

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS**

**Процедурная генерация аренных игровых уровней для персонажей разного типа в
шутере от первого лица**

Обучающийся / Student Артемьев Федор Антонович
Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр
Группа/Group J4223
Направление подготовки/ Subject area 09.04.03 Прикладная информатика
Образовательная программа / Educational program Технологии разработки
компьютерных игр 2022
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Магистр
Руководитель ВКР/ Thesis supervisor Загарских Александр Сергеевич, кандидат
технических наук, Университет ИТМО, школа разработки видеоигр, доцент
(квалификационная категория "доцент практики")

Обучающийся/Student

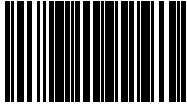
Документ подписан	
Артемьев Федор Антонович	
22.05.2024	

(эл. подпись/ signature)

Артемьев
Федор
Антонович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Загарских Александр Сергеевич	
22.05.2024	

(эл. подпись/ signature)

Загарских
Александр
Сергеевич

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Артемьев Федор Антонович
Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр
Группа/Group J4223
Направление подготовки/ Subject area 09.04.03 Прикладная информатика
Образовательная программа / Educational program Технологии разработки компьютерных игр 2022
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Процедурная генерация аренных игровых уровней для персонажей разного типа в шутере от первого лица
Руководитель ВКР/ Thesis supervisor Загарских Александр Сергеевич, кандидат технических наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная категория "доцент практики")

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

Основные вопросы, подлежащие разработке / Key issues to be analyzed

- Изучить паттерны создания NPC в шутерах;
- Изучить паттерны создания игровых уровней в шутерах;
- Изучить методы генерации игровых уровней;
- Создание алгоритма оценки противников для формирования входных данных алгоритма генерации
- Создание алгоритма генерации игровых уровней для FPS;
- Реализация алгоритмов оценки и генерации

1. I3D'19Zhang Y., Zhang G., Huang X. A survey of procedural content generation for games //2022 International Conference on Culture-Oriented Science and Technology (CoST). – IEEE, 2022. – С. 186-190.
2. Henrik K. Minecraft terrain generation in a nutshell // YouTube [Electronic resource] – URL: <https://www.youtube.com/watch?v=CSa5O6knuwI>
3. Global Competitive Gaming Market Surges // excorp.gg [Electronic resource] – URL: <https://excorp.gg/news/global-competitive-gaming-market-surges-reached-64.4>
4. Rivera G., Hullett K., Whitehead J. Enemy NPC design patterns in shooter games

- //Proceedings of the First Workshop on Design Patterns in Games. – 2012. – С. 1-8.
5. Qu J., Song Y., Wei Y. Design patterns applied for game design patterns //2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). – IEEE, 2016. – С. 351-356.
 6. Rivera G., Hullett K., Whitehead J. Enemy NPC design patterns in shooter games //Proceedings of the First Workshop on Design Patterns in Games. – 2012. – С. 1-8.
 7. Hullett K., Whitehead J. Design patterns in FPS levels //proceedings of the Fifth International Conference on the Foundations of Digital Games. – 2010. – С. 78-85.
 8. Allen P. M., Alonso W., Applebaum D. Alexander, C.(1964), Notes on the Synthesis of Form, Harvard University Press, Cambridge, MA. Alexander, C.(1965), 'A city is not a tree', Architectural Forum 122, 58–62. Alexander, C., Ishikawa, S. & Silverstein, M.(1977), A Pattern Language: Towns, Buildings, Construction, Oxford University Press, New York. With M. Jacobson, I //system. – Т. 11. – №. 3. – С. 256-272.
 9. Gamma E. et al. Design patterns: Abstraction and reuse of object-oriented design //ECOOP'93 —Object-Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings 7. – Springer Berlin Heidelberg, 1993. – С. 406-431.
 10. Dahlskog S., Togelius J. A multi-level level generator //2014 IEEE Conference on Computational Intelligence and Games. – IEEE, 2014. – С. 1-8.
 11. Cachia W., Liapis A., Yannakakis G. Multi-level evolution of shooter levels //Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. – 2015. – Т. 11. – №. 1. – С. 115-121.
 12. Viana B. M. F., dos Santos S. R. Procedural dungeon generation: A survey //Journal on Interactive Systems. – 2021. – Т. 12. – №. 1. – С. 83-101.
 13. Shaker N., Togelius J., Nelson M. J. Procedural content generation in games. – 2016.

Форма представления материалов ВКР / Format(s) of thesis materials:

Пояснительная записка (ВКР). Презентация.

Дата выдачи задания / Assignment issued on: 22.02.2024

Срок представления готовой ВКР / Deadline for final edition of the thesis 24.05.2024

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Загарских Александр Сергеевич	
30.04.2024	

Загарских
Александр
Сергеевич

(эл. подпись)

Задание принял к
исполнению/ Objectives
assumed BY

Документ подписан	
Артемьев Федор Антонович	
03.05.2024	

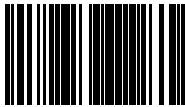
Артемьев
Федор
Антонович

(эл. подпись)

Руководитель ОП/ Head

Документ	
----------	--

of educational program

подписан	
Карсаков Андрей Сергеевич	
22.05.2024	

(эл. подпись)

Карсаков
Андрей
Сергеевич

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Артемьев Федор Антонович

Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр

Группа/Group J4223

Направление подготовки/ Subject area 09.04.03 Прикладная информатика

Образовательная программа / Educational program Технологии разработки компьютерных игр 2022

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Магистр

Тема ВКР/ Thesis topic Процедурная генерация аренных игровых уровней для персонажей разного типа в шутере от первого лица

Руководитель ВКР/ Thesis supervisor Загарских Александр Сергеевич, кандидат технических наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная категория "доцент практики")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Разработка алгоритма процедурной генерации аренных игровых уровней зависящего от специфики агентов – противников в шутере от первого лица.

Задачи, решаемые в ВКР / Research tasks

1. Изучить паттерны создания NPC в шутерах; 2. Изучить паттерны создания игровых уровней в шутерах; 3. Изучить методы генерации игровых уровней; 4. Создание алгоритма генерации игровых уровней для FPS; 5. Реализация алгоритма генерации игровых уровней для FPS; 6. Экспериментальное тестирование полученного алгоритма;

Краткая характеристика полученных результатов / Short summary of results/findings

В ходе работы был проведен анализ существующих работ, направленных на изучение используемых в видеоиграх в жанре FPS паттернов проектирования игровых уровней и паттернов создания противников. Также были исследованы некоторые подходы и алгоритмы, используемые при генерации двух- и трехмерных игровых уровней. Исходя из данного анализа был разработан теоретический алгоритм процедурной генерации уровня для игры в жанре Roguelike FPS с учетом особенностей набора противников и применяемых при проектировании игровых уровней паттернов. Результатам данной работы стал разработанный алгоритм процедурной генерации аренных игровых уровней зависящего от специфики агентов – противников в шутере от первого лица. Эффективность его работы была подтверждена экспериментальными данными.

Обучающийся/Student

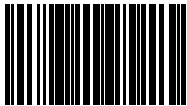
Документ подписан	
Артемов Федор Антонович	
22.05.2024	

(эл. подпись/ signature)

Артемов
Федор
Антонович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Загарских Александр Сергеевич	
22.05.2024	

(эл. подпись/ signature)

Загарских
Александр
Сергеевич

(Фамилия И.О./ name
and surname)

СОДЕРЖАНИЕ

СЛОВАРЬ ТЕРМИНОВ	9
ВВЕДЕНИЕ	10
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	13
1.1 Паттерны проектирования	13
1.1.1 Паттерны создания NPC в шутерах.....	13
1.1.2 Паттерны дизайна игровых уровней в шутерах.....	20
1.2 Методы генерации локаций	24
1.2.1 Традиционный подход.....	24
1.2.2 Подход на основе поиска.....	24
1.2.3 Алгоритмы, основанные на использовании МО.....	25
1.3 Алгоритмы генерации.....	26
1.3.1 Генерация с использованием BSP	26
1.3.2 Клеточные автоматы	27
1.3.3 WFC	28
1.3.4 Диаграмма Вороного.....	29
1.3.5 Алгоритмы с использованием шума Перлина и симплексного шума 30	
1.3.6 Генетические алгоритмы	30
1.3.7 Итоги анализа различных способов генерации локаций	31
1.4 Вывод.....	31
2 Описание собственного алгоритма	33
2.1 Параметры, вычисление и оценка эффективности NPC	34
2.1.1 Оценка базовой эффективности SE	35
2.1.2 Коэффициент предпочтительности выбора макропаттерна kp	36
2.1.3 Коэффициент влияния на эффективность NPC в зависимости от наличия мезопаттерна kpe	38
2.1.4 Коэффициент влияния на эффективность NPC в зависимости от типа движения и наличия определенного паттерна $kmt - pe$	39
2.2 Формирование входных данных.....	39
2.2.1 Выбор макропаттерна	39

2.2.2	Выбор мезопаттернов	40
2.3	Генерация уровня	40
2.3.1	Генерация основания	41
2.3.2	Добавление мезопаттернов	41
2.3.3	Добавление микропаттернов.....	48
3	Практическая реализация	49
3.1	Формирование и оценка набора противников	49
3.2	Генерация макропаттерна	49
3.3	Генерация мезопаттернов.....	51
3.4	Генерация микропаттернов	55
4	ЭКСПЕРЕМЕНТАЛЬНАЯ ПРОВЕРКА ГИПОТЕЗЫ.....	59
4.1	Формирование параметров оценки	59
4.2	Проведение эксперимента	61
4.3	Результаты.....	62
4.3.1	Самоубийца.....	63
4.3.2	Ловкач.....	63
4.3.3	Берсерк	64
4.3.4	Снайпер	64
4.3.5	Пехотинец	64
	ЗАКЛЮЧЕНИЕ	65
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	66

СЛОВАРЬ ТЕРМИНОВ

FPS – First-person shooter, шутер от первого лица.

AAA – термин используется в индустрии компьютерных игр для обозначения высокобюджетного класса игр, рассчитанных на массовую аудиторию и требующих колоссальных затрат на разработку самой игры и ее маркетинг.

NPC – Non-player character, неигровой персонаж.

PCG – Procedural Content Generation, процедурная генерация контента.

МО – машинное обучение.

WFC – Wave Function Collapse, коллапс волновой функции, один из методов, применяемый в процедурной генерации контента.

ВВЕДЕНИЕ

С развитием видеоигр сложность их создания постоянно возрастает, вместе с этим возрастает количество необходимого для каждой игры контента. Цена на создание различного рода ассетов, уровней, игровой логики и остального наполнения игры также не уменьшается. В процессе создания AAA игр на производство игрового контента тратятся десятки миллионов долларов. Популярным способом для сокращения расходов и времени, затрачиваемого на ручное создание наполнения игр, стала процедурная генерация контента (Procedural Content Generation), которая, в последние время, приобрела огромную популярность в рядах разработчиков игр. [1]

Помимо сокращения затрат, процедурная генерация контента позволяет создавать бесконечные, уникальные и непредсказуемые уровни, миры [2]. Хорошими примерами игр с использованием процедурной генерации миров являются Minecraft и No Man's Sky. Процедурная генерация позволяет повысить реиграбельности игры. Игроки могут возвращаться к игре многократно и каждый раз получать уникальные и разнообразные игровые ситуации, что оживляет игру и позволяет наслаждаться ею вновь и вновь.

В целом, процедурная генерация контента является мощным инструментом, который значительно расширяет и обогащает возможности игрового мира, создавая интересный, разнообразный и уникальный игровой опыт.

Одним из самых популярных жанров в игровой индустрии является FPS, доходы от продаж в этом жанре начиная с 2019 года превышают доходы любого другого жанра. Помимо это число уникальных пользователей в месяц в данном жанре стало наибольшим в 2018 и с тех пор держит свои лидирующие позиции [3].

Процедурная генерация контента широко используется в подвиде FPS – Rouge like FPS, в котором очень часто применяют и процедурную генерацию игровых локаций, в частности. При детальном анализе различных подходов к генерации уровней в различных играх данного жанра было обнаружено, что

уровни зачастую сильно похожи между собой, а также слабо зависят от специфики врагов, представленных на уровне. Из-за этого уменьшается эффективность одних агентов, сильно повышает эффективность других, а самое главное, однообразие игровых уровней редко побуждает игрока придумывать новые тактики боя, даже при наличии разных наборов противников на поле боя. В связи с этим, кажется актуальным вопрос создания алгоритма процедурной генерации уровней для Roguelike FPS, учитывающий количество, а также параметры противников, которые будут находиться на вновь сгенерированной арене.

Объект исследования: Процедурная генерация игровых уровней.

Предмет исследования: Процедурная генерация уровней в зависимости от набора персонажей.

Цель работы: разработка алгоритма процедурной генерации аренных игровых уровней, зависящего от специфики агентов – противников в шутере от первого лица.

Задачи:

- 1) Изучить паттерны создания NPC в шутерах;
- 2) Изучить паттерны создания игровых уровней в шутерах;
- 3) Изучить методы генерации игровых уровней;
- 4) Создание алгоритма генерации игровых уровней для FPS;
- 5) Реализация алгоритма генерации игровых уровней для FPS;
- 6) Экспериментальное тестирование полученного алгоритма.

Актуальность: Использование процедурной генерации контента в играх – тренд последних лет. Такой подход позволяет сократить время разработки, увеличить разнообразие и реиграбельность игры.

Научная новизна: На данный момент было обнаружено несколько аналогичных исследований, связанных с процедурной генерацией игровых уровней, однако ни в одном из них не учитывается особенность находящегося на ней набора агентов – противников.

Решаемая проблема: Алгоритмы процедурной генерации уровней в Roguelike FPS играх практически не учитывают специфику набора противников, что негативно сказывается на игровом опыте. В данной работе мы хотим повысить его качество путем создания более осмысленного и разнообразного алгоритма генерации аренных игровых уровней.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Паттерны проектирования

Первоначально паттерны, связанные с архитектурой, для решения проблем были описаны в работе Александра и др. [8]. В объектно-ориентированной разработке проблема, решаемая использованием паттерна, является распространённым и повторяющимся элементом проектирования [9]. В программном приложении шаблоны проектирования дают дизайнерам представление об архитектурных знаниях и служат шаблоном для многих ситуаций. Коллекция возможных вариантов дизайна и паттернов в играх весьма обширна. Изучение различных вариантов проектирования может помочь автоматизировать разработку игр.

1.1.1 Паттерны создания NPC в шутерах

Для создания принципов построения уровней, а также методов их оценки необходимо определить и разобрать основные паттерны создания противников в жанре FPS.

В данном разделе внимания достойны работы [4] и [5] в которых выделяются основные типы противников, а также их параметры.

1.1.1.1 Основные параметры противников

Тип передвижения – Характеристика, описывающая как NPC обычно будет передвигаться в боевой ситуации. Некоторые NPC могут использовать сразу несколько типов движения или же могут переключаться между ними в зависимости от ситуации. Среди них выделяются:

Обходящий с флангов – NPC будет атаковать с неожиданных направлений, т.е. NPC будет пытаться приблизиться к игроку с той стороны, на которую не направлено его внимание.

Пассивный – NPC практически не двигается при атаке и никогда не отходит слишком далеко от места появления и доступного укрытия.

Тип с медленным продвижением – NPC будет медленно продвигаться к позиции игрока и, как правило, делать это по прямой. Происходить это может быть без необходимости в укрытии, однако, иногда NPC может использовать

их, при продвижении вперед. Главное отличие этого типа от осторожного заключается в том, что он будет постоянно пытаться сократить дистанцию до своей цели, а не пытаться держать дистанцию.

Стремительный – NPC совершает рывок к определенной цели, не заботясь о своей безопасности, как правило, по прямой. Однако, главным аспектом этого типа передвижения является то, что они атакуют очень быстро и часто пытаются сократить дистанцию между собой и своей целью как можно быстрее.

Осторожный – NPC с данным типом движения предпочитает передвигаться по полю боя, но старается держаться на расстоянии от своей цели. По возможности старается использовать укрытие и не сокращать дистанцию. Отличие от типа с медленным продвижением заключается в том, что этот NPC пытается поддерживать определенный радиус вокруг своей цели, не продвигаясь вперед.

Способ перемещения – то, каким способом NPC перемещается по уровню: по поверхности или по воздуху

Дальность перемещения – От нее зависит, как далеко NPC будет перемещаться во время боя.

Частота передвижения – то, как часто NPC будет менять позицию во время боя.

Скорость перемещения – то, с какой скоростью NPC перемещается по уровню.

Частота атак – Описывает, как часто NPC будет инициировать атаку.

Тип оружия – В этой общей категории используются известные паттерны проектирования оружия в шутерных играх. Каждый тип достаточно подробно разобран в работе [6], авторы выделяют следующие:

- 1) Снайперское оружие;
- 2) Взрыв на близкой дистанции;
- 3) Штурмовое оружие;
- 4) Взрывные снаряды;

- 5) Силовое оружие (лазер);
- 6) Оружие ближнего боя.

Однако такое разделение не сильно информативно в контексте параметрических оценок, поэтому стоит выделить несколько базовых характеристик:

Дальность атаки – Ближний бой, средняя / высокая дальность

Тип урона – По цели или по площади

Урон оружия – Глобальный показатель того насколько какой урон наносит тот или иной NPC по щитам или здоровью игрока.

Точность атаки – Показатель того насколько вероятно попадание при нанесении атаки по игроку.

Броня и здоровье – Описывает значение урона, которое NPC может получить до своей смерти.

Мотив – указывает на то, какой тип боевого столкновения создал бы NPC, и показывает его цель в дизайне. Три основных фактора, на которые может повлиять NPC:

- 1) Вызов – степень сложности в боевом столкновении;
- 2) Напряжение – степень психического напряжения, которое игрок испытывает во время боевого столкновения;
- 3) Темп – степень движения, в которой игрок будет участвовать во время боевого столкновения.

1.1.1.2 Основные шаблоны NPC

Основываясь на параметрах NPC, можно выделить следующие шаблоны и их подвиды:

Солдат – NPC, который оказывает давление на игрока с средней дистанции:

- 1) Пехотинец – слабый NPC с небольшим количеством здоровья и урона, который атакует со среднего расстояния, часто группами.

2) Элита – улучшенная версия пехотинца, сильный NPC, который действует со среднего расстояния и зачастую обладает дополнительными способностями.

3) Гренадер – NPC, параметры которого часто совпадают с пехотинцем, однако, он держится на большом расстоянии, и использует взрывные снаряды, чтобы повысить напряжение игрока и побудить его двигаться вперед.

4) Снайпер – NPC, наносящий высокий урон с большого расстояния.

Агрессор – NPC, который пытается сократить расстояние между собой и своей целью:

1) Самоубийца – бросается на игрока ценой своей жизни, зачастую обладает низким здоровьем и может взрываться при сближении.

2) Рой – атакует игрока группой, но индивидуальный урон невелик.

3) Берсерк – сильный NPC, наносящий большое количество урона в течение длительного периода времени, также обладает большим количеством здоровья.

Носитель – NPC, который порождает других NPC во время столкновения:

1) Жертвенный – NPC создает больше NPC после своей смерти.

2) Призыватель – порождает больше NPC на расстоянии.

3) *Танк* – NPC, который представляет значительную единичную угрозу и мешает игроку продолжить игру.

4) Турель – медленно движущийся или статичный NPC, наносящий высокий урон на большом расстоянии.

5) Защищенный – NPC с большим количеством брони, но только в одном направлении.

Так же стоит отметить, что в представленных выше работах отсутствует паттерн *Поддержка*, их основная роль – усиление или защита остальных NPC. Хорошим примером такого NPC является Arch-vile из игры Doom Eternal

(2020), он способен защищать своих союзников путем установки щита, а также усиливать их характеристики.

Теперь, после описания всех типов врагов, составим таблицу с примерным описанием характеристик каждого типа (характеристики не всех паттернов могут быть оценены даже примерно так как их дизайн и параметры очень сильно варьируется от игры к игре):

Таблица 1 – Описание характеристик противников для каждого паттерна

	Тип передвижения	Скорость перемещения	Частота перемещения	Дальность атаки	Тип урона	Урон атак	Частота атак	Точность атак	Здоровье	Броня
Пехотинец	Медленное продвижение / пассивный / осторожный	Средняя	Средняя	Средняя	По цели	Слабый	Средняя	Низкая	Низкое	Нет
Элита	Медленное продвижение / пассивный / осторожный	Средняя	Средняя	Средняя	По цели	Средний	Средняя	Средняя	Среднее	Слабая
Гренадер	Медленное продвижение / пассивный / осторожный	Средняя	Средняя	Высокая	По области	Высокий	Низкая	Средняя	Низкое	Слабая
Снайпер	Пассивный / осторожный	Низкая	Низкая	Очень высокая	По цели	Высокий	Низкая	Очень высокая	Низкое	Нет
Самоубийца	Стремительный	Высокая	Высокая	Ближний бой	По области	Средний	-	Высокая	Низкое	Нет
Рой	Стремительный	Высокая	Высокая	Ближний бой	По цели	Слабый	Средняя	Высокая	Очень низкий	Нет
Берсерк	Стремительный	Высокая	Высокая	Ближний бой	По цели / по области	Сильный	Средняя	Высокая	Высокое	Высокая
Жертвенный	-	-	-	-	-	-	-	-	-	-
Призыватель	Медленное продвижение / пассивный / осторожный	Низкая	Низкая	-	-	-	-	-	-	-
Турель	Пассивный	-	-	Высокая	По цели / по области	Высокий	Средняя	Высокая	Высокое	Высокая
Защищенный	Медленное продвижение	Низкая	Низкая	Ближний бой / средняя	По цели	Высокий	Средняя	Средняя	Среднее	Высокая
Поддержка	Медленное продвижение / пассивный / осторожный	Средняя	Средняя	-	-	-	-	-	-	-

1.1.2 Паттерны дизайна игровых уровней в шутерах

Паттерны создания игровых уровней – это заранее разработанные шаблоны или стратегии, используемые для создания уровней, которые предлагают интересный, увлекательный и сбалансированный геймплей. Эти паттерны помогают гарантировать, что уровни создаются с определенными ожиданиями и стандартами качества, и дают возможность игрокам пройти игру с увлечением и удовольствием.

Шаблоны проектирования ранее применялись для генерации уровней для конкретных игр. В работе [10] авторы использовали шаблоны дизайна для определить модели врагов, промежутков, долин, множества путей и лестниц для создания уровней для Марио, предложили наивный способ объединения обнаруженных шаблонов проектирования в игровые уровни. Кроме того, они представили метод поуровневой генерации карты, разделяя процесс создания на микро-, мезо- и макроуровни. Где микроуровень – вертикальные срезы карты, мезоуровень – их объединение в конкретные функциональные блоки, например – лестница, а макроуровень – целый игровой уровень. Несмотря на то, что алгоритм генерации рассматривается в рамках 2D платформера, такое многоуровневое разделение может применяться и в 3D.

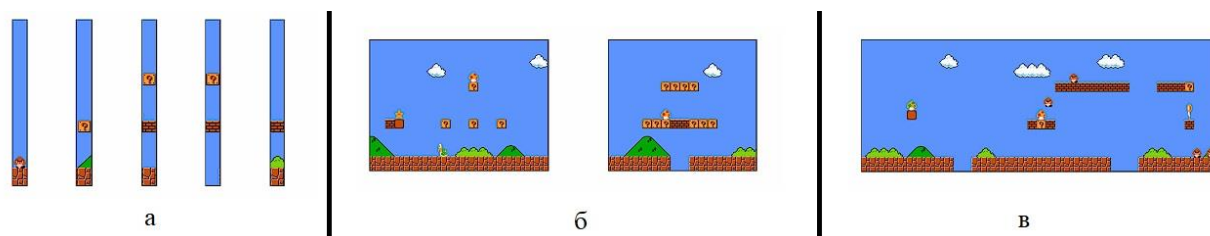


Рисунок 1 – а) микро-; б) мезо-; в) макропаттерн

На данный момент существует не так много работ, в которых рассматриваются паттерны для жанра FPS. Фундаментальной можно считать работу К. Халлетт и Дж. Вайтхеда [7], в ней описаны все основные паттерны создания игровых уровней, используемые в рассматриваемом жанре.

1.1.2.1 Паттерны позиционного преимущества

Основная концепция данного паттерна заключается в том, что один субъект получает преимущество перед другим за счет более высокого шанса совершить атаку и не подвергнуться, при этом, контратаке. Выделяют 3 основных типа:

Снайперская позиция – подразумевает собой локацию, из которой возможно вести огонь с большой дистанции и при этом оставаться защищенным.

Галерея – это возвышенная площадка, параллельная узкому проходу и примыкающая к нему. Персонажи, находящиеся на возвышенности, могут легко атаковать с персонажами в проходе.

Узкий проход – а это узкая область без альтернативных маршрутов. Как правило, это вход в помещение, также это может быть проход на открытой местности ограниченный препятствиями, например, ландшафтом или пропастями. Когда игрок или NPC перемещаются через узкую точку, они подвергаются атаке

1.1.2.2 Паттерны для больших битв

Эти шаблоны предоставляют области для боевого геймплея, где игрок либо вступает в бой с большим количеством вражеских NPC, либо с одним мощным вражеским NPC (битва с боссом). Есть два основных типа:

Арена – это открытая площадка или широкий коридор, где игрок сталкивается с той или иной формой сильного сопротивления, будь то волны вражеских NPC или битва с боссом. Дизайнеры должны позаботиться о том, чтобы игрок не был перегружен. Укрытия и предметы могут быть размещены таким образом, чтобы игрок мог постоянно продвигаться к своей цели от укрытия к укрытию.

Крепость – это ограниченная территория, как правило, с хорошим прикрытием. Персонажи в крепости могут защищаться от нападающих, оставаясь при этом защищенными. В крепости есть ограниченные точки доступа, поэтому обороняющиеся персонажи могут легко их прикрывать.

1.1.2.3 Паттерны для альтернативного геймплея

Эти шаблоны используются для разбивки уровней на разделы игрового процесса, которые существенно отличаются от стандартного игрового процесса:

- 1) Использование игроком турели;
- 2) Секция с транспортом.

1.1.2.4 Паттерны альтернативных маршрутов

Эти шаблоны предоставляют игрокам возможность выбора того, как они хотят проходить уровень:

Разделённый уровень – Двухуровневый коридор представляет собой коридор с верхней и нижней секциями. Персонажи на верхнем уровне могут атаковать персонажей на нижнем уровне. Игроки могут выбрать верхний или нижний маршрут, или переключаться между ними

Обходной маршрут – Область с сильным сопротивлением может содержать альтернативный маршрут, который позволяет персонажам получить позиционное преимущество. Как правило, вражеские NPC располагаются в главном коридоре, в то время как у игрока есть возможность использовать обходной маршрут, чтобы оказаться позади вражеских NPC или рядом с ними. Обходной маршрут может представлять собой отдельный коридор или элементы прикрытия на большой арене.

Важно отметить, что каждый паттерн не обязан полностью определять геометрию уровня, наиболее оптимальным видится их комбинация на разных уровнях абстракции, макро и микро. Например, в уровне, созданном с использованием паттерна арена на макроуровне, могут присутствовать небольшие зоны с применением паттерна снайперская позиция – микроуровень.

Подобный подход описывается в работе [11], в ней авторы задействовали разбиения уровня по паттернам проектирования игровых уровней, представленных ранее, но подход не включил в себя их явное выделение, также ничего не было сказано про их уровни. При этом работа

предполагает использование результатов генерации для многопользовательских шутеров и, как следствие, не учитывает возможности использования различных видов NPC.

1.2 Методы генерации локаций

На данный момент существует огромное множество алгоритмов процедурной генерации контента, в том числе и локаций. Данные алгоритмы разделяют на три типа в зависимости от использованных в них методах: традиционные, алгоритмы, основанные на поиске и методы с использованием машинного обучения [26].

1.2.1 Традиционный подход

При традиционном подходе генерация процедурных уровней предполагает создание уровней с использованием predetermined алгоритмов или правил. Эти алгоритмы генерируют уровни, задавая планировку, размещение объектов, врагов и других элементов на основе predetermined параметров. Например, простой алгоритм может сгенерировать уровень, похожий на лабиринт, случайным образом размещая стены и соединяя их с помощью дорожек.

Типичным примером использования традиционного подхода в генерации трехмерных уровней можно назвать работу [21]. В ней авторы разработали эвристический алгоритм для генерации арен с использованием стыкующихся между собой мешей, на подобии паззла, и сводом правил для приоритетного направления размещения.

1.2.2 Подход на основе поиска

При процедурной генерации контента на основе поиска для поиска контента с желаемыми качествами используется эволюционный алгоритм или какой-либо другой алгоритм стохастического поиска/оптимизации. Основная метафора заключается в том, что дизайн – это процесс поиска: достаточно хорошее решение проблемы проектирования существует в некотором пространстве решений, и, если мы продолжим повторять и корректировать одно или несколько возможных решений, сохраняя те изменения, которые делают решение лучше, и отбрасывая те, которые вредны, мы получим в конце концов приходите к желаемому решению [13].

Основными компонентами основанного на поиске подхода к решению проблемы генерации контента являются:

Алгоритм поиска. Это “движок” метода, основанного на поиске. Часто относительно простые эволюционные алгоритмы работают достаточно хорошо, хотя иногда есть существенные преимущества в использовании более сложных алгоритмов, которые учитывают, например, ограничения или специализированы для конкретного представления контента.

Представление контента. Это представление артефактов, которые необходимо сгенерировать, например, игровой уровень, квест или 3D модель. Представление содержимого может быть любым - от массива вещественных чисел до графика и строки. Содержание представление определяет (и, следовательно, также ограничивает), какой контент может быть сгенерирован, и определяет, возможен ли эффективный поиск.

Одна или несколько функций оценки. Функция оценки - это функция от артефакта (отдельной части контента) к числу, указывающему на качество артефакта. Результат функции оценки может указывать, например, на играбельность уровня, сложность задания или эстетическую привлекательность модели. Разработка оценочной функции, которая надежно измеряет тот аспект качества игры, который она предназначена измерять, часто является одной из самых сложных задач при разработке метода PCG, основанного на поиске.

1.2.3 Алгоритмы, основанные на использовании МО

Машинное обучение сейчас находится на пике своего развития и популярности в связи с чем количество попыток применить его методы в различных областях постоянно растет. Процедурная генерация контента в играх, а в частности в игровых уровнях – не исключение. Баррига [26] собрал некоторые работы, связанные с генерацией игровых уровней с использованием МО и выделил методы и тип выборки данных (Рисунок 2).

Level Type	Game	Method	Training Data
Sequence	Super Mario Bros.	LSTM	Original game levels; simulated and human player trayectories.
	Super Mario Bros., Loderunner, Kid Icarus.	Multi-dimensional Markov Chains (MdMC), Hierarchical MdMC	Original game levels
Grid	Super Mario Bros., Loderunner, Kid Icarus.	Markov Random Field	Original game levels
	Super Mario Bros. StarCraft II resource locations	Autoencoders CNN	Original game levels Human authored maps
Graph	Super Mario Bros. Generic interactive fiction	K-means clustering OPTICS clustering	Gameplay videos Crowdsourced stories

Рисунок 2 – Некоторые методы МО для генерации уровней

Методы МО показывают хорошие результаты генерации, однако большой проблемой для их использования является сбор и подготовка большого количества консистентных данных, возможности собрать или получить которые зачастую может не быть.

1.3 Алгоритмы генерации

С момента создания игры Rouge (1998), являющейся первой в жанре Roguelike, развитие алгоритмов для процедурной генерации уровней и данжей не стояло на месте, в работах [12, 13] описываются различные алгоритмы, которые применяют при генерации 2-х и 3-х мерных пространств. Рассмотрим некоторые из них.

1.3.1 Генерация с использованием BSP

Алгоритм размещения комнат BSP (Binary space partition) в основном используется для генерации уровней данжей и представляет собой реализацию бинарного разбиения пространства [14]. Он разбивает граф на “листовые” объекты, каждый из которых хранится в древовидной структуре данных, начиная со всего графа как “корня” дерева. Листы разделяются до тех пор, пока у разделенного листа не появятся дочерние элементы определенного минимального размера, который напрямую связан с размерами графа и

предназначен для предотвращения перекрытия помещений. У листа Бошона нет дочерних элементов, так как он не расщепляется. Листьям, не имеющим детей, назначаются комнаты (максимальный размер которых меньше минимального размера листа), и эти комнаты добавляются в массив-список в порядке, в котором создаются объекты листа.

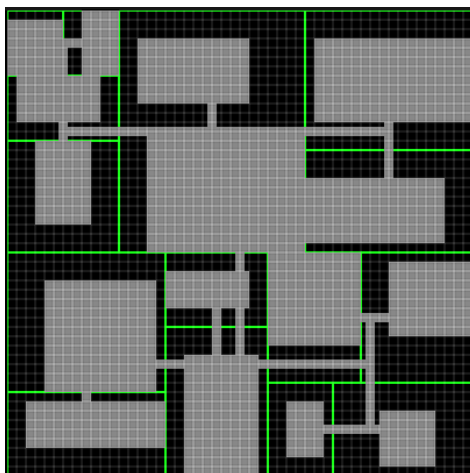


Рисунок 3 – Результат генерации уровня методом BSP

1.3.2 Клеточные автоматы

Клеточные автоматы – это системы, включающие локализованные, распределенные вычисления, часто по неограниченной двумерной квадратной сетке. Простые клеточные автоматы иногда использовались в исследованиях процедурной генерации контента: создание пещер, карт влияния и уровней – лабиринтов. Многие клеточные автоматы способны создавать сложную, возникающую динамику на длительных временных горизонтах, что делает их естественным выбором для генерации контента в пространственных областях [24].

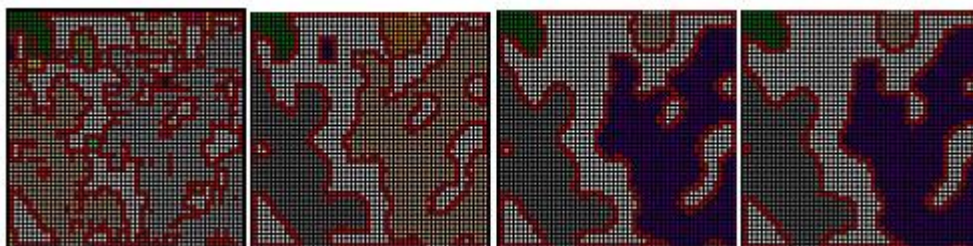


Рисунок 4 – Варианты генерации уровней с использованием клеточных автоматов

Клеточные автоматы имеют множество вариаций реализации, в свежих работах рассматриваются даже реализации с использованием нейронных сетей [25].

1.3.3 WFC

Смысл работы алгоритм WFC (Wave-function collapse) делит игровые уровни на сетку и определяет все ячейки в сетке, которые имеют все возможные значения. Во время операции он определяет значения в каждой ячейке одно за другим посредством "наблюдения", а затем распространяет эффекты на соседние ячейки и, наконец, достигает цели определения выполнимого решения, удовлетворяющего всем ограничениям. Иными словами, работа алгоритма заключается в подборе подходящих элементов из набора в каждую клетку сетки карты. Хотя этот алгоритм имеет вероятность сбоя, вызванного конфликтом, его эффективность выше, чем у традиционного метода решения ограничений на основе поиска.

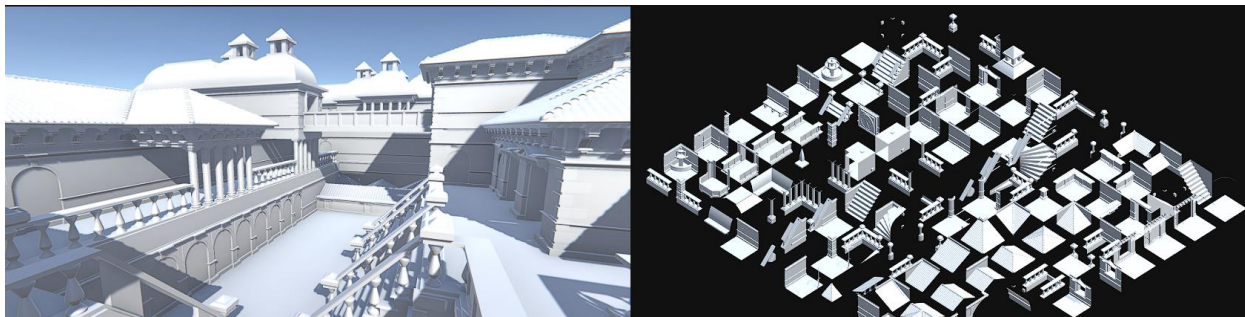


Рисунок 5 – Результат генерации уровня методом WFC

В последнее время данный алгоритм начал пользоваться большой популярностью. В работе [19] рассматривается вариант генерации городов. В [20] авторы рассматривают вариант генерации террейна с использованием WFC и иерархического сематического разбиения.

В целом, данный алгоритм, с одной стороны, является достаточно универсальным и удобным в использовании для генерации как террейна, так и построек в 2D и 3D, с другой, его эффективность и скорость работы сильно падает при большом количестве вариантов сегментов. Также стоит упомянуть, что создание сегментов сложных форм является проблематичной работой для дизайнера.

1.3.4 Диаграмма Вороного

Диаграмма Вороного представляет собой математическую конструкцию, которая делит плоскость на области в зависимости от расстояния до заданного набора точек на плоскости. Каждая область на диаграмме Вороного содержит все точки, которые находятся ближе к определенной входной точке, чем к любой другой точке в наборе. Эти области часто называют ячейками или полигонами Вороного. Диаграммы Вороного универсальны и находят применение в областях, где требуется пространственное разбиение, анализ близости или генерация шаблонов. Результирующие ячейки Вороного обеспечивают естественный способ деления пространства на основе конфигурации входной точки [15].

Данный метод также широко используется при генерации игровых уровней для создания поверхностей и их разбиения. В источнике [16] показан вариант использования граней, построенных с помощью диаграммы Вороного для создания дорог, а пространства внутри в качестве тайлов игрового уровня.

В работах [17 и 18] диаграммы Вороного в комбинации с другими методами используются для генерации правдоподобных пещер.

Стоит отметить, что данный метод является достаточно универсальным и удобным в использовании и, как следствие находит применение в разных областях.



Рисунок 6 – Вариант генерации уровня с использованием диаграмм Вороного

1.3.5 Алгоритмы с использованием шума Перлина и симплексного шума

Шум Перлина и его модификация – симплексный шум – это функции градиентного шума, обычно используемые при генерации процедурного контента, включая создание ландшафта, текстур и других естественно выглядящих узоров в компьютерной графике и разработке игр. Главной особенностью этих шумов является гладкий, непрерывный внешний вид, за счет чего они эффективны в вычислительном отношении. Данные шумы создаются путем интерполяции случайных значений (градиентов), определенных на сетке. Интерполяция создает плавный переход между значениями, придавая данным шумам их характерный внешний вид [22].

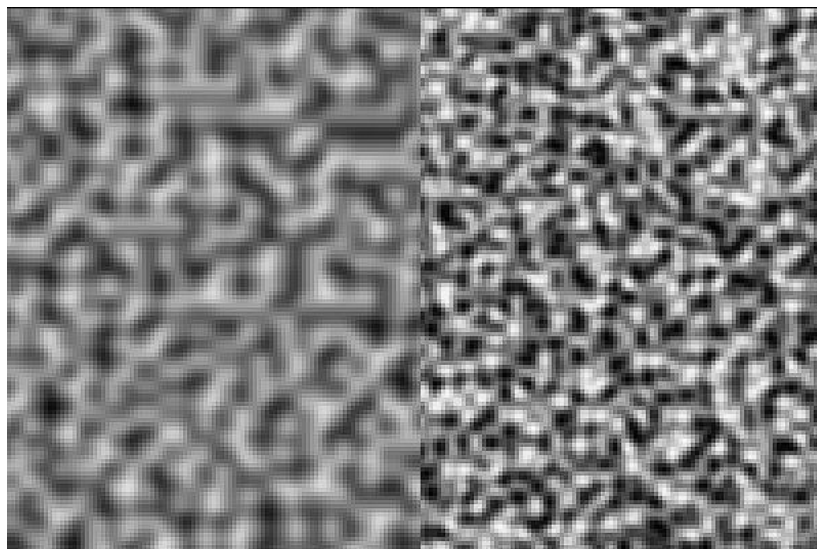


Рисунок 7 – Шум Перлина / симплексный шум

Ярчайшим примером использования этих шумов при генерации трехмерных уровней является игра Minecraft. В ней используется несколько уровней такого шума в комбинации с различными кривыми пороговых значений, за счет чего получается создавать практически бесконечный, уникальный и интересный для исследования мир [2].

1.3.6 Генетические алгоритмы

Генетические алгоритмы представляют собой оптимизационный метод, вдохновленный процессами естественного отбора и генетики в биологической

эволюции. Такие алгоритмы очень часто выступают в роли алгоритма поиска в подходе, описанном в пункте 1.2.2.

Существует несколько работ, в которых данный алгоритм рассматривается как ядро для генерации трехмерных уровней для игр в жанре FPS [27 – 29]. С его помощью получается достичь достаточно качественного результата, однако процесс генерации занимает значительное время и больше нацелен на получение нескольких конечных результатов, соответствующих определенным метрикам, нежели чем на постоянный повтор процесса генерации для получения различных результатов.

1.3.7 Итоги анализа различных способов генерации локаций

После рассмотрения некоторых алгоритмов процедурной генерации можно сделать вывод, что каждый из них имеет свои достоинства и недостатки, а также наиболее предпочтительные варианты использования. Это говорит о том, что при создании модели генерации лучше всего использовать комбинацию данных методов. Например, в работе [23] были использованы диаграммы Вороного для создания базовой разметки и комбинация WFC с графом для создания разметки.

1.4 Вывод

На данный момент основным способом генерации локаций в Roguelike F/TPS является простая случайная расстановка геометрии на уровне, будь то укрытия или заготовленные блоки карты. Такая система представлена в таких популярных играх жанра как Risk of Rain 2, Gunfire Reborn, Returnal и многих других. Такой подход является более чем приемлемым, однако после более детального изучения игры, пользователь замечает уже знакомые локации и закономерности, от чего интерес к игре может упасть.

В ходе исследования, проведенного в данной главе, удалось обнаружить работы в которых используются более комплексные и сложные алгоритмы, однако их анализ показал, что на данный момент не существует работ, направленных на процедурную генерацию игровых уровней в зависимости от набора противников различных типов. Из этого следует что создание

подобного алгоритма может повлиять на качество игрового опыта в играх жанра Roguelike.

Таким образом, исходя из анализа существующих алгоритмов и методов генерации локаций, а также изучения паттернов создания NPC и уровней, рассмотренных выше, кажется возможным объединение знаний в области геймдизайна и различных методов генерации для более осмысленной и логичной процедурной генерации игровых уровней для жанра FPS Roguelike.

2 Описание собственного алгоритма

В данной работе метод генерации должен представлять собой алгоритм создания игрового уровня для игры в жанре FPS в следующем виде:

- 1) Процедурная генерация основания уровня в виде арены или коридора на макроуровне.
- 2) Выделение зон: галерея, снайперская позиция, крепость, альтернативный путь на мезоуровне, модификация базовой геометрии уровня.
- 3) Расстановка статической и процедурно сгенерированной геометрии в зависимости от нахождения в определенной зоне на микроуровне.

На данном этапе работы все экспертные оценки и пороговые значения приблизительные и будут требовать корректировки после реализации алгоритма и игровых тестов.

Набор противников на уровне является ограниченно случайным и определяется пользователем алгоритма.

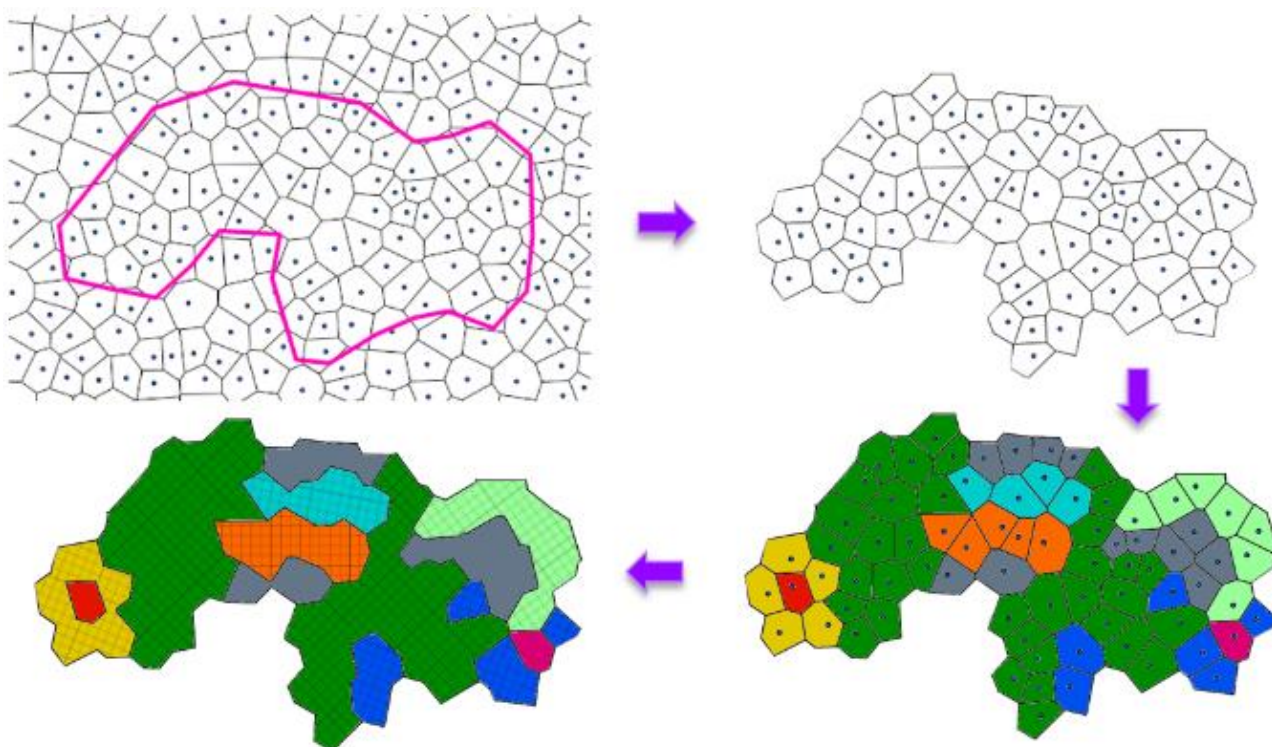


Рисунок 8 – Визуализация работы алгоритма

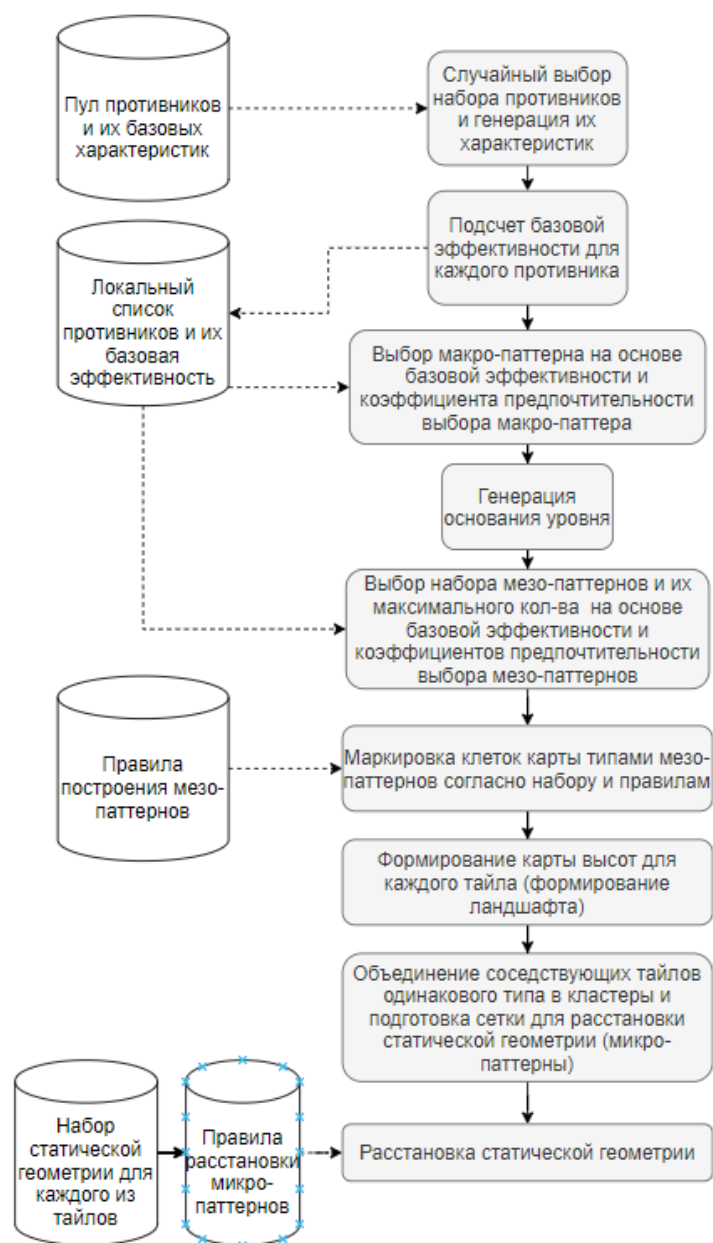


Рисунок 9 – Глобальный вид алгоритма

2.1 Параметры, вычисление и оценка эффективности NPC

Исходя из рассмотренных ранее паттернов создания NPC и дизайна игровых уровней, личного игрового опыта, а также характеристик NPC в проекте «Hell over me», в рамках которого планируется применить разрабатываемый алгоритм, можно вывести характеристики, по которым, впоследствии будет происходить генерация локаций.

2.1.1 Оценка базовой эффективности S_E

$$\begin{aligned} \text{Оценка базовой эффективности } S_E = \\ w_m * S_m + w_a * S_a + w_p * S_p + w_A * S_A, \end{aligned} \quad (1)$$

состоит из следующих значений:

$$\begin{aligned} S_m - \text{оценка передвижения} = \\ \text{скорость перемещения}(S_{m-s}) * \text{частота перемещения}(S_{m-f}), \end{aligned} \quad (2)$$

где $S_{m-f} = [0; 1]$, 0 – статичный, 1 – постоянно в движении.

Выбор такой формулы связан с тем, что в шутерах попасть во врага сложнее, если он быстро и часто перемещается.

$$\begin{aligned} S_a - \text{оценка атакующей способности (при наличии)} = \\ \text{частота атак}(S_{a-f}) * \text{урон от атак}(S_{a-d}) * \text{точность атак}(S_{a-a}), \end{aligned} \quad (3)$$

где $S_{a-a} = [0; 1]$, 0 – каждая атака изначально направлена мимо игрока, 1 – каждая атака направлена в игрока.

При помощи подобных формул балансируют параметры оружия в онлайн играх, значение $S_{a-d} * S_{a-f}$ дает знакомый параметр DPS (урон в секунду).

S_A – оценка специальной способности (при наличии).

Данный параметр не представляется возможным оценить с помощью какой-либо универсальной математической формулы, поэтому данное значение должно быть вычислено пользователем алгоритма.

$$\begin{aligned} S_p - \text{оценка защищенности} = \\ \frac{\text{количество здоровья}(S_{p-h}) + \text{количество брони}(S_{p-a})}{1 + \text{размер NPC}(S_{p-s})}, \end{aligned} \quad (4)$$

где $(S_{p-s}) = [0; 1]$, 0 – самый мелкий противник в игре, 1 – самый большой противник в игре.

С увеличением размер противника увеличивается шанс попадания в него, следовательно, уменьшается защищенность.

Также введем дополнительные коэффициенты важности параметра для каждого NPC: w_m, w_a, w_p, w_A

Таблица 2 – Коэффициенты важности параметров

	w_m	w_a	w_p	w_A
Пехотинец	1	1	1	0
Элита	1	1	1	0
Гренадер	1	1	1	0
Снайпер	1,3	1,5	1	0
Самоубийца	0,8	0,2	1,3	0
Рой	0,6	0,4	1,3	0
Берсерк	1,5	1,5	1,5	0
Жертвенный	1,1	0,9	0,8	1,3
Призыватель	0,5	0,5	1,3	2
Турель	0,2	1,7	1,5	0
Защищенный	1,1	0,8	2	0
Поддержка	1,3	0	1,5	1,5

2.1.2 Коэффициент предпочтительности выбора макropаттерна k_p

Данный параметр отвечает за то, в какая макрозона, предположительно, более предпочтительна каждому из видов NPC. Данный параметр является коэффициентом и не может превышать 1 для каждого оцениваемого типа NPC.

Локации в шутерах в основном имеют два вида: Арена, где враги нападают со всех сторон, но у игрока есть достаточное пространство для маневра и Коридор, где враги атакуют с одного направления, но при этом у игрока меньше места для тактического перемещения.

Таблица 3 – Оценка предпочтительности выбора макropаттерна для NPC

	Арена	Коридор
Пехотинец	0,5	0,5
Элита	0,5	0,5
Гренадер	0,35	0,65
Снайпер	0,3	0,7
Самоубийца	0,8	0,2
Рой	0,6	0,4

Берсерк	0,5	0,5
Жертвенный	0,75	0,25
Призыватель	0,5	0,5
Турель	0,3	0,7
Защищенный	0,2	0,8
Поддержка	0,5	0,5

2.1.3 Коэффициент влияния на эффективность NPC в зависимости от наличия мезопаттерна k_{pe}

В этом подразделе будет приведена таблица коэффициентов, отражающая влияние наличия различных паттернов игрового уровня на эффективность NPC с разными типами. Оценка может варьироваться от -1 (крайне негативно) до 1 (крайне положительно).

Таблица 4 – Оценка влияния мезопаттерна на эффективность NPC

	Узкий проход	Снайперская позиция	Галерея	Альтернативный путь
Пехотинец	0	0	0	0
Элита	0	0	0	0
Гренадер	1	0,5	0,5	0
Снайпер	0,5	1	1	0
Самоубийца	-1	0	0	0
Рой	-0,5	0	0	0
Берсерк	-0,5	0	0	0
Жертвенный	0	0	0	0
Призыватель	0	0	0	0
Турель	1	0,5	0,5	0
Защищенный	1	0	0	-1
Поддержка	0	0	0	0

2.1.4 Коэффициент влияния на эффективность NPC в зависимости от типа движения и наличия определенного паттерна k_{mt-pe}

В данном подразделе будет приведена таблица коэффициентов, отражающая влияние наличия различных паттернов игрового уровня на эффективность NPC с разными типами движения. Оценка может варьироваться от -1 (крайне негативно) до 1 (крайне положительно).

Таблица 5 – Оценка влияния мезопаттерна на эффективность NPC с выбранным типом движения

	Узкий проход	Снайперская позиция	Галерея	Альтернативный путь
Обходящий с флангов	-0,5	0	0	1
Пассивный	0	1	1	-1
Тип с медленным продвижением	0,5	0	0	-1
Осторожный	0	1	1	0,5

2.2 Формирование входных данных

Для начала работы генератора необходимо задать изначальные входные параметры, основывающиеся на количестве и характеристиках NPC.

Ширина и высота путей – для исключения возможности эксплуатации игроком мест, где различные NPC не могут достичь игрока необходимо определить высоту самого высокого NPC, а также ширину самого крупного. Таким образом, при создании окружения и его оценке

2.2.1 Выбор макропаттерна

Выбор макропаттерна должен зависеть от параметра базовой эффективности S_E каждого NPC и коэффициента предпочтительности выбора k_p , каждого i – го макропаттерна. Исходя из этого составим формулу:

S_{pi} – Оценка предпочтительности выбора для каждого i – го паттерна

$$S_{pi} = \sum_{j=1}^n S_{Ej} * k_{pi}, \quad (5)$$

где n-количество NPC

Тогда конечный макропаттерн определяется соответствующим индексом для значения $\text{Max}(S_{p0}, \dots, S_{pn})$

2.2.2 Выбор мезопаттернов

Мезопаттерн должен появляться на уровне в таком количестве, которое имеет смысл в контексте набора противников:

Оценка максимального количества N_{max} тайлов для конкретного мезопаттерна будет оцениваться по формуле:

$$N_{max} = N_T * \frac{\sum_{i=1}^n [S_{Ei} * (k_{pei} + k_{mt-pei})]}{\sum_{i=1}^n S_{Ei}} \quad (6)$$

2.3 Генерация уровня

Перед описанием самого алгоритма введем цветовые обозначения зон с использованием конкретных паттернов для более удобного графического обозначения:



Рисунок 10 – Цветовые обозначения

2.3.1 Генерация основания

После анализа существующих алгоритмов в качестве основного метода генерации основания был выбран метод диаграмм Вороного. Данный выбор связан с тем, что он позволяет получить более натуральную форму карты в отличие от любых видов разбиения по клеткам. Также он не требует большого времени на вычисление и позволяет легко модифицировать полученные в результате генерации данные.

Для определения начальной формы уровня возьмем случайно сгенерированную диаграмму Вороного и наложим на нее границы геометрии выбранного макропаттерна, грани и вершины, принадлежащие точкам внутри границы, будут использованы как изначальная форма.

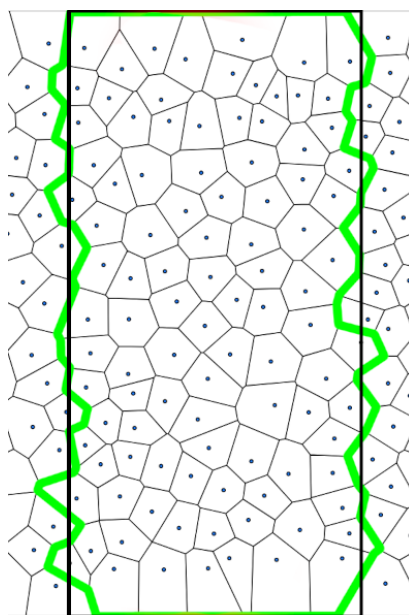


Рисунок 11 – Пример начальной генерации для паттерна коридор

2.3.2 Добавление мезопаттернов

Следующим этапом станет разбиение карты на логические зоны, внедрение в них мезопаттернов и правил их размещения.

Диаграмма Вороного легко позволяет представить локацию в виде связного графа за счет наличия корневых точек каждого получившегося тайла. Связи каждого конкретного тайла будут определяться только для соседних тайлов, имеющих смежные грани с текущим.

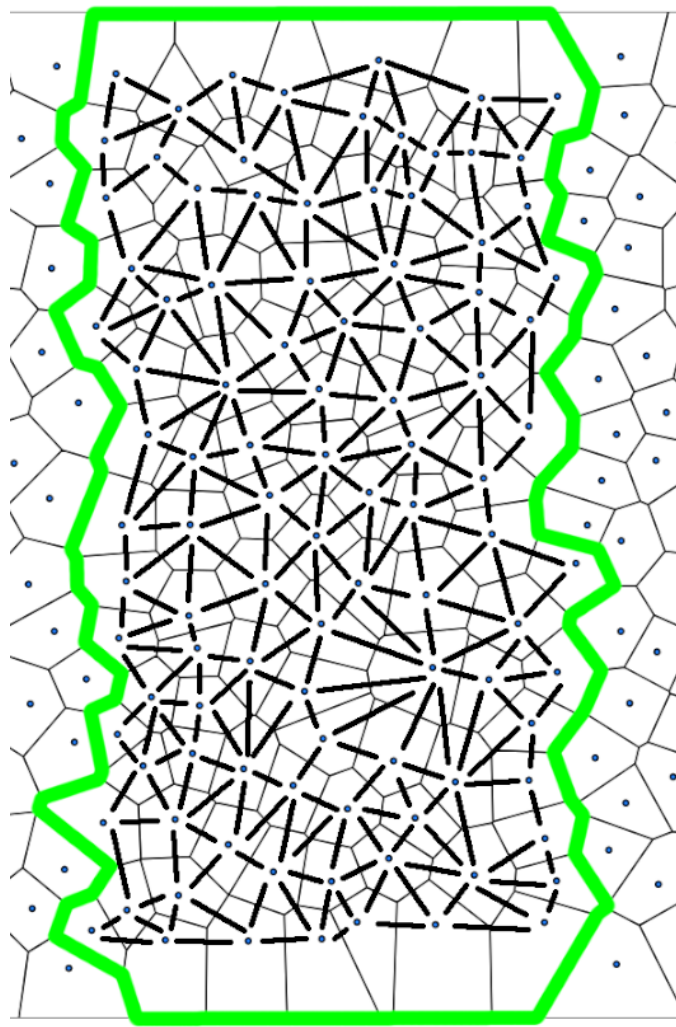


Рисунок 12 – Пример создания связного графа 1

После этого мы можем использовать модифицированный алгоритм WFC для графов, представленный в источнике [23]. Модификация будет заключаться в том, что выбор типа тайла для каждой связи из набора доступных будет иметь свой динамический вес и вычисляться только при работе с прилегающей к ней вершиной. Таким образом, по задумке, можно добавить больше контроля при генерации и получать более осмысленное расположение зон на карте, т.е., при свертке, очередной тайл может либо заблокировать появление какого-либо типа тайла рядом или изменить шанс его появления.

Исходя из рассмотренных в подразделе 1.1 особенностей паттернов проектирования игровых уровней и NPC и соотнеся их с особенностями типа генерируемого уровня, мы можем сделать следующие предположения:

- 1) Тайлы паттерна альтернативный путь должны находиться на значительном удалении от основного пути (начло – конец уровня), а также в достаточной степени параллельными ему.
- 2) Тайлы паттерна снайперская позиция должны быть значительно удалены от старта игрового уровня.
- 3) Паттерн узкий проход должен разграничивать уровень на 2 части значимого размера, вследствие чего он должен располагаться ближе к середине основного пути и не быть сильно удаленным от него.
- 4) Паттерн галерея может иметь хорошую синергию с паттернами узкий проход и альтернативный путь и иметь больший шанс располагаться неподалеку.
- 5) Так же стоит отметить, что паттерн крепость в его ослабленном виде будет необходим всегда. Такое решение связано с тем, чтобы при начале битвы дать игроку пространство и время для оценки обстановки. В этой зоне невозможно появление противников, однако они могут достичь его.

Для того, чтобы соблюсти этот подход введем глобальные зоны, регулирующие шанс появления необходимых паттернов (w_{gl}):

- 1) Основной путь (красный);
- 2) Край уровня (синий);
- 3) Середина уровня (маджента).

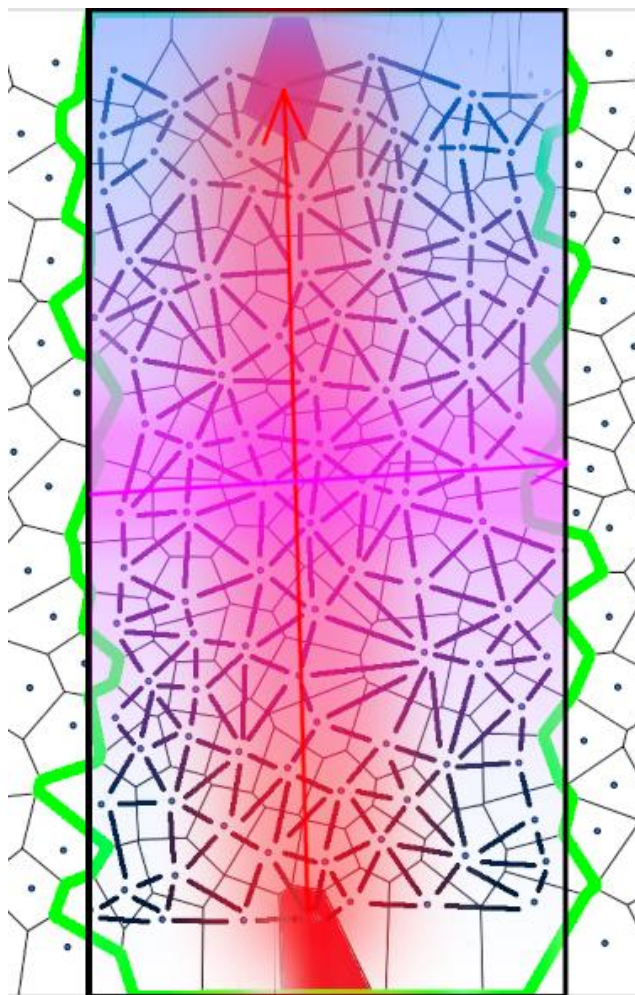


Рисунок 13 – Пример создания глобальных зон, повышающих шанс появления паттернов

Помимо простого добавления глобальных зон введем фактор синергии тайлов, основанный на их потенциальном взаимном расположении относительно глобальных зон (w_{gl}). Он будет определяться следующим образом: зная векторы направленности глобальных зон, мы можем ввести повышающий вероятность справна коэффициент для очередного тайла конкретного типа в необходимом направлении, относительно глобальной зоны, в заданном угловом диапазоне. Данный фактор задается отдельно для каждой связи наравне с базовым коэффициентом.

Приведем графический пример (Рисунок 12). При выборе очередного тайла для типа узкий проход (CP) фактор (w_{ts}) будет выше для тайла в направлении вектора основного пути.

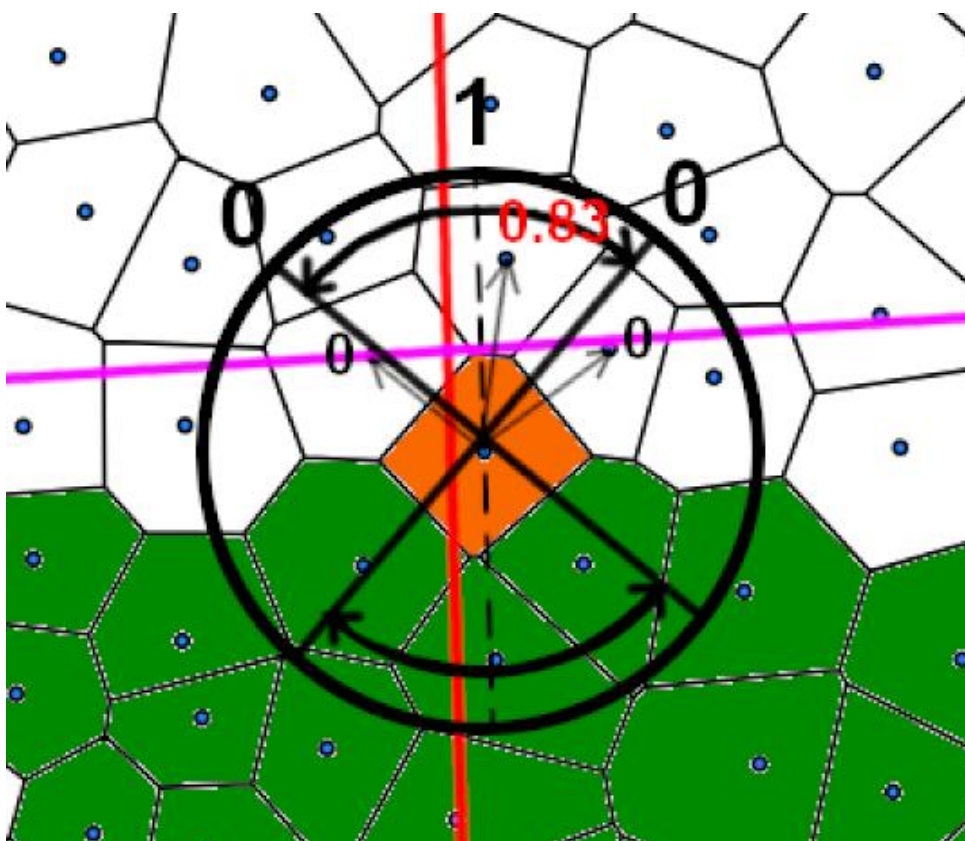


Рисунок 14 – Графический пример вычисления фактора синергии тайлов

Теперь создадим матрицу базовых (w_b) весов для каждого тайла:

Таблица 6 – Матрица весов

	S	E	F	BI	SP	B	CP	AP	G
S	x	x	1	x	x	x	x	x	x
E	x	x	x	x	0	0	0	0	0
F	0	x	0	0	0	0	0	0	0
BI	x	x	0	0	0	0	0	0	0
SP	x	x	0	0	1	1	1	1	0
B	x	x	0	0	1	1	1	1	1
CP	x	x	0	1	0	1	1	x	1
AP	x	x	x	1	0	1	x	1	0
G	x	x	x	1	1	1	1	0	1

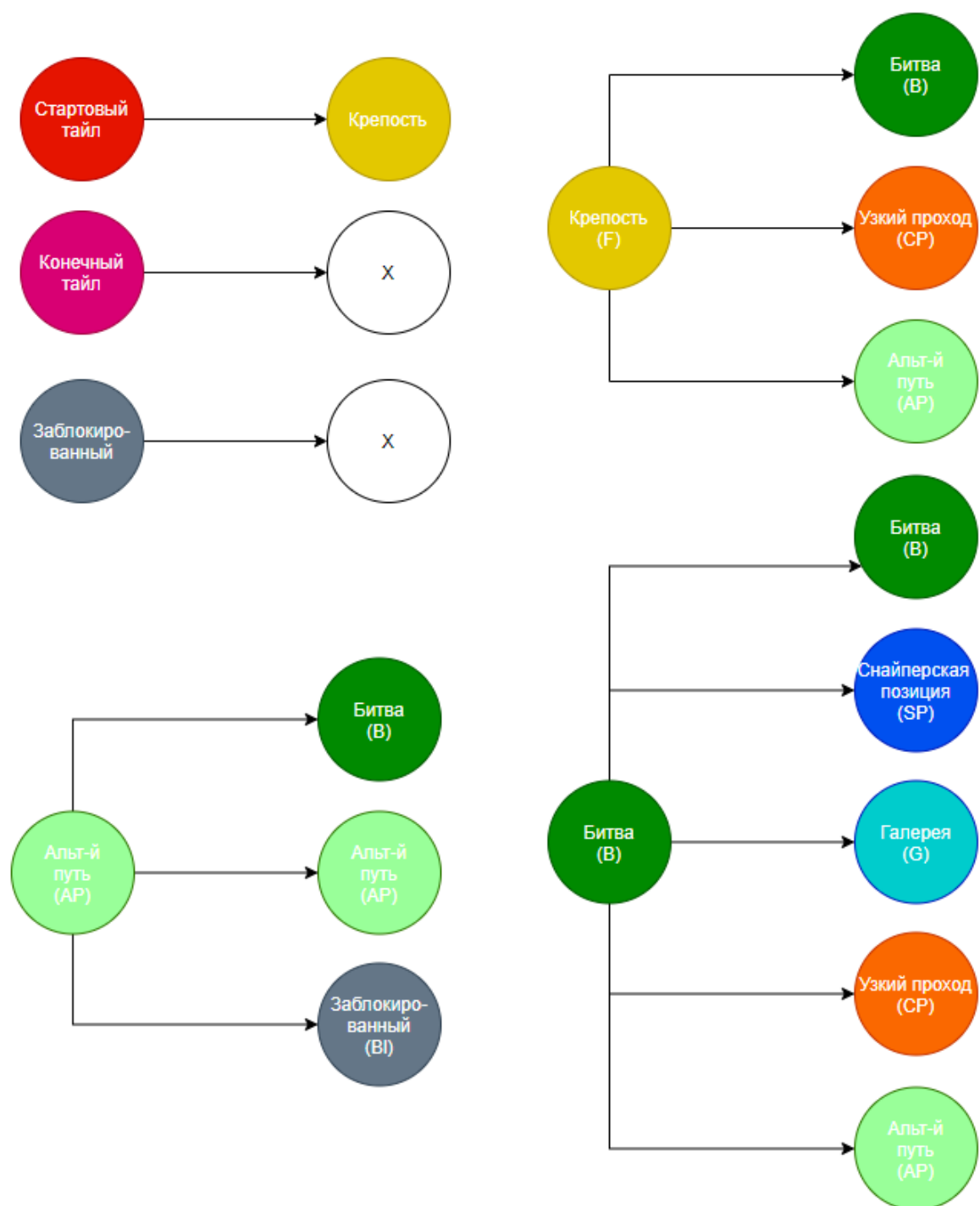


Рисунок 15 – Возможные связи тайлов 1

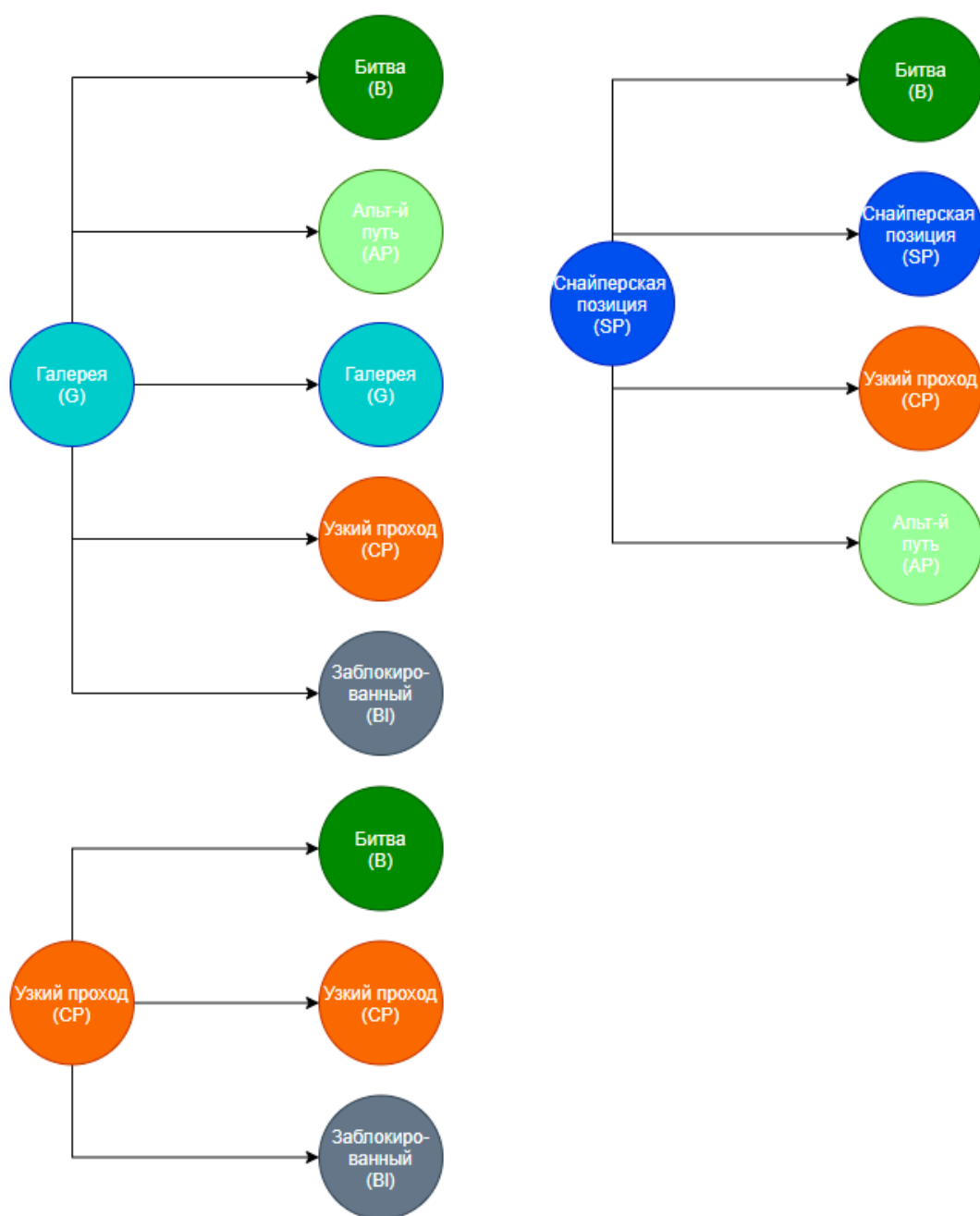


Рисунок 16 – Возможные связи тайлов 2

Итого, при выборе очередного тайла вес каждого из возможных вариантов будет считаться по формуле:

$$W = w_b * w_{gl} * w_d * w_{ts} \quad (7)$$

$$w_d = \frac{N_{max} - N_{spawned}}{N_{max}}, \quad (8)$$

где w_d всегда ≥ 0

При свертке будет выбран тип тайла с наибольшим показателем W .

2.3.3 Добавление микропаттернов

Предположим, что у нас имеется кластер тайлов одного типа. Для расстановки статической геометрии воспользуемся взвешенным алгоритмом WFC. Взвешенность поможет регулировать плотность генерации препятствий. Для каждого мезопаттерна будет необходим свой функциональный набор геометрии.

- 1) Для того, чтобы обеспечить потенциальную проходимость путей для всех NPC зададим размер клетки сетки равный $2R_{\max NPC \text{ size}} * 1,1$.
- 2) Чтобы обеспечить разнообразие при генерации развернем сетку вдоль наиболее длинной грани кластера.
- 3) Оставим свободными только клетки, не пересекающиеся границами кластера.
- 4) Работа алгоритма WFC.

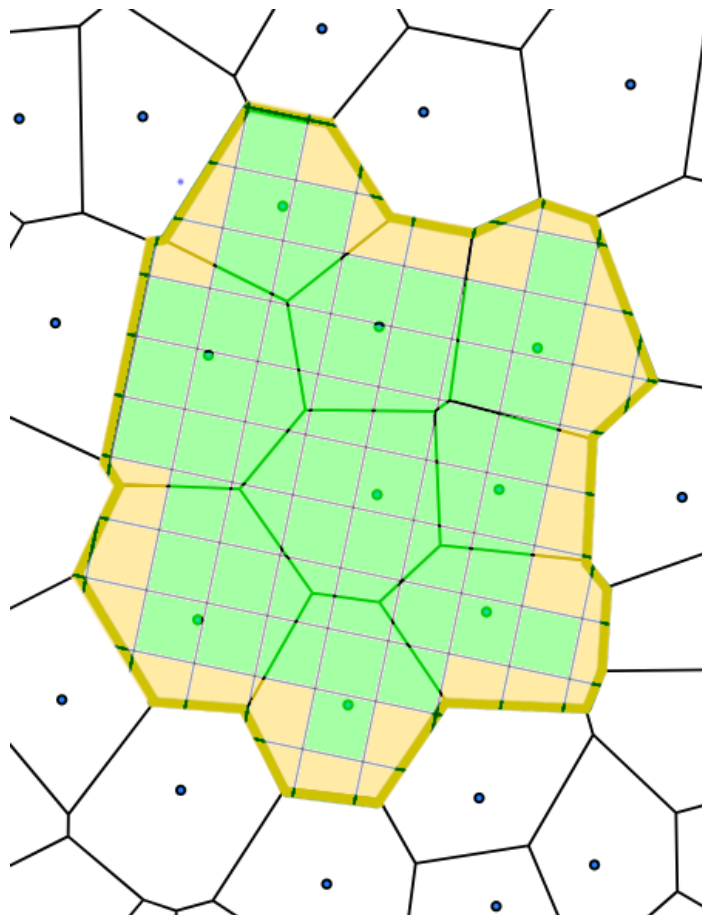


Рисунок 17 – Пример размещения сетки в кластере.

3 Практическая реализация

Разрабатываемый метод, не зависит от конкретной среды выполнения и может быть имплементирован с использованием любого языка программирования, который поддерживается выбранным игровым движком. В ходе работы было принято решение практически реализовать данный метод в виде части отдельного модуля для процедурной генерации карты с использованием движка Unreal Engine 5 на языке C++.

3.1 Формирование и оценка набора противников

В связи с тем, что практическая реализация алгоритма происходит в рамках работы над проектом «Hell over me», набор противников будет состоять из имплементированных на данный момент видов:

- 1) Пехотинец;
- 2) Самоубийца;
- 3) Берсерк;
- 4) Снайпер;
- 5) Ловкач.

Ловкач является нестандартным паттерном поведения противника и представляет собой обычного пехотинца ближнего боя с типа движения – *Обходящий с флангов*.

3.2 Генерация макропаттерна

Для использования диаграммы Вороного был применен внутренней класс Unreal Engine – Voronoi, с его помощью можно сгенерировать диаграмму Вороного на основе заданных в пространстве точек.

Для внесения разнообразия в изначальную форму уровня алгоритм позволяет внести неограниченный список пресетов форм для каждого из макропаттернов. При этом один и тот же пресет будет давать похожий, но не идентичный результат при очередной генерации. Форма, главный путь уровня и другие пространственные параметры могут задаваться кривыми в специальном классе.

▼ Arenas Pool	2 Map elements	⊕	🗑
▼ Corridor	1 members	▼	
▼ Array	2 Array elements	⊕	🗑
Index [0]	BP_NewArena_CorridorSmall	↶	🔍 ⊕ ✕ ▼
Index [1]	BP_NewArenaisland_SplieSmall	↶	🔍 ⊕ ✕ ▼
▼ Arena	1 members	▼	
▼ Array	2 Array elements	⊕	🗑
Index [0]	BP_NewArena_CircleSmall	↶	🔍 ⊕ ✕ ▼
Index [1]	BP_NewArena_TorusSmall	↶	🔍 ⊕ ✕ ▼

Рисунок 18 – Пример набора пресетов макрпаттернов.



Рисунок 19 – Пример генерации макрпаттернов для разных пресетов 1.



Рисунок 20 – Пример генерации макропаттернов для разных пресетов 2.

3.3 Генерация мезопаттернов

Для генерации мезопаттернов был реализован взвешенный алгоритм WFC, регулируемый, как базовыми весами для каждого из соседей, так и специальными правилами, учитывающими взаимное расположение тайлов относительно друг друга, а также общее положение каждого тайла в пространстве макропаттерна. Помимо этого, в зависимости от набора противников, согласно формуле (6), определяется максимальное количество тайлов для каждого из мезопаттернов. По окончании работы алгоритма соседние тайлы с одинаковым типом объединяются в кластеры.

Базовый набор правил генерации для тайлов представлен в следующем виде:

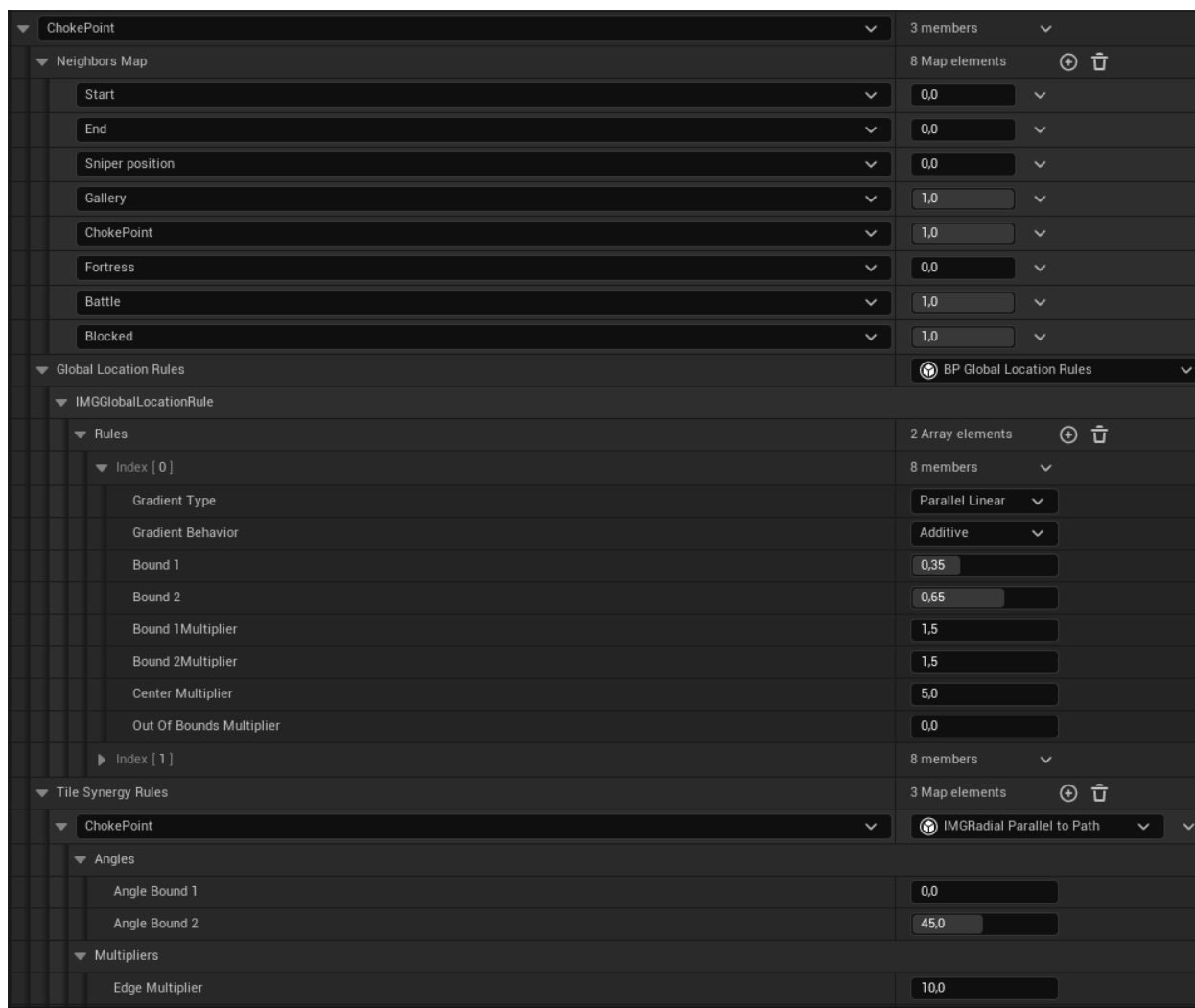


Рисунок 21 – Пример набора базовых правил тайлов для генерации мезопаттернов.

На этапе генерации мезопаттернов мы можем получить следующие результаты для следующих случайных наборов противников:

Набор 1:

Таблица 7 – Набор противников 1

Вид противника	Число противников
Самоубийца	30

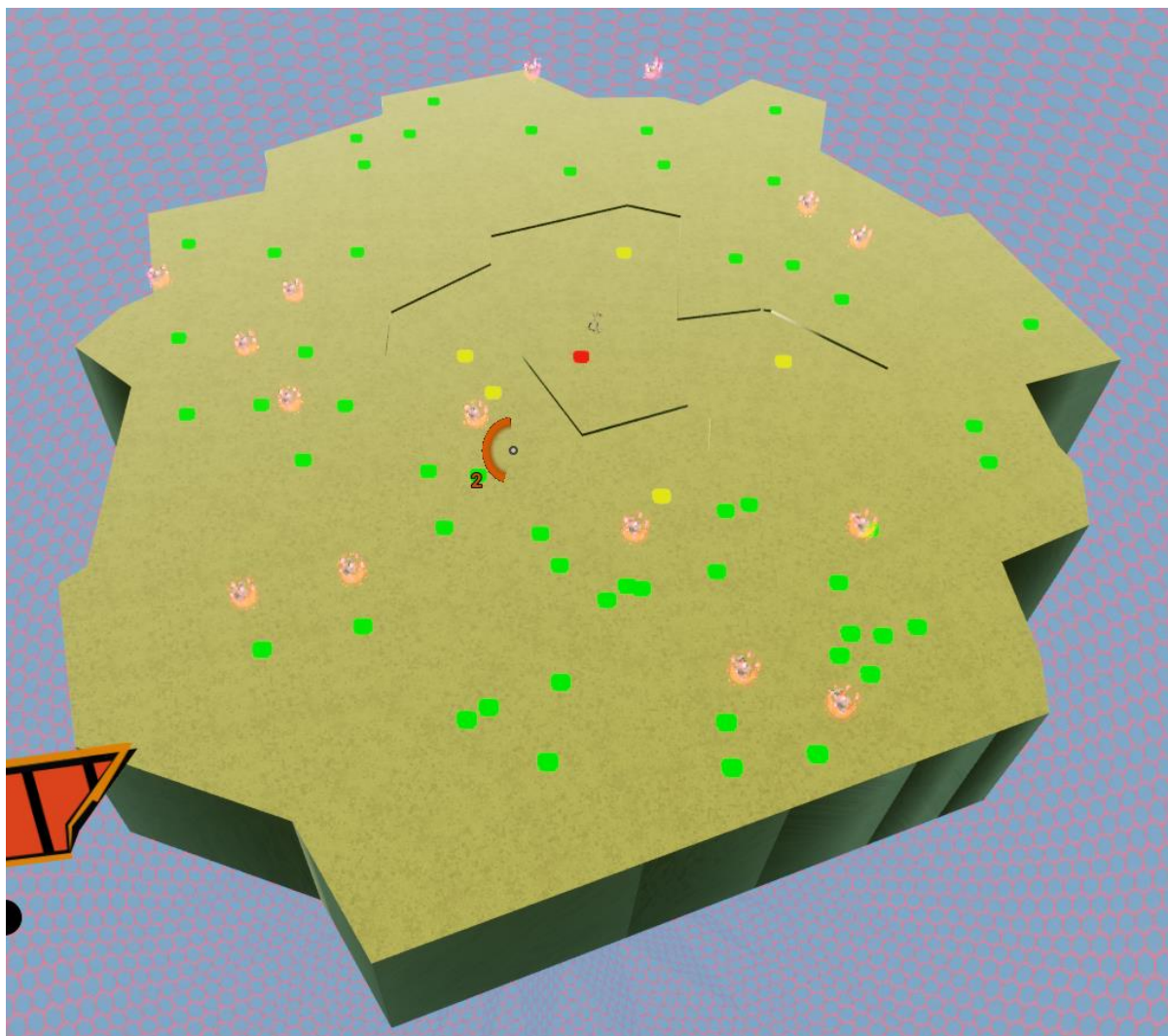


Рисунок 22 – Пример работы алгоритма для случайного набора 1.

Набор 2:

Таблица 8 – Набор противников 2

Вид противника	Число противников
Пехотинец	7
Снайпер	2

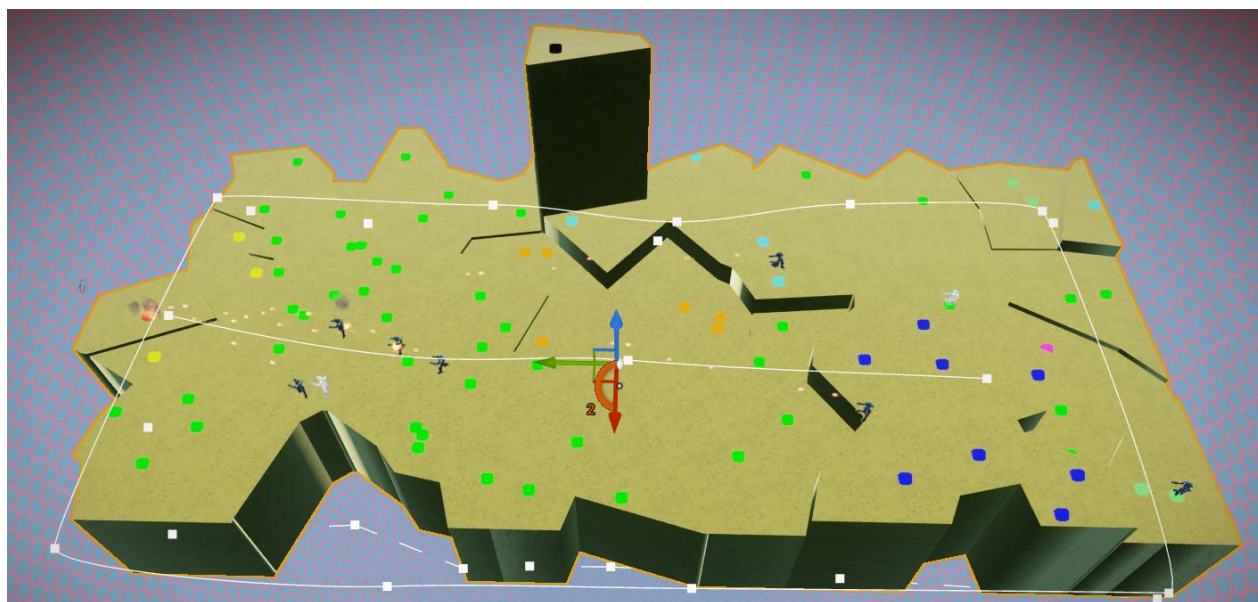


Рисунок 23 – Пример работы алгоритма для случайного набора 2.

Набор 3:

Таблица 9 – Набор противников 3

Вид противника	Число противников
Пехотинец	2
Самоубийца	4
Ловкач	2
Снайпер	4
Берсерк	1



Рисунок 24 – Пример работы алгоритма для случайного набора 3.

3.4 Генерация микропаттернов

Для генерации микропаттернов был также применен взвешенный, модифицированный алгоритм WFC. Для тестирования алгоритма были созданы несколько черновых тайлсетов для каждого типа кластеров, состоящих из следующих статических мешей:

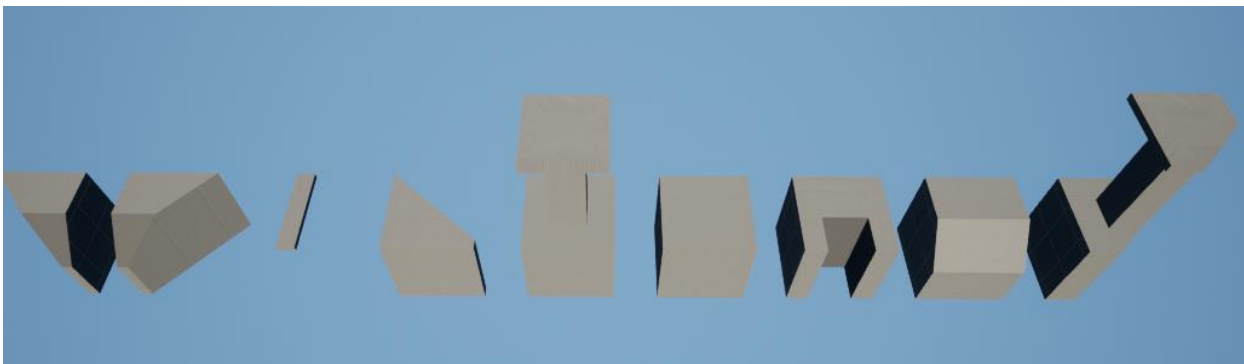


Рисунок 25 – Тестовый набор статических мешей

Помимо этого, были сформированы дата ассеты для организации правил работы WFC:

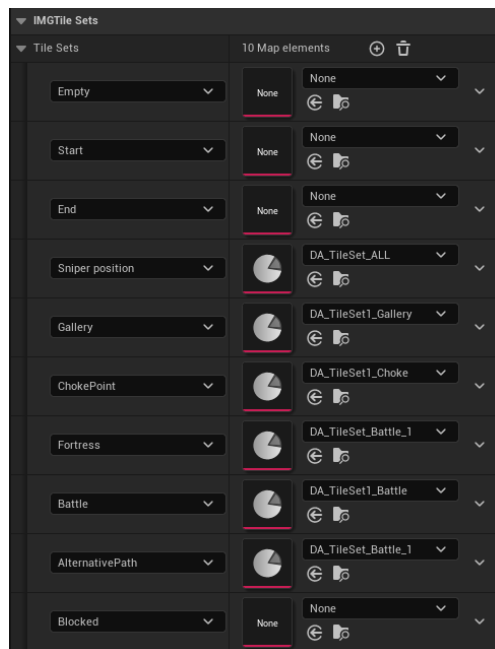


Рисунок 26 – Дата ассет: тип мезопаттерна – тайлсет.

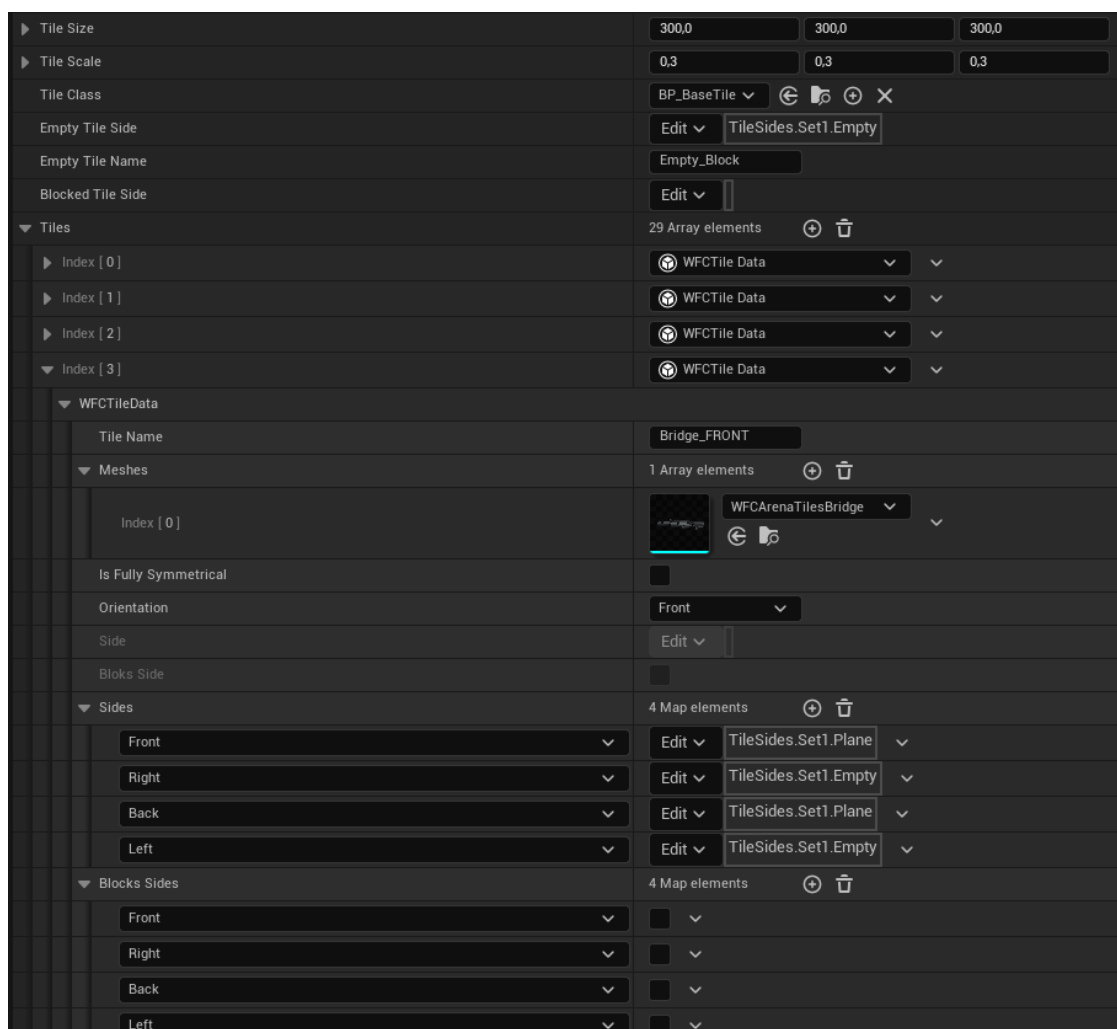


Рисунок 27 – Пример тайлсета

Каждый кластер имеет свой тайлсет и сетку, размер которой определяется размерами самого кластера.

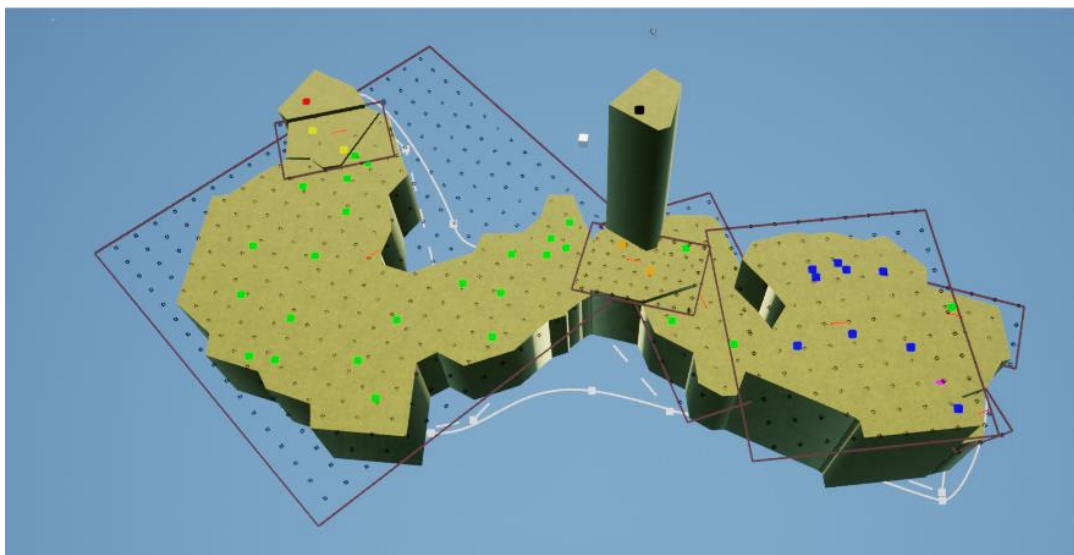


Рисунок 28 – Пример установки сетки алгоритма WFC для каждого из кластеров.

Клетки, не входящие в границы кластера, отсекаются, после чего начинается работа алгоритма – свертка:

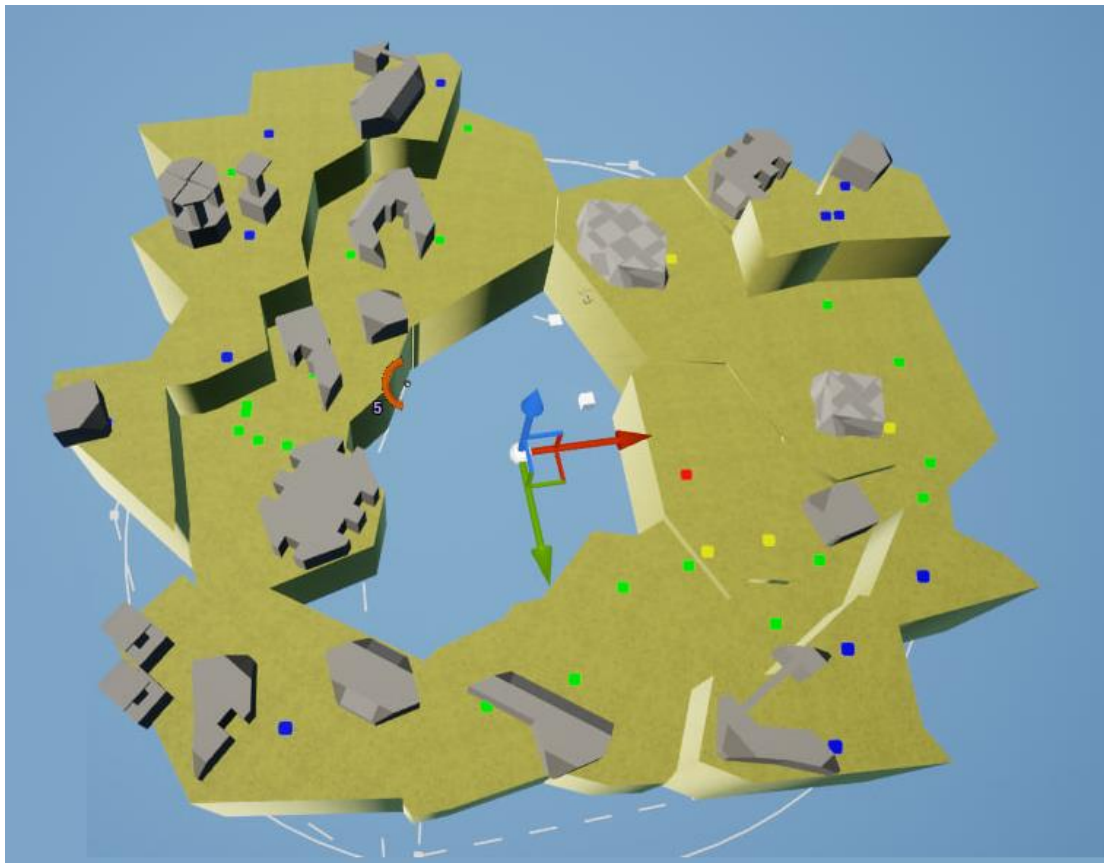


Рисунок 29 – Пример работы алгоритма расстановки микропаттернов 1

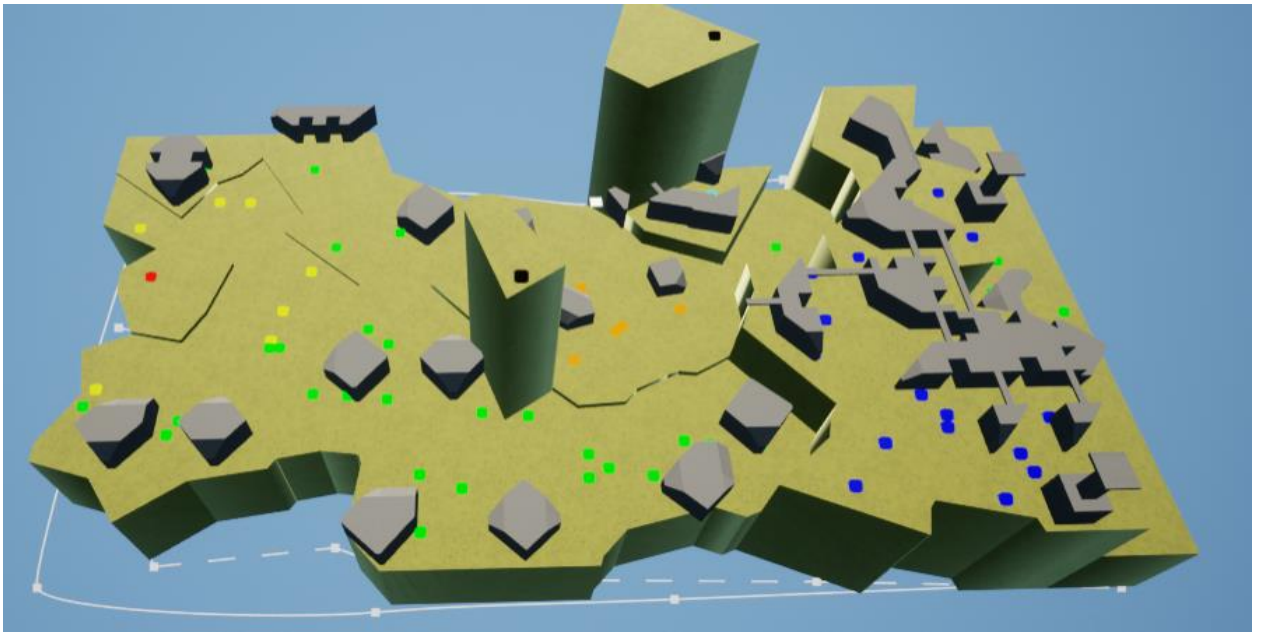


Рисунок 30 – Пример работы алгоритма расстановки микропаттернов 2

4 ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА ГИПОТЕЗЫ

Для проверки гипотезы о том, что возможно создать алгоритм генерации игровых уровней, способный повысить общую эффективность противников на основе учета их особенностей, необходимо экспериментальным путем оценить его текущую реализацию.

На данный момент не представляется реальным автоматическое тестирование, так как реализация компьютерной симуляции прохождения игроком уровня, приближенной к реальному опыту, является очень нетривиальной и сложной задачей. Тем не менее, вполне реализуемой задачей кажется сбор статистики во время игры и проведение А-Б тестов с различными параметрами на реальных игроках.

Первый – прохождение игроком сгенерированных аренных игровых уровней без учета специфики противников.

Второй – прохождение игроком сгенерированных аренных игровых уровней с учетом специфики противников.

4.1 Формирование параметров оценки

Общая цель работы – разработать алгоритм генерации аренных игровых уровней для врагов разного типа, это значит, что аренны, сгенерированные специально для заданного набора противников должны повышать КПД данных противников.

Наиболее важными параметрами для оценки КПД каждого противника являются:

- 1) Время жизни противника;
- 2) Время визуального контакта с игроком;
- 3) Урон по игроку.

Эти данные необходимо собирать для каждого типа противника на каждой очередной сгенерированной локации. Помимо этого, также необходимо знать количество противников каждого типа. Еще одним немаловажным фактором является смерть игрока при прохождении уровня, которая может говорить о высокой эффективности противников.

Таким образом структура данных для хранения информации об эффективности противников на каждой из арен будет иметь следующий вид:

The screenshot shows a hierarchical data structure in GHStats DB. It is organized into three main sections, each representing an arena type. Each section contains a 'Data Array Depends on Enemies' and a 'Data Array Not Depends on Enemies'.

Section	Sub-section	Item	Value
BP_Striker	Data Array Depends on Enemies	Index [0]	2 members
		Index [1]	2 members
	Data Map	Player Died	<input checked="" type="checkbox"/>
		BP_Striker	5 members
	Data Map	Player Died	<input type="checkbox"/>
		Enemies Count	4
		Total Damage	0,0
		Alive Time	6,291186
		Visual Contact Time	1,935702
	Data Array Not Depends on Enemies	BP_Bomber	5 members
		Player Died	<input type="checkbox"/>
		Enemies Count	4
		Total Damage	10,0
		Visual Contact Time	5,588062
BP_Sniper	Data Array Depends on Enemies	Index [2]	2 members
		Index [3]	2 members
	Data Array Not Depends on Enemies	Index [4]	2 members
		Index [5]	2 members
	Data Array Depends on Enemies	Index [0]	2 members
		Index [1]	2 members
	Data Map	Player Died	<input type="checkbox"/>
		BP_Sniper	5 members
	Data Map	Index [2]	2 members
		Index [3]	2 members

Рисунок 31 – Данные об эффективности противников на каждой арене

Данные фиксировались только при полном прохождении арены или смерти игрока.

4.2 Проведение эксперимента

Для проведения эксперимента был создан специальный игровой уровень и система логирования в рамках проекта Hell Over Me. Для сбора данных игрокам предлагалось поочередно проходить случайно сгенерированные арены, при этом никто не знал, какой тип генерации использовался при создании каждой арены.

По окончании прохождения произвольной серии уровней на компьютере игрока формировался пакет с усредненными данными в следующем виде:

Mean Not Depends on Enemy	
Arenas Num	10
Player Deaths	2
Data Map	
5 Map elements	
BP_Striker	4 members
Enemies Count	56
Total Damage	1.785714
Alive Time	8.882244
Visual Contact Time	0.313673
BP_Berserk	4 members
Enemies Count	31
Total Damage	6.774193
Alive Time	20.702751
Visual Contact Time	3.388477
BP_Bomber	4 members
Enemies Count	98
Total Damage	0.612245
Alive Time	15.753111
Visual Contact Time	0.95139
BP_Sniper	4 members
Enemies Count	41
Total Damage	0.292683
Alive Time	11.017117
Visual Contact Time	0.300773
BP_Trickster	4 members
Enemies Count	7
Total Damage	3.571429
Alive Time	17.756092
Visual Contact Time	1.212328

Рисунок 32 – Пакет усредненных данных

Собранные пакеты заносились в общую таблицу исходя из которой были получены результаты эксперимента.

4.3 Результаты

В проведении эксперимента участвовало 10 человек. Всего за время проведения эксперимента было сгенерировано и пройдено 138 аренных игровых уровней.

Для 69 аренных игровых уровней, сгенерированных без учета набора противников, процент смертей игрока составил 16%, а средний урон по игроку составил 12,3 единицы здоровья. Для каждого из типа противников были получены следующие данные:

Таблица 10 – Данные для генерации без учета характеристик противников

	Средний урон (ед. здоровья))	Среднее время жизни (сек.)	Среднее время визуального контакта (сек.)	Всего противников
Самоубийца	0,908731385	27,05467642	15,60929765	387
Пехотинец	0,818434409	23,44772989	11,06673422	198
Ловкач	1,379506481	28,36506651	17,5947821	79
Берсерк	3,388983	50,73673308	36,63873797	59
Снайпер	0,294231519	24,26969843	11,954847	108

Для 69 аренных игровых уровней, сгенерированных с учетом набора противников, процент смертей игрока составил 21%, а средний урон по игроку составил 21,7 единицы здоровья. Для каждого из типа противников были получены следующие данные:

Таблица 11 –Данные для генерации с учетом характеристик противников

	Средний урон (ед. здоровья)	Среднее время жизни (сек.)	Среднее время визуального контакта (сек.)	Всего противников
Самоубийца	1,202754	23,498763	13,891824	374
Пехотинец	1,432824	21,572290	10,477025	224
Ловкач	2,369419	36,026297	18,452279	86
Берсерк	4,649535	46,294207	37,038608	86
Снайпер	0,458065	26,704318	18,248929	159

В результате были получены следующие данные, сформированные из отношения генерации с учетом противников и генерации без учета противников:

Количество смертей игрока выросло на 36%, процентное отношение смертей выросло также на 36%, а средний получаемый урон на 69%. В относительных показателях характеристик противников, также можно наблюдать значительный прирост эффективности:

Таблица 12 – Относительные данные

С учетом, относительно без учета	Относ-е кол-во урона урон	Относ-е среднее время жизни	Относ-е среднее время визуального контакта	Всего противников
Самоубийца	32%	-13%	-11%	761
Пехотинец	75%	-8%	-5%	422
Ловкач	72%	27%	5%	165
Берсерк	37%	-9%	1%	145
Снайпер	56%	10%	53%	267

Интерпретируем полученные данные относительно каждого типа противников:

4.3.1 Самоубийца

Можно смело говорить, что эффективность данного типа врага повысилась, так как количество нанесенного урона увеличилось на 32%. Общее снижение времени жизни и визуального контакта лишь говорит о том, что NPC данного типа понадобилось меньше времени чтобы достичь игрока и нанести ему урон и самоуничтожиться.

4.3.2 Ловкач

Размер нанесенного урона для класса ловкач выросло почти на три четверти, также заметно выросли средняя продолжительность жизни с игроком, что легко объясняется тем, что для класса противника с типом передвижения “Обходящий с флангов” было предоставлено достаточное количество укрытий и пространства при генерации арены.

4.3.3 Берсерк

Средняя эффективность класса берсерк также возросла. Наносимое игроку кол-во урона выросло на 37%, а время визуального контакта снизилось, что свидетельствует о том, что для данного класса противника были предоставлены лучшие условия, обеспечивающие возможность быстрого сближения с игроком и нанесения урона.

4.3.4 Снайпер

КПД типа противника снайпер так же повысился. Являясь противником с типом перемещения “Осторожный” и ведущим огонь с большой дистанции, для снайпера особенно важно иметь прямой визуальный контакт с игроком. Данный параметр вырос на 53%, как следствие, кол-во нанесенного урона также значительно выросло.

4.3.5 Пехотинец

Среднее количество нанесенного урона по игроку у противника класса пехотинец выросло на 75%, а среднее время жизни снизилось за счет того, что увеличилось среднее время взаимодействия с игроком.

ЗАКЛЮЧЕНИЕ

В ходе работы был проведен анализ существующих работ, направленных на изучение используемых в видеоиграх в жанре FPS паттернов проектирования игровых уровней и паттернов создания противников. Также были исследованы некоторые подходы и алгоритмы, используемые при генерации двух- и трехмерных игровых уровней.

Исходя из данного анализа был разработан теоретический алгоритм процедурной генерации уровня для игры в жанре Roguelike FPS с учетом особенностей набора противников и применяемых при проектировании игровых уровней паттернов.

Результатам данной работы стал разработанный алгоритм процедурной генерации аренных игровых уровней зависящего от специфики агентов – противников в шутере от первого лица. Эффективность его работы была подтверждена экспериментальными данными.

В будущем данный метод будет усовершенствован за счет создания более детальной системы логирования действий противников и игрока, что может позволить создать модуль, отвечающий за автоматическую корректировку параметров генерации уровня.

Протестировать геймплей в рамках экспериментальной сборки проекта “Hell over Me”, а также изучить работу алгоритма можно в репозитории по ссылке: <https://github.com/arthe0/ArenaGeneration>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. I3D'19Zhang Y., Zhang G., Huang X. A survey of procedural content generation for games //2022 International Conference on Culture-Oriented Science and Technology (CoST). – IEEE, 2022. – С. 186-190.
2. Henrik K. Minecraft terrain generation in a nutshell // YouTube [Electronic resource] – URL: <https://www.youtube.com/watch?v=CSa5O6knuwI> (дата обращения 11.11.2023)
3. Global Competitive Gaming Market Surges // excorp.gg [Electronic resource] – URL: <https://excorp.gg/news/global-competitive-gaming-market-surges-reached-64.4> (дата обращения 11.11.2023)
4. Rivera G., Hullett K., Whitehead J. Enemy NPC design patterns in shooter games //Proceedings of the First Workshop on Design Patterns in Games. – 2012. – С. 1-8.
5. Qu J., Song Y., Wei Y. Design patterns applied for game design patterns //2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). – IEEE, 2016. – С. 351-356.
6. Rivera G., Hullett K., Whitehead J. Enemy NPC design patterns in shooter games //Proceedings of the First Workshop on Design Patterns in Games. – 2012. – С. 1-8.
7. Hullett K., Whitehead J. Design patterns in FPS levels //proceedings of the Fifth International Conference on the Foundations of Digital Games. – 2010. – С. 78-85.
8. Allen P. M., Alonso W., Applebaum D. Alexander, C.(1964), Notes on the Synthesis of Form, Harvard University Press, Cambridge, MA. Alexander, C.(1965),‘A city is not a tree’, Architectural Forum 122, 58–62. Alexander, C., Ishikawa, S. & Silverstein, M.(1977), A Pattern Language: Towns, Buildings, Construction, Oxford University Press, New York. With M. Jacobson, I //system. – Т. 11. – №. 3. – С. 256-272.

9. Gamma E. et al. Design patterns: Abstraction and reuse of object-oriented design //ECOOP'93—Object-Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings 7. – Springer Berlin Heidelberg, 1993. – C. 406-431.
10. Dahlskog S., Togelius J. A multi-level level generator //2014 IEEE Conference on Computational Intelligence and Games. – IEEE, 2014. – C. 1-8.
11. Cachia W., Liapis A., Yannakakis G. Multi-level evolution of shooter levels //Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. – 2015. – T. 11. – №. 1. – C. 115-121.
12. Viana B. M. F., dos Santos S. R. Procedural dungeon generation: A survey //Journal on Interactive Systems. – 2021. – T. 12. – №. 1. – C. 83-101.
13. Shaker N., Togelius J., Nelson M. J. Procedural content generation in games. – 2016.
14. Baron J. R. Procedural dungeon generation analysis and adaptation //Proceedings of the SouthEast Conference. – 2017. – C. 168-171.
15. Aurenhammer F. Voronoi diagrams—a survey of a fundamental geometric data structure //ACM Computing Surveys (CSUR). – 1991. – T. 23. – №. 3. – C. 345-405.
16. Herbert Wolverson - Procedural Map Generation Techniques // YouTube [Electronic resource] – URL https://youtu.be/TILIOgWYVpI?si=iLMU92C7he_LCDBA&t=593
17. Santamaría-Ibirika A. et al. Procedural playable cave systems based on voronoi diagram and delaunay triangulation //2014 International Conference on Cyberworlds. – IEEE, 2014. – C. 15-22.
18. Shivanasab P., Abbaspour R. A. An incremental algorithm for simultaneous construction of 2D Voronoi diagram and Delaunay triangulation based on a face-based data structure //Advances in Engineering Software. – 2022. – T. 169. – C. 103129.

19. Gaisbauer W. et al. Procedural generation of video game cities for specific video game genres using WaveFunctionCollapse (WFC) //Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts. – 2019. – C. 397-404.
20. Alaka S., Bidarra R. Hierarchical Semantic Wave Function Collapse //Proceedings of the 18th International Conference on the Foundations of Digital Games. – 2023. – C. 1-10.
21. Silva R. C. E. et al. Procedural generation of 3D maps with snappable meshes //IEEE Access. – 2022. – T. 10. – C. 43093-43111.
22. Andrian R., Permana A. A., Kusnadi A. Design Adventure Role Playing Game with Procedural Content Generation using Perlin Noise Algorithm //Journal of Theoretical and Applied Information Technology. – 2023. – T. 101. – №. 9.
23. Kim H. et al. Automatic generation of game content using a graph-based wave function collapse algorithm //2019 IEEE Conference on Games (CoG). – IEEE, 2019. – C. 1-4.
24. Johnson L., Yannakakis G. N., Togelius J. Cellular automata for real-time generation of infinite cave levels //Proceedings of the 2010 Workshop on Procedural Content Generation in Games. – 2010. – C. 1-4.
25. Earle S. et al. Illuminating diverse neural cellular automata for level generation //Proceedings of the Genetic and Evolutionary Computation Conference. – 2022. – C. 68-76.
26. Barriga N. A. A short introduction to procedural content generation algorithms for videogames //International Journal on Artificial Intelligence Tools. – 2019. – T. 28. – №. 02. – C. 1930001.
27. Kruse J., Connor A. M., Marks S. An interactive multi-agent system for game design //The Computer Games Journal. – 2021. – T. 10. – C. 41-63.
28. Loiacono D., Arnaboldi L. Multiobjective evolutionary map design for cube 2: Sauerbraten //IEEE Transactions on Games. – 2018. – T. 11. – №. 1. – C. 36-47.

29. Bailly R., Leveux G. Genetic-WFC: Extending Wave Function Collapse With Genetic Search //IEEE Transactions on Games. – 2022. – T. 15. – №. 1. – C. 36-45.