МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет (национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

Разработка компьютерной игры в жанре «Платформер» с процедурной генерацией игрового мира на платформе Unity

Научный руководитель:
ст. преподаватель кафедры СП
П.Г. Верман

Автор:
студентка группы КЭ-404
Е.М. Блинова

Челябинск, 2024 г.

Актуальность

- Игровая индустрия является наиболее популярным способом времяпрепровождения, 60% опрошенных россиян играют в видеоигры регулярно или эпизодически
- Игры жанра «Платформер» наиболее популярны среди однопользовательских компьютерных игр и имеют низкий порог вхождения для широкой аудитории
- Процедурная генерация решает проблему обеспечения реиграбельности и позволяет автоматически создавать неограниченное количество уровней

Цель и задачи исследования

Разработка компьютерной игры в жанре «Платформер» с процедурной генерацией игрового мира на платформе Unity

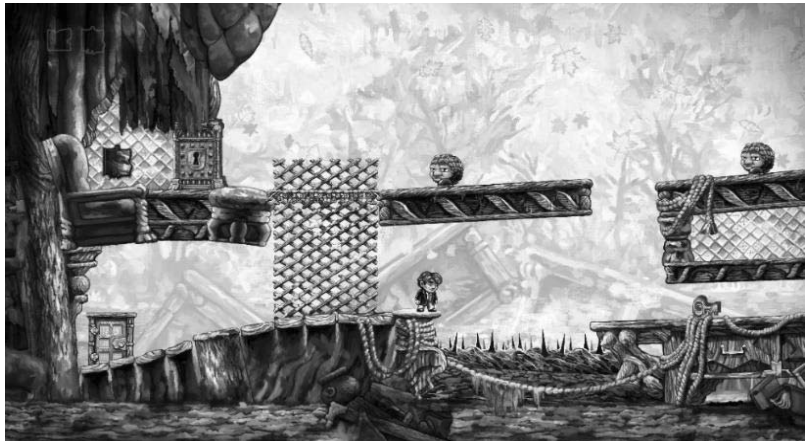
Задачи:

1) произвести анализ предметной области
2) выполнить обзор алгоритмов процедурной генерации
3) спроектировать игровое приложение
4) реализовать игровое приложение
5) провести тестирование игрового приложения

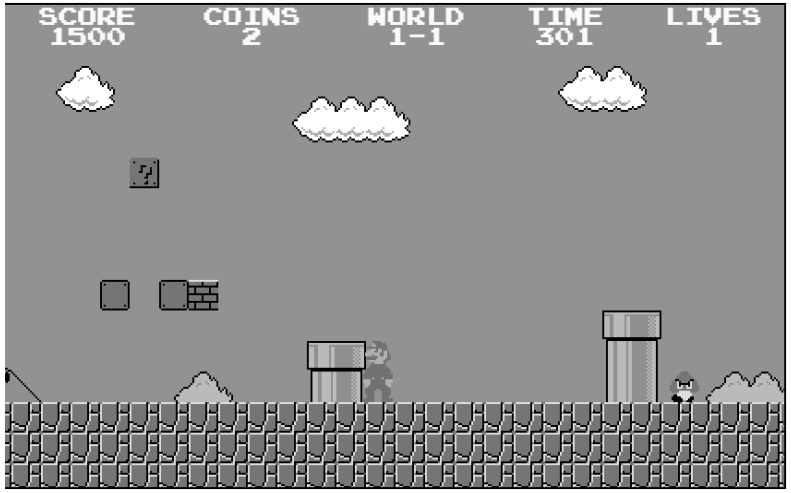
Обзор существующих решений

Fez

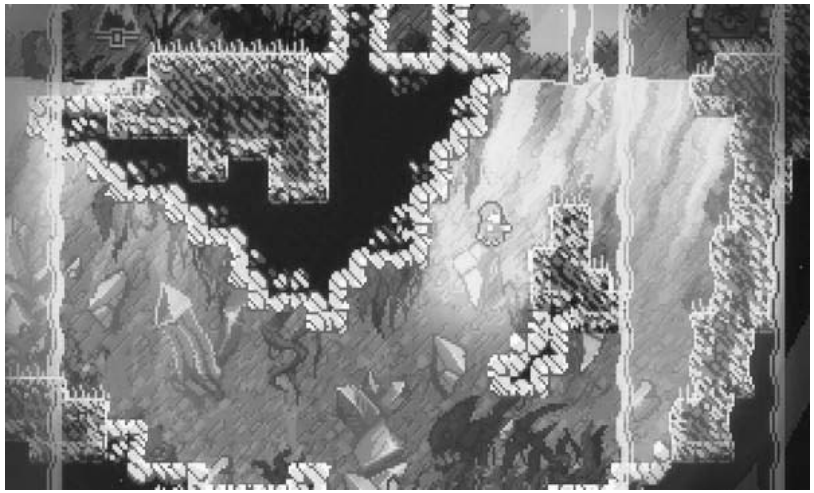
Braid



Super Mario Bros



Celeste



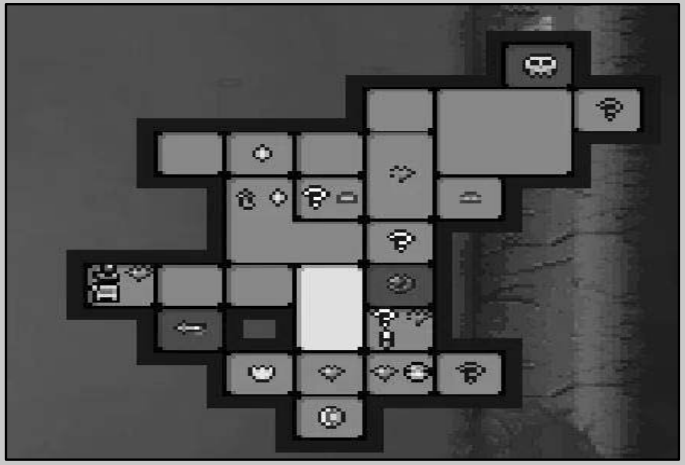
Перемещение:

- Прыжки
- Бег
- Взбирание
- Преодоление препятствий

Механики:

- Платформы
- Собирание предметов
- Головоломки
- Игровые экраны

Обзор игр с процедурной генерацией



The Binding of Isaac

Darkest Dungeon

Rogue Legacy

- Создание случайных подземелий
- Расположение предметов
- Враги, монстры
- Создание конфигурации подземелий и событий
- Создание конфигурации комнат

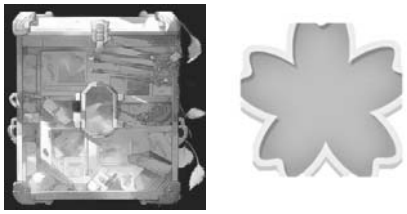
Концепция – цели и задачи

Особенности игры:

- 4 типа локаций, на каждой локации особый тип платформ
- 3 уровня сложности
- Процесс прохождения игры основан на изменении цвета
- Процедурная генерация игрового мира
  - Комнат и переходов
  - Размещении кристаллов и платформ

Цель игры:

- Открыть сундук



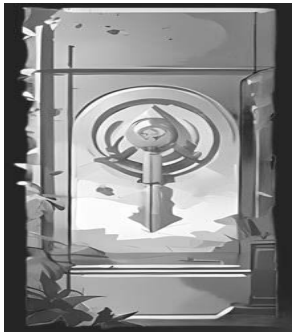
Задачи игры (достижение цели):

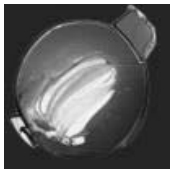
- Посетить как можно больше комнат
- Собрать как можно больше очков
- Сделать черно-белый игровой мир цветным

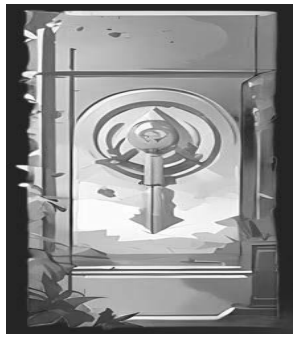
Концепция – объекты и правила

Игровые объекты:

- Двери
- В инвентаре: кристаллы 4-х типов и особый предмет
- Сундук
- Платформы статичные и не-статичные (по типам локаций):
  - «Облачное пространство» – исчезающие платформы
  - «Зеленый лес» – подвижные платформы
  - «Огненные земли» – изменяют направление гравитации
  - «Морские глубины» – изменяют направление движения



-N 



+1 

Игровые правила:

- Дверь открывается кристаллом того типа, в локацию которого ведет или локации, в которой находится
- За открытие двери дается особый предмет
- Для открытия сундука требуется определенное количество особых предметов
- Проигрыш возможен на локации «Облачное пространство»

Процедурная генерация контента (ПГК)

Классификация видов ПГК*

- Методы распределения (врагов, объектов, очков)
- Параметрические методы (наборы характеристик, адаптивность)
- Методы на основе плиток (подземелья, карты)
- Методы на основе формальных грамматик (города, объекты, квесты, сюжеты)
- Использование решателей ограничений (подземелья, карты, головоломки, правила)
- Методы на основе агентов и моделирования (поведение, экосистемы)

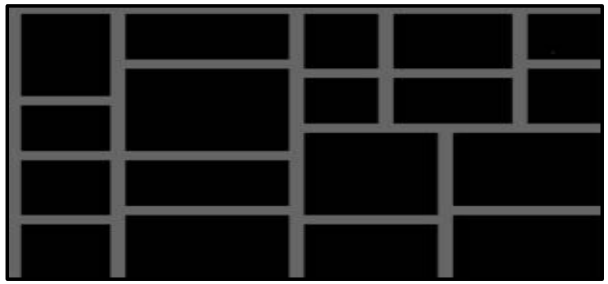
Особенности применения алгоритмов:

- Графическая перспектива «Top-Down» и«Side-Scrolling»
- Назначение генерации: сценарии, ландшафт, персонажи, размещение объектов игрового мира
- Другие механизмы: (не рассматривались в данной работе) нейронные сети, генетические алгоритмы

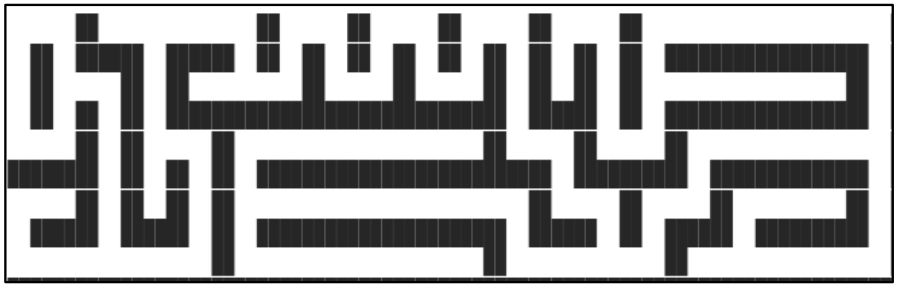
*Шорт Т.Х., Адамс Т. Процедурная генерация в гейм-дизайне. // М.: ДМК Пресс, 2020. – 344 с.

Алгоритмы генерации игрового мира

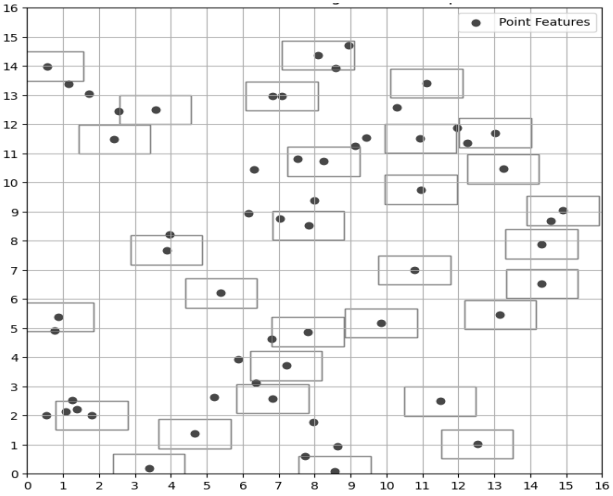
(3а) Алгоритм двоичного разбиения пространства



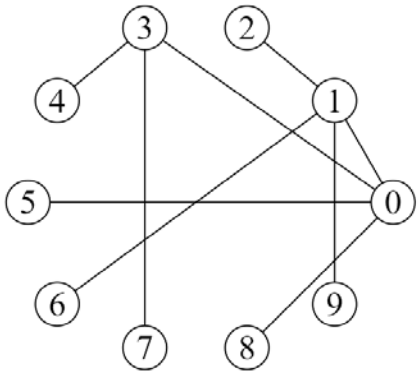
(4) Генерация лабиринта методом поиска в глубину



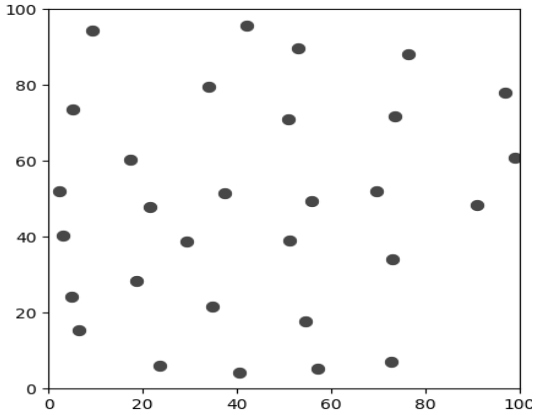
(3б) Алгоритм размещения меток на основе сетки



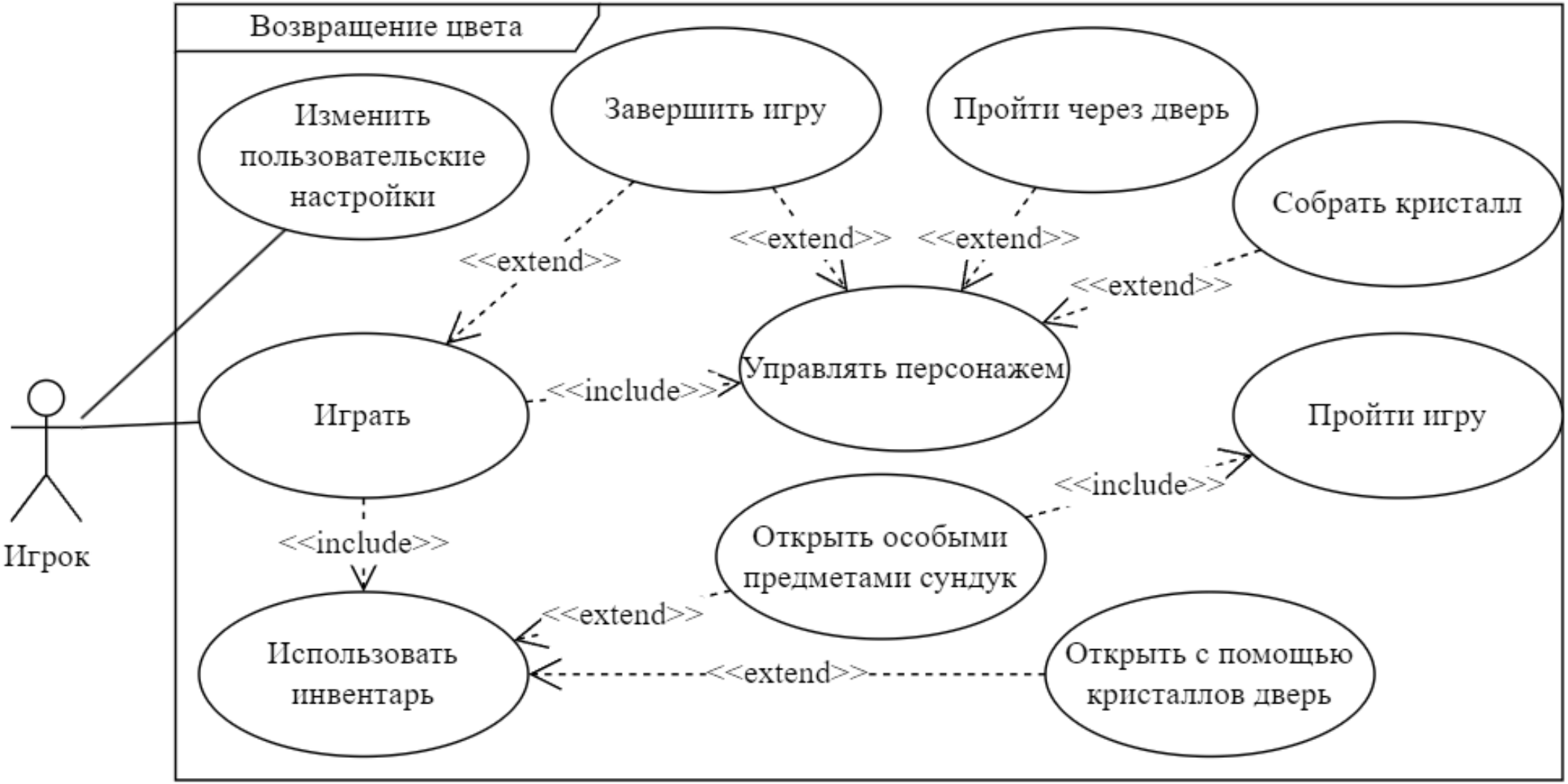
(1) Случайный граф Барабаши-Альберта



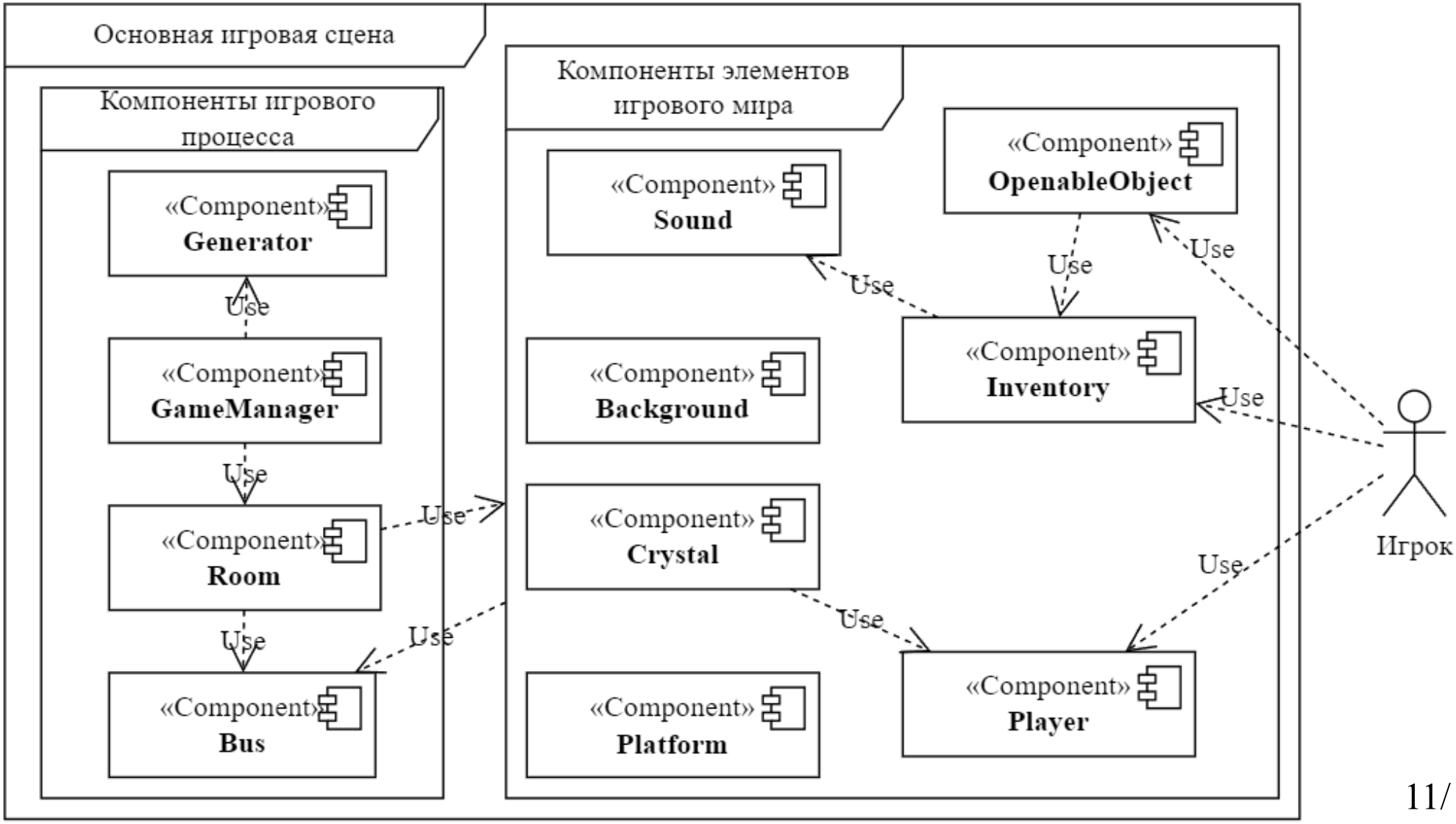
(2) Процесс упаковки дисков Пуассона



Варианты использования

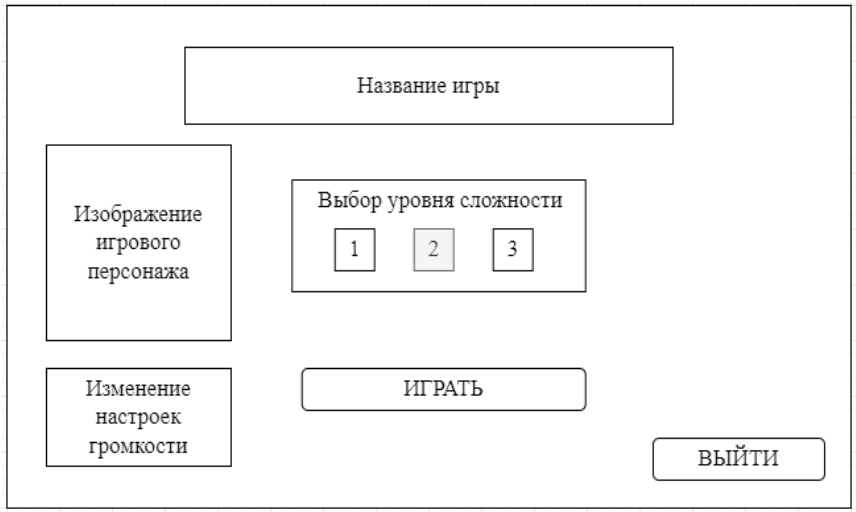


Архитектура игрового приложения

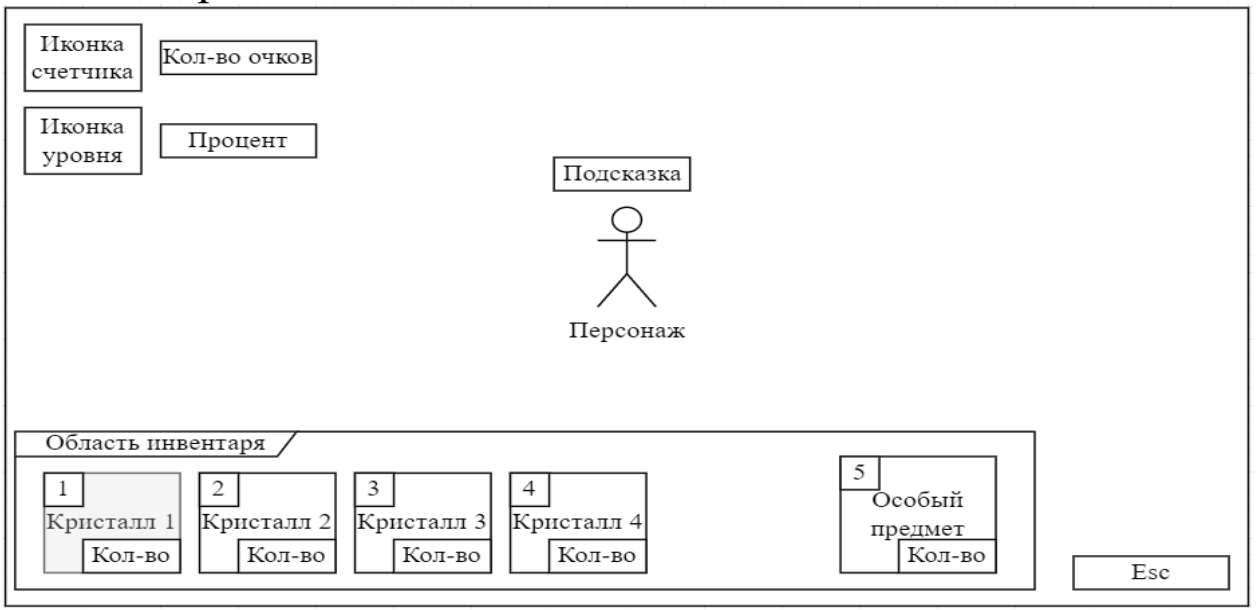


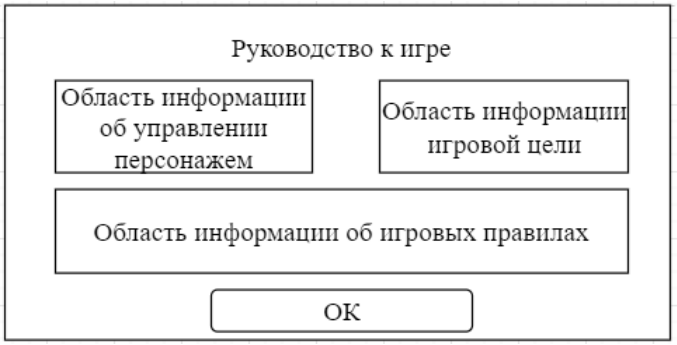
Макеты интерфейса

Макет экрана StartScreen

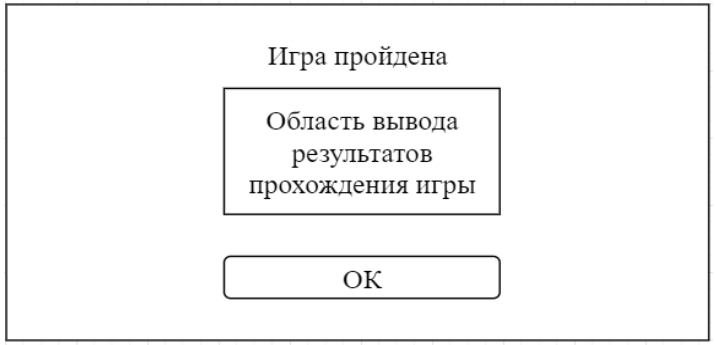


Макет экрана GameScreen

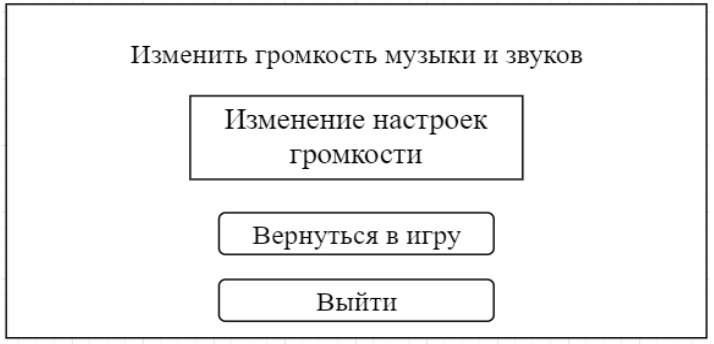




Макет окна игровой справки



Макет окна прохождения игры

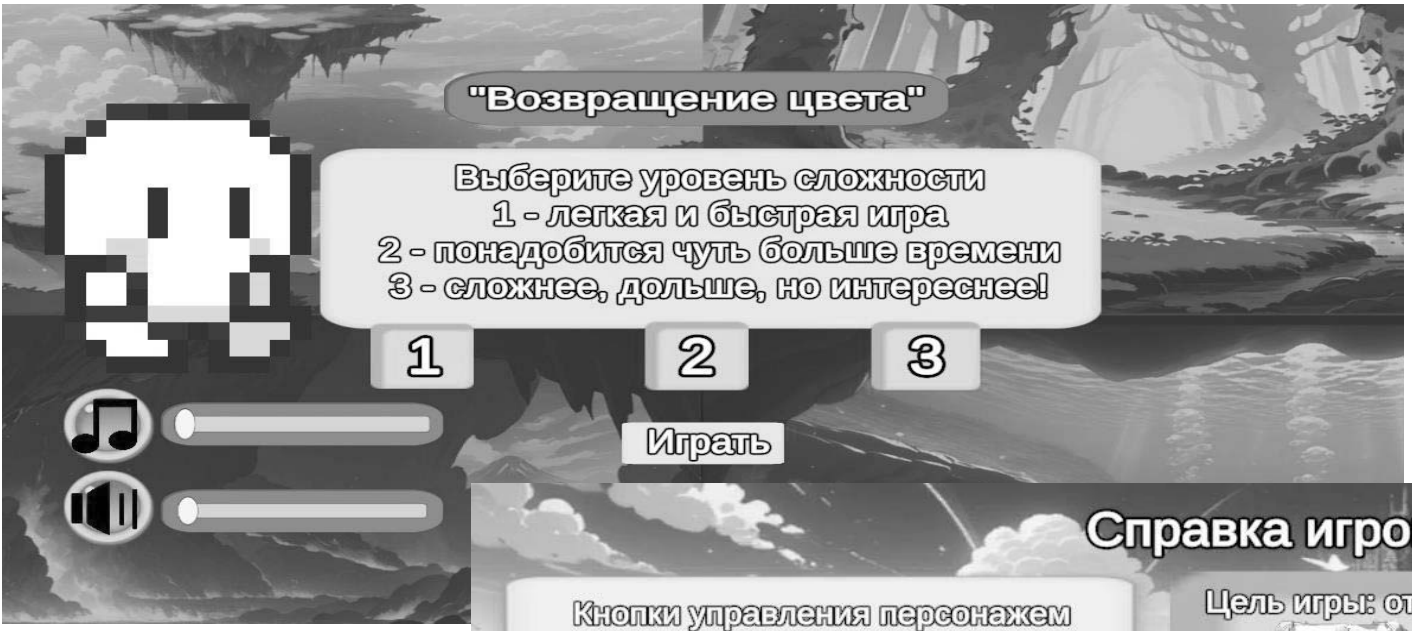


Макет окна меню

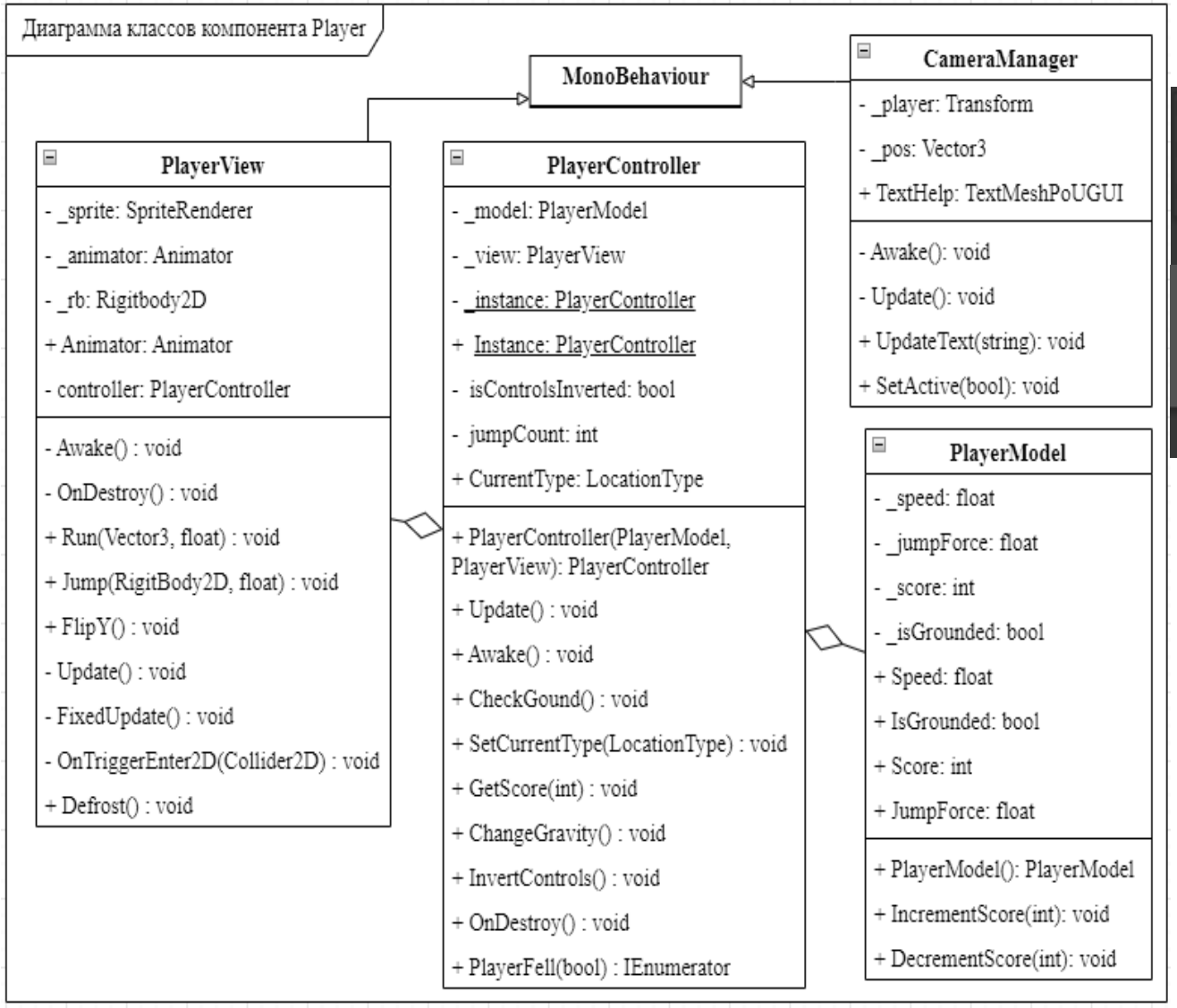
Средства реализации

1. Unity 2022.3.19f1
2. UnityHub
3. Текстовый редактор кода Visual Studio Code
4. Github
5. Библиотека QuikGraph 2.5.0
6. Playground – ресурс для генерации изображений

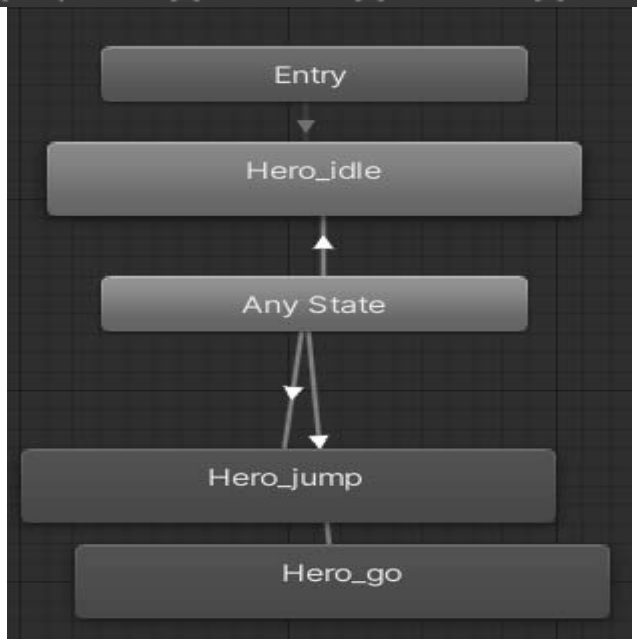
Стартовая сцена



Реализация персонажа

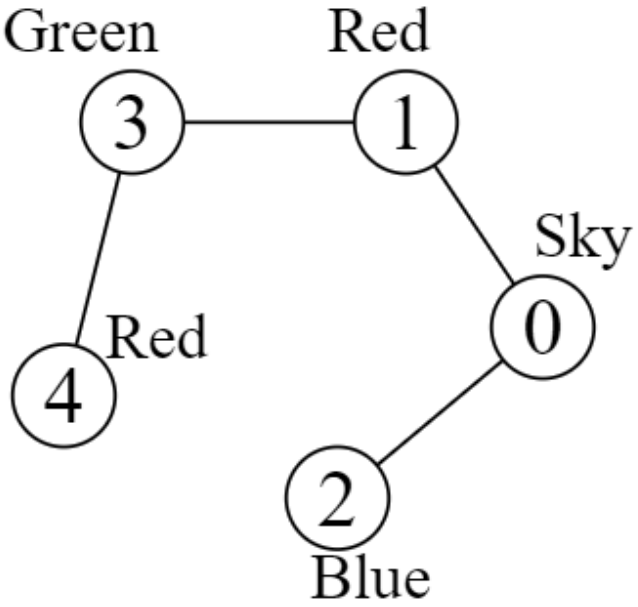


3 состояния анимации

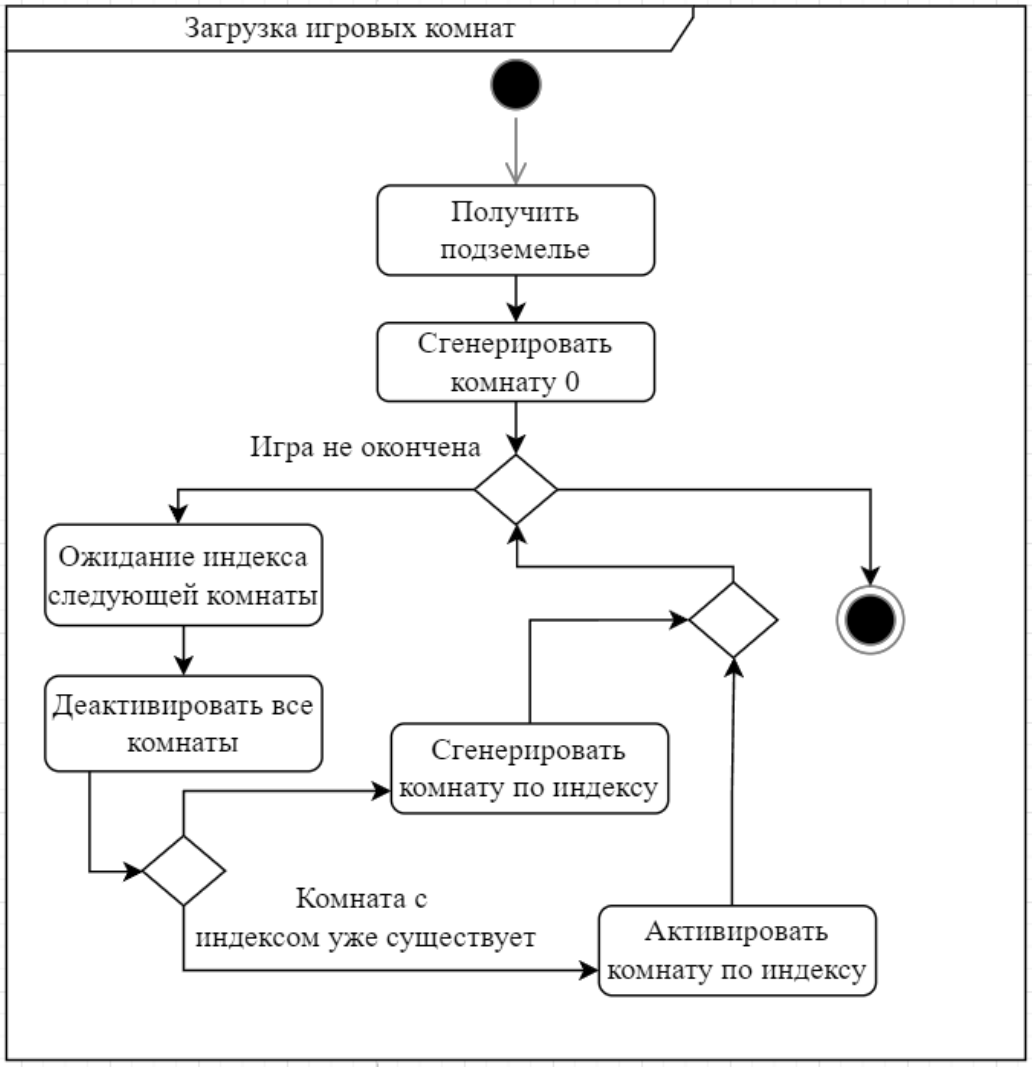


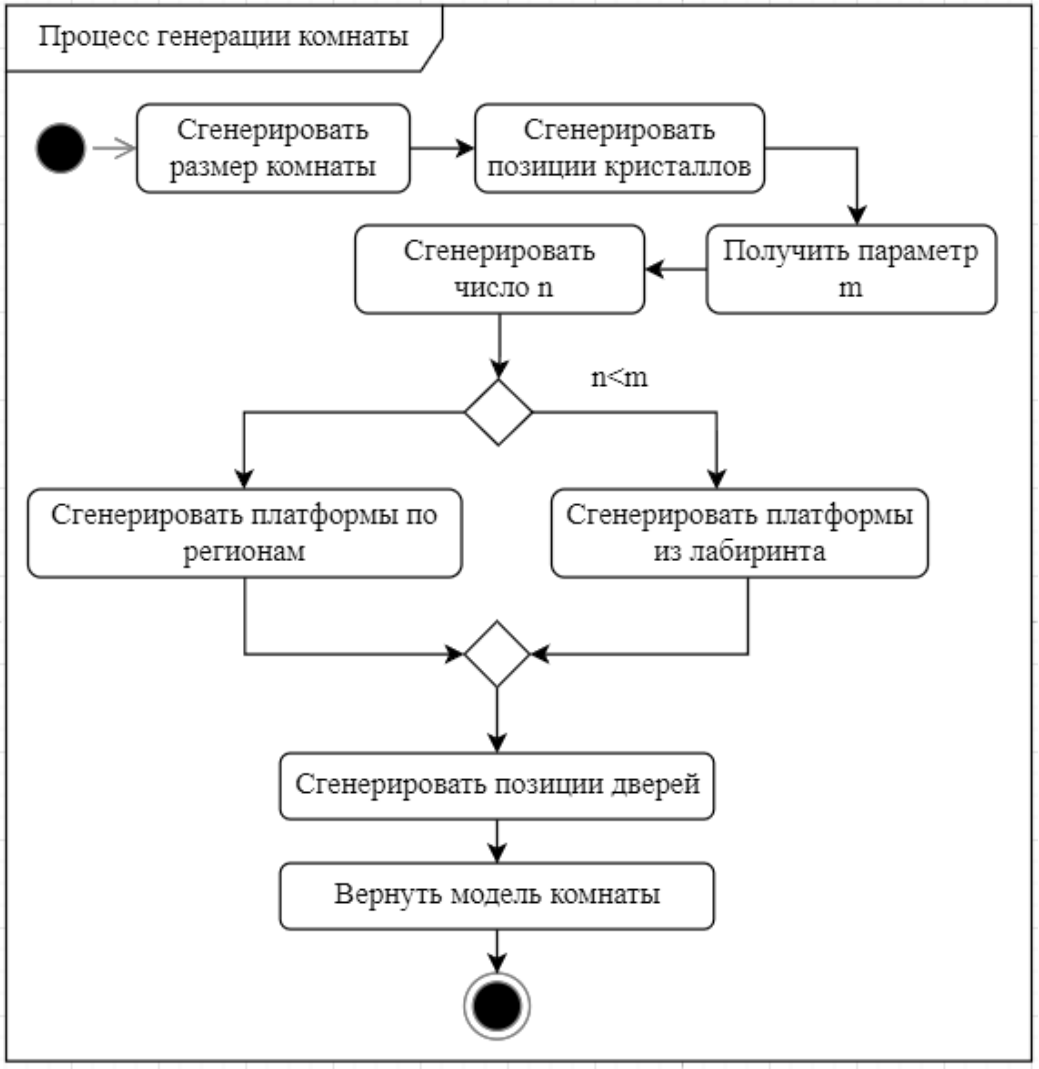
Генерация игрового подземелья

Количество комнат: 5
[GameManager] Выбран уровень сложности: 1
[Generator] [GeneratorGraph] Vertices: 0, 1, 2, 3, 4
[Generator] [GeneratorGraph] Edges: (1, 0), (2, 0), (3, 1), (4, 3)
[Generator] [GeneratorGraph] Levels: Level 0: Sky, Level 1: Red,
Level 2: Blue, Level 3: Green, Level 4: Red,
Переходы:
Level 0: Sky -> Level 1: Red, Level 2: Blue
Level 1: Red -> Level 0: Sky, Level 3: Green
Level 2: Blue -> Level 0: Sky
Level 3: Green -> Level 1: Red, Level 4: Red
Level 4: Red -> Level 3: Green
[GameManager] Door Index: 0, Current Location: 0, Sky,
Next Location: 1, Red
[GameManager] Door Index: 1, Current Location: 0, Sky,
Next Location: 2, Blue
[GameManager] Door Index: 2, Current Location: 1, Red,
Next Location: 0, Sky
[GameManager] Door Index: 3, Current Location: 1, Red,
Next Location: 3, Green
[GameManager] Door Index: 4, Current Location: 2, Blue,
Next Location: 0, Sky
[GameManager] Door Index: 5, Current Location: 3, Green,
Next Location: 1, Red
[GameManager] Door Index: 6, Current Location: 3, Green,
Next Location: 4, Red
[GameManager] Door Index: 7, Current Location: 4, Red,
Next Location: 3, Green



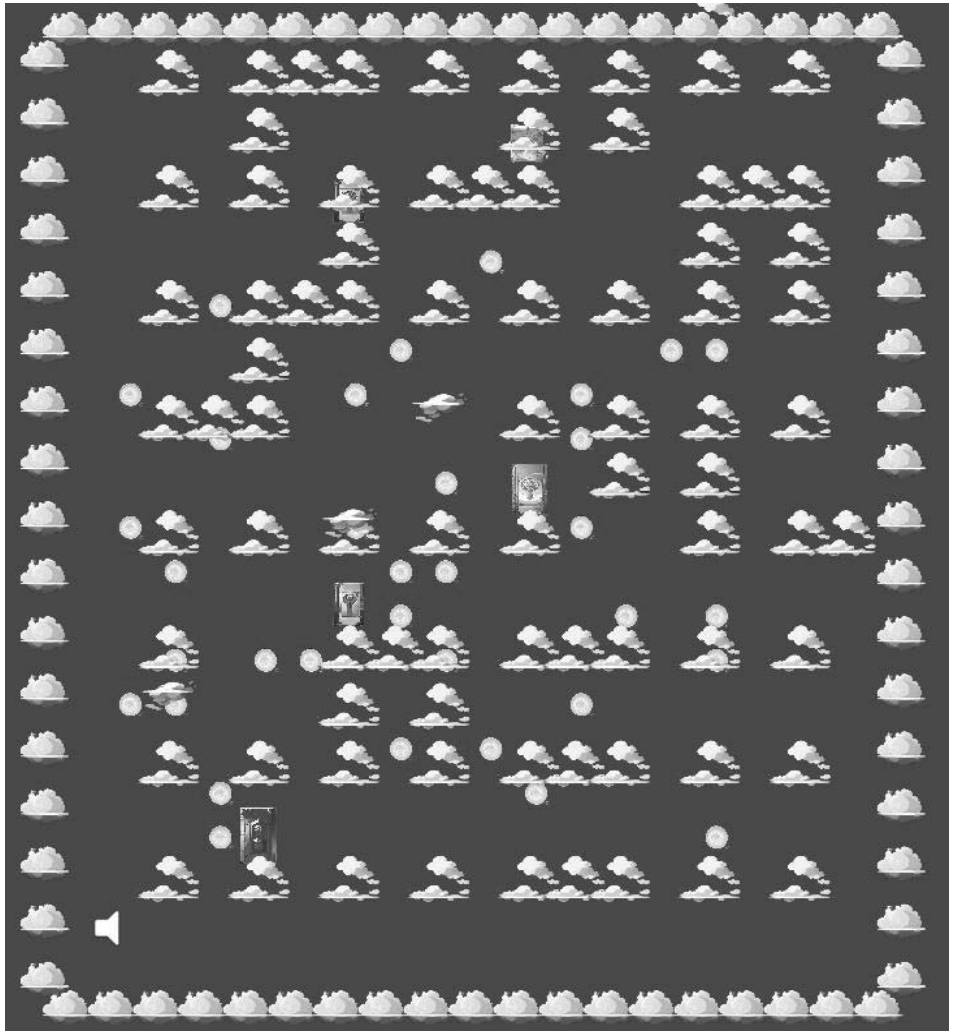
Загрузка и генерация комнат



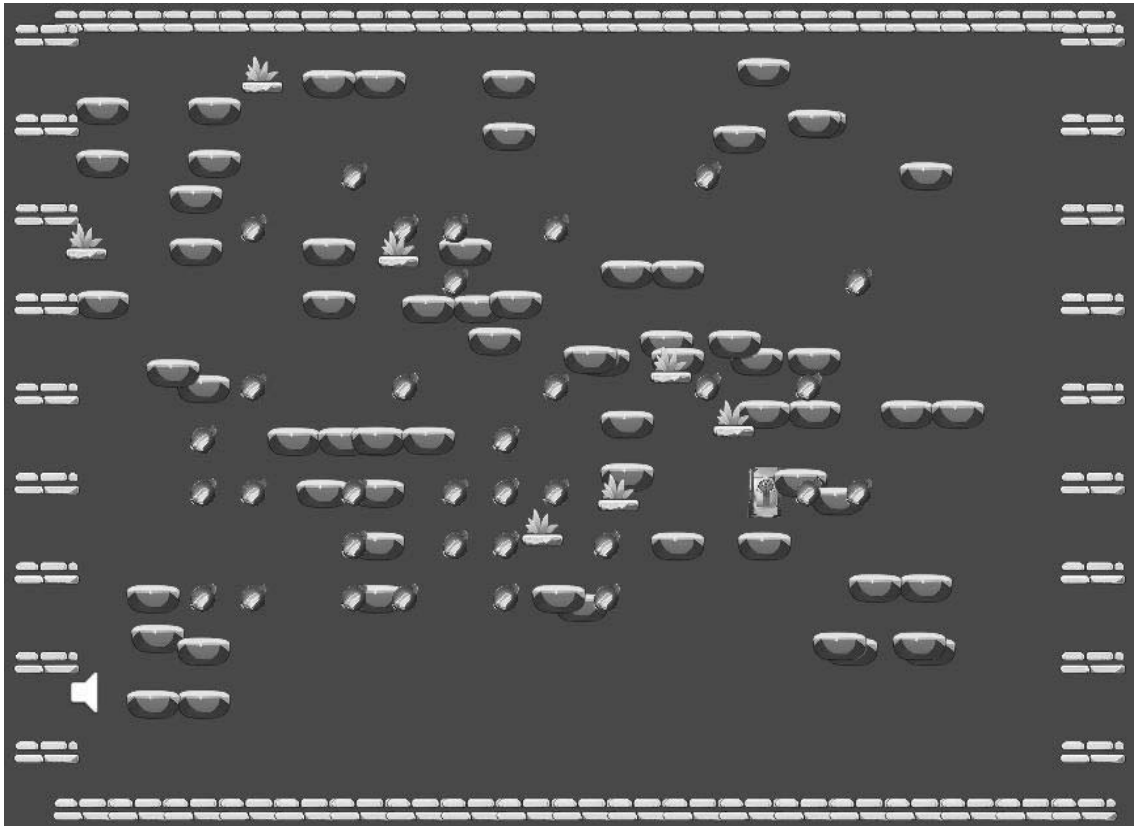


Результат генерации комнат

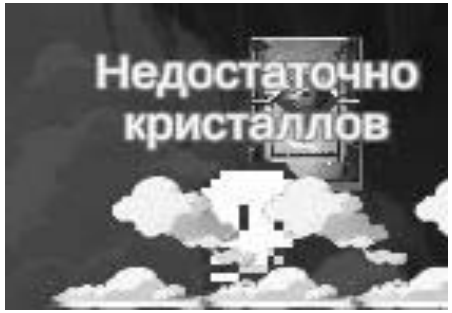
Платформы из лабиринта



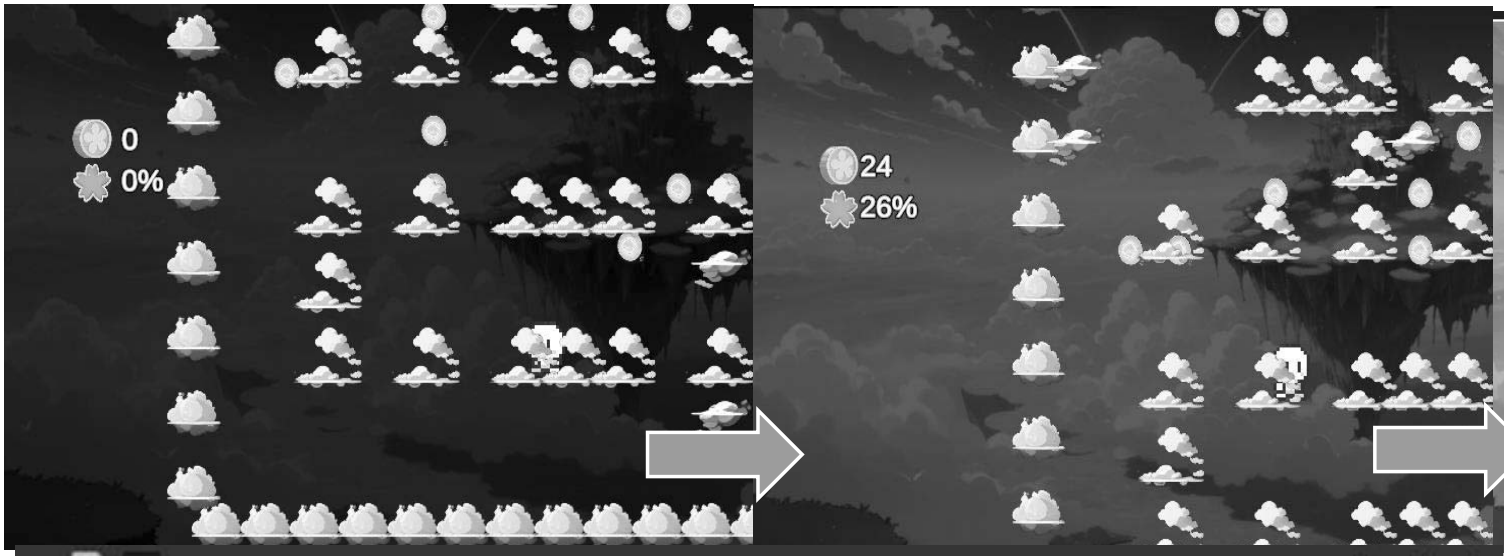
Платформы по регионам

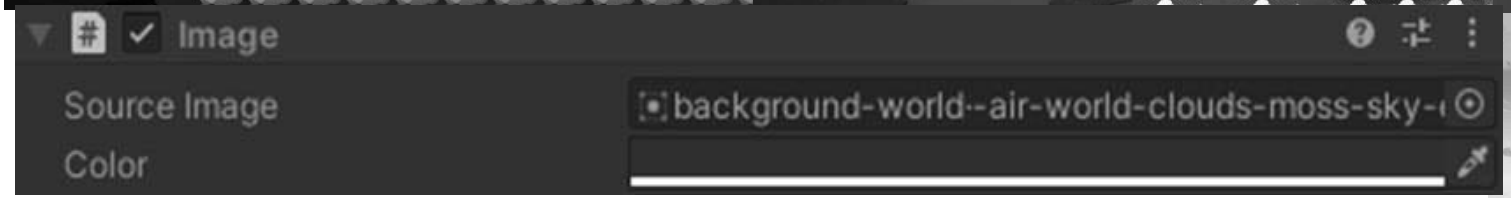


Реализация основного игрового экрана и подсказок для игрока



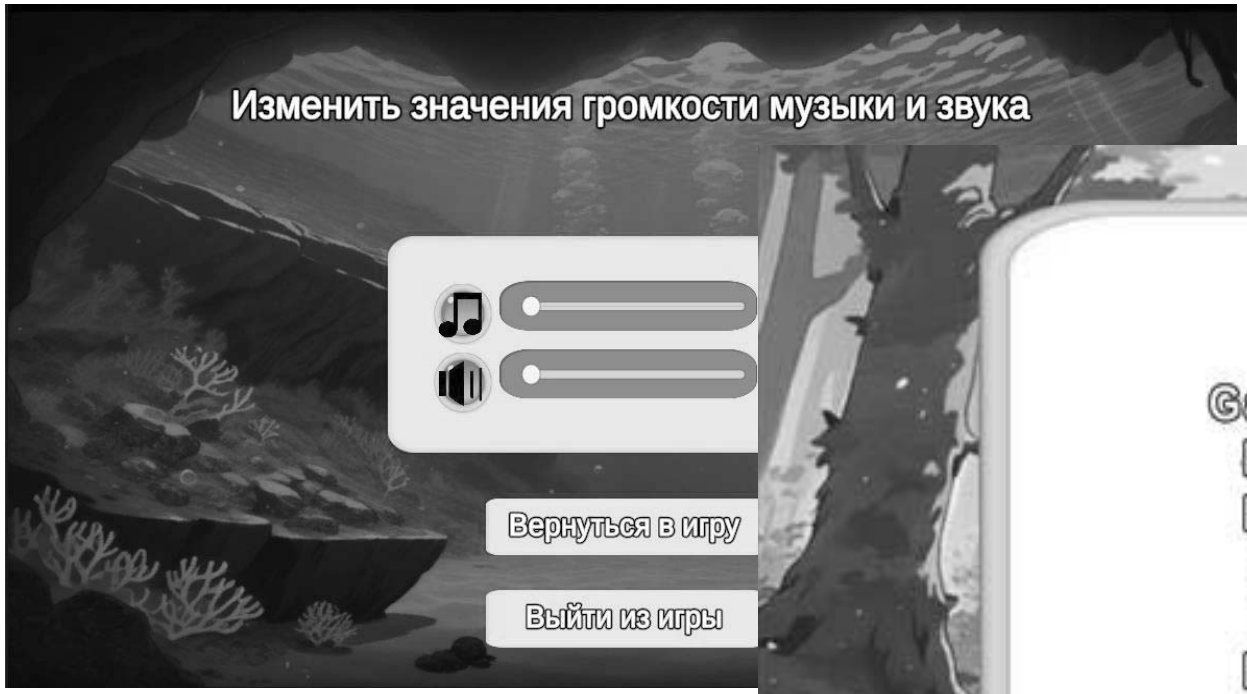
Процесс изменения игрового мира






```
// Вычисление новой яркости фона на основе процента прохождения уровня
Color newColor = new Color(
    Mathf.Lerp(view.CurrentColor.r, model.TargetColor.r, percent),
    Mathf.Lerp(view.CurrentColor.g, model.TargetColor.g, percent),
    Mathf.Lerp(view.CurrentColor.b, model.TargetColor.b, percent),
    view.CurrentColor.a
);
view.ChangeColor(newColor);
```

Экраны паузы и результатов



Тестирование

Тестирование требований:

- ❖ Проведено 6 тестов управления внутри стартовой сценой
- ❖ Проведено 15 тестов управления внутри игровой сцены

- ❖ Проверена работоспособность на операционных системах Windows и Linux

Юзабилити-тестирование:

- ❖ 4 группы пользователей по 2 человека
- ❖ При первом знакомстве с приложением время прохождения составило в среднем: 10, 20 и 25 минут для трех уровней сложности соответственно.
- ❖ Повторное прохождении игры на третьем уровне сложности составлялось в среднем 15 минут.
- ❖ Общее субъективное впечатление пользователей: расслабляющий увлекательный игровой процесс.
- ❖ Было отмечено, что отсутствие элементов сражения и противников позволяет сосредоточиться на посещение новых комнат, сборе предметов и исследовании игрового мира.

Изменение баланса игры

Параметр	Уровень сложности 1	Уровень сложности 2	Уровень сложности 3	Уровень сложности 1	Уровень сложности 2	Уровень сложности 3
Количество комнат	10	15	20	5	10	15
Количество очков за кристалл	1	2	3	2	2	2
Количество кристаллов в комнате	(20-30)*1.1	(30-50)*1.2	(50-55)*1.3	(20-35)*1.1	(35-50)*1.2	(50-65)*1.3
Процент количества кристаллов для открытия двери	20%	30%	50%	20%	25%	35%
Процент количества не-статичных платформ	10%	20%	30%	10%	20%	35%
Количество особых предметов для открытия сундука	3	7	10	3	5	7

- На первом уровне сложности игра является проходимой по всем комнатам
- На третьем уровне сложности могут быть посещены не все комнаты, если игрок не оптимально распределит ресурсы

Достижения

Верман П.Г., **Блинова Е.М**. Применение процедурной генерации при создании игрового мира // Наука ЮУрГУ: материалы 76-й научной конференции профессорско-преподавательского состава. Секции технических наук. (принята к публикации)

Основные результаты

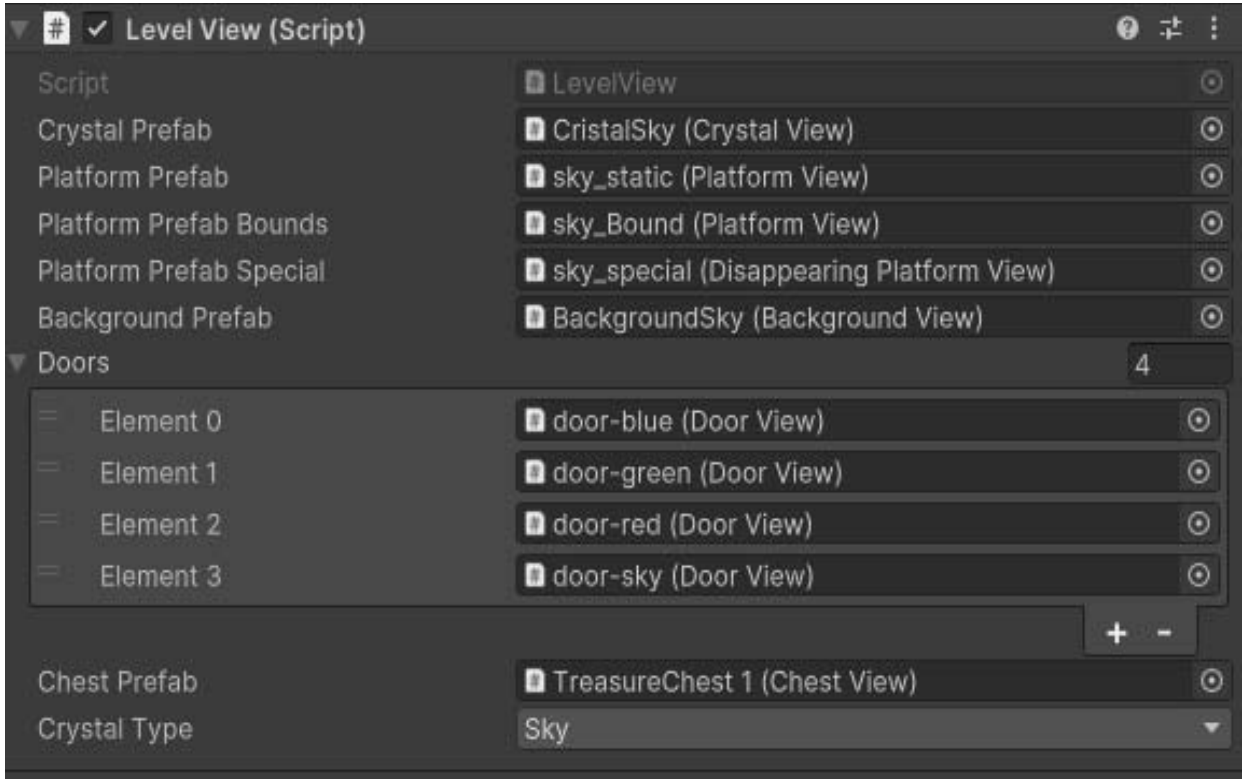
1. Проведен анализ предметной области
2. Выполнен обзор алгоритмов процедурной генерации
3. Спроектировано игровое приложение
4. Реализовано игровое приложение
5. Проведено тестирование игрового приложения

Интегрировано 5 алгоритмов процедурной генерации для создания различного контента игрового мира.

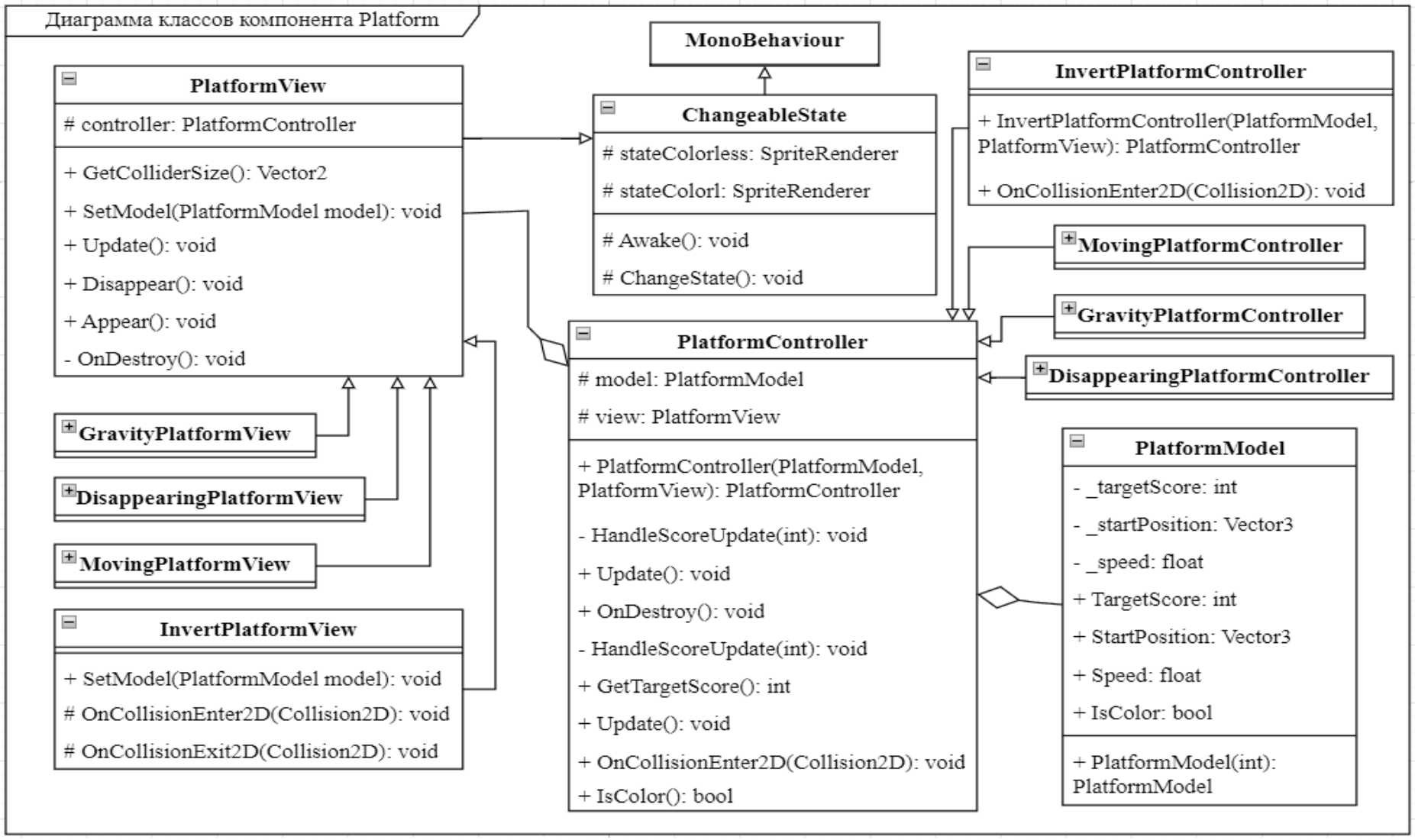
НАПРАВЛЕНИЕ ДАЛЬНЕЙШЕЙ РАЗРАБОТКИ

1. Внедрение системы противников на третьем уровне сложности
2. Внедрение процедурной генерации квестов и сценариев

Реализованные префабы проекта



Реализация платформ



Реализация инвентаря

Параметры результатов генерации комнат

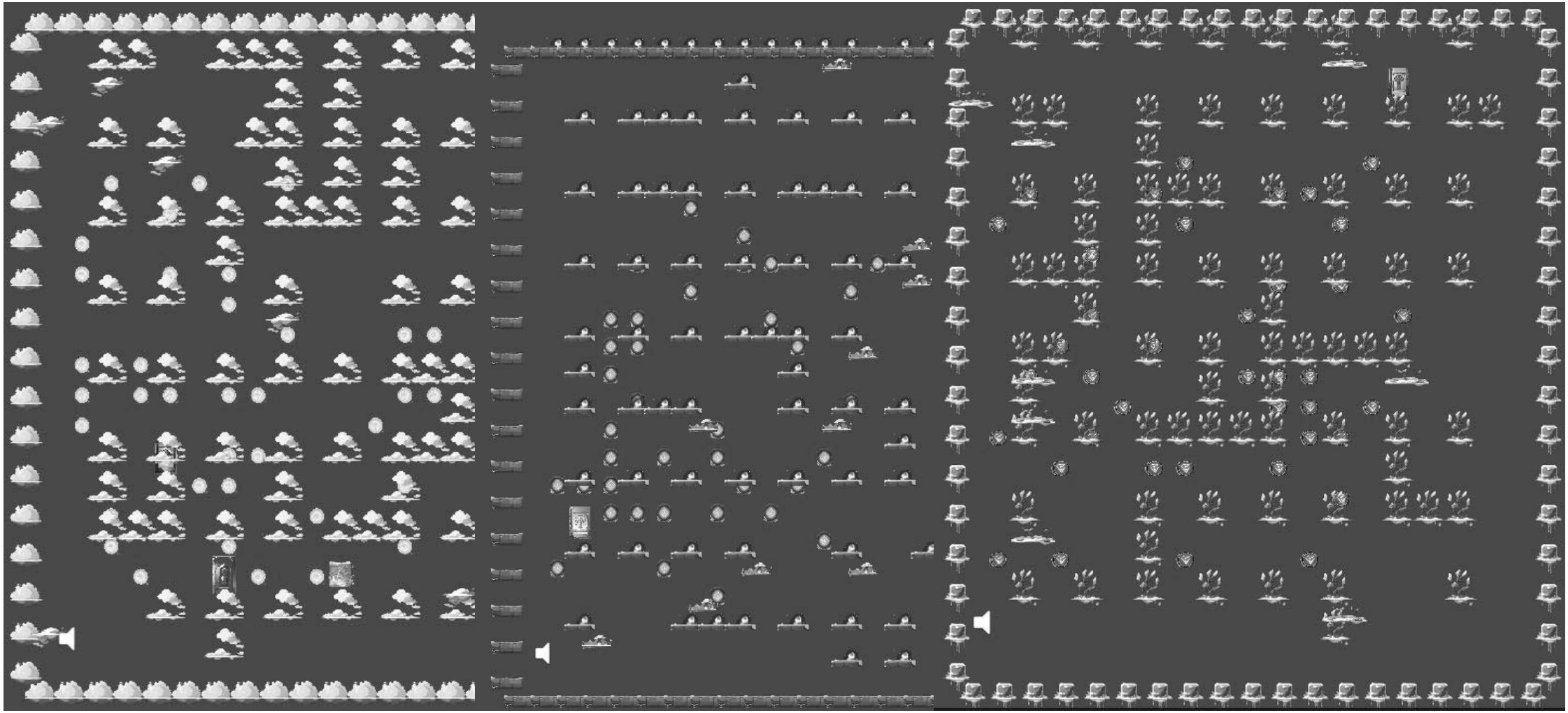
Платформы из лабиринта

[Generator] RandomGridOrMazePlatforms: randomNumber = 2
[Generator] Generate MazePlatforms
[Generator][MazePlatform] labelSize = (1.00, 1.30)
[Generator][MazePlatform] width = 17
[Generator][MazePlatform] height = 17
[Generator][MazePlatform] removalProbability = 0,9
[Generator] GetScorePerCrystal = 2
[Generator] countCrystal = 39
[Generator] countForOpenDoor = 7

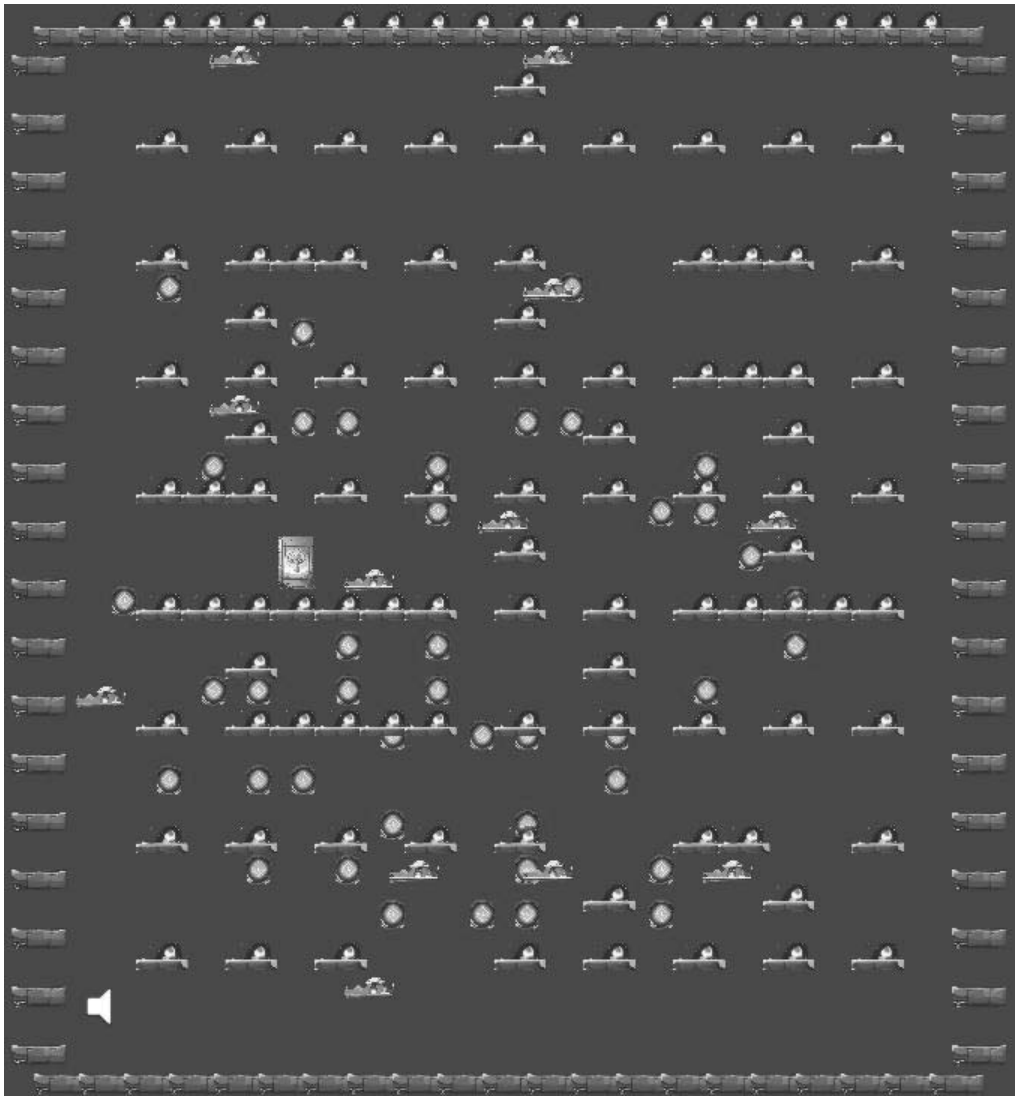
Платформы по регионам

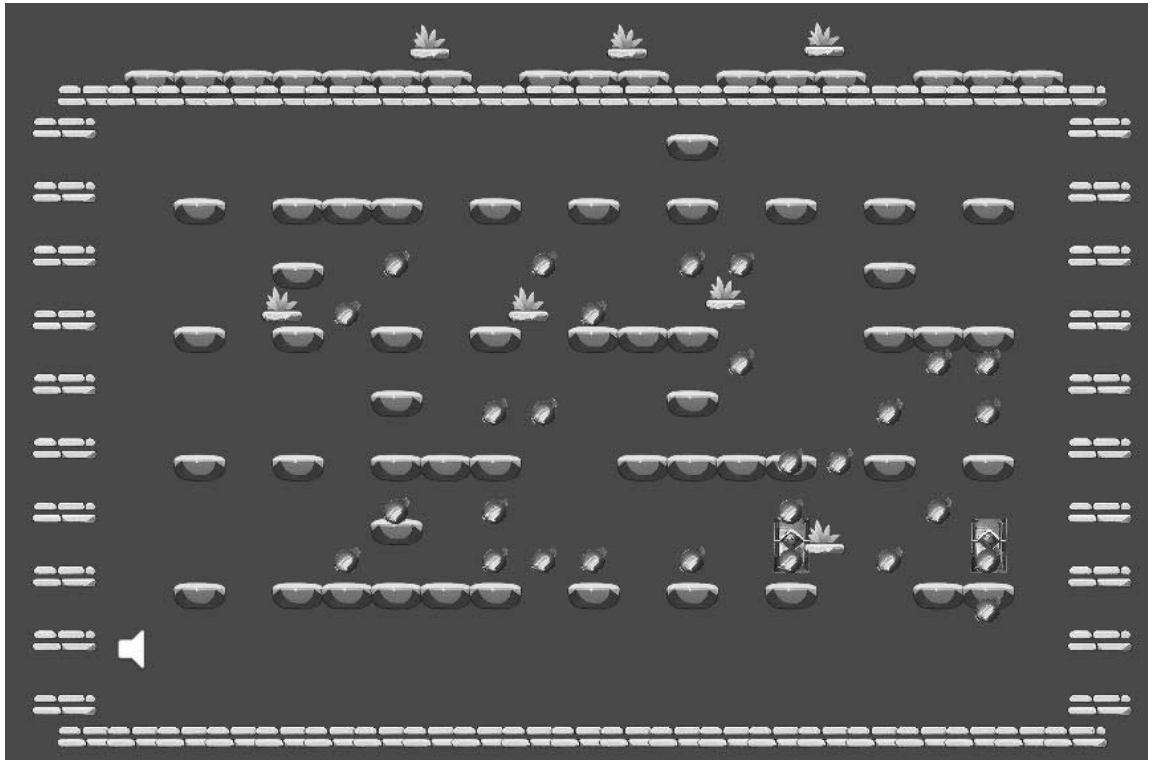
[Generator] RandomGridOrMazePlatforms: randomNumber = 9
[Generator] Generate RandomGridPlatforms
[Generator][GeneratorGridPlatform] labelSize = (2.00, 3.00)
[Generator][GeneratorGridPlatform] grid = {Width=17, Height=12}
[Generator][GeneratorGridPlatform] regions = 32
[Generator] GetScorePerCrystal = 3
[Generator] countCrystal = 37
[Generator] countForOpenDoor = 6

Пример – игровой мир комнаты 0-2

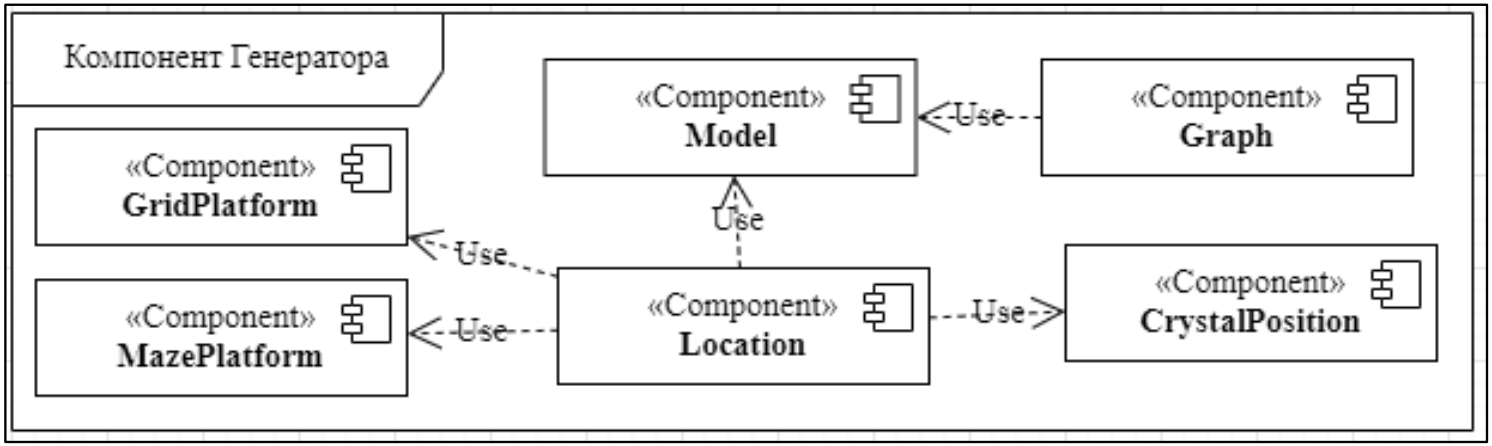


Пример – игровой мир комнаты 3-4





Компонент генератора игрового мира

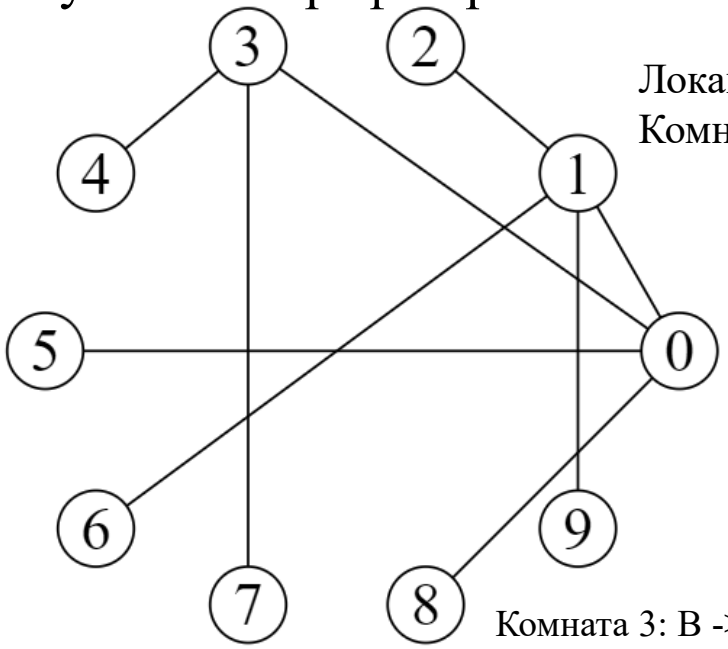


Действия:

- ❖ Связи между локациями
- ❖ Размер комнаты
- ❖ Расположение игровых объектов
- ❖ Количество кристаллов в комнате
- ❖ Количество очков за кристалл
- ❖ Точки переходов между комнатами расположены на заданном расстоянии

Создание карты подземелья

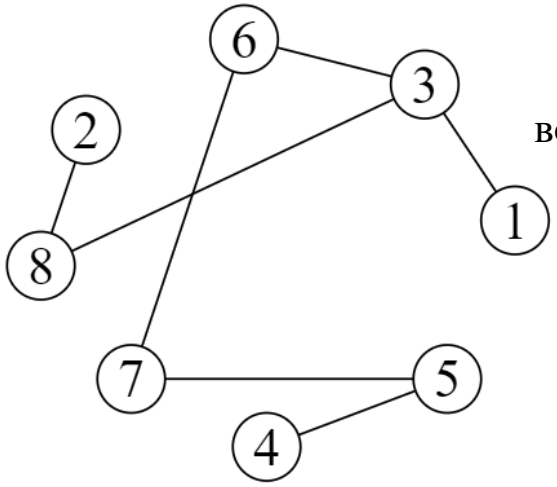
Случайный граф Барабаши-Альберта



Локации: [‘А’, ‘Б’, ‘В’, ‘Г’]
Комнаты 1-10

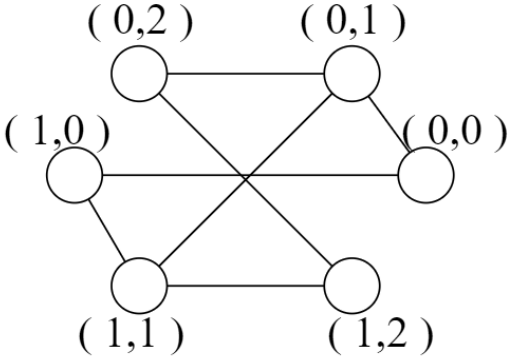
Переходы между комнатами:
Комната 1: В -> Комната 2: Г,
Комната 4: А,
Комната 6: В,
Комната 9: А
Комната 2: Г -> Комната 1: В,
Комната 3: В,
Комната 7: В,
Комната 10: В
Комната 3: В -> Комната 2: Г
Комната 4: А -> Комната 1: В,
Комната 5: Б,
Комната 8: А
Комната 5: Б-> Комната 4: А
Комната 6: В-> Комната 1: В
Комната 7: В-> Комната 2: Г
Комната 8: А-> Комната 4: А
Комната 9: А-> Комната 1: В
Комната 10: В-> Комната 2: Г

Модель случайного графа



Количество начальных точек: n=4 Количество всевозможных пар точек: C(2n,2) = n*(2n-1)=28
Сгенерированное количество ребер:7

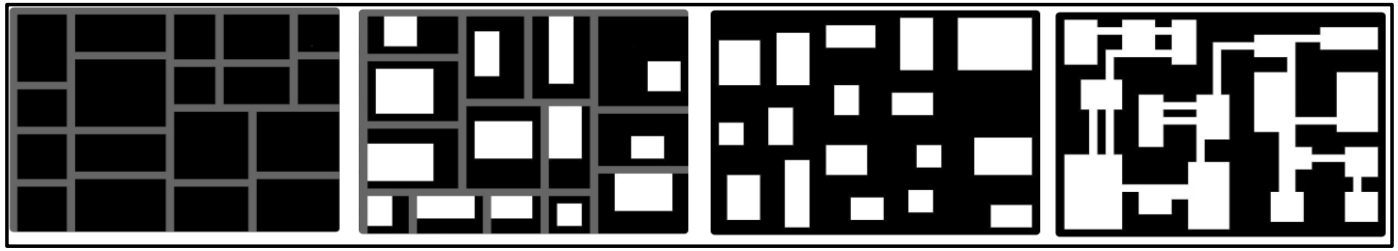
По модели графа на основе сетки



Количество строк: 2
Количество столбцов: 3

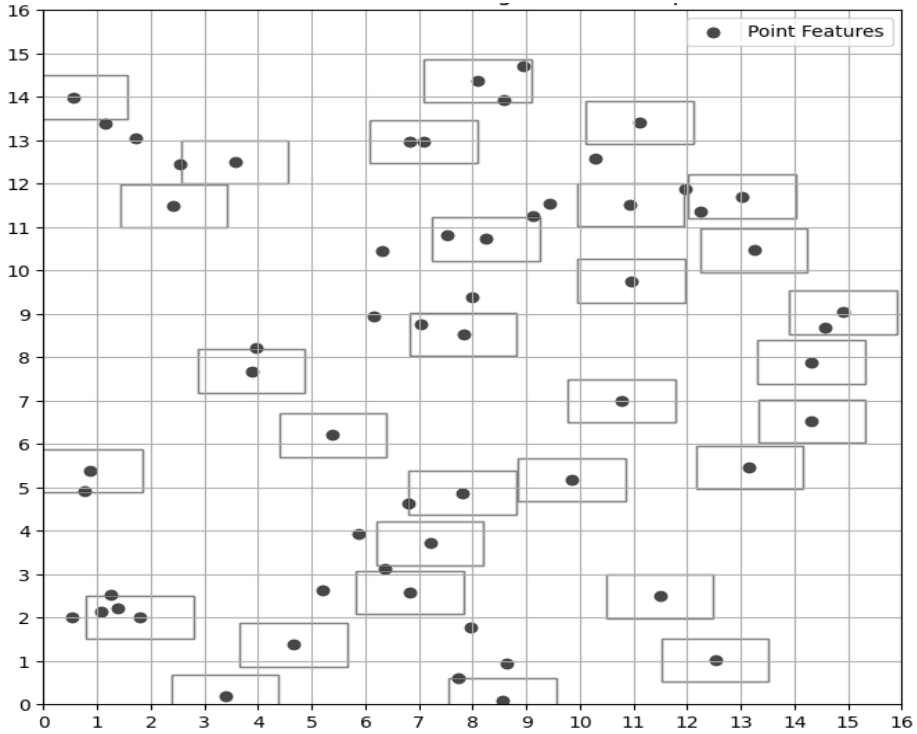
Создание платформ по регионам

Алгоритм двоичного разбиения пространства



https://gamedevelopment.tutsplus.com/how-to-use-bsp-trees-to-generate-game-maps--gamedev-12268t

Алгоритм размещения меток на основе сетки



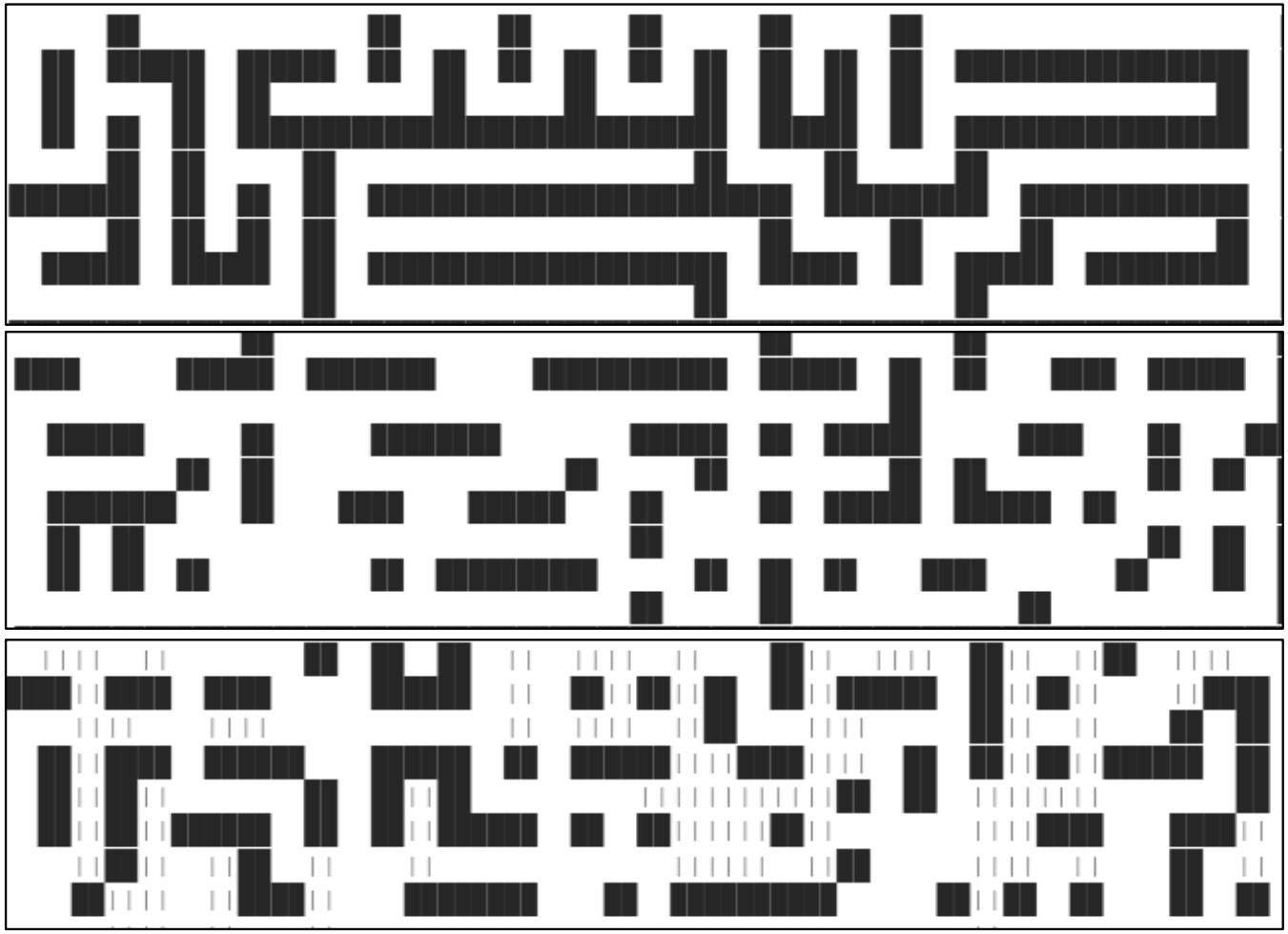
Особенности:

- ✓ Игровое пространство разбивается на сегменты
- ✓ Простота интеграции
- ✓ Комбинация структурированных и случайных платформ

Размер сетки: 16 x 16
Размер метки: 2 x 1
Количество точек: 60

Создание платформ из лабиринта

Генерация лабиринта методом поиска в глубину



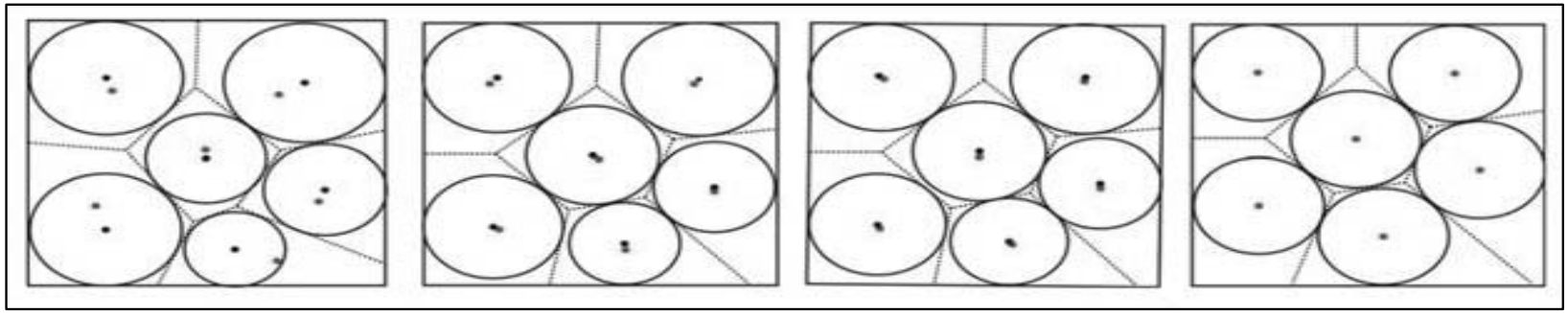
Особенности:

- ✓ Регулировка сложности игрового окружения
- ✓ Простота реализации и интеграции
- ✓ Высокий контроль над расположением платформ
- ✓ Возможность внедрения дополнительных структурных элементов (лестницы, лифты)

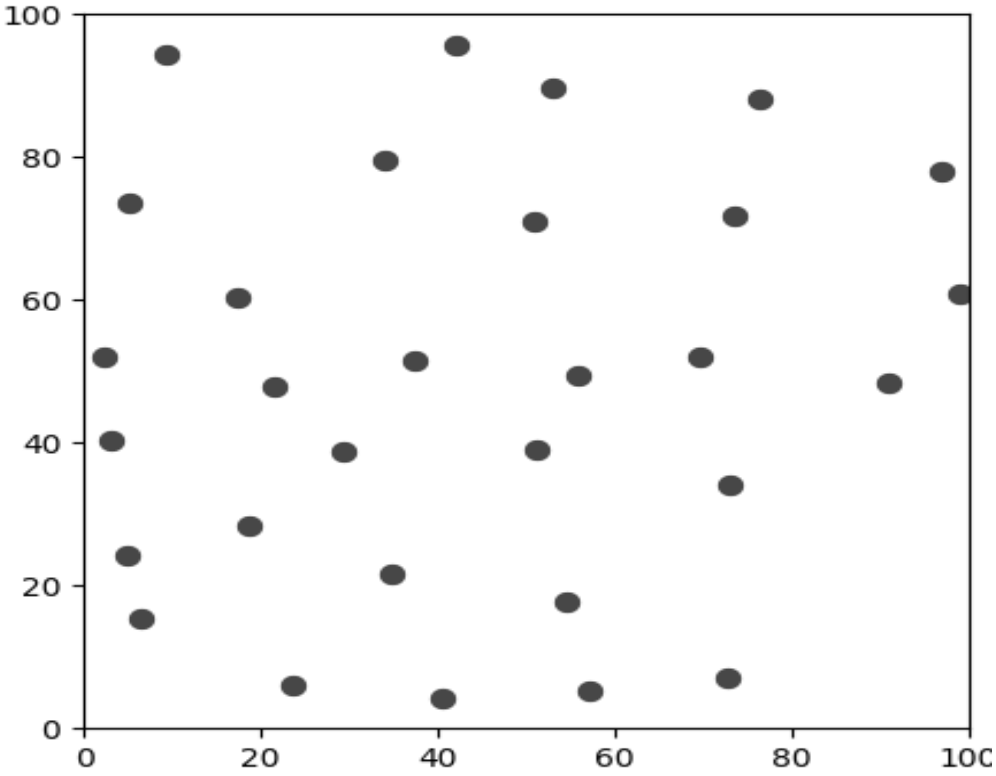
Размер поля: 40 x 10
Вероятность создания прорези: 0.4
Вероятность создания лестницы: 0.3

Размещение кристаллов

Процесс упаковки дисков Пуассона



Poisson disk sampling through disk packing / G. Liang, L. Lu, Zh. Chen, Ch. Yang. // Computational Visual Media. – 2015. – Vol. 1, No. 1. – P. 17–26.



Особенности:

- ✓ Равномерное распределение кристаллов
- ✓ Без перекрытия
- ✓ Гибкость настройки параметров

Размер поля: 100 x 100
Радиус: 5
Количество точек: 30
Количество шагов релаксации: 5

Оценка сложности алгоритмов

Алгоритм генерации графа
Временная сложность:
$O(n*m + m^2)$, где: n – количество комнат, а m – количество начальных узлов
Пространственная сложность:
O(n * m)

Алгоритм генерации точек
Временная сложность:
O(n * r) где: n – количество образцов (кристаллов), r – количество шагов релаксации
Пространственная сложность:
O(n + w * h)

Алгоритм генерации лабиринта
Временная и пространственная сложность:
O(width* height)

Алгоритм двоичного разбиения пространства
Временная и пространственная сложность:
$O(2^{depth})$