НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Факультет компьютерных наук
Департамент программной инженерии
Выпускная квалификационная работа

Игра “Dark Lab”

Dark Lab Game

Выполнил: студент группы БПИ141
образовательной программы 09.03.04
«Программная инженерия»
Фазли Моазам Джан Карим
Научный руководитель: к.т.н., доцент
Департамента Программной Инженерии
Ахметсафина Римма Закиевна

Концепт игры Dark Lab

База игры:

- Управляемый персонаж
- Нагнетание атмосферы страха
- Ограниченность ресурсов
- Монстр преследует игрока
- Уязвимость монстра к свету
- Исследование помещения
- Цель – найти выход

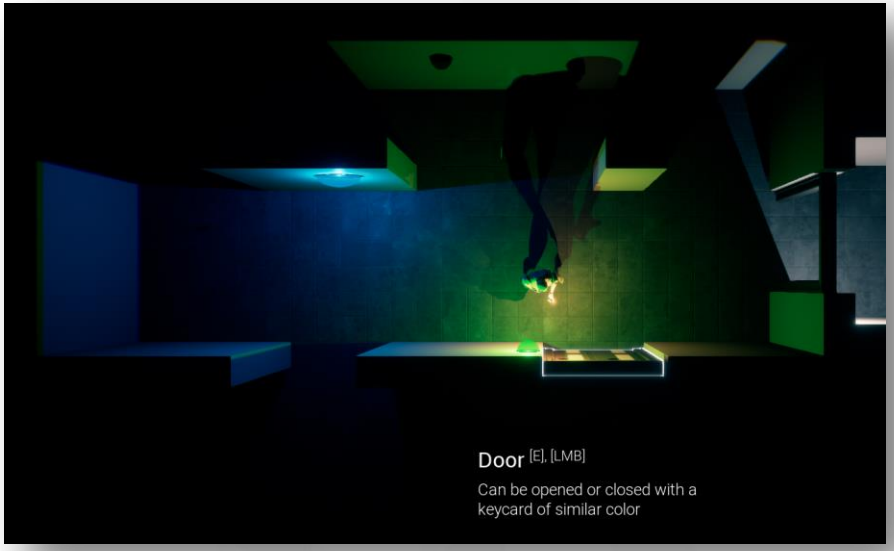

Рисунок 1. Dark Lab, стандартный вид (3D сверху)

Основная особенность:

- Лаборатория постоянно динамически меняется в процессе игры
- Изменения происходят когда область карты остается неосвещенной

Концепт игры Dark Lab

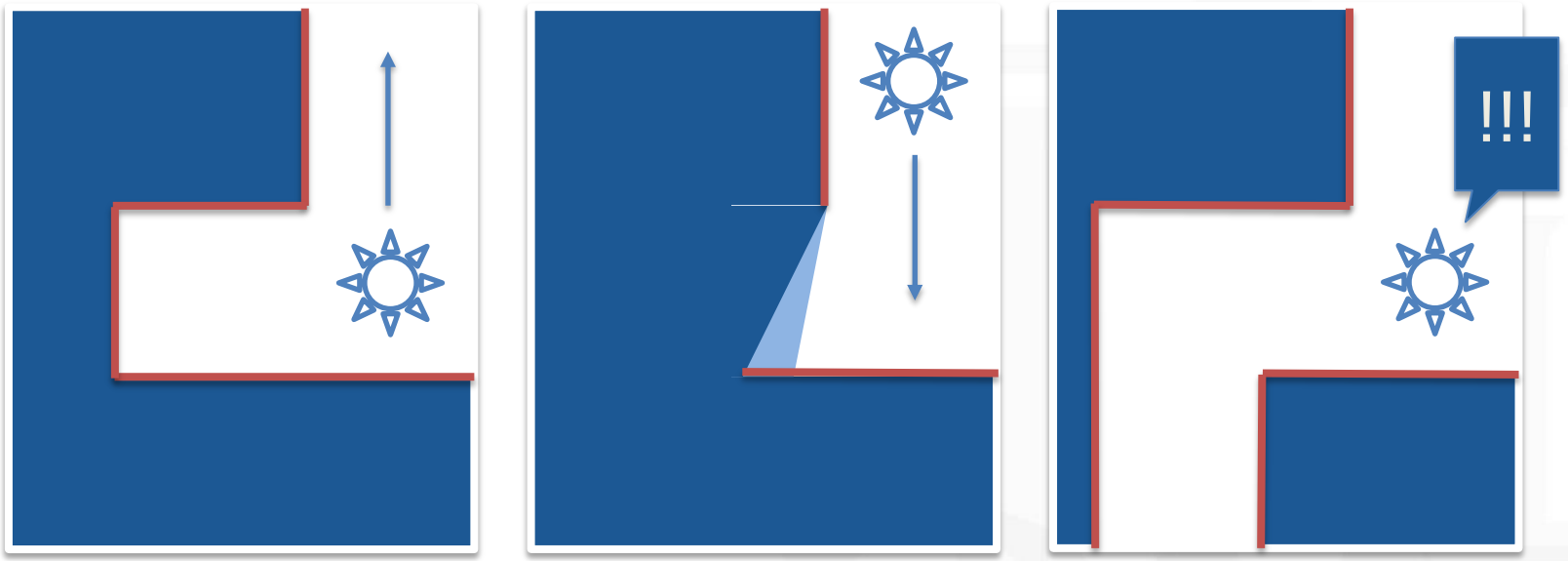

Рисунок 2. Схематичное изображение примера работы основной механики

Основная особенность:

- Лаборатория постоянно динамически меняется в процессе игры
- Изменения происходят когда область карты остается неосвещенной

Основные определения

Survival horror – это жанр видеоигр, в которых протагонист должен спастись из пугающей и/или жестокой среды. Игры жанра стремятся создать атмосферу похожую на фильмы ужасов.

Блупринты (blueprints) – это продвинутая технология визуального скриптинга, встроенная в Unreal Engine и используемая для прототипирования или для полноценной разработки.

Динамическая генерация – процедурная генерация, которая происходит во время игры и на которую могут повлиять действия игрока в отличие от статической генерации, запускаемой перед загрузкой уровня.

Игровой движок (game engine) – это коллекция библиотек и инструментов, собранных вместе для создания игр, также обладающая графом объектов сцены и редактором мира/уровня.

Контроллер (controller) – управляющий класс, контролирующий другой класс (в Unreal Engine).

Подземелье – в процедурной генерации это структура карты состоящая из набора комнат (зачастую прямоугольных) и соединяющих их коридоров.

Процедурная генерация – это процесс создания контента с использованием какого-то алгоритма. При вызове одного и того же алгоритма с одинаковыми параметрами выдается одинаковый результат. Однако процедурной генерацией часто называют случайную.

Актуальность работы

Рост индустрии игр в целом (в прибыли) за 2017 год – **10.7%**.

Доля игр на персональных компьютерах – **28%**.

Количество survival horror игр в магазине Steam – **460**.

Уникальная механика, которая нигде не реализована, в сочетании с привычными игрокам игровыми элементами!

Рисунок 3. Некоторые популярные survival horror игры 2017-го года

Цель и задачи работы

Цель работы:

Создание игры в реальном времени для ПК в жанре survival horror, в которой игрок пытается найти выход из помещения-лаборатории, в то время как за ним охотится монстр, а сама лаборатория постоянно меняется в темноте незаметно для игрока.

Цель и задачи работы

Задачи работы:

1. Разработать основной концепт, сценарий и правила игры;
2. Изучить и проанализировать существующие survival horror игры и выделить основные черты жанра;
3. Изучить и проанализировать существующие алгоритмы случайной генерации карт;
4. Разработать функциональные требования к программе;
5. Разработать структуры данных, используемые в программе;
6. Разработать собственный алгоритм случайной генерации для динамического изменения карты в процессе игры;
7. Выбрать средства и инструменты разработки;
8. Изучить и использовать в разработке Unreal Engine 4;
9. Разработать архитектуру программы;
10. Разработать программу для ПК с полным функционалом игры;
11. Провести тестирование программы;
12. Разработать техническую документацию.

Существующие алгоритмы генерации карт

Варианты модели:

- Лабиринты
- Системы пещер
- Подземелья.

Комнаты:

- Прямоугольные
- Параллельные.

Требования к алгоритму генерации:

1. Эффективность работы генератора
2. Достижимость несгенерированной области из любой сгенерированной
3. Неидеальность – между двумя точками может быть несколько путей
4. Открытые комнаты
5. Коридоры
6. Настраиваемость всех частей генератора
7. Генерация вокруг «закрепленных» комнат
8. Генерация только части карты
9. «Бесконечные» коридоры и циклы
10. Сложность для прохождения

Существующие алгоритмы генерации карт

Search-based approach

Не конкретный алгоритм, а применяется к другим.

Определяются свойства результата.

Генерация пока не «хороший» результат.

Требует:

- Алгоритм поиска (часто использует генетические алгоритмы)
- Репрезентация создаваемого контента
- Функция оценки качества созданного объекта

Почему нет: неэффективный в runtime.

Существующие алгоритмы генерации карт

Space partioning

Основан на рекурсивном разделении пространства на клетки (cells), которые в конце соединяются создавая подземелье.

В процессе создается дерево – space partioning tree.

Создает строгие организованные уровни.

Почему нет: создают всю карту, а не часть, а также не работают вокруг существующих комнат.

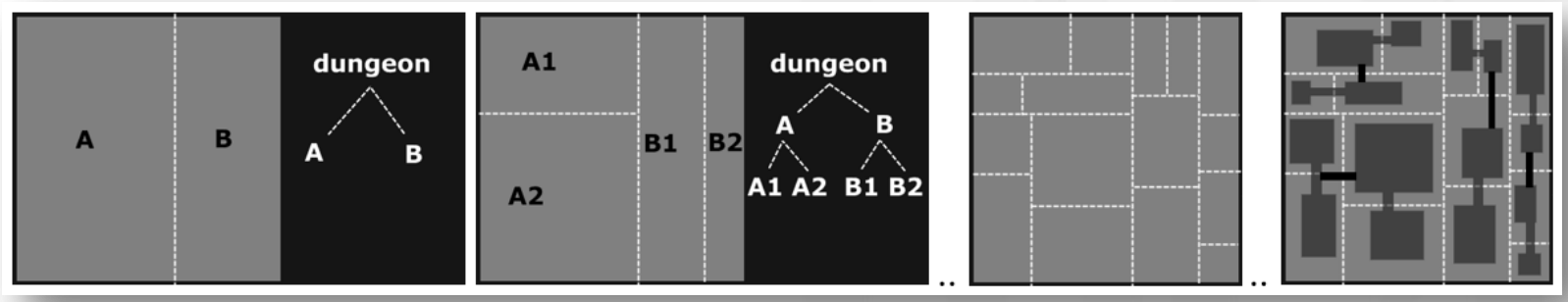

Рисунок 4. Пример работы space-partioning алгоритма

Существующие алгоритмы генерации карт

Cellular automata

Клеточные автоматы.

Основаны на решетке ячеек с состояниями, окрестностями и правилами. По правилам меняют состояния.

Простые эффективные алгоритмы.

Почему нет: обычно работают на всей карте, а еще больше подходят для создания пещер.

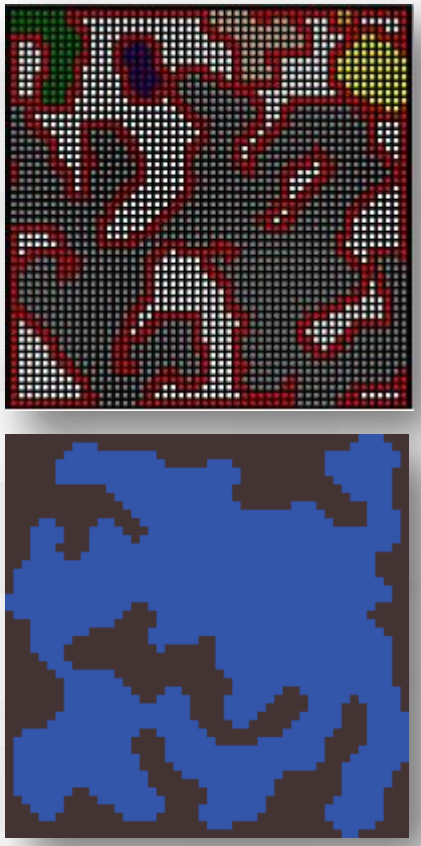

Рисунок 5. Результаты работы алгоритмов на клеточных автоматах

Существующие алгоритмы генерации карт

Agent-based algorithms

Основаны на «копающих» агентах (обычно один). Получаются органичные хаотичные подземелья. Не нарушает поставленные требования. **Создание карты изнутри!**

Подходят если доработать.

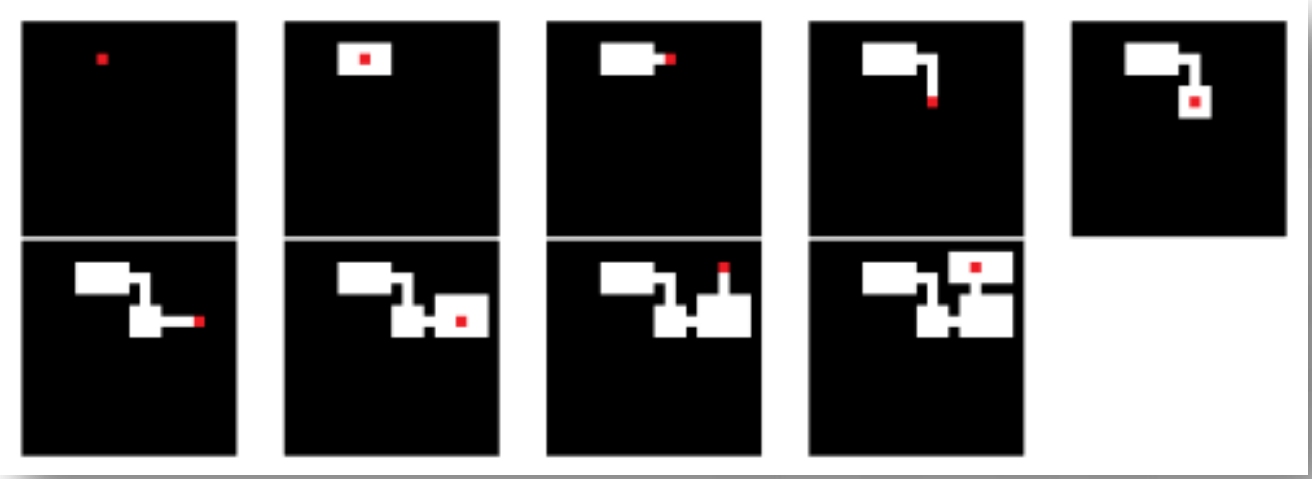

Рисунок 6. Пример работы agent-based алгоритма

Структура карты

Комната:

- Прямоугольное помещение
- Включает множество проходов
- Окружена единичными стенами везде кроме мест расположения проходов

Проход:

- Может быть дверью или проемом
- Соединяет две комнаты (**в** и **из**)
- Существует пока связан хотя бы с одной комнатой

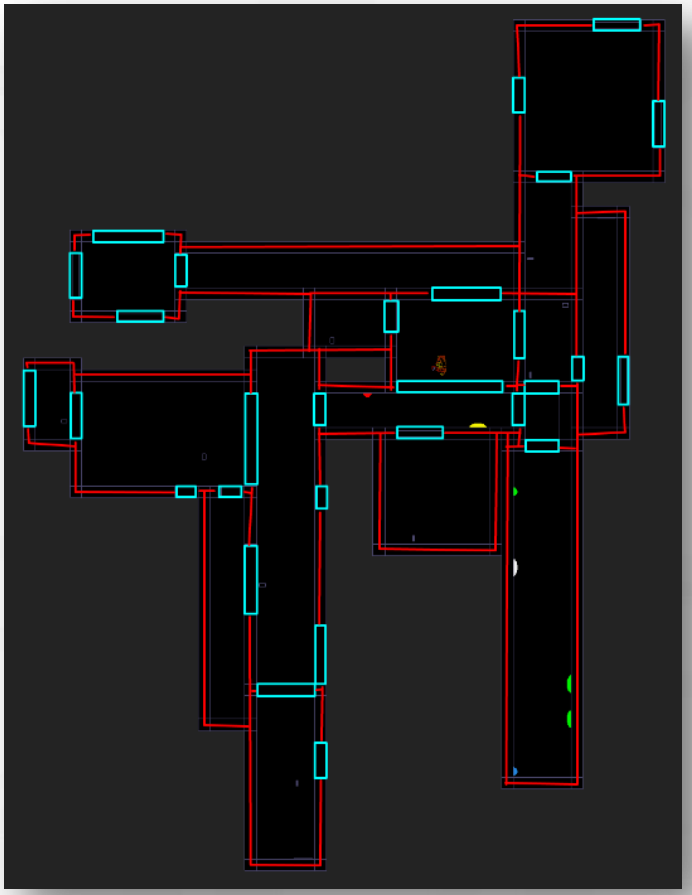

Рисунок 7. Пример связи комнат через проходы

Разработанный алгоритм генерации карты

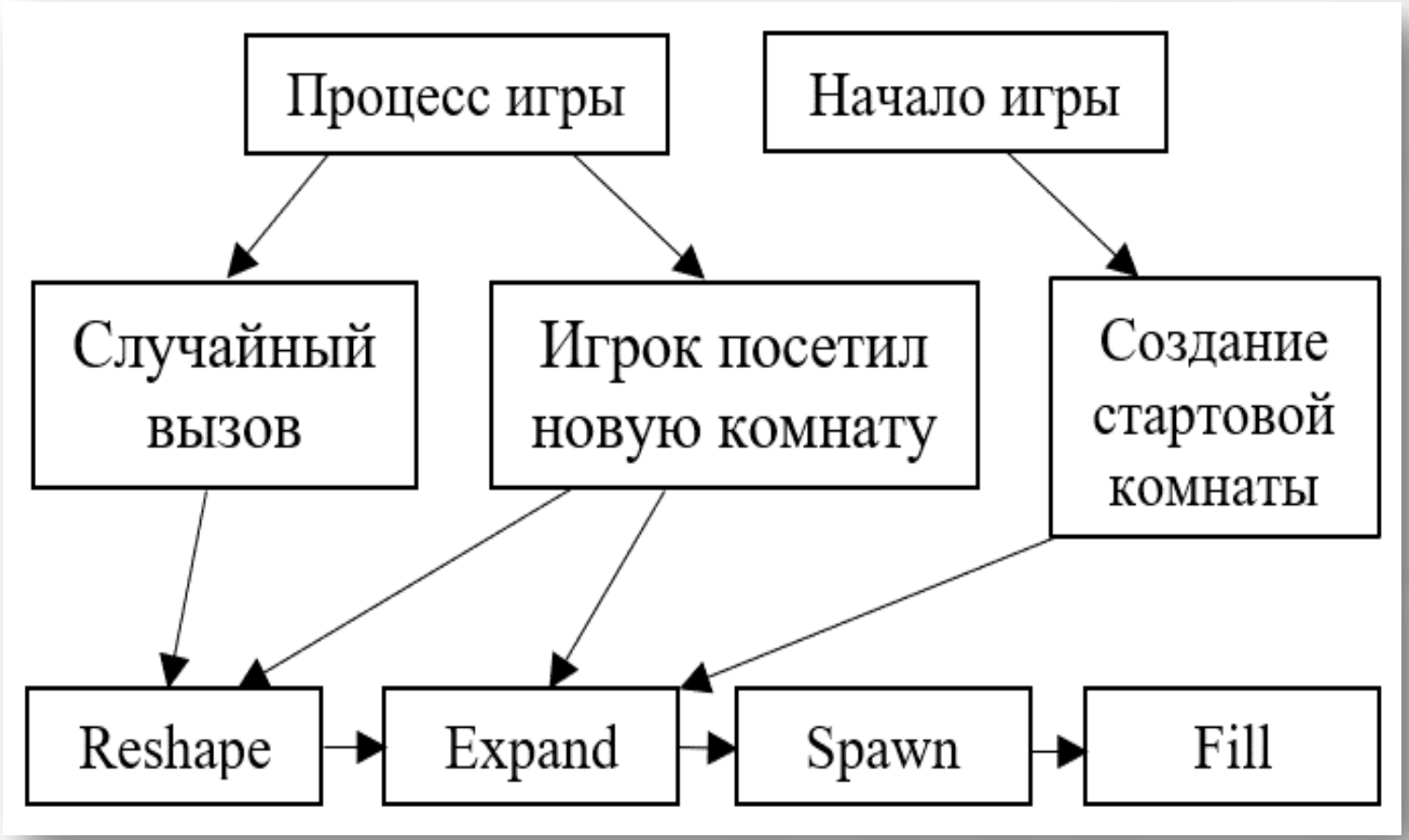

Рисунок 8. Схема вызова генерации карты

Разработанный алгоритм генерации карты

Reshape:

- Области в темноте по всей карте удаляются
- «Сломанные» комнаты «чинятся»

«Сломанные» проходы – проходы, соединенные только с одной комнатой.

«Сломанные» комнаты – комнаты, в которых есть сломанные проходы.

Рисунок 9. Reshape: Начало. Нижняя и правая комнаты находятся в темноте

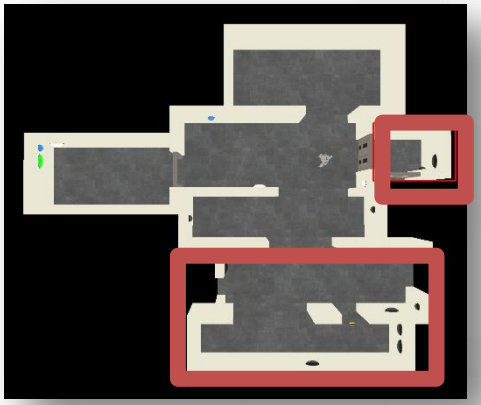

Рисунок 10. Reshape: Комнаты в темноте обведены

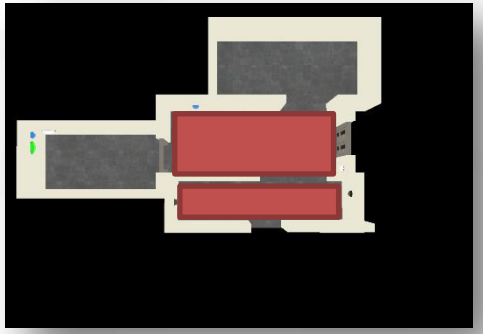

Рисунок 11. Reshape: Комнаты в темноте удаляются, сломанные комнаты выделены

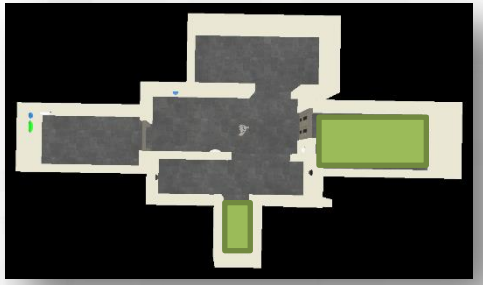

Рисунок 12. Reshape: Комнаты чинятся созданием новых, новые комнаты выделены

Разработанный алгоритм генерации карты

Expand:

- Начиная с комнаты игрока рекурсивно создаются новые проходы и комнаты
- Проверяется, что игрок всегда имеет путь в несгенерированную область

Рисунок 13. Наличие/отсутствие пути

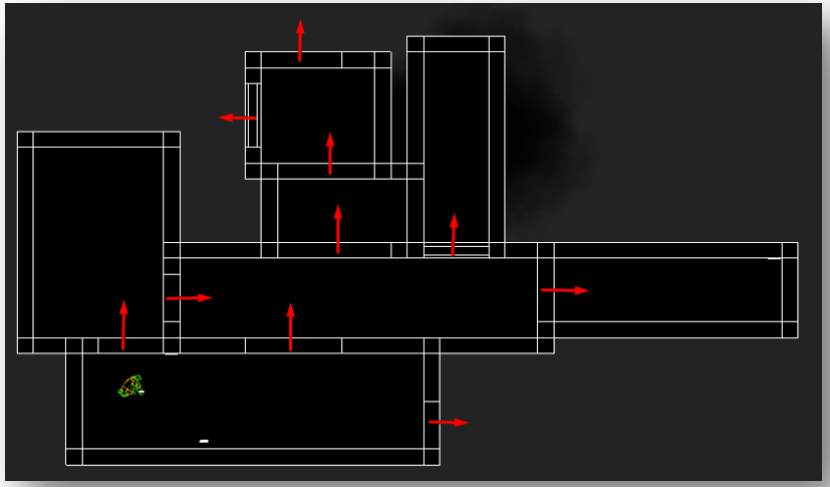

Рисунок 14. Expand: Пример обычной работы

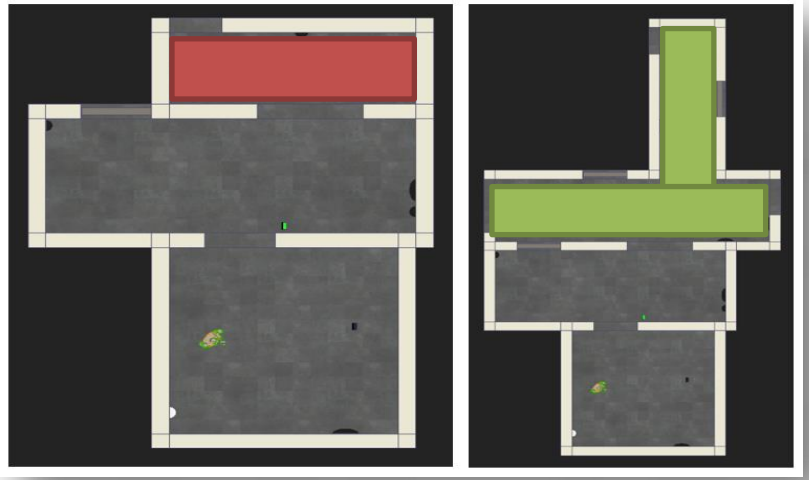

Рисунок 15. Expand: Пример работы с Reshape

Разработанный алгоритм генерации карты

Expand одной комнаты:

- Создается случайный проход
- Для него создается пространство минимальной комнаты
- Проверяется его пересечение с существующими комнатами
- При пересечении проход обычно отбрасывается, но иногда можно соединиться с другой комнатой
- Если такого пересечения нет, то на основе мин. пространства создается случайная комната

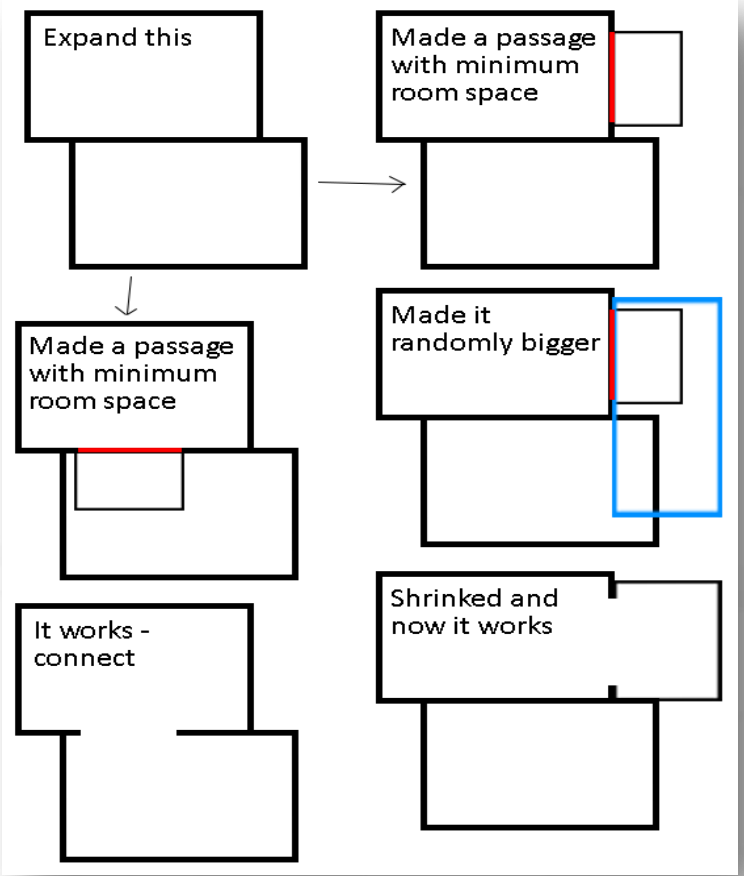

Рисунок 16. Два основных варианта успешного завершения расширения комнаты

Разработанный алгоритм генерации карты

Spawn:

- Начиная с комнаты игрока комнаты преображаются в объекты трехмерного пространства
- Для ускорения используются «умный» выбор комнат для спауна и переиспользование объектов (pooling)

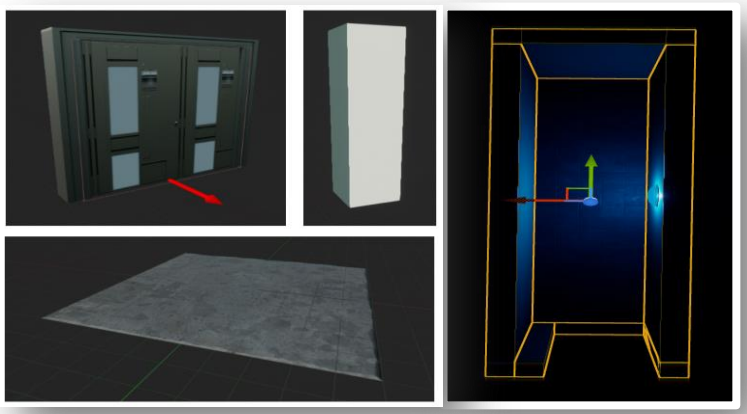

Рисунок 17. Составные части комнаты

Рисунок 18. Проверка видимости во время спауна

Разработанный алгоритм генерации карты

Fill:

- Начиная с комнаты игрока комнаты рекурсивно наполняются объектами и предметами (лампами, ключ-картами и т.п.)

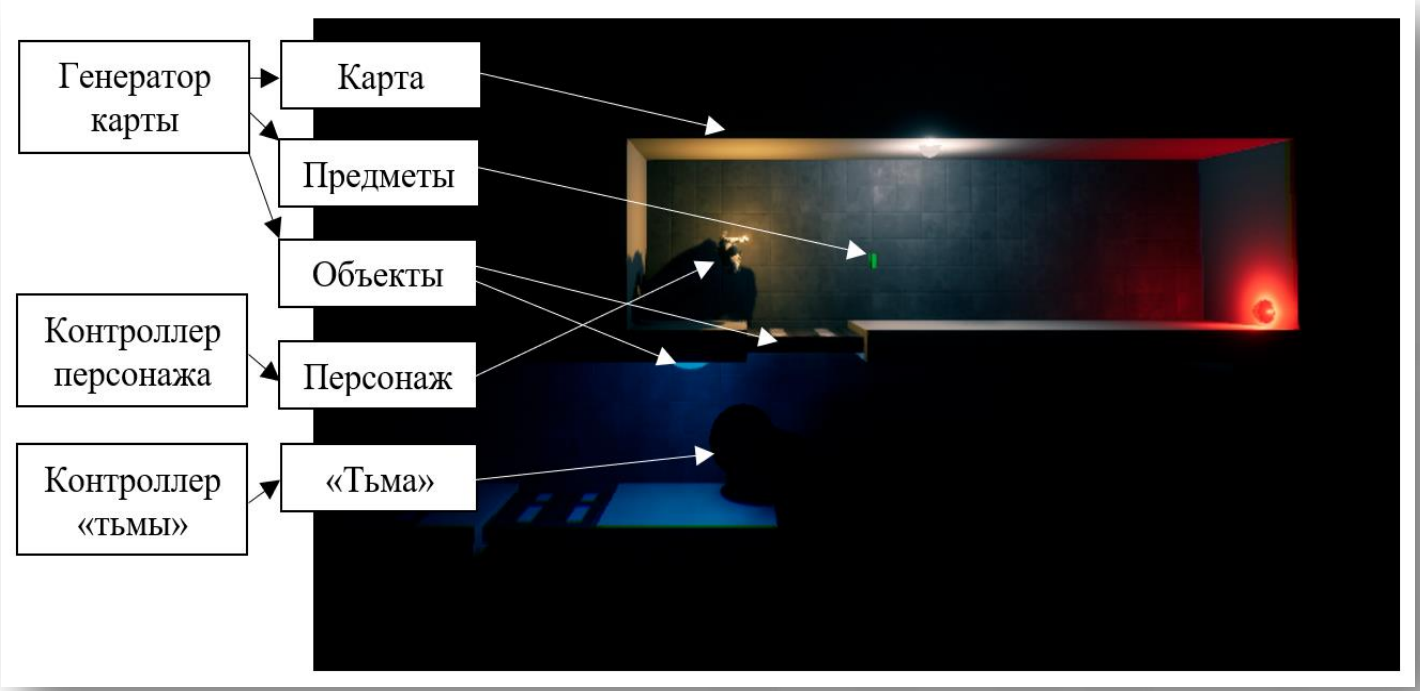

Рисунок 19. Основные компоненты программы

Персонаж и монстр

Персонаж:

- Передвигается по комнатам
- Активирует объекты
- Подбирает предметы
- Использует предметы
- Контролируется игроком

Монстр:

- Передвигается через стены
- Телепортируется
- Боится света
- Имеет сопротивление свету
- Может убить игрока
- Автономен

Рисунок 20. Персонаж и монстр (обведен)

Архитектура программы

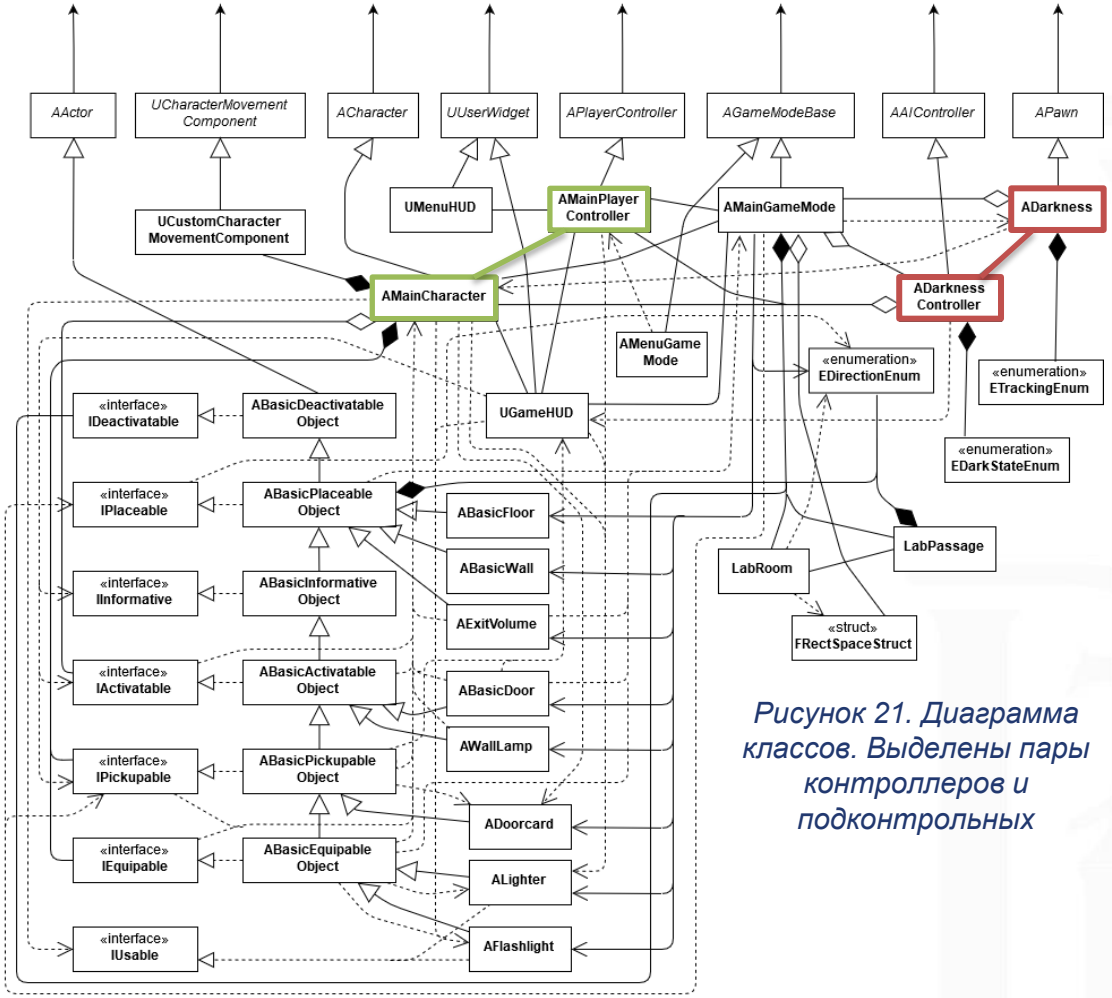

Рисунок 21. Диаграмма классов. Выделены пары контроллеров и подконтрольных

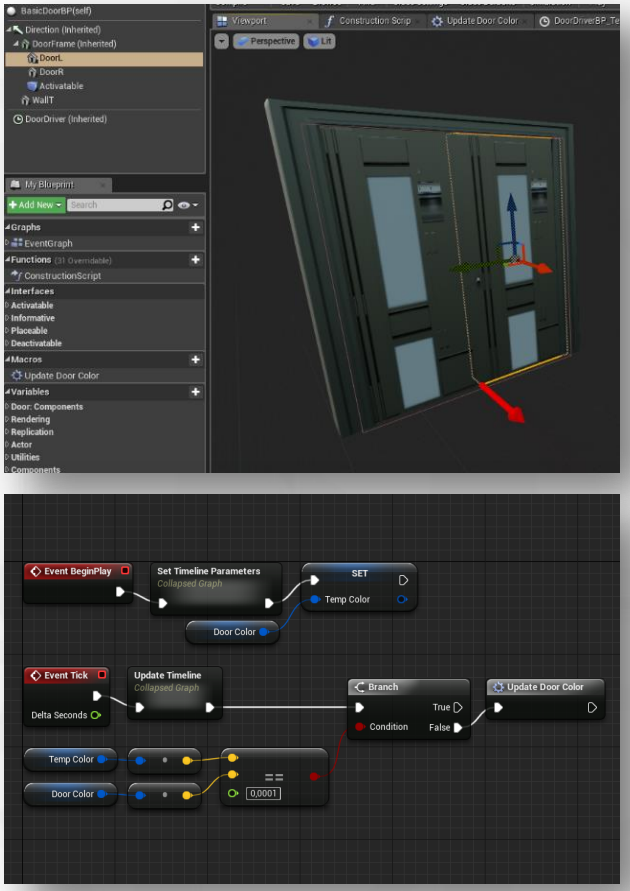

Рисунок 22. Примеры работы с Blueprints

Технологии и инструменты реализации

Игровой движок – Unreal Engine 4

Язык программирования – C++

Среда разработки – Visual Studio 2017

Система контроля версий – Git

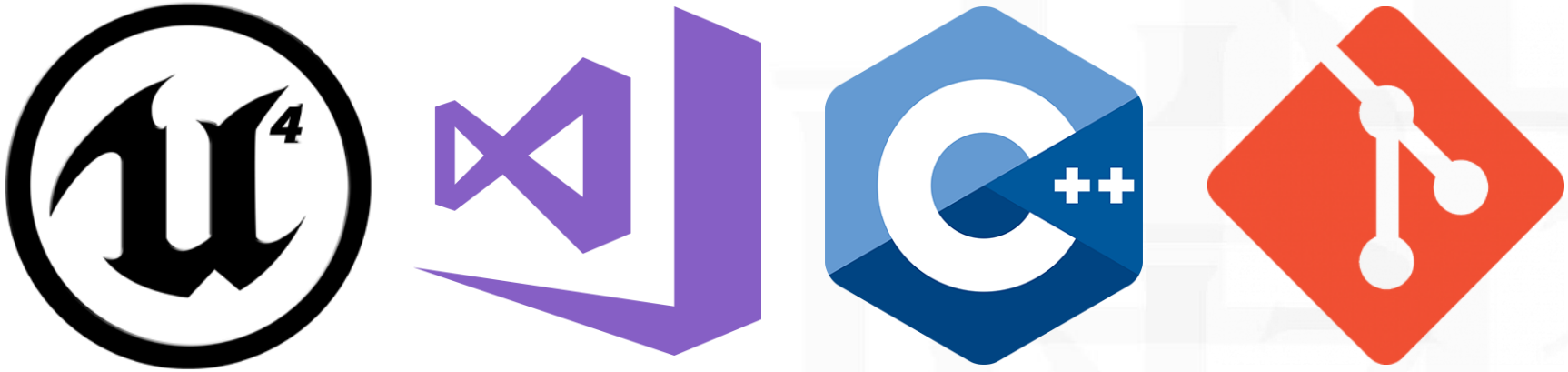

Рисунок 23. Используемые технологии (слева направо: Unreal Engine 4, Visual Studio 2017, C++, Git)

ДЕМОНСТРАЦИЯ

Результаты работы

- Проанализированы другие игры жанра и опираясь на них и на основную идею разработаны функциональные требования проекта
- Выделены требования для алгоритма генерации карты и на их основе проанализированы существующие алгоритмы
- Разработаны структура карты и собственный алгоритм генерации с рядом улучшений производительности
- Разработаны другие структуры данных и алгоритм поведения монстра
- Выбраны и изучены средства разработки программы
- Разработана архитектура программы с учетом возможности будущего расширения контента и функционала
- Программа реализована и удовлетворяет поставленным требованиям

Практическая значимость:

Пути дальнейшей работы

Расширение контента:

- Новые объекты в лаборатории
- Собственная модель для персонажа
- Анимации использования предметов и т.п.

Улучшение алгоритма генерации карты:

- Новые структуры карты, закрытые зоны
- Несколько уровней лаборатории
- Уменьшение затрат по времени/памяти

Улучшение монстра:

- Продвинутый AI
- Новые возможности
- Уменьшение затрат по времени/памяти

Вывод игры на рынок!!

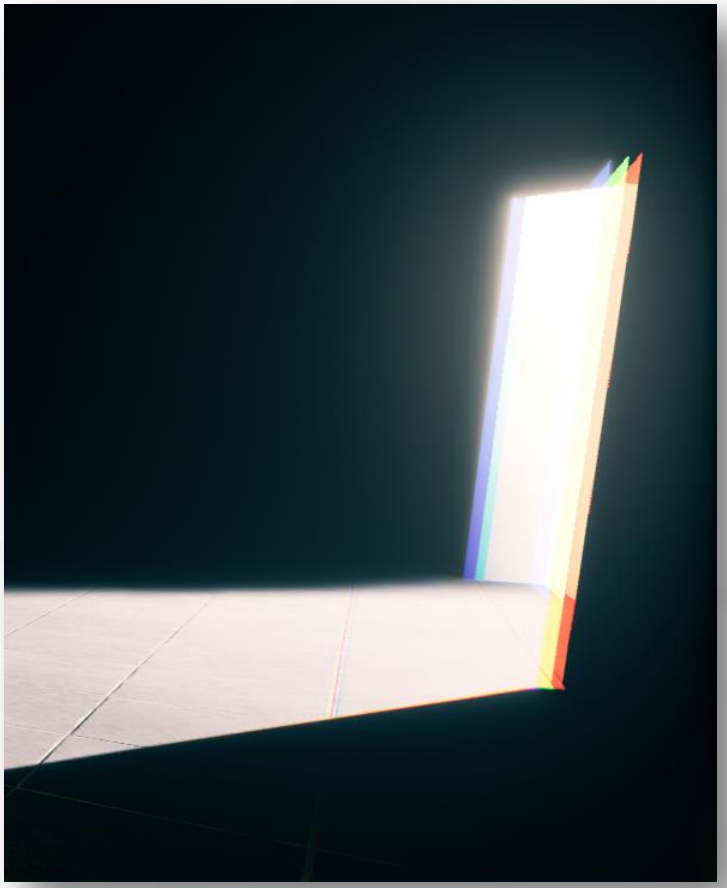

Рисунок 24. Выход из лаборатории

Использованные источники

1. Basic BSP Dungeon generation [Электронный ресурс] / RogueBasin. Режим доступа: roguebasin.roguelikedevelopment.org/index.php?title=Basic_BSP_Dungeon_generation, свободный. (дата обращения: 28.05.18)
2. Blueprints Visual Scripting [Электронный ресурс] / Unreal Engine. Режим доступа: docs.unrealengine.com/en-us/Engine/Blueprints, свободный. (дата обращения: 28.05.18)
3. Dynamic vs Static Procedural Generation [Электронный ресурс] / Medium. Режим доступа: medium.com/@eigenbom/dynamic-vs-static-procedural-generation-ed3e7a7a68a3, свободный. (дата обращения: 28.05.18)
4. Gamedev Glossary: Library VS Framework VS Engine [Электронный ресурс] / GameFromScratch. Режим доступа: www.gamefromscratch.com/post/2015/06/13/GameDev-Glossary-Library-Vs-Framework-Vs-Engine.aspx, свободный. (дата обращения: 28.05.18)
5. Generate Random Cave Levels Using Cellular Automata [Электронный ресурс] / Tutsplus. Режим доступа: gamedevelopment.tutsplus.com/tutorials/generate-random-cave-levels-using-cellular-automata--gamedev-9664, свободный. (дата обращения: 28.05.18)
6. Genetic Algorithm [Электронный ресурс] / PCG Wiki. Режим доступа: pcg.wikidot.com/pcg-algorithm:genetic-algorithm, свободный. (дата обращения: 28.05.18)
7. Git [Электронный ресурс] / Software Freedom Conservancy. Режим доступа: git-scm.com, свободный. (дата обращения: 28.05.18)
8. Green, D. Procedural Content Generation for C++ Game Development / A. Neil, D. Indrajit, M. Priyanka, M. Vishal, and N. Vedangi. Birmingham, UK: Packt Publishing Ltd., 2016 – 279 c.

Использованные источники

9. Object Pool [Электронный ресурс] / Game Programming Patterns. Режим доступа: gameprogrammingpatterns.com/object-pool.html, свободный. (дата обращения: 28.05.18)
10. Procedural Content Generation In Games [Электронный ресурс] / N. Shaker, J. Togelius, M. J. Nelson. Режим доступа: pcgbook.com, свободный. (дата обращения: 28.05.18)
11. Rooms and Mazes: A Procedural Dungeon Generator [Электронный ресурс] / B. Nystrom. Режим доступа: journal.stuffwithstuff.com/2014/12/21/rooms-and-mazes/, свободный. (дата обращения: 28.05.18)
12. Steam [Электронный ресурс] / Valve. Режим доступа: store.steampowered.com/, свободный. (дата обращения: 28.05.18)
13. Survival horror [Электронный ресурс] / Oxford Living Dictionaries. Режим доступа: en.oxforddictionaries.com/definition/survival_horror, свободный. (дата обращения: 28.05.18)
14. The Year In Numbers 2017 [Электронный ресурс] / gameindustry.biz. Режим доступа: www.gamesindustry.biz/articles/2017-12-20-gamesindustry-biz-presents-the-year-in-numbers-2017, свободный. (дата обращения: 28.05.18)
15. Unreal Engine 4 [Электронный ресурс] / Epic Games. Режим доступа: www.unrealengine.com/en-US/what-is-unreal-engine-4, свободный. (дата обращения: 28.05.18)
16. Visual Studio [Электронный ресурс] / Microsoft. Режим доступа: www.visualstudio.com, свободный. (дата обращения: 28.05.18)

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Спасибо за внимание!

Фазли Д.М.К.
FJMK2011@gmail.com
Москва - 2018