент и каждый раздел основной части отчета начинают с новой страницы.
7. Презентация для доклада на защите (не более 12 слайдов), длительность доклада – 5 минут.
8. Нормокнотроль за руководителем.

*Приложение 1. Техническое задание (ТЗ)*

*Приложение 2. График выполнения задания.*

**Диплом (ВКР – выпускная квалификационная работа) (весна 4 курс)**

Титульный лист оформляется согласно п.2.1 методического пособия (с.7).

Текст ВКР в объеме 40-50 стр, шрифт Times New Roman, 14 pt, полтора интервала. Текст следует печатать соблюдая следующие размеры полей: левое - 30мм, правое - 15 мм, верхнее и нижнее – 20мм.

ПЗ содержит реферат, содержание, введение, разделы работы, приложения с фрагментами кода программы.

Реферат оформляется согласно ГОСТ 7.32 – 2017 (п.5.3.2.2, п. 6.12).

В содержании номера подразделов приводят после абзацного отступа, равного двум знакам, относительно номеров разделов. Номера пунктов приводят после абзацного отступа, равного двум знакам относительно номеров подразделов.

Введение должно отражать актуальность и новизну темы работы, оценку современного состояния решаемой научной проблемы, основания и исходные данные для написания работы. Во введении должны быть отражены объект и предмет, цель, задачи и методы исследования, теоретическая и практическая значимость работы, возможность использования полученных результатов.

Каждый структурный элемент и каждый раздел основной части отчета начинают с новой страницы.

Презентация для доклада на защите (12-15 слайдов), длительность доклада – 7 минут.

Прохождение нормоконтроля у нормоконтролера.

*Приложение 1. Техническое задание (ТЗ)*

*Приложение 2. График выполнения задания.*

*ОТЗЫВ Руководителя с оценкой.*

**Требования к выполнению курсовых и ВКР**

**(магистратура)**

**Курсовая (весна 1 курс)**

1. Титульный лист оформляется согласно п.2.1 методического пособия (с.7).
2. Текст курсовой работы (ПЗ-пояснительная записка) в объеме 35-40 стр, шрифт Times New Roman, 14 pt, полтора интервала. Текст следует печатать соблюдая следующие размеры полей: левое - 30мм, правое - 15 мм, верхнее и нижнее – 20мм.
3. ПЗ содержит реферат, содержание, введение, разделы работы, приложения с фрагментами кода программы.
4. Реферат оформляется согласно ГОСТ 7.32 – 2017 (п.5.3.2.2, п. 6.12).
5. В содержании номера подразделов приводят после абзацного отступа, равного двум знакам, относительно номеров разделов. Номера пунктов приводят после абзацного отступа, равного двум знакам относительно номеров подразделов.
6. Каждый структурный элемент и каждый раздел основной части отчета начинают с новой страницы.
7. Презентация для доклада на защите (не более 12 слайдов), длительность доклада – 5 минут.
8. Нормокнотроль за руководителем.

**Диплом (ВКР – выпускная квалификационная работа) (весна 2 курс)**

Титульный лист оформляется согласно п.2.1 методического пособия (с.7).

Текст ВКР в объеме 50-60 стр, шрифт Times New Roman, 14 pt, полтора интервала. Текст следует печатать соблюдая следующие размеры полей: левое - 30мм, правое - 15 мм, верхнее и нижнее – 20мм.

ПЗ содержит реферат, содержание, введение, разделы работы, приложения с фрагментами кода программы.

Реферат оформляется согласно ГОСТ 7.32 – 2017 (п.5.3.2.2, п. 6.12).

В содержании номера подразделов приводят после абзацного отступа, равного двум знакам, относительно номеров разделов. Номера пунктов приводят после абзацного отступа, равного двум знакам относительно номеров подразделов.

Введение должно отражать актуальность и новизну темы работы, оценку современного состояния решаемой научной проблемы, основания и исходные данные для написания работы. Во введении должны быть отражены объект и предмет, цель, задачи и методы исследования, теоретическая и практическая значимость работы, возможность использования полученных результатов.

Каждый структурный элемент и каждый раздел основной части отчета начинают с новой страницы.

Презентация для доклада на защите (12-15 слайдов), длительность доклада – 7 минут.

Прохождение нормоконтроля у нормоконтролера.

*Приложение 1. Техническое задание (ТЗ)*

*Приложение 2. График выполнения задания.*

*ОТЗЫВ Руководителя с оценкой.*

*Рецензия научного специалиста (доцент, профессор) не со своего факультета.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Текст курсовой работы в соответствии с методическими указаниями «Структура и оформление бакалаврской, дипломной и курсовой работ», 2019 г. и ГОСТ 7.32 – 2017, ГОСТ Р 7.0.100 – 2018.

***ТЗ и график оформляются и подписываются при получении задания на курсовую работу (диплом).***

ПЗ-пояснительная записка.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**КОМБИНИРОВАННЫЙ АЛГОРИТМ ПОИСКА ПУТИ В ЛАБИРИНТЕ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. И. Иванов

(подпись)

Направление подготовки 02.03.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ курс\_\_\_\_\_3\_\_\_\_\_

Направленность (профиль)\_\_\_\_Математическое и программное обеспечение компьютерных технологий\_\_\_\_

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Е. Полупанова

(подпись, дата)

Нормоконтролер

преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Нигодин

(подпись, дата)

Краснодар

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**КОМБИНИРОВАННЫЙ АЛГОРИТМ ПОИСКА ПУТИ В ЛАБИРИНТЕ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Иванов

(подпись)

Направление подготовки 02.04.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ курс\_\_\_\_\_1\_\_\_\_\_

Направленность (профиль) \_\_\_Интеллектуальные системы и технологии\_\_\_

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Е. Полупанова

(подпись, дата)

Нормоконтролер

преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Нигодин

(подпись, дата)

Краснодар

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

Допустить к защите

И.о. заведующего кафедрой

канд. физ-мат. наук, доц.,

\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Еремин

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(БАКАЛАВРСКАЯ РАБОТА)**

**КОМБИНИРОВАННЫЙ АЛГОРИТМ ПОИСКА ПУТИ В ЛАБИРИНТЕ**

Работу выполнил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Иванов

(подпись)

Направление подготовки 02.03.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность (профиль) \_\_\_\_Математическое и программное обеспечение компьютерных технологий\_\_\_

Научный руководитель

канд.техн.наук., доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.Е. Полупанова

(подпись)

Нормоконтролер

преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Нигодин

(подпись)

Краснодар

2024МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

Допустить к защите

И.о. заведующего кафедрой

канд. физ-мат. наук, доц.,

\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Еремин

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г.

Руководитель ООП

д-р техн. наук, проф.,

\_\_\_\_\_\_\_\_\_\_\_\_ Ю.М. Вишняков

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)**

**КОМБИНИРОВАННЫЙ АЛГОРИТМ ПОИСКА ПУТИ В ЛАБИРИНТЕ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Иванов

(подпись)

Направление подготовки 02.04.02 — «Фундаментальная информатика и\_\_\_\_\_

(код, наименование)

информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность (профиль) \_\_\_Интеллектуальные системы и технологии\_\_\_

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Е. Полупанова

(подпись)

Нормоконтролер

преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Нигодин

(подпись)

Краснодар

2024

**РЕФЕРАТ**

Выпускная квалификационная работа 38 с., 4 ч., 29 рис., 1 табл., 8 источн., 4 прил.

КРАТЧАЙШИЙ ПУТЬ, ЛАБИРИНТ, МУРАВЬИНЫЙ АЛГОРИТМ, ВОЛНОВОЙ АЛГОРИТМ, СТОХАСТИЧЕСКИЙ АЛГОРИТМ

Объектом исследования в данной работе является комбинированный алгоритм поиска кратчайшего пути в заданном лабиринте произвольных размеров.

Цель работы: разработка комбинированного алгоритма поиска кратчайшего пути в лабиринте.

Методологическая основа исследования включает в себя эмпирический метод, сбор статистической информации, графический метод, анализ полученных зависимостей, синтез выводов, аналогию.

В результате работы было разработано приложение, которое строит лабиринт заданного размера, находит кратчайший путь между двумя заданными клетками с помощью комбинированного алгоритма и выделяет полученный путь в лабиринте.

Научная новизна работы заключается в том, что предлагается новый приближённый алгоритм решения задачи поиска кратчайшего пути в лабиринте.

В результате решения поставленных задач были определены оптимальные параметры для муравьиного алгоритма, которые позволяют оптимизировать время и качество работы комбинированного алгоритма поиска кратчайшего пути в лабиринте. Проведённые экспериментальные исследования показали преимущество разработанного алгоритма над его обычным аналогом. Разработанный алгоритм в большинстве случаев решает поставленную задачу нахождения кратчайшего пути в лабиринте быстрее и не хуже точного.

**СОДЕРЖАНИЕ**

[Введение 12](#_Toc74084969)

[1 Общие сведения о лабиринтах и графах 15](#_Toc74084970)

[1.1 Понятие лабиринта 15](#_Toc74084971)

[1.2 Представление клеток лабиринта 18](#_Toc74084972)

[1.3 Сведение задачи к поиску кратчайшего пути в графе 19](#_Toc74084973)

[1.3.1 Планарность 20](#_Toc74084974)

[1.3.2 Маршруты, пути и связность 22](#_Toc74084975)

[2 Математическая постановка задачи о пути в лабиринте 24](#_Toc74084976)

[2.1 Алгоритм Дейкстры 25](#_Toc74084977)

[2.2 Алгоритм А\* 28](#_Toc74084978)

[2.3 Алгоритм волновой трассировки 30](#_Toc74084979)

[2.4 Алгоритм следования вдоль стен 32](#_Toc74084980)

[2.5 Муравьиный алгоритм 33](#_Toc74084981)

[2.6 Комбинированный алгоритм 35](#_Toc74084982)

[2.7 Пример реализации 36](#_Toc74084983)

[3 Реализация приложения 38](#_Toc74084984)

[3.1 Основные сведения о программе 38](#_Toc74084985)

[3.2 Особенности реализации 43](#_Toc74084986)

[4 Анализ эффективности алгоритма волновой трассировки 44](#_Toc74084987)

[Заключение 47](#_Toc74084988)

[Список использованных источников 48](#_Toc74084989)

[Приложение А Основная программа 50](#_Toc74084990)

[Приложение Б Файл генерации текстуры 67](#_Toc74084991)

[Приложение В Файл хранения матрицы феромонов 68](#_Toc74084992)

[Приложение Г Файл инициализации программы 69](#_Toc74084993)

# ВВЕДЕНИЕ

Актуальность данной работы состоит в необходимости разработки новых алгоритмов и методов решения трудноразрешимых задач оптимизации, которые имеют широкое практическое применение. Такими методами являются, в частности, различные эвристические алгоритмы (жадные, генетические, эволюционные, алгоритмы, основанные на природных явлениях). В настоящее время существуют задачи, которые на определённых входных данных не решаются с помощью точных алгоритмов за приемлемое время даже на распределённых системах с тысячами узлов. Задача поиска кратчайшего пути в лабиринте имеет сложность 𝑂(𝑛6), если для размера лабиринта 𝑛×𝑛. Поэтому эта задача тоже нуждается в приближённом алгоритме решения. Из всего вышеуказанного следует, что разработка эвристических алгоритмов решения различных задач сейчас востребована.

Основная цель работы – разработка комбинированного алгоритма поиска кратчайшего пути в лабиринте, на основе волнового и муравьиного алгоритмов, определение значения свободных параметров, которые позволят оптимизировать время и качество работы разработанного алгоритма поиска кратчайшего пути в лабиринте.

Для реализации поставленной цели предполагается решить следующие задачи:

* изучить теорию алгоритмов, решающих задачу поиска пути в лабиринте и на планарных графах;
* разработать программу, предназначенную для решения задачи нахождения кратчайшего пути в лабиринте с помощью точного и комбинированного алгоритмов;
* провести ряд экспериментов для исследования зависимости времени и качества работы комбинированного алгоритма от размера лабиринта и доли тупиков в лабиринте;
* сравнить комбинированный алгоритм с точным по времени и качеству работы.

Объектом исследования в данной работе является комбинированный алгоритм поиска кратчайшего пути в заданном лабиринте произвольных размеров.

Предметом исследования является время и качество работы комбинированного алгоритма поиска кратчайшего пути в заданном лабиринте произвольных размеров.

Информационной базой исследования являются результаты работы программы. Методологическая основа исследования включает в себя эмпирический метод (многократный запуск программы на разных лабиринтах и получение результатов её работы), сбор статистической информации (подсчёт среднего времени работы точного и комбинированного алгоритмов и среднего значения длины пути, найденного комбинированным алгоритмом), графический метод (построение графиков зависимостей времени и качества работы комбинированного алгоритма от размера лабиринта и доли тупиков в лабиринте), анализ полученных зависимостей, синтез выводов, аналогия (механизм работы программы сравнивается с механизмом эволюции живых организмов).

Научная новизна работы заключается в том, что предлагается новый приближённый алгоритм решения задачи поиска кратчайшего пути в лабиринте.

Теоретическая и практическая значимость работы состоит в широком применении алгоритма решения задачи нахождения кратчайшего пути в заданном лабиринте произвольных размеров в различных областях. Например, данная задача необходима в игровой индустрии, пошаговых стратегиях или стратегиях в реальном времени. Кроме того, подобная задача применяется при трассировке печатных плат, когда необходимо быстро и эффективно расположить на микросхемах электропроводящие цепи. Также, алгоритм с поиском нескольких маршрутов необходим при проектировке планов пожарных выходов, для быстрого поиска оптимальных маршрутов, например, в больших зданиях или когда доступ к кратчайшим ограничен.

# Общие сведения о лабиринтах и графах

## Понятие лабиринта

Лабиринт - какая-либо структура (обычно в двухмерном или трёхмерном пространстве), состоящая из запутанных путей к выходу или в тупики [1]. Лабиринт можно представить, как граф, в котором каждая вершина является достижимой или нет. Т.е. задача может свестись к поиску пути в неориентированном планарном графе с невзвешенными ребрами.

Для понимания условий необходимо классифицировать лабиринты.

В зависимости от генерации лабиринта они могут различаться по маршрутизации.

Лабиринт с одиночным соединением – лабиринт без петель или замкнутых цепей и без недостижимых областей [2]. В таком лабиринте из одной точки в любую другую существует только один единственный путь. Отсюда можно понять, что решение у данного лабиринта будет существовать всегда и притом только одно. Пример такого лабиринта представлен на рисунке 1. С точки зрения программирования такой лабиринт можно описать как дерево, связующее множество ячеек или вершин.

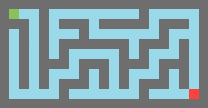


Рисунок 1 - Идеальный лабиринт

Лабиринт с многократными соединениями – в таком лабиринте используются проходы, замыкающиеся и возвращающиеся друг к другу, также в данном лабиринте отсутствуют тупики [2]. В подобной структуре решений будет несколько, т.к. во многие развилки можно попасть несколькими путями, из-за чего некоторые решение будет короче либо длиннее других. Данный лабиринт представлен на рисунке 2. Такой лабиринт можно представить, как связный граф, в котором не существует вершин с одним смежным ребром.



Рисунок 2 - Плетенный лабиринт

Одномаршрутный лабиринт – в нем содержится один длинный извивающийся проход, который меняет направление на всем протяжении лабиринта [2]. В данной постановке решение будет единственным. Таким образом, что игрок должен пройти лабиринт целиком. Подобный лабиринт показан на рисунке 3. Данную структуру можно представить в виде простой цепи.



Рисунок 3 - Одномаршрутный лабиринт

Переплетенный – в рамках прохождения лабиринта данная структура не отличается от идеального с одиночным соединением, в которых, однако, пути могут накладываться друг на друга. Подобный лабиринт можно представить на рисунке 4.

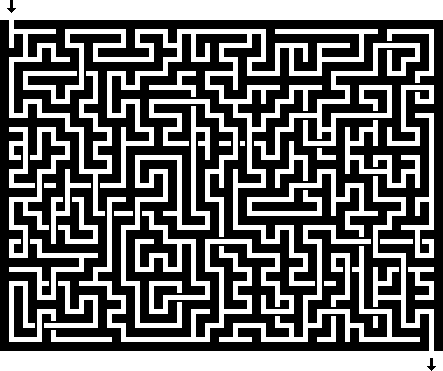


Рисунок 4 - Переплетенный лабиринт

Если представить каждую клетку как узел, который является достижимым (проход) или недостижимым (стена), то задачу поиска пути в лабиринте можно свести к поиску пути в неориентированном планарном графе. В случае нескольких решений появляется необходимость нахождения кратчайшего пути в лабиринте и соответственно на графе.

## Представление клеток лабиринта

Для упрощения работы с лабиринтом было принято решение в представлении клеток лабиринта в окрестностях Мура – в двумерном пространстве, совокупность восьми клеток на квадратном паркете, имеющих общую вершину с данной клеткой. Пример представления окрестности Мура показан на рисунках 5 и 6. Для более точных расчетов расстояния между соседними клетками будут представлены следующим образом: вертикальные и горизонтальные как – 1, а диагональные как – 1.45.

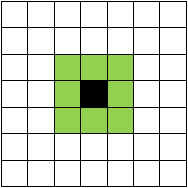


Рисунок 5 - Двумерная окрестность Мура порядка 1

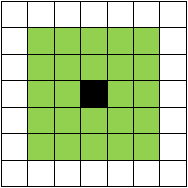


Рисунок 6 - Двумерная окрестность Мура порядка 2

## Сведение задачи к поиску кратчайшего пути в графе

Граф G(V,E) называется совокупность двух множеств – непустого множества V(множества вершин) и множества E двухэлементных подмножеств множества V(E – множество ребер) [7]. В зависимости от наличия или отсутствия ориентированных ребер в графе, графы можно поделить на:

* ориентированные графы (рисунок 7);

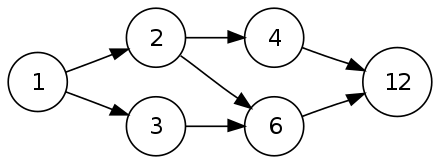


Рисунок 7 - Пример ориентированного графа

* неориентированные графы (рисунок 8).

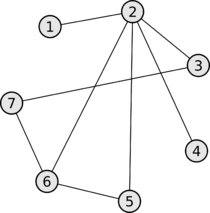


Рисунок 8 - Пример неориентированного графа

Вес ребра – значение, которое ставят в соответствие данному ребру. Обычно под весом подразумевают «длину» ребра.

## Планарность

Планарный граф — структура, которая на плоскости может быть изображена таким образом, чтобы не было пересечений рёбер. Другими словами, граф является планарным, если он изоморфен некоторому плоскому графу, то есть графу, изображённому на плоскости так, что его вершины – это точки плоскости, а рёбра — непересекающиеся кривые на ней.

Не все обыкновенные графы являются планарными. Среди непланарных графов выделяются два графа — полный граф с пятью вершинами K5, представленный на рисунке 9.

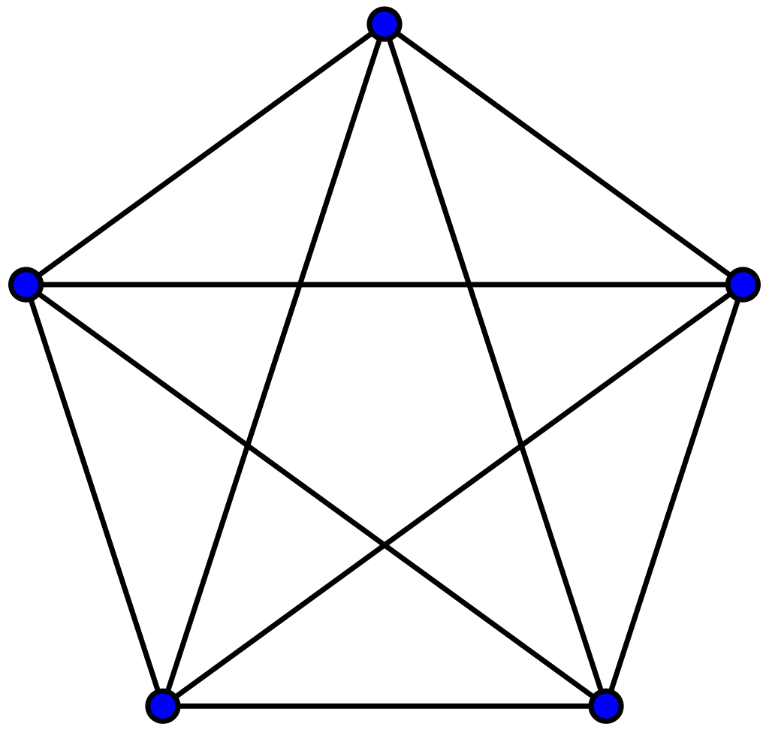


Рисунок 9 - Полный K5 граф

Пример полного двудольного графа с тремя вершинами в каждой доле K3,3 представлен на рисунке 10.

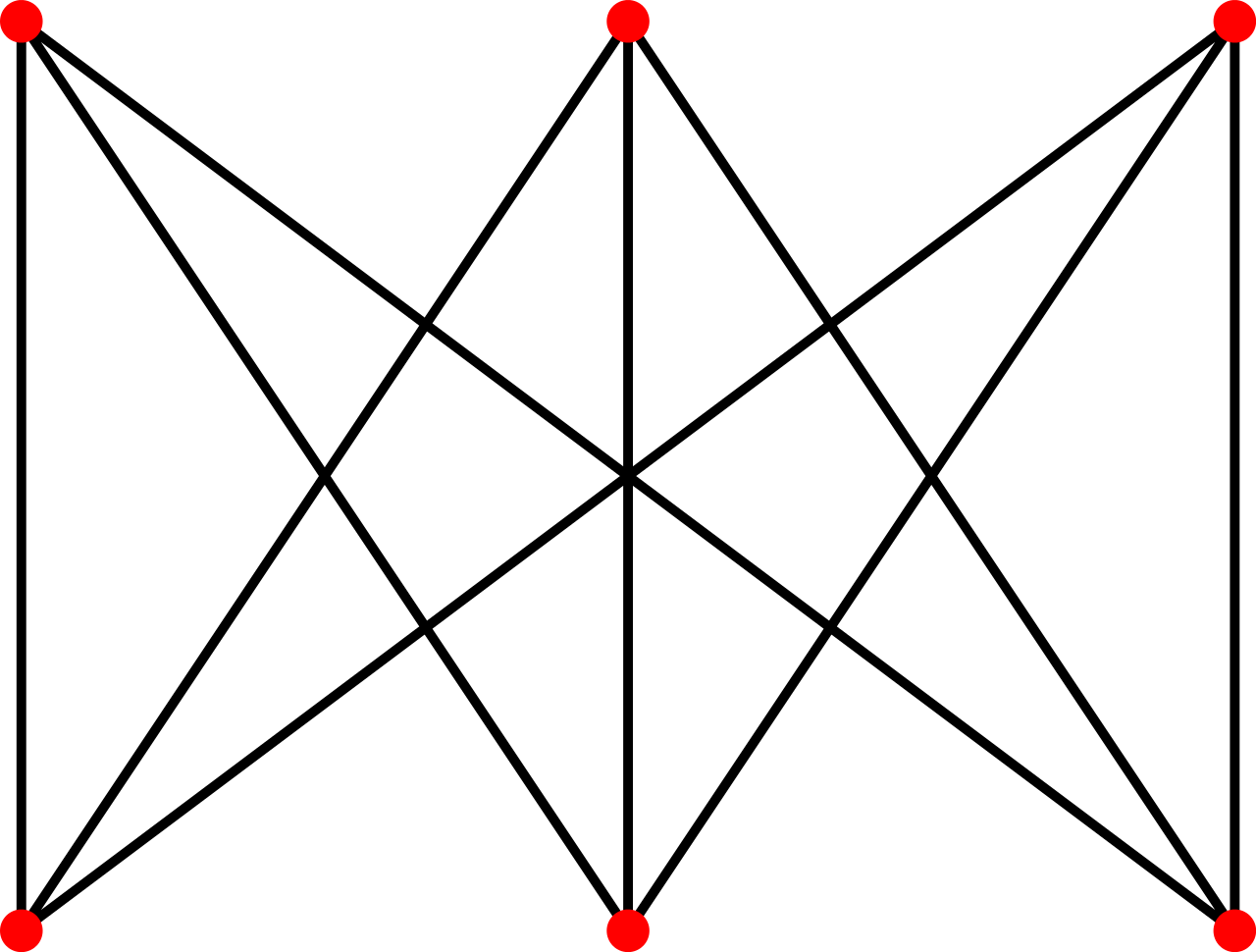


Рисунок 10 - Полный двудольный K3,3 граф

Теорема Понтрягина-Куратовского – графы K5 и K3,3 являются единственными минимальными непланарными графами [3].

Теорема Вагнера – конечный граф является планарным тогда и только тогда, когда его миноры не включают ни K5, ни K3,3 [3].

## Маршруты, пути и связность

Маршрут в графе – чередующаяся последовательность вершин и рёбер, в которой любые два соседних элемента инцидентны. Если вершина начала и конца совпадают, то маршрут является замкнутым, иначе открытым.

Цепь – маршрут, все ребра которого различны. Также есть простая цепь, когда различны еще и вершины [7].

Путь – последовательность рёбер, такая, что конец одного ребра является началом другого [7].

Две вершины называются связными, если существует соединяющая их цепь. Граф называется связным, если между всеми вершинами выполняется условие связности [7]. Связным графом на примере лабиринта, в котором все узлы связны, показан на рисунке 11.

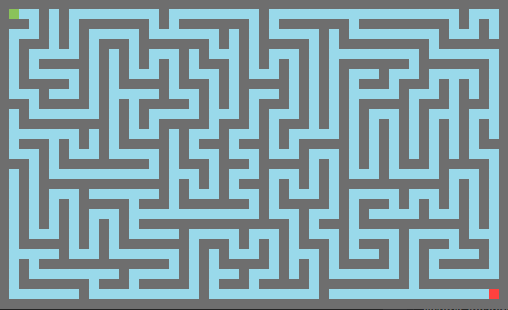


Рисунок 11 - Лабиринт со связными вершинами

Если в лабиринте существуют изолированные области то, соответствующий ему граф является несвязным. Несвязный граф на примере лабиринта показан на рисунке 12.

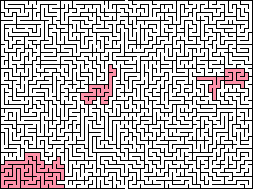


Рисунок 12 - Лабиринт с изолированными вершинами

На данном рисунке можно увидеть области, отмеченные розовым цветом, эти области изолированы от общего лабиринта.

# Математическая постановка задачи о пути в лабиринте

Задачу поиска кратчайшего пути называют классической задачей теории графов. Ее суть заключается в нахождении пути между двумя вершинами в графе, таким образом, чтобы сумма весов ребер, составляющих путь, была минимальна.

Существуют различные постановки задачи о кратчайшем пути.

* 1. Кратчайший путь, проходящий через заданное множество вершин. В данном случае рассматривается два ограничения: кратчайший маршрут проходит через выбранные вершины, и кратчайший маршрут должен проходить через минимальное число невыделенных вершин.
  2. Кратчайший путь между всеми парами узлов графа. В данной постановке необходимо найти кратчайший маршрут из любой вершины u в любую вершину v.
  3. Кратчайший путь между выбранной парой вершин. Необходимо найти кратчайший маршрут из выбранной вершины u в выбранную вершину v.

В данной работе будет рассмотрена задача поиска пути между двумя выбранными вершинами.

Существует несколько алгоритмов решения данной задачи:

* алгоритм Дейкстры находит кратчайшие пути от одной из вершин до всех остальных;
* алгоритм поиска А\* находит кратчайший маршрут от одной вершины к другой, используя эвристическую функцию стоимости и расстояния вершин;
* алгоритм волновой трассировки основан на алгоритме поиска в ширину находит маршрут из стартовой вершины в конечную в невзвешенном графе;
* алгоритм следования вдоль стен - простой алгоритм для решения лабиринта, единственным преимуществом которого над другими алгоритмами, является то, что он не требует затрат в памяти и применим игроком или роботом.

## Алгоритм Дейкстры

Алгоритм Дейкстры – находит кратчайшее расстояние от одной из вершин графа до всех остальных [4].

Каждую ячейку графа ставим в соответствие минимальное известное расстояние из этой вершину в стартовую. Данный алгоритм работает пошагово, посещая на каждом этапе единственную вершину. Алгоритм завершает свою работу в тот момент, когда заканчиваются все непосещенные вершины графа. Начальный граф представлен на рисунке 13.

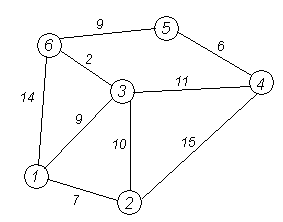


Рисунок 13 - Начальный граф

1. На этапе инициализации, стартовая вершина помечается расстоянием равным 0, оставшиеся вершины помечаются каким-либо большим числом, например, суммой весов всех ребер графа. Также все ячейки помечаются как не посещенные. Граф с помеченными вершинами после инициализации показан на рисунке 14.

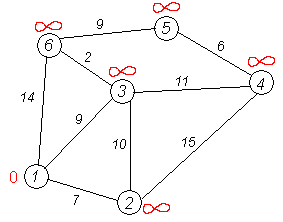


Рисунок 14 - Граф после инициализации

1. На каждом шаге происходит проверка, если все проходимые узлы графа пройдены, то алгоритм завершает свою работу. Иначе выбирает из еще не посещенных вершин следующую ячейку u с минимальным расстоянием, показан на рисунке 15. Находятся все соседние вершины данного узла. Так как нам известно кратчайшее расстояние до u вычисляются все маршруты в соседние вершины из u складывая расстояния u + вес ребра, ведущего в соседний узел. Если полученное расстояние меньше значения метки соседа, то заменяем расстояние на наименьшее. Обход всех соседних вершин показан на рисунках 16 и 17. Перебрав все соседние вершины, текущий узел u помечается, как посещенный – рисунок 18, и алгоритм повторяет пункт 2.

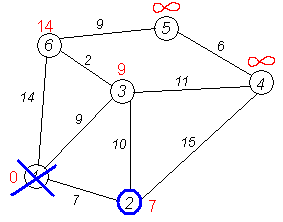


Рисунок 15 - Выбрана текущая вершина 2

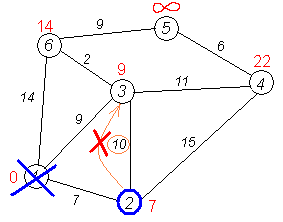


Рисунок 16 - Сравнение расстояния из вершины 2 в 3 с текущим значением в вершине 3

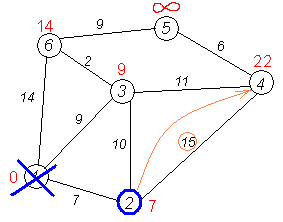


Рисунок 17 - Сравнение расстояния из вершины 2 в 4 с текущим значением в вершине 4

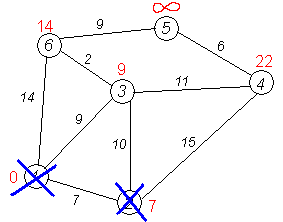


Рисунок 18 - Вершина 2 помечается как посещенная

Алгоритм Дейкстры не является эффективным для данной задачи, т.к. для его завершения необходим полный обход матрицы узлов.

## Алгоритм А\*

Поиск A\* в информатике и математике– это алгоритм поиска по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от одной вершины к другой [5].

Порядок обхода вершин определяется эвристической функцией, представленной формулой (1.1):

f(x)=g(x)+h(x) (1.1)

где

g(x) - функция стоимости достижения рассматриваемой вершины x из начальной;

h(x) - функция эвристической оценки расстояния от рассматриваемой вершины к конечной.

Пример представления и инициализации данного алгоритма показан на рисунке 19.

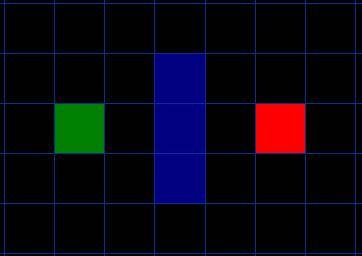


Рисунок 19 – Начальное представление. Зеленой клеткой указан стартовый узел, красным – финишный. Синими клетками обозначены непроходимые элементы – стены

Приведем шаги выполнения алгоритма:

1. заносим начальный узел в открытый список;
2. находим в открытом списке ячейку с наименьшей стоимостью f(x), она становится текущей;
3. удаляем текущую клетку из открытого списка и добавляем в закрытый;
4. для всех соседних узлов проводим проверку, если не выполняются пропустить 4 шаг:

* если она является проходимой и не находится в закрытом спискеаппро;
* если она не находится в открытом списке;

1. добавляем ячейку в открытый список, с вычислением функции стоимости f(x), помечается родитель для данной клетки;
2. цикл повторяется с 1 шага, пока:

* пока в открытый список не добавлен финишный узел, путь найден;
* открытый список не пуст, путь не найден;

1. если путь найден, то восстанавливается маршрут от финишной клетки до стартовой от каждой клетки к родителю;
2. если путь не найден вывести ошибку.

Решение начального примера показан на рисунке 20.

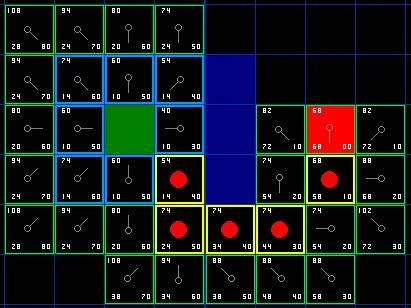


Рисунок 20 - Найденное решение показано красными точками

Данный алгоритм является модификацией алгоритма Дейкстры, с добавлением эвристики.

## Алгоритм волновой трассировки

Алгоритм Ли — алгоритм поиска кратчайшего пути на планарном графе. Принадлежит к алгоритмам, основанным на методах поиска в ширину [6].

Принцип выполнения волнового алгоритма можно разделить на 3 шага:

* инициализация;
* распространение волны;
* восстановление пути.

Во время инициализации строится образ множества ячеек обрабатываемого поля, каждой ячейке приписываются атрибуты проходимости и непроходимости, запоминаются стартовая и финишная ячейки.

Далее, от стартовой ячейки порождается шаг в соседнюю ячейку, при этом проверяется, проходима ли она, и не принадлежит ли ранее меченной в пути ячейке.

При выполнении условий проходимости и непринадлежности её к ранее помеченным в пути ячейкам, в атрибут ячейки записывается число, равное количеству шагов от стартовой ячейки, от стартовой ячейки на первом шаге это будет 1. Каждая ячейка, меченная числом шагов от стартовой ячейки, становится стартовой и из неё порождаются очередные шаги в соседние ячейки. Очевидно, что при таком переборе будет найден путь от начальной ячейки к конечной, либо очередной шаг из любой порождённой в пути ячейки будет невозможен.

Восстановление кратчайшего пути происходит в обратном направлении: при выборе ячейки от финишной ячейки к стартовой на каждом шаге выбирается ячейка, имеющая атрибут расстояния от стартовой на единицу меньше текущей ячейки. Очевидно, что таким образом находится кратчайший путь между парой заданных ячеек. Результат работы алгоритма показан на рисунке 21.

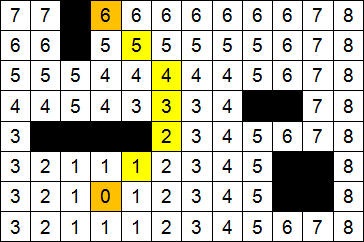


Рисунок 21 - Пример работы алгоритма Ли

## Алгоритм следования вдоль стен

Один из наиболее простых алгоритмов решения лабиринта. Суть работы заключается в постоянном следовании вдоль одной стены. Начиная обход лабиринта и при достижении развилки всегда поворачивать направо (либо всегда налево). Решение данным методом не является эффективным, а также данный алгоритм не находит решение, если выход находится в центре лабиринта и его окружает замкнутая цепь. В таком случае алгоритм вернется в стартовую клетку откуда и вышел.

Главными преимуществами такого подхода являются:

* отсутствие дополнительных затрат памяти;
* применение в реальной жизни человеком или в робототехнике.

Пример обхода лабиринта с помощью алгоритма одной руки показан на рисунке 22.

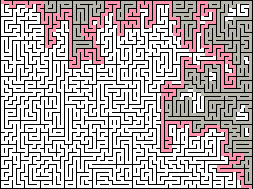


Рисунок 22 - Обход лабиринта методом следования вдоль стен

## Муравьиный алгоритм

Алгоритм муравьиной колонии – полиномиальный алгоритм для нахождения приближенного решения поиска маршрутов в графах. Подход данного алгоритма, заключается в моделировании поведения колонии муравьев, которые ищут путь от муравейника к источнику пищи.

В природе муравьи двигаются в случайном направлении и после того, как найдут источник пищи возвращаются обратно в муравейник, прокладывая феромонную дорожку. Если другие муравьи найдут эту тропу, с большей вероятностью выберут эту тропу. Находя еду, они больше укрепляют текущую тропу. Со временем тропа из феромонов испаряется, в следствие чего привлекательность для муравьев снижается, а также уменьшая вероятность выбора таких троп. Чем длиннее маршрут от муравейника до еды и обратно, тем выше испарение феромонов. Однако короткие маршруты будут проходить быстрее и их феромонные тропы будут сохранятся. Испарение феромонов с троп нужно для отсутствия локально-оптимальных решений. Без испарения феромонов, муравьи использовали бы только первый найденный путь до еды и для них он был единственным привлекательным. Пример поведения муравьиной колонии представлен на рисунке 23.

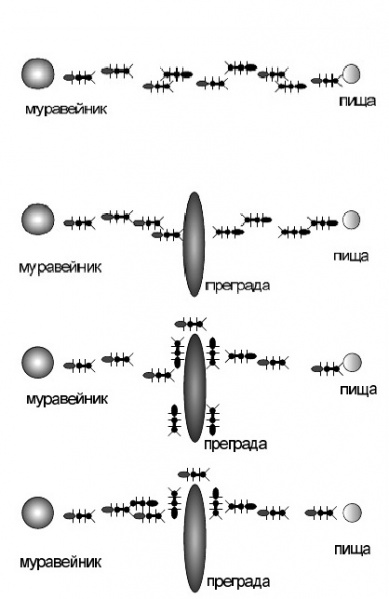


Рисунок 23 - Поведение муравьиной колонии

Алгоритм муравьиной колонии можно представить, как цикл содержащий 3 этапа.

1. Создание муравьев: выбирается стартовая точка, куда размещаются муравьи. Они могут располагаться все в одной точке, в разных с повторением или в разных без повторений. Также на данном этапе инициализируется начальное значение феромонов, чтобы на первом шаге вероятности перехода муравьев в соседние вершины не были нулевыми.
2. Поиск решения: вычисляется вероятность перехода муравья из вершины i в вершину j по следующей формуле (2.1):

[8] (2.1)

где

– уровень феромона;

– расстояние из i в j вершину;

– константные параметры.

1. Обновление феромонов: уровень феромонов изменяется с помощью следующие формулы (2.2):

[8] (2.2)

где

– интенсивность испарения;

– цена текущего маршрута для k-го муравья;

Q – параметр, имеющий значение порядка цены оптимального решения.

## Комбинированный алгоритм

В данной работе было принято решение комбинации волнового алгоритма и алгоритма муравьиной колонии.

Данные методы были выбраны в следствие эффективности волнового алгоритма на решении лабиринтов, а также адаптивность и общий концепт муравьиного алгоритма для решения задачи поиска кратчайшего пути на графе.

В случае начально заданной матрицы феромонов вероятностный подход муравьиного алгоритма не имеет значительного эффекта, т.к. выбор пути на развилке будет равным в любом направлении. Из-за чего множество итераций муравьев не будет достигать выхода из лабиринта.

Следовательно, необходимо задать муравьям хотя бы один достижимый маршрут. Для этого используется этап распространения волны из алгоритма волновой трассировки. Данный этап проходит по лабиринту, находя все маршруты до финишной клетки. Далее по полученным данным формируются феромонные тропы на матрице узлов инцидентной заданному лабиринту. После чего запускается этап муравьиного алгоритма.

Задаются начальные параметры эвристик:

* α – параметр влияния феромонов на клетке;
* β – параметр влияния веса ребра;
* ρ – скорость испарения феромона с клетки.

Имея начальные параметры феромонных троп, по которым муравьи могут следовать, мы уменьшаем вероятность блуждания муравьев по лабиринту, но также позволяем алгоритму находить более оптимальные маршруты. Также для ухода от локальных минимумов, мы задаем обновление феромонов только для тех муравьев, которые достигли финишной клетки, иначе из-за большого количество переборов, феромоны на клетках будут быстро испаряться, что потеряет всю эффективность этапа распространения волны.

## Пример реализации

Дана матрица узлов (двумерный массив), каждый элемент массива представляет узел (клетку массива), а его значением является проходимость этой клетки. Необходимо найти путь от стартовой клетки до конечной, учитывая непроходимость некоторых клеток (стен).

Добавляем стартовую клетку в очередь присваиваем ей значение d=1. Проверяем всех ближайших соседей, по принципу:

* является ли клетка проходимой;
* не посещена ли клетка ранее.

Добавляем всех соседей в очередь и присваиваем им значения d+1, удаляем стартовую клетку из очереди.

Если были посещены все достижимые клетки алгоритм переходит к этапу муравьев.

По полученным маршрутам создается феромонная тропа на матрице вершин лабиринта. Создаются муравьи на стартовой клетке. Задаются значения эвристик α, β, ρ. Т.к. в случае комбинированного алгоритма начальные феромоны уже существуют большее значение будет иметь параметр α, также следует понимать, что длины ребер в подобном лабиринте будут равными, кроме случаев, когда алгоритм может идти по диагонали. Следовательно, параметр β не должен превышать 1,2 значения.

Далее происходит этап поиска решения. Так как начальная феромонная тропа уже существует, то муравьи с большей вероятностью будут двигаться в правильном направлении, а не блуждая по лабиринту.

После прохождения муравьями итерации происходит обновление феромонных троп. Запоминается кратчайший найденный путь на итерации.

Алгоритм заканчивает свою работу, после того как будет найден кратчайший путь через некоторое количество итераций.

# Реализация приложения

## Основные сведения о программе

Для исследования алгоритма волновой трассировки была разработана программа с использованием следующих фреймворков: XNA Framework и MonoGame, в которой можно реализовать произвольный планарный граф с помощью 2D Текстуры.

Результатом работы программы является визуальное представление кратчайшего пути.

На рисунке 8 показана начальное окно программы. Генерируется сетка для последующей генерации лабиринта.

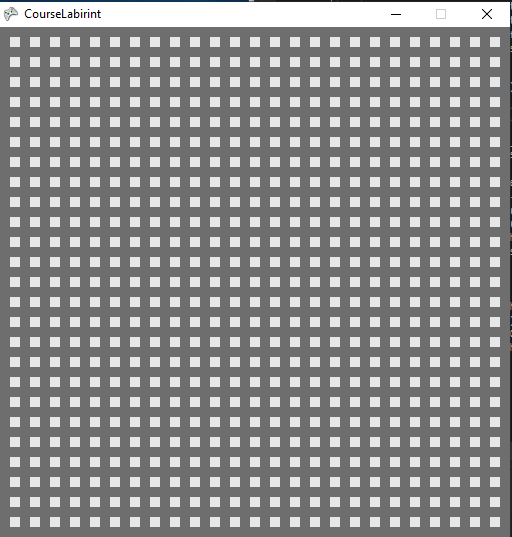


Рисунок 24 - Начальное состояние программы

Далее с помощью алгоритма поиска в глубину начинается генерация лабиринта. В конечном результате получается идеальный лабиринт без петель и изолированных областей. Для большего количества маршрутов выполняется удаление которых непроходимых клеток. В результате работы получается конечная матрица узлов, изображенная на рисунке 25. Данный метод генерации позволяет избежать наличия недостижимых областей в графе.

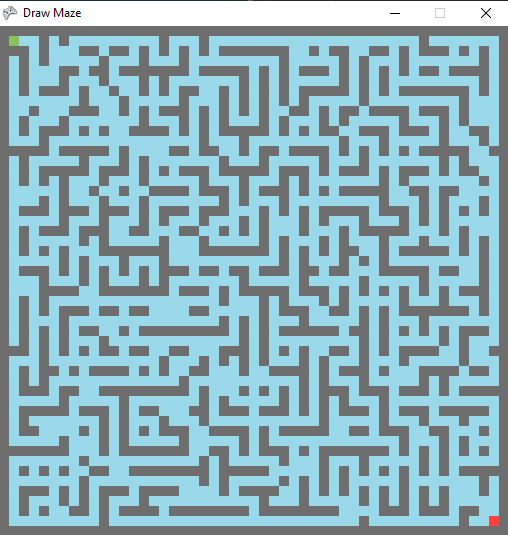


Рисунок 25 - Конечная матрица узлов

После нажатия клавиши “Пробел” происходит обход всей матрицы, с пометкой посещенных узлов и расстояния до стартового узла. Это необходимо для последующего построения феромонных троп. Один из этапов прохода матрицы показан на рисунке 26.

Далее запускается алгоритм муравьиной колонии, задаются начальные параметры метода, запускается движение муравьев. После некоторого количества итераций находится кратчайший найденный путь от старта до финиша и алгоритм завершает свою работу. Если необходимо оптимизировать маршрут или начать поиск другого пути, нажимается кнопка “F2”.

Изображение конечного результата показано на рисунке 27.

При реализации данного приложения был использован язык программирования C# и среда разработки Visual Studio. Этапы выполнения программы показаны на рисунке 28.

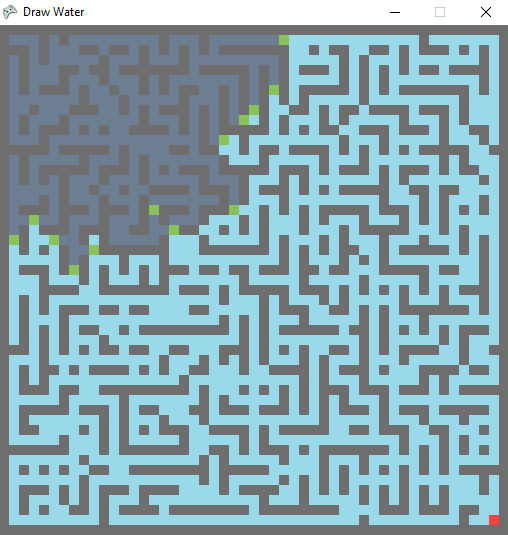


Рисунок 26 - Распространение волны

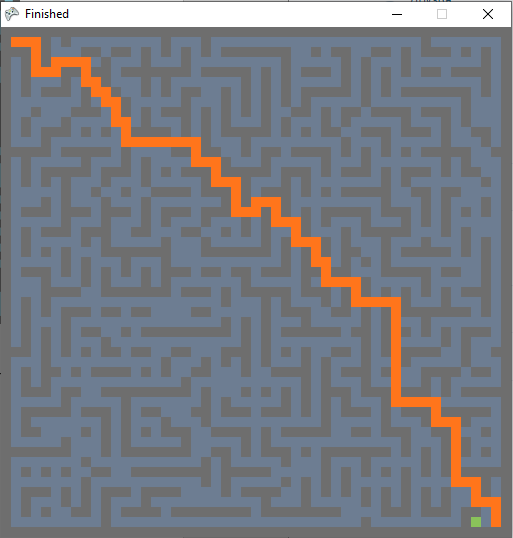


Рисунок 27 - Конечный результат

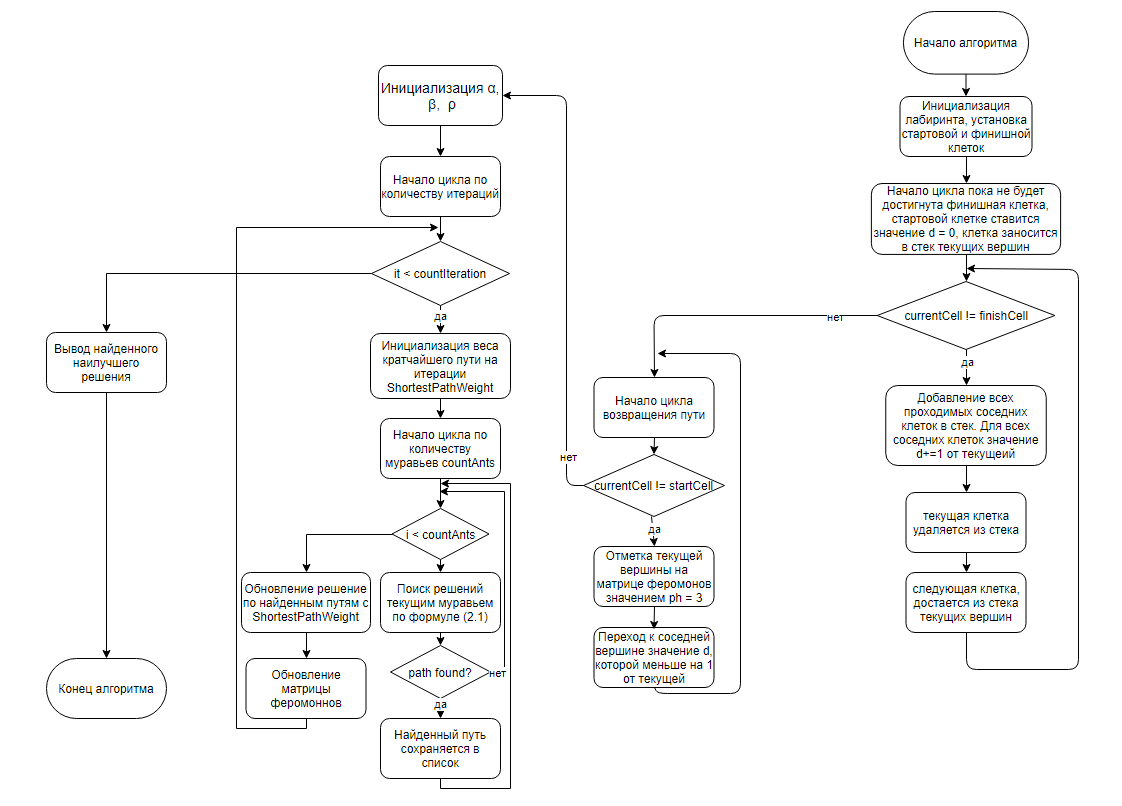


Рисунок 28 - Блок схема комбинированного алгоритма

## Особенности реализации

Программа включает следующие основные модули(файлы):

а) Program.cs – основной файл, отвечающий за инициализацию игрового окна;

б) Game1.cs – файл игрового окна, основной файл программы, с предустановленными методами XNA Framework содержит в себе:

1) Game1 – конструктор класса, задаёт первоначальные настройки окна, задает координаты для стартового и финишного узлов;

2) Initialize – предустановленный метод фреймворка, отвечающий за инициализацию;

3) MainPoint – метод, визуализирующий стартовый и финишный узлы;

4) DrawMaze – метод, рисующий идеальный лабиринт;

5) BreakWalls – метод, удаляющий непроходимые узлы;

6) GetNeighbours – метод, проверяющий не посещённых соседей для генерации лабиринта;

7) RemoteWalls – метод, соединяющий узлы, путем удаления между ними стен;

8) Path – метод, реализующий этап алгоритма Ли – распространение волны;

9) GetNeighboursPath – метод, возвращающий всех достижимых соседей текущего узла;

10) PathBack – метод, который формирует феромонные тропы;

11) GetNeighboursPathBack – функция, возвращающая соседний узел, помеченный на 1 меньше числа текущего узла;

13) AntsColony – метод, реализующий муравьиный алгоритм;

14) GetAntbours – метод, возвращающий всех достижимых соседей для текущего муравья;

12) LoadContent – предустановленный метод фреймворка, отвечает за первоначальную загрузку текстуры, в данном случае сетки;

13) Update – предустановленный метод фреймворка, основной в программе, отвечает за обновление окна программы в зависимости от времени, также в нем реализованы кнопки перехода между этапами на “Пробел” и запуск муравьиного алгоритма снова на “F2”;

14) Draw – предустановленный метод фреймворка, отвечает за отрисовку текстур во время работы;

в) Matrix.cs –файл для отрисовки сетки текстуры;

г) Ants.cs – файл, хранящий феромонные тропы на матрице узлов.

# Анализ эффективности алгоритма волновой трассировки

Для анализа эффективности работы разработанного комбинированного алгоритма использовалось два параметра – время, затраченное программой на решение задачи и размерность поля лабиринта.

Сравнение разработанного алгоритма будем проводить с различными известными алгоритмами:

* комбинированный алгоритм – происходит начальная генерация феромонных троп;
* муравьиный алгоритм – лабиринт с несколькими маршрутами.

На рисунке 29 показана зависимость времени работы волнового алгоритма от размерности матрицы узлов для трех вариаций сложности.

Рисунок 29 - Зависимость времени работы алгоритмов от размерности матрицы узлов

Также в таблице 1 представлены сравнительные значения работы алгоритма.

Таблица 1 - Сравнительные значения работы алгоритма

|  |  |  |
| --- | --- | --- |
| Размерность матрицы | Муравьиный алгоритм | Комбинированный алгоритм |
| 50 | 25,3 мс | 30,2 мс |
| 100 | 70,6 мс | 61,2 мс |
| 250 | 841,1 мс | 311,2 мс |
| 500 | 3512,7 мс | 1120,3 мс |
| 1000 | 10501,2 мс | 4515,4 мс |

По полученным данным можно сделать вывод, что комбинированный алгоритм имеет сложность O(t\*n2\*m), где t – время жизни колонии, т.е. число итераций, n – количество вершин графа, m – количество муравьев в колонии. Также можно понять, что разработанный алгоритм работает эффективнее муравьиного, т.к. без начальных данных вероятностный подход к решению теряет свою эффективность, т.к. алгоритму требуется больше времени для генерации хотя бы первого маршрута, который бы в последствии улучшался.

# ЗАКЛЮЧЕНИЕ

Задача поиска кратчайшего пути в лабиринте является NP-полной, т.е. гарантировано решается только полным перебором. Поэтому разработка и реализация новых методов ее решения является актуальной. Дальнейшее изучение эвристических алгоритмов, основанных на природных явлениях, для планарных графов является важным направлением.

В данной работе были изучены наиболее актуальные для данной задачи алгоритмы поиска. Был реализован алгоритм на основе волнового и муравьиного алгоритмов для решения задачи о кратчайшем пути в лабиринте. Произведено сравнение разработанного алгоритма с алгоритмом муравьиной колонии, а также проведено сравнение по времени выполнения и результатам работы. По результатам экспериментальных исследований удалось выяснить что полученный алгоритм эффективнее обычного муравьиного, однако выяснилось, что подобные эвристические методы, не являются эффективными на планарных графах.

Разработанный комбинированный алгоритм решения задачи нахождения кратчайшего пути в заданном лабиринте произвольных размеров можно использовать при трассировке печатных плат, когда необходимо быстро и эффективно расположить на микросхемах электропроводящие цепи. Также, алгоритм с поиском нескольких маршрутов необходим при проектировании планов пожарных выходов, для быстрого поиска оптимальных маршрутов, например, в больших зданиях, или, когда доступ к кратчайшим маршрутам ограничен.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Керн, Г.** Лабиринты мира / Германн Керн ; [пер. с англ. А. Рудаковой, Л. Шведовой ; пер. с лат. Н. Горелова ; пер. с итал. и нем. И. Хмелевских]. — Санкт-Петербург : Азбука-классика, 2007. — 430, [1] с. : ил., портр., табл., факс.; 30 см. — ISBN 978-5-352-02036-4. — Текст: непосредственный. [Это учебное пособие с тремя авторами]
2. **Варламова, Л. Н.** Управление документацией : англо-русский аннотированный словарь стандартизированной терминологии / Л. Н. Варламова, Л. С. Баюн, К. А. Бастрикова. — Москва : Спутник+, 2017. — 398 с . ; 21 см. —Библиогр.: с. 358—360. — 100 экз. — ISBN 978-5-9973-4489-4. — Текст: непосредственный. [Это книга-словарь с тремя авторами]
3. Распределенные интеллектуальные информационные системы и среды : монография / А. Н. Швецов, А. А. Суконщиков, Д. В. Кочкин [и д р .]; Министерство образования и науки Российской Федерации, Вологодский государственный университет. — Курск: Университетская книга, 2017. — 196 с . : и л .; 20 см. — Библиогр.: с. 192—196. —500 экз. — ISBN 978-5-9909988-3-4. — Текст : непосредственный. [Это монография более четырех авторов]
4. **Жукова, Н. С.** Инженерные системы и сооружения. Учебное пособие. В 3 частях. Часть 1. Отопление и вентиляция / Н. С. Жукова, В. Н. Азаров; Министерство образования и науки Российской Федерации, Волгоградский государственный технический университет. — Волгоград : ВолгГТУ, 2017. — 89, [3] с. : ил. ; 21 см. — Библиогр.:с. 92. — 65 экз. — ISBN 978-5-9948-2526-6. — Текст : непосредственный. [Учебное пособие несколько разделов или частей]
5. Российская книжная палата : [сайт]. - 2018. - URL: http://bookchamber.ru/isbn.html (дата обращения: 22.05.2018). - Текст : электронный. [Сайт]
6. **Грязев, А.** Пустое занятие : кто лишает Россию права вето в СБ ООН : в ГА ООН возобновлены переговоры по реформе Совета Безопасности / А. Грязев. - Текст : электронный // Газета.ru : [сайт]. - 2018. - 2 февр. - URL: https://www.gazeta.ru/politics/2018/02/02\_a\_11634385.shtml (дата обращения: 09.08.2019). [Статья с сайта в сети интернет]
7. Мониторинг общественного мнения: экономические и социальные перемены / учредитель Всероссийский центр изучения общественного мнения ; главный редактор журнала Федоров В. В. — 1992 —. — Москва, 2015 — . — 200—350 с. — Выходит 6 раз в год. — ISSN 2219-5467. — Текст : электронный. 2015, № 1 (125) — 6 (130). — URL: <https://wciom.ru/books_magazines/zhurnal_monitoring/arkhiv_vypuskov> (дата обращения: 29.12.2015). [Ссылка на журнал размещенный в интернете]
8. **Скрипник, К. Д.** Лингвистический поворот и философия языка Дж. Локка: интерпретации, комментарии, теоретические источники / К. Д. Скрипник. -Текст : непосредственный // Вестник Удмуртского университета. Серия: Фило-софия. Психология. Педагогика. - 2017. - Т. 27, вып. 2. - С. 139-146. [Это статья из обычного журнала]
9. **Якишин, Ю. В.** Управление структурой экономики региона в нестабильной среде / Ю. В. Якишин. - Текст : электронный // Вестник Самарского государственного экономического университета. - 2019. - № 5 (175). - С. 9-16. - URL: http://vestnik.sseu.ru/index.php?cnt=1&idv=359 (дата обращения: 05.12.2019). [Статья из журнала размещенного в интернете]
10. **Янушкина, Ю. В.** Исторические предпосылки формирования архитектурного образа советского города 1930—1950-х гг. / Ю. В. Янушкина. — Текст: электронный //Архитектура Сталинграда 1925—1961 гг. Образ города в культуре и его воплощение : учебное пособие / Ю. В. Янушкина ; Министерство образования и науки Российской Федерации, Волгоградский государственный архитектурно-строительный университет. — Волгоград : ВолГАСУ, 2014. — ISBN 978-5-982766-693-9. — Раздел 1. — С. 8—61. — URL: <http://vgasu.ru/attachments/oi_yanushkina_01.pdf> (дата обращения: 20.06.2018). [Раздел учебного пособия размещенного в интернете]

# ПРИЛОЖЕНИЕ А

**Основная программа**

Файл Game1.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Net;

using System.Runtime.Remoting.Metadata.W3cXsd2001;

using System.Threading;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Content;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

using System.Diagnostics;

namespace CourseLabirint

{

public class Game1 : Game

{

public struct Cells

{

public int X;

public int Y;

public Cells(int newX, int newY)

{

X = newX;

Y = newY;

}

}

private int cell\_size = 10;

int index=-1;

private SpriteBatch \_spriteBatch;

private readonly Texture2D[,] \_maze;

private static Cells \_size;

private readonly Cells \_start;

private readonly Cells \_finish;

private int \_shc;

private int \_fhc;

private int \_whc;

private int \_vhc;

private int \_vpath;

private readonly List<Cells> \_neighbours;

private readonly List<double> \_weights;

private readonly Stack<Cells> \_path;

private readonly Queue<Cells> \_dfs;

private int \_status;

private int check\_key=0;

private int winWidth;

private int winHeight;

private int[,] mat;

int d;

Stopwatch stopwatch = new Stopwatch();

Ants ants;

public Game1(List<Cells> neighbours, Stack<Cells> path)

{

\_neighbours = neighbours;

\_weights = new List<double>();

\_path = path;

\_dfs = new Queue<Cells>();

var graphics = new GraphicsDeviceManager(this);

Content.RootDirectory = "Content";

winWidth = graphics.PreferredBackBufferWidth = 20\*cell\_size + cell\_size;

winHeight = graphics.PreferredBackBufferHeight = 20\*cell\_size + cell\_size;

\_size = new Cells(winWidth / cell\_size, winHeight / cell\_size);

\_start = new Cells(1, 1);

\_finish = new Cells(\_size.X-2, \_size.Y - 2);

\_maze = new Texture2D[\_size.X, \_size.Y];

\_path.Push(\_start);

\_path.Push(\_start);

IsMouseVisible = true;

mat = new int[\_size.X, \_size.Y];

}

protected override void Initialize()

{

base.Initialize();

}

protected override void UnloadContent()

{

}

//Generation

private void MainPoint()

{

\_maze[\_start.X, \_start.Y] = Content.Load<Texture2D>("start");

\_shc = \_maze[\_start.X, \_start.Y].GetHashCode();

\_maze[\_finish.X, \_finish.Y] = Content.Load<Texture2D>("finish");

\_fhc = \_maze[\_finish.X, \_finish.Y].GetHashCode();

ants = new Ants(\_maze, \_whc, \_size);

}

private void DrawMaze()

{

var rand = new Random();

while(\_path.Count != 0)

{

GetNeighbours(\_path.Peek());

if (\_neighbours.Count != 0)

{

var a = \_neighbours[rand.Next(0, \_neighbours.Count)];

rand.Next(0, 2);

RemoteWall(\_path.Peek(), a);

\_path.Push(a);

\_maze[\_path.Peek().X, \_path.Peek().Y] = Content.Load<Texture2D>("now");

\_neighbours.Clear();

}

else

{

\_maze[\_path.Peek().X, \_path.Peek().Y] = Content.Load<Texture2D>("visited");

\_path.Pop();

if (\_path.Count <= 0) return;

\_maze[\_path.Peek().X, \_path.Peek().Y] = Content.Load<Texture2D>("now");

}

}

MainPoint();

\_status = 1;

\_dfs.Enqueue(\_start);

d = 1;

mat[\_start.X, \_start.Y] = d;

\_neighbours.Clear();

}

private void BreakWalls()

{

var rand = new Random();

for(int k=0;k<50;k++)

{

int x = rand.Next(1, \_size.X - 2);

int y = rand.Next(1, \_size.Y - 2);

if (\_maze[x, y].GetHashCode() == \_whc)

\_maze[x, y] = Content.Load<Texture2D>("visited");

else

k--;

}

\_status = 2;

}

private void GetNeighbours(Cells localcell)

{

var x = localcell.X;

var y = localcell.Y;

const int distance = 2;

var d = new[]

{

new Cells(x, y - distance),

new Cells(x + distance, y),

new Cells(x, y + distance),

new Cells(x - distance, y)

};

for (var i = 0; i < 4; i++)

{

var s = d[i];

if (s.X <= 0 || s.X >= \_size.X || s.Y <= 0 || s.Y >= \_size.Y) continue;

if (\_maze[s.X, s.Y].GetHashCode() == \_whc || \_maze[s.X, s.Y].GetHashCode() == \_vhc) continue;

\_neighbours.Add(s);

}

}

private void RemoteWall(Cells first, Cells second)

{

var xDiff = second.X - first.X;

var yDiff = second.Y - first.Y;

Cells target;

Cells newCells;

var addX = (xDiff != 0) ? xDiff / Math.Abs(xDiff) : 0;

var addY = (yDiff != 0) ? yDiff / Math.Abs(yDiff) : 0;

target.X = first.X + addX;

target.Y = first.Y + addY;

\_maze[target.X, target.Y] = Content.Load<Texture2D>("visited");

\_maze[\_path.Peek().X, \_path.Peek().Y] = Content.Load<Texture2D>("visited");

\_vhc = \_maze[target.X, target.Y].GetHashCode();

newCells.X = first.X + 2 \* addX;

newCells.Y = first.Y + 2 \* addY;

\_path.Push(newCells);

\_maze[\_path.Peek().X, \_path.Peek().Y] = Content.Load<Texture2D>("visited");

}

//Generation

//Draw Path

private void Path()

{

while (\_dfs.Peek().X != \_finish.X || \_dfs.Peek().Y != \_finish.Y)

{

GetNeighboursPath(\_dfs.Peek());

if (\_neighbours.Count != 0)

{

for (int i = 0; i < \_neighbours.Count; i++)

{

\_dfs.Enqueue(\_neighbours[i]);

\_maze[\_neighbours[i].X, \_neighbours[i].Y] = Content.Load<Texture2D>("start");

mat[\_neighbours[i].X, \_neighbours[i].Y] = mat[\_dfs.Peek().X, \_dfs.Peek().Y] + 1;

}

\_maze[\_dfs.Peek().X, \_dfs.Peek().Y] = Content.Load<Texture2D>("vpath");

\_vpath = \_maze[\_dfs.Peek().X, \_dfs.Peek().Y].GetHashCode();

\_dfs.Dequeue();

\_neighbours.Clear();

}

else

{

\_maze[\_dfs.Peek().X, \_dfs.Peek().Y] = Content.Load<Texture2D>("vpath");

\_dfs.Dequeue();

}

}

\_neighbours.Clear();

\_status = 4;

\_dfs.Clear();

\_dfs.Enqueue(\_finish);

return;

}

private void GetNeighboursPath(Cells localCells)

{

var x = localCells.X;

var y = localCells.Y;

const int distance = 1;

var d = new[]

{

new Cells(x, y - distance),

new Cells(x + distance, y),

new Cells(x, y + distance),

new Cells(x - distance, y)

};

for (var i = 0; i < 4; i++)

{

var s = d[i];

if (s.X <= 0 || s.X >= \_size.X || s.Y <= 0 || s.Y >= \_size.Y) continue;

if (\_maze[s.X, s.Y].GetHashCode() != \_whc && \_maze[s.X, s.Y].GetHashCode() != \_shc && \_maze[s.X, s.Y].GetHashCode() != \_vpath)

{

\_neighbours.Add(s);

}

}

}

private void PathBack()

{

while (\_dfs.Count != 0)

{

if (\_dfs.Peek().X == \_start.X && \_dfs.Peek().Y == \_start.Y)

{

\_maze[\_dfs.Peek().X, \_dfs.Peek().Y] = Content.Load<Texture2D>("now");

\_status = 5;

return;

}

GetNeighboursPathBack(\_dfs.Peek());

\_maze[\_dfs.Peek().X, \_dfs.Peek().Y] = Content.Load<Texture2D>("now");

\_dfs.Dequeue();

}

}

private void GetNeighboursPathBack(Cells localCells)

{

var x = localCells.X;

var y = localCells.Y;

const int distance = 1;

var d = new[]

{

new Cells(x, y - distance),

new Cells(x + distance, y),

new Cells(x, y + distance),

new Cells(x - distance, y)

};

for (var i = 0; i < 4; i++)

{

var s = d[i];

if (s.X <= 0 || s.X >= \_size.X || s.Y <= 0 || s.Y >= \_size.Y) continue;

if (mat[s.X, s.Y] + 1 == mat[x, y])

{

\_dfs.Enqueue(s);

return;

}

}

}

//Draw Path

//Ants Algorithm

private void AntsColony()

{

for (var l = 1; l < \_size.X - 1; l++)

for (var j = 1; j < \_size.Y - 1; j++)

if (\_maze[l, j].GetHashCode() != \_whc)

\_maze[l, j] = Content.Load<Texture2D>("visited");

\_maze[1, 1] = Content.Load<Texture2D>("start");

\_maze[\_finish.X, \_finish.Y] = Content.Load<Texture2D>("finish");

double alpha = 1.2;

double beta = 1.5;

double ro = 0.1;

Random random = new Random();

int countAnts = 10;

var massOfPath = new List<Stack<Cells>>();

var allPathWeight = new List<double>();

double shortestPathWeight = ants.n \* ants.m;

Stack<Cells> shortestPath = new Stack<Cells>();

int countIteration = 200;

for (int it = 0; it < countIteration; it++)

{

double shortestPathWeightIteration = ants.n \* ants.m;

for (int i = 0; i < countAnts; i++)

{

var \_point = \_start;

double weightPath = 0;

while (\_point.X != \_finish.X || \_point.Y != \_finish.Y)

{

GetAntbours(\_point);

if (\_neighbours.Count == 0) break;

var Pv = new List<double>();

double sumWeightAndTau = 0;

for (int kekw = 0; kekw < \_neighbours.Count; kekw++)

{

double checkTau = Math.Pow(ants.matrix[\_neighbours[kekw].X, \_neighbours[kekw].Y], beta);

double checkNu = Math.Pow(\_weights[kekw], alpha);

sumWeightAndTau += checkTau \* checkNu;

}

for (int k = 0; k < \_neighbours.Count; k++)

{

var tau = Math.Pow(ants.matrix[\_neighbours[k].X, \_neighbours[k].Y], alpha);

var nu = Math.Pow(\_weights[k], beta);

Pv.Add((tau \* nu) / sumWeightAndTau);

}

var randWay = random.NextDouble();

double checkWay = 0;

for (int k = 0; k < Pv.Count; k++)

{

checkWay += Pv[k];

if (checkWay >= randWay)

{

\_maze[\_point.X, \_point.Y] = Content.Load<Texture2D>("vpath");

\_vpath = \_maze[\_point.X, \_point.Y].GetHashCode();

\_path.Push(\_point);

weightPath += \_weights[k];

\_point = \_neighbours[k];

break;

}

}

Pv.Clear();

\_neighbours.Clear();

\_weights.Clear();

}

if (\_point.X == \_finish.X && \_point.Y == \_finish.Y)

{

\_path.Push(\_point);

massOfPath.Add(new Stack<Cells>(\_path));

allPathWeight.Add(weightPath);

}

\_path.Clear();

for (var l = 1; l < \_size.X - 1; l++)

for (var j = 1; j < \_size.Y - 1; j++)

if (\_maze[l, j].GetHashCode() != \_whc)

\_maze[l, j] = Content.Load<Texture2D>("visited");

\_maze[1, 1] = Content.Load<Texture2D>("start");

\_maze[\_finish.X, \_finish.Y] = Content.Load<Texture2D>("finish");

}

index = -1;

for (int i = 0; i < allPathWeight.Count; i++)

{

if (allPathWeight[i] <= shortestPathWeightIteration)

{

shortestPathWeightIteration = allPathWeight[i];

index = i;

}

}

if (shortestPathWeightIteration < shortestPathWeight)

shortestPath = new Stack<Cells>(massOfPath[index]);

if (index >= 0)

{

for (int i = 0; i < ants.n; i++)

for (int j = 0; j < ants.m; j++)

ants.matrix[i, j] = (1 - ro) \* ants.matrix[i, j];

for (int i = 0; i < massOfPath.Count; i++)

foreach (var passCell in massOfPath[i])

{

ants.matrix[passCell.X, passCell.Y] += shortestPathWeight / allPathWeight[i];

}

//foreach (var pathes in massOfPath)

}

}

foreach (var path in shortestPath)

{

\_maze[path.X, path.Y] = Content.Load<Texture2D>("now");

\_status = 5;

}

Console.WriteLine("kekw");

}

private void GetAntbours(Cells localcell)

{

var x = localcell.X;

var y = localcell.Y;

const int distance = 1;

var d = new[]

{

new Cells(x, y - distance),

new Cells(x + distance, y),

new Cells(x, y + distance),

new Cells(x - distance, y),

new Cells(x - distance, y - distance),

new Cells(x + distance, y + distance),

new Cells(x - distance, y + distance),

new Cells(x + distance, y - distance),

};

for (var i = 0; i < 4; i++)

{

var s = d[i];

if (s.X <= 0 || s.X >= \_size.X || s.Y <= 0 || s.Y >= \_size.Y) continue;

if (\_maze[s.X, s.Y].GetHashCode() != \_whc && \_maze[s.X, s.Y].GetHashCode() != \_shc && \_maze[s.X, s.Y].GetHashCode() != \_vpath)

{

\_neighbours.Add(s);

\_weights.Add(1.0);

}

}

for (var i = 4; i < 8; i++)

{

var s = d[i];

if (s.X <= 0 || s.X >= \_size.X || s.Y <= 0 || s.Y >= \_size.Y) continue;

if (\_maze[localcell.X, s.Y].GetHashCode() == \_whc && \_maze[s.X, localcell.Y].GetHashCode() == \_whc) continue;

if (\_maze[s.X, s.Y].GetHashCode() != \_whc && \_maze[s.X, s.Y].GetHashCode() != \_shc && \_maze[s.X, s.Y].GetHashCode() != \_vpath)

{

\_neighbours.Add(s);

\_weights.Add(1.45);

}

}

}

//Ants Algorithm

//Main Function

protected override void LoadContent()

{

\_spriteBatch = new SpriteBatch(GraphicsDevice);

for (var i = 0; i < \_size.X; i++)

{

for (var j = 0; j < \_size.Y; j++)

{

if (i==\_size.X)

\_maze[i, j] = Content.Load<Texture2D>("wall");

if (j == \_size.Y)

\_maze[i, j] = Content.Load<Texture2D>("wall");

if ((i % 2 != 0 && j % 2 != 0) && (i < \_size.Y-1 || j < \_size.X-1))

{

\_maze[i, j] = Content.Load<Texture2D>("flat");

}

else

{

\_maze[i, j] = Content.Load<Texture2D>("wall");

\_whc = \_maze[i, j].GetHashCode();

}

}

}

}

protected override void Update(GameTime gameTime)

{

if (Keyboard.GetState().IsKeyDown(Keys.Escape))

this.Exit();

if (Keyboard.GetState().IsKeyDown(Keys.Space))

check\_key++;

if (Keyboard.GetState().IsKeyDown(Keys.F2))

\_status = 3;

if (Keyboard.GetState().IsKeyUp(Keys.Space) && check\_key > 0)

{

\_status++;

check\_key = 0;

}

if (index >= 0)

Console.WriteLine();

switch (\_status)

{

case 0:

Window.Title = "Draw Maze";

DrawMaze();

break;

case 1:

BreakWalls();

break;

case 3:

Window.Title = "Draw Water";

stopwatch.Start();

AntsColony();

//Path();

break;

case 4:

Window.Title = "Draw Path";

PathBack();

break;

case 5:

Window.Title = "Finished";

break;

}

base.Update(gameTime);

if (\_status == 5)

{

stopwatch.Stop();

double time = stopwatch.ElapsedMilliseconds;

}

}

protected override void Draw(GameTime gameTime)

{

GraphicsDevice.Clear(Color.White);

\_spriteBatch.Begin();

var matrix = new Matrix(\_size, \_maze);

matrix.Draw(\_spriteBatch,cell\_size);

\_spriteBatch.End();

base.Draw(gameTime);

}

//Main Function

}

}

# ПРИЛОЖЕНИЕ Б

**Файл генерации текстуры**

Файл Matrix.cs

using System;

using System.Linq.Expressions;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

namespace CourseLabirint

{

class Matrix

{

private struct Size

{

public int Width { get; set; }

public int Height { get; set; }

}

readonly Size \_size;

readonly Texture2D[,] \_maze;

public Matrix(Game1.Cells size, Texture2D[,] cell2D)

{

\_maze = cell2D;

\_size.Width = size.X;

\_size.Height = size.Y;

}

public void Draw(SpriteBatch spriteBatch)

{

for (var i = 0; i < \_size.Width; i++)

{

for (var j = 0; j < \_size.Height; j++)

{

spriteBatch.Draw(\_maze[i, j], new Rectangle(i \* 10, j \* 10, 10, 10), Color.White);

}

}

}

}

}

# ПРИЛОЖЕНИЕ В

**Файл хранения матрицы феромонов**

Файл Ants.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

namespace CourseLabirint

{

class Ants

{

public double[,] matrix;

public int n;

public int m;

public Ants(Texture2D[,] maze,int wall,Game1.Cells size)

{

n = size.X;

m = size.Y;

matrix = new double[n, m];

for(int i = 0; i < n; i++)

for(int j = 0; j < m; j++)

matrix[i, j] = 1;

matrix[n - 2, m - 2] = 50;

}

}

}

# ПРИЛОЖЕНИЕ Г

**Файл инициализации программы**

Файл Program.cs

using System;

using System.Collections.Generic;

namespace CourseLabirint

{

#if WINDOWS || XBOX

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

static void Main(string[] args)

{

using (Game1 game = new Game1(new List<Game1.Cells>(), new Stack<Game1.Cells>()))

{

game.Run();

}

}

}

#endif

}