УДК 004.021:004.75

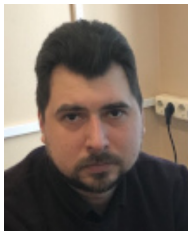
РАБОТА АЛГОРИТМА КОЛЛАПСА ВОЛНОВОЙ ФУНКЦИИ НА ПРИМЕРЕ ИГРЫ «TOWER OF ELEVATION»

С.Ю. Слюсарь
Учащийся Учреждения образования «Национальный детский технопарк», учащийся ГУО «Гимназия г. Логойска»

Д.С. Коваленко
Учащийся Учреждения образования «Национальный детский технопарк», учащийся ГУО «Гимназия г. Логойска»

М.С. Ильясова
магистрант кафедры инженерной психологии и эргономики БГУИР

Ф.В. Усенко
магистрант кафедры инженерной психологии и эргономики БГУИР



Л.Р. Коркин
магистр технических наук, ассистент кафедры инженерной психологии и эргономики БГУИР

А.М. Прудник
доцент кафедры инженерной психологии и эргономики БГУИР, кандидат технических наук, доцент
aleksander.prudnik@bsuir.by

С.Ю. Слюсарь
Обучается в гимназии г. Логойска. Область научных интересов связана с технологиями виртуальной и дополненной реальности.

Д.С. Коваленко
Обучается в гимназии г. Логойска. Область научных интересов связана с технологиями виртуальной и дополненной реальности.

М.С. Ильясова
Окончила Белорусский государственный университет информатики и радиоэлектроники. Область научных интересов связана с автоматизированным тестированием информационных систем.

Ф.В. Усенко
Окончил Белорусский государственный университет информатики и радиоэлектроники. Область научных интересов связана с разработкой манипуляторов для дистанционного управления.

Л.Р. Коркин
Окончил Белорусский государственный университет информатики и радиоэлектроники. Область научных интересов связана с системами распознавания снимков.

А.М. Прудник
Окончил Белорусский государственный университет информатики и радиоэлектроники. Область научных интересов связана с взаимодействием человека с компьютером, интерфейсами информационных систем, пользовательскими интерфейсами, front-end web development.

Аннотация. В статье описывается алгоритм коллапса волновой функции и пример его работы на примере игры «Tower of elevation».

Ключевые слова: Алгоритм коллапса волновой функции, игра «Tower of elevation», , Unity, MagicaVoxel, тайл, мир, карта,Visual Studio.

Введение.

Алгоритмы имеют большую популярность в «мире» разработчиков. Их используют в играх разного масштаба: от совсем коротеньких и с относительно маленькой картой, до бесконечных миров, где игрок вряд ли сможет дойти до границы мира.

Цели доклада:

1. Рассказать о алгоритме коллапса волновой функции;
2. Показать работу алгоритма на примере разработанной игры;
3. Рассказать о плюсах и минусах алгоритма.

Основная часть.

В настоящее время разработка игр является одним из наиболее распространенных направлений в компьютерной индустрии. Некоторые из них создаются просто путем ручной расстановки элементов карты дизайнерами, другие ограничиваются небольшой локацией, в которой игрок находится, а третьи используют компьютер для создания огромных миров. Мы относимся к последней категории разработчиков, что особенно полезно для игр с большой картой, генерируемой в произвольном порядке, имеющей множество вариаций. Примером такой игры является «Minecraft» [1], где большое количество вариаций блоков расставляется в произвольном порядке в соответствии с используемым алгоритмом. Практически все эти алгоритмы работают по принципу, который заключается в сопоставлении границы модели с другой моделью, чтобы получить единый и гармоничный результат.

Описание алгоритма коллапса волновой функции.

Алгоритм коллапса волновой функции (Wavefunction Collapse Algorithm) учит компьютер «импровизировать». На входе задаются архетипичные данные и создаются процедурно-генерируемые данные, похожие на исходные и приведенные на рисунке 1.Чаще всего алгоритм используется для создания изображений, но может также строить города, скейтпарки и другие объекты. В нашей игре алгоритм строит локации, на которых происходят все действия игрока. Каждая часть локации была создана не человеком, а алгоритмом. Это означает, что игроку будет интересно исследовать мир игры, так как он не сможет предугадать, где что находится [2].

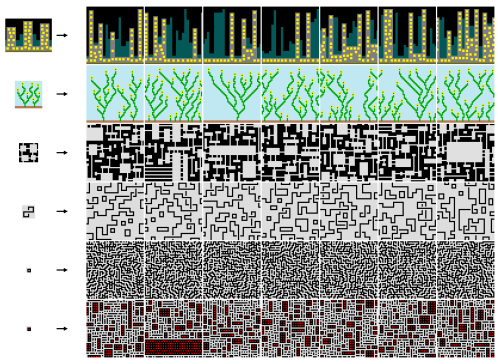


Рисунок 1. Создание процедурно-генерируемых данных, похожих на исходные

Коллапс волновой функции — это очень «независимо мыслящий» алгоритм, не требующий практически никакой помощи или инструкций извне. Необходим только пример стиля, которого нужно достичь, а всё остальное алгоритм сделает сам. Несмотря на свою самодостаточность, алгоритм является простым и не использует никаких нейронных сетей, случайных лесов или чего-то другого, похожего на машинное обучение [3].

Реализация работы алгоритма на примере игры «Tower of elevation».

Игра «Tower of Elevation» была создана для исследования работы алгоритма коллапса волновой функции. В процессе создания игры были выполнены следующие этапы:

1. Создание в MagicaVoxel тайлов, соответствующих требованиям алгоритма коллапса волновой функции.
2. Разработка алгоритма в среде программирования Visual Studio.
3. Интеграция алгоритма с ранее разработанными тайлами и тестирование его работы.
4. Анализ плюсов и минусов работы алгоритма коллапса волновой функции.

Следует рассмотреть каждый этап более подробно.

Первым этапом являлось создание тайлов, которые должны были соответствовать требованиям алгоритма коллапса волновой функции. Для этого каждый тайл должен был иметь определенную границу, чтобы алгоритм мог взаимодействовать с ними [4]. Если границы у каждого тайла разные, то алгоритм не сможет их обрабатывать. На рисунке 2 показаны все тайлы, которые были использованы в работе.

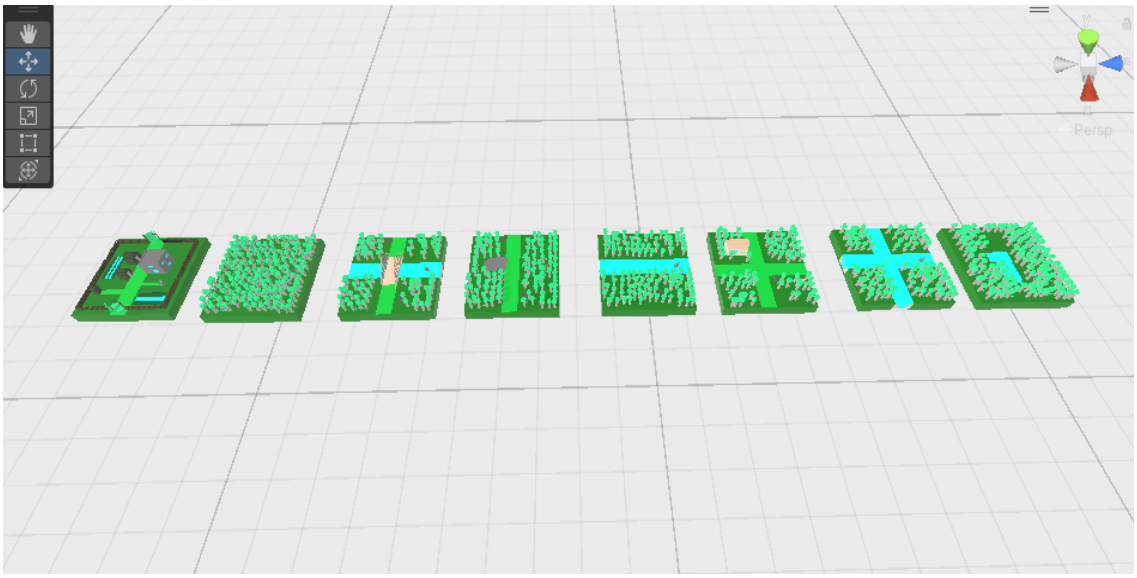


Рисунок 2. Виды тайлов, которые были использованы в работе разработки игры

Затем приступили к разработке скриптов в интегрированной среде разработки Visual Studio. Написанный программный код предназначался для тайлов и места их расположения. Примеры начала написания кода продемонстрированы на рисунках 3 и 4, а фрагмент кода приведён ниже.

Скрипт, представленный на рисунке 3, выполняет роль «конструктора». То есть скрипт считывает из каких цветов состоит боковая сторона определённого тайла и потом благодаря этому скрипту программа может соединять различные тайлы логически и без резких изменений для глаза человека. Одним словом, с помощью этого скрипта делаются плавные и логические переходы между разными тайлами.

```
        public void CalculateSideColors()
    {
        ColorsRight = new byte[TilesSideVoxels * TilesSideVoxels];
        ColorsForward = new byte[TilesSideVoxels * TilesSideVoxels];
        ColorsLeft = new byte[TilesSideVoxels * TilesSideVoxels];
        ColorsBack = new byte[TilesSideVoxels * TilesSideVoxels];
        for (int y = 0; y < TilesSideVoxels; y++)
        {
            for (int i = 0; i < TilesSideVoxels; i++)
            {
                ColorsRight[y * TilesSideVoxels + i] =
GetVoxelColor(verticalLayer y, horizontalOffset i, Direction.right);
                ColorsForward[y * TilesSideVoxels + i] =
GetVoxelColor(verticalLayer y, horizontalOffset i, Direction.forward);
                ColorsLeft[y * TilesSideVoxels + i] =
GetVoxelColor(verticalLayer y, horizontalOffset i, Direction.left);
                ColorsBack[y * TilesSideVoxels + i] =
GetVoxelColor(verticalLayer y, horizontalOffset i, Direction.back);
            }
        }

    }
```

Рисунок 3. Скрипт для плавного и логического перехода между тайлами

Скрипт, представленный на рисунке 4, устанавливает определённые координаты для расположения тайлов. То есть при задании определённых координат алгоритм сам решает, какой тайл из списка тайлов подходит в определенное положение на сцене.

```
    public void PlaceTile(int x, int y)
    {
        List<VoxelTile> availableTiles = new List<VoxelTile>();
        foreach (VoxelTile tilePrefab in TilePrefabs)
        {
            if (CanAppendTile(existingTile: spawnedTiles[x - 1, y],
tileToAppend: tilePrefab, Direction.Left) &&
                CanAppendTile(existingTile: spawnedTiles[x + 1, y],
tileToAppend: tilePrefab, Direction.Right) &&
                CanAppendTile(existingTile: spawnedTiles[x, y - 1],
tileToAppend: tilePrefab, Direction.Back) &&
                CanAppendTile(existingTile: spawnedTiles[x, y + 1],
tileToAppend: tilePrefab, Direction.Forward))
            {
                availableTiles.Add(tilePrefab);
            }
        }
        if (availableTiles.Count == 0) return;
        VoxelTole selectedTile = GetRandomTile(availableTiles);
        Vector3 position = new Vector3(x, y: 0, z: y) * selectdTile.VoxelSide
* selectedTile.TileSideVoxels;
        spawnedTiles[x, y] = Instatiate(selectedTile, position,
Quaternion.Identity);
    }
        }
    }
```

Рисунок 4. Скрипт для автоматического расположения тайлов по координатам

И заключительным этапом разработки игры является применение алгоритма. Результат выполнения алгоритма показан на рисунке 5.

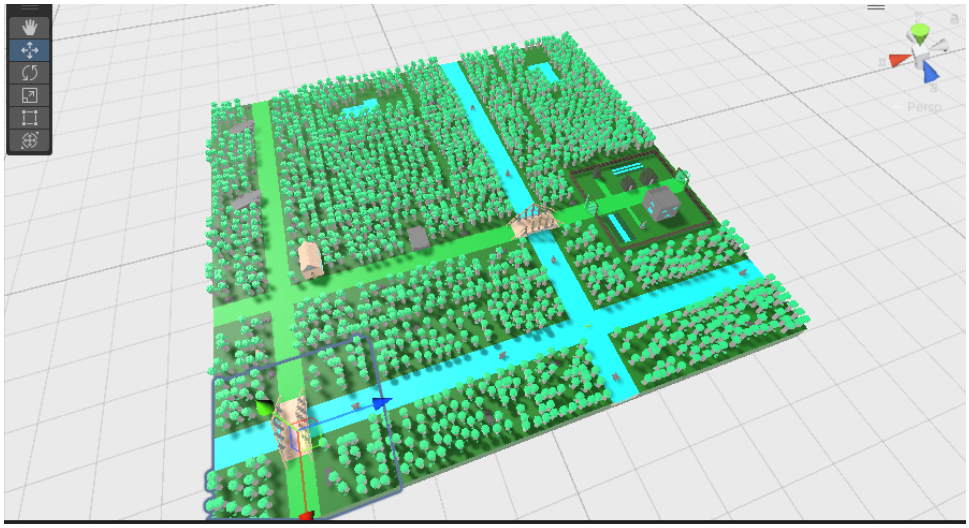


Рисунок 5. Результат выполнения алгоритма коллапса волновой функции

Перечислим следующие преимущества алгоритма коллапса волновой функции:

1. Простота работы алгоритма, что делает его привлекательным для использования.
2. Удобство в использовании, что упрощает процесс его применения..
3. Быстрота достижения результата действий алгоритма и возможность его перезапуска с дополнительными настройками.

К недостаткам алгоритма коллапса волновой функции можно отнести:

1. Ограниченность возможностей создаваемых объектов.
2. Необходимость точного определения границ тайлов.
3. Ограниченность разнообразия создаваемых объектов.
4. Необходимость многочисленных экспериментов для определения наиболее подходящих правил и параметров алгоритма.
5. Сложность создания и отладки алгоритма для новых типов объектов.

Заключение.

В результате выполнения проекта разработана игра виртуальной реальности «Tower of elevation» с генерацией карты при помощи алгоритма коллапса волновой функции. Игра рассчитана на аудиторию до 18 лет, хотя в неё может играть и взрослый человек. Так как приложение представляет собой игру с некоторым сюжетом, пользователю будет интересно «продвигаться» по игре, разыскивая пасхалки, сделанные разработчиками. Так как в игре после начального тестирования не обнаружено никаких багов и всё работает правильно, то можно начать крупное тестирование, набирая бета-тестеров и разрабатывать обновления для игры.

Список литературы

[1] Minecraft [Электронный ресурс]. URL: https://www.minecraft.net/ru-ru. (Дата обращения: 12.04.2023).

[2] Генерация уровня из 3D тайлов: часть 1 [Электронный ресурс]. URL: https://www.youtube.com/watch?v=iG_n23W952Y&list=RDCMUC6wnai488mwec_FUVfdl84w&index=10. (Дата обращения: 12.04.2023).

[3] Генерация уровня из 3D тайлов: часть 2 [Электронный ресурс]. URL: https://www.youtube.com/watch?v=fMtbvWCGke4&t=636s. (Дата обращения: 12.04.2023).

[4] Генерация уровня из 3D тайлов: часть 3 [Электронный ресурс]. URL: https://www.youtube.com/watch?v=EG0iERtXc0U&list=RDCMUC6wnai488mwec_FUVfdl84w&index=1. (Дата обращения: 12.04.2023).

OPERATION OF THE WAVE FUNCTION COLLAPSE ALGORITHM ON THE EXAMPLE OF THE GAME "TOWER OF ELEVATION"

S.Y. Slusar
Student of the educational Institution "National Children's Technopark", student GUO "Gymnasium of Logoysk"

D.S. Kovalenko
Student of the educational Institution "National Children's Technopark", student GUO "Gymnasium of Logoysk"

M.S. Ilyasova
Master's student of the Department of Engineering Psychology and Ergonomics of BSUIR

F.V. Usenko
Master's student of the Department of Engineering Psychology and Ergonomics of BSUIR

L.R. Korkin
Master of Technical Sciences, Assistant of the Department of Engineering Psychology and Ergonomics of BSUIR

A.M. Prudnik
Associate Professor of Engineering Psychology and Ergonomics Department of BSUIR, Candidate of Technical Sciences, Associate Professor aleksander.prudnik@bsuir.by

Department of Engineering Psychology and Ergonomics
Faculty of Computer-Aided Design
Belarusian State University of Informatics and Radio Electronics, Republic of Belarus
E-mail: aleksander.prudnik@bsuir.by