***И.Н. Паршиков, В.С. Иванов,  
ст. преп. Н.С. Васильев***

**Генерация карт с использованием WFC**

**Аннотация**. Показано использование алгоритма коллапса волновой функции для генерации игровых карт на языке C#.

**Ключевые слова:** алгоритм коллапса волновой функции, C#, генерация, игровой мир.

***I.N. Parshikov, V.S. Ivanov,  
ст. преп. N.S. Vasiliev***

**Generating maps using WFC**

**Annotation.**  The use of the Wave Function Collapse algorithm for generating game maps in C# is demonstrated.

**Keywords:** Wave Function Collapse algorithm, C#, generation, game world.

Игровые миры требуют разнообразия и уникальности, с чем помогают различные процедурные генерации.

В данной работе описывается генерация карт при помощи алгоритма коллапса волновой функции.

Wfc (Wavefunction Collapse Algorithm) – алгоритм коллапса волновой функции — это процесс, описывающий, как система квантовых частиц переходит из состояния суперпозиции в одно определенное состояние при измерении. В квантовой механике частицы могут находиться в множестве состояний одновременно, что выражается в виде волновой функции. Когда происходит измерение, волновая функция "коллапсирует" в одно из возможных состояний, что приводит к наблюдаемому результату.

Алгоритм генерации карт с использованием WFC делится на 6 шагов:

1. Разбиение карты на части
2. Поиск элемента с наибольшей энтропией
3. Выполнение коллапса для найденного элемента
4. Обновление состояний
5. Выполнение шагов 2-4 пока есть частицы в суперпозиции

Для реализации данного алгоритма использовался язык программирования C#.

Рассмотрим реализацию некоторых шагов алгоритма в коде.

За разбиение карты на части отвечает фрагмент:

public class Region

{

public List<(int x, int y)> Pixels = new List<(int x, int y)>();

public List<Region> Neighbors = new List<Region>();

public Biomes? Biome = null;

public List<Biomes> PossibleBiomes = new List<Biomes>();

public void FindNeighbors(List<Region> regions)

{

foreach (var region in regions)

{

if (region != this) // Исключаем сам регион

{

foreach (var pixel in Pixels)

{

// Проверяем соседние пиксели

if (region.Pixels.Contains((pixel.x + 1, pixel.y)) || // Сосед справа region.Pixels.Contains((pixel.x - 1, pixel.y)) || // Сосед слева region.Pixels.Contains((pixel.x, pixel.y + 1)) || // Сосед сверху region.Pixels.Contains((pixel.x, pixel.y - 1))) // Сосед снизу

{

Neighbors.Add(region);

break; // Выходим из цикла, если нашли соседа

}

}

}

}

}

}

Коллапс для найденного элемента осуществляется так:

// Находим максимальное количество соседей без биома

var maxNeighbors = regions.Max(region =>

region.Biome == null ? region.Neighbors.Count(neighbor => neighbor.Biome == null) : 0);

if (maxNeighbors == 0 && regions.All(x=>x.Biome != null)) break; // Если нет доступных соседей, выходим из цикла

if (maxNeighbors == 0)

{

var region = regions.FirstOrDefault(x=> x.Biome == null);

region.Biome = region.PossibleBiomes.FirstOrDefault();

Console.WriteLine($"установлен биом для региона {n}");

n++;

continue;

}

// Находим регион с максимальным количеством соседей без биома

var currentRegion = regions.First(region =>

region.Neighbors.Count(neighbor => neighbor.Biome == null) == maxNeighbors && region.Biome == null);

if (currentRegion.PossibleBiomes.Count == 0)

{

Console.WriteLine($"установлен биом для региона {n}");

n++;

currentRegion.Biome = (Biomes)\_random.Next(5);

currentRegion.PossibleBiomes = \_biomRules[currentRegion.Biome.Value];

UpdateNeighborsFirs(currentRegion);

}

В результате выполнения программы создается картинка формата .png в корне проекта.

Пример сгенерированной карты можно увидеть на рис.1.

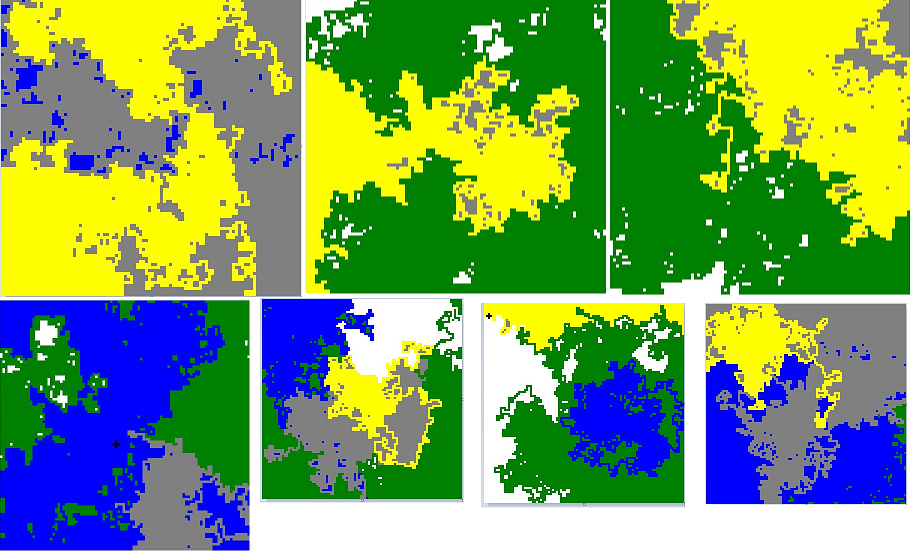


Рис.1. Пример сгенерированной карты 100\*100.

Разработанная программа позволяет генерировать карты биомов, что ускоряет разработку.