Створення гри на Unity з автоматичною генерацією рівнів

Виконав Янченко Б. І.

Науковий керівник Жежерун О.П.

Мета роботи

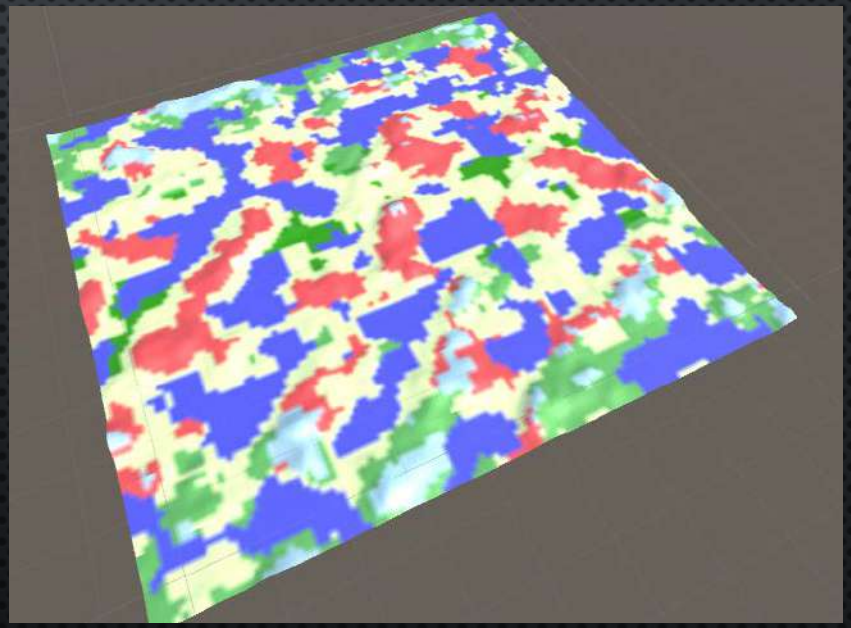
- Розробка прототипу 2D-гри з автоматичною генерацією рівнів на основі різних методів формування ландшафту — зокрема перлинного, фрактального, симплексного шумів, синусоїдальних функцій та їх комбінацій.

Що таке процедурна генерація

Процедурна генерація — це автоматизований процес створення ігрового контенту (рівнів, ландшафтів, об’єктів) за допомогою алгоритмів.

Переваги:

- Створення унікального контенту «на льоту»
- Економія часу та ресурсів розробників
- Нескінченна варіативність та реіграбельність
- Масштабованість без додаткових витрат
- Адаптація рівнів під гравця

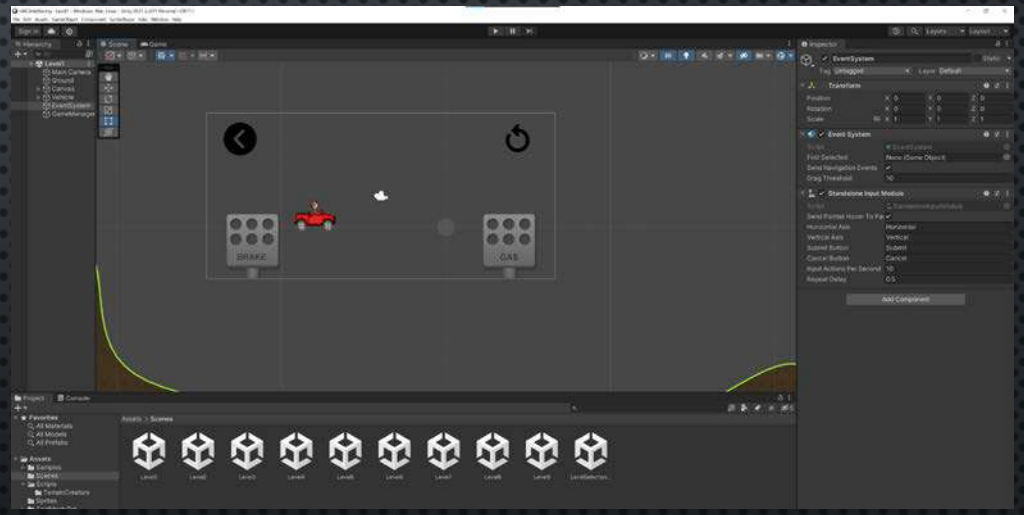


Чому Came Hill Climb Racing

- Має просту, але захоплюючу механіку керування (газ / гальмо)
- Добре реалізована фізика та відчуття прогресу
- Підходить для реалізації процедурної генерації рівнів
- Недолік оригіналу — фіксовані траси, які швидко набридають
- У дипломному проєкті реалізовано динамічну генерацію рельєфу, яка робить кожну спробу унікальною

Unity як рушій для створення

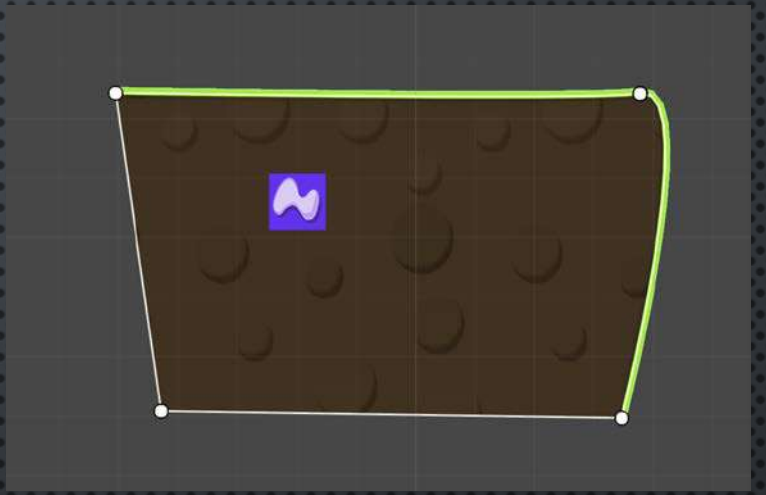
- Потужний інструмент для створення 2D-ігор
- Підтримка мови C#
- Великий обсяг документації
- Зручний редактор



Реалізація рівнів

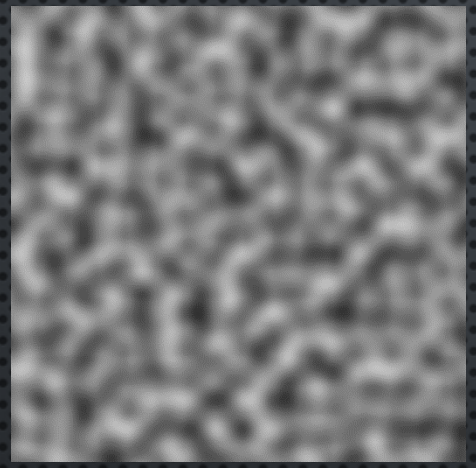
- У грі реалізовано 9 типів рівнів, кожен із унікальним підходом до генерації рельєфу
- Використано різні алгоритми: Перлинний шум, фрактальний шум, синусоїдальні функції, Симплексний шум та їх комбінації
- Деякі рівні мають динамічну складність — ландшафт ускладнюється під час гри

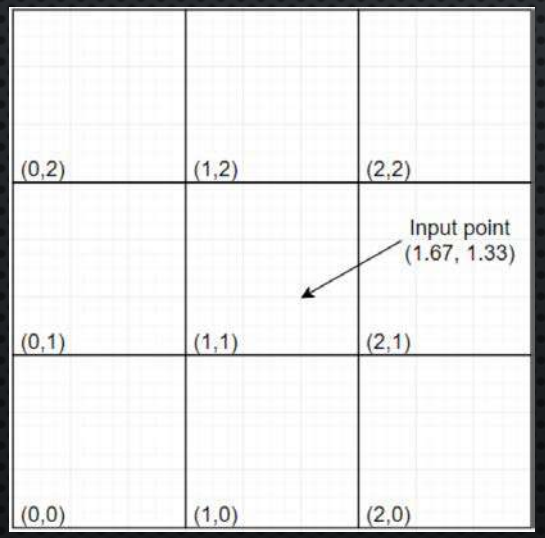
SPRITE SHAPE CONTROLLER



- Ініціально створюється сплайн з 4-ма стартовими точками
- Далі в циклі:
  - Обчислюється висота нової точки за допомогою, наприклад, Перлін шуму
  - Нова точка додається в сплайн методом InsertPointAt(...)
  - Задаються тангенси для плавності вигину (SetTangentMode, SetLeftTangent, SetRightTangent)
  - В кінці встановлюється фінальна точка, яка логічно завершує трасу

PERLIN NOISE



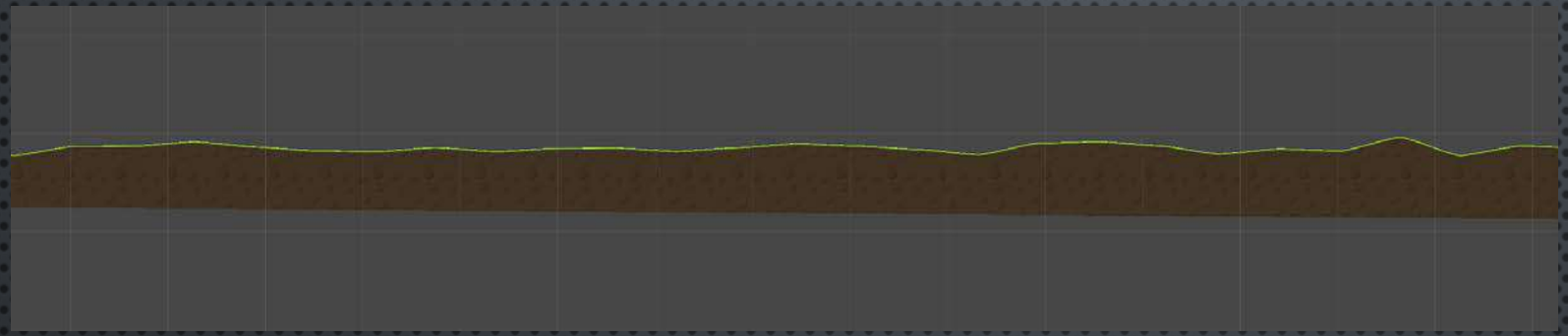


Алгоритм Перлін шуму створює **плавний псевдовипадковий шум**, який широко застосовується в процедурній генерації рельєфів, хмар, текстур тощо.

Основні принципи роботи:

- Простір ділиться на **ґратку з фіксованими градієнтами**
- Для кожної точки:
  - Визначаються **вектори до сусідніх вершин**
  - Обчислюється **скалярний добуток** цих векторів із градієнтами
  - Результати **інтерполюються** за згладжувальною функцією (наприклад, fade)

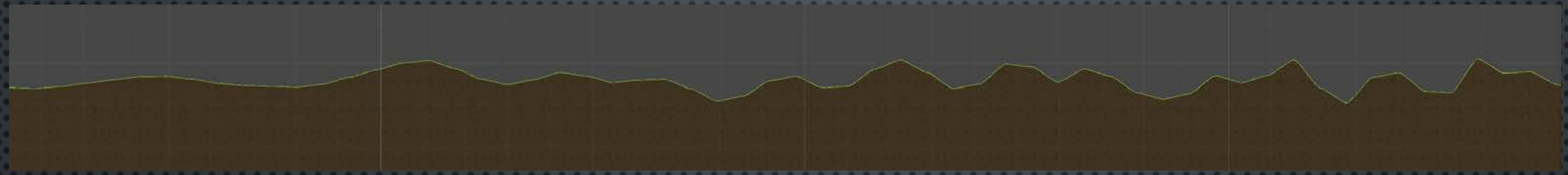
РЕАЛІЗАЦІЯ РІВНЯ



$$y_i = PerlinNoise(x_i * k, offset) * h$$

- y_i – ВИСОТА ТОЧКИ РЕЛЬЄФУ;
- x_i - КООРДИНАТА ТОЧКИ ПО ОСІ X;
- *K* – МАСШТАБ;
- *OFFSET* - ВИПАДКОВИЙ ЗСУВ ПО Y, ЩОБ УНИКНУТИ ОДНАКОВИХ ШАБЛОНІВ;
- *H* — КОЕФІЦІЄНТ АМПЛІТУДИ.

ПОСТУПОВЕ ПІДВИЩЕННЯ СКЛАДНОСТІ



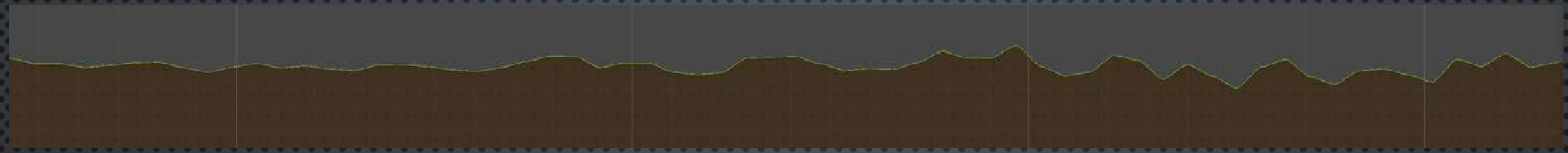
$$y_i = y_{base} + ([PerlinNoise(x_i * k_i, offset) - 0,5] * 2) * h_i$$

$$k_i = k_0 + i * \Delta k$$

$$h_i = h_0 + i * \Delta h$$

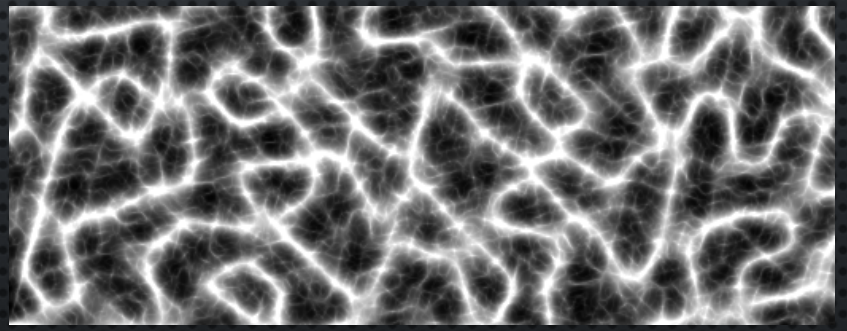
- y_i – висота точки на координаті x_i;
- k_i - поточний масштаб Перлинного шуму (що зростає з кожною ітерацією);
- h_i - поточна амплітуда шуму (також зростає);
- $\Delta k, \Delta h$ - малі прирости, які задають швидкість ускладнення рельєфу;
- *offset* випадковий зсув по другій координаті.

ФРАКТАЛЬНИЙ ШУМ



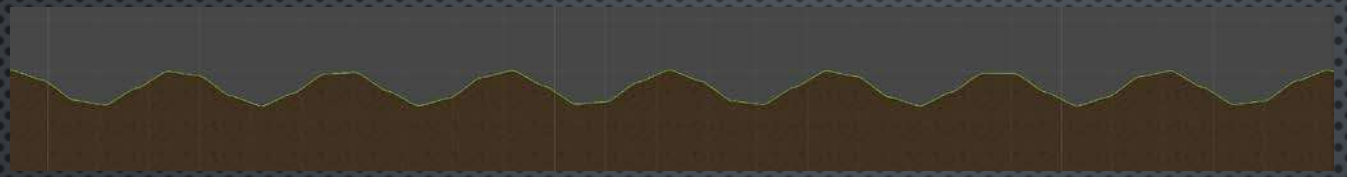
$$f(x) = \frac{1}{A}\sum_{i=0}^{n-1} PerlinNoise\left(x * scale * \lambda^i\right) * p^i$$

- *N* – КІЛЬКІСТЬ ОКТАВ (РІВНІВ ШУМУ);
- *Λ (LACUNARITY)* – КОЕФІЦІЄНТ ЗРОСТАННЯ ЧАСТОТИ НА КОЖНУ ОКТАВУ;
- *P (PERSISTENCE)* – КОЕФІЦІЄНТ ЗМЕНШЕННЯ АМПЛІТУДИ;
- *SCALE* – БАЗОВИЙ МАСШТАБ ШУМУ;
- $A = \sum_{i=0}^{n-1} p^i$ – НОРМАЛІЗУЮЧИЙ МНОЖНИК

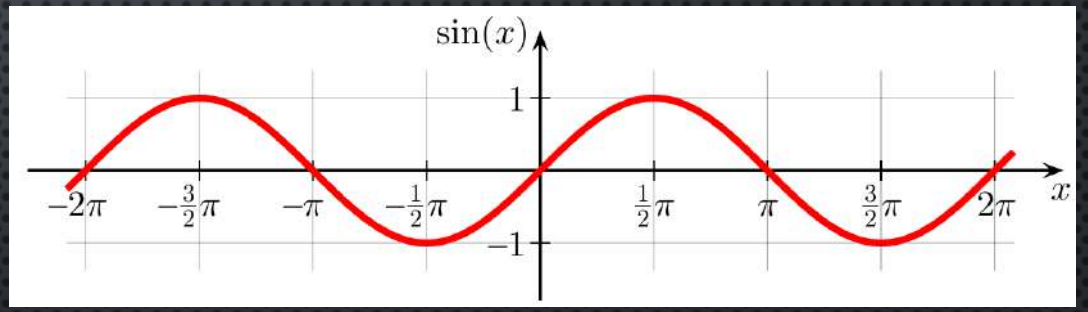


```
float y = baseGroundY + ((GenerateFractalNoise(x, offset, baseScale, octaves, lacunarity,
persistence) - 0.5f) * 2f * heightMultiplier);
```

СИНУСОЇДА



$$y(x) = A * sin(f * x) + y_0$$



- *A* – АМПЛІТУДА;
- *F* – ЧАСТОТА;
- *X* – КООРДИНАТА ПО ГОРИЗОНТАЛІ:
- y_0 - ВЕРТИКАЛЬНЕ ЗМІЩЕННЯ, АБО БАЗОВИЙ РІВЕНЬ ЗЕМЛІ.

```
float y = baseGroundY + Mathf.Sin(x * frequency) * amplitude;
```

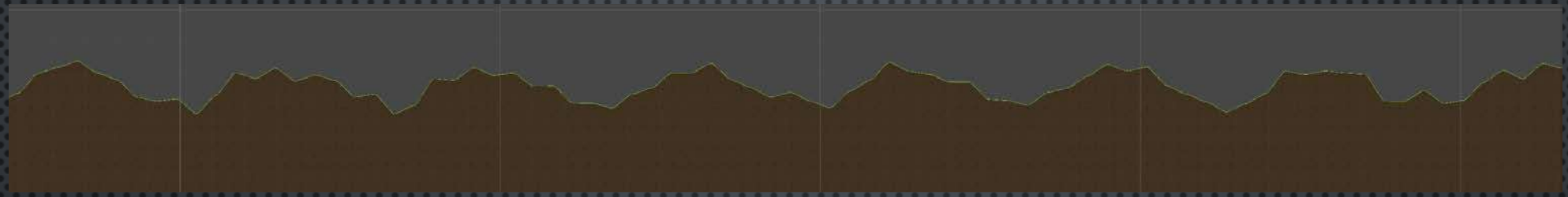
«УСКЛАДНЕНА» СИНУСОЇДА

$$y(x) = A(x) * sin(f(x) * x) + y_0$$

```
float y = baseGroundY + Mathf.Sin(x * currentFrequency) * currentAmplitude;
```

```
currentAmplitude += amplitudeIncreaseRate;
currentFrequency += frequencyIncreaseRate;
```

КОМБІНАЦІЇ

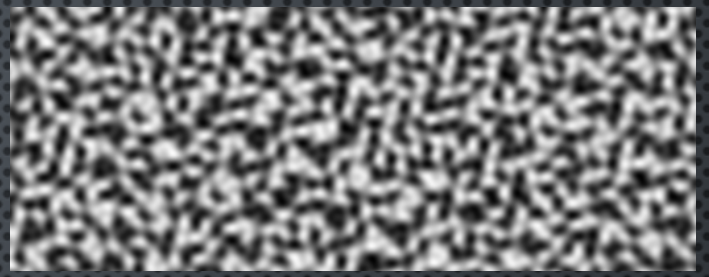


$$y(x) = A * sin(f * x) + N(x) + y_0$$

- $A * \text{SIN}(f * x)$ – СИНУСОЇДА;
- $N(x)$ – ПЕРЛИННИЙ ШУМ, ЯКИЙ В СВОЮ ЧЕРГУ ЯВЛЯЄ СОБОЮ;
  - $N(x) = (Perlin(x * s) - 0.5) * 2 * S$3 МАСШТАБОМ S ТА СИЛОЮ S;
- y_0 - БАЗОВИЙ РІВЕНЬ ЗЕМЛІ.

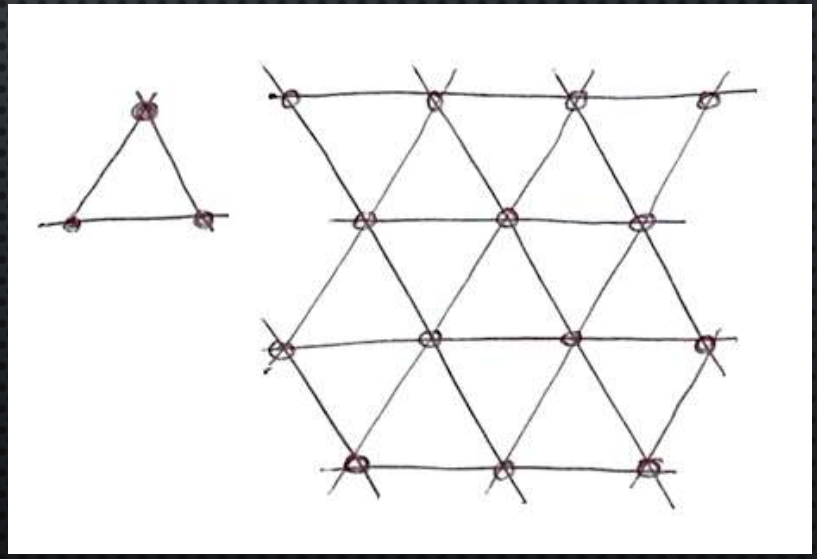
```
float sineWave = Mathf.Sin(x * baseFrequency) * baseAmplitude;
float noise = (Mathf.PerlinNoise(x * noiseScale, noiseOffset) - 0.5f) * 2f *
noiseStrength;
float y = baseGroundY + sineWave + noise;
```

Simplex Noise



Алгоритм **Simplex Noise** був розроблений Кеном Перліном у 2001 році як **ефективніша альтернатива** класичному Perlin Noise.

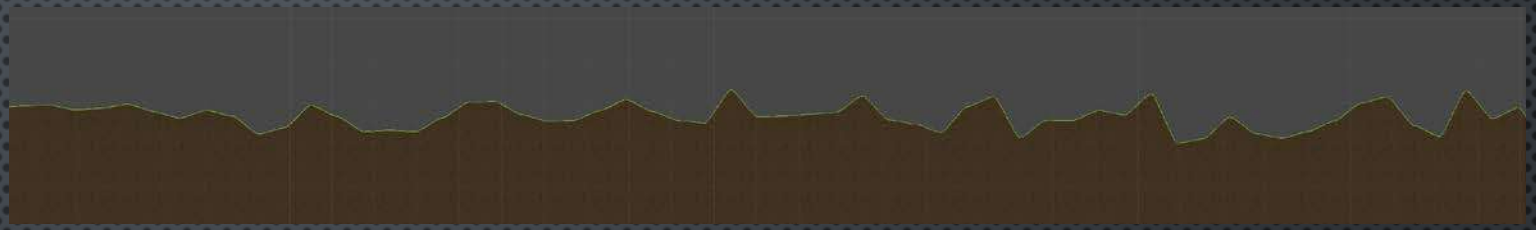
- Працює не з квадратами/кубами, а з **симплексами:**
  - ▲ Трикутники в 2D
  - ◆ Тетраедри в 3D
- Дає **менше артефактів**, особливо у вищих вимірах
- **Швидший** для обчислення при великій кількості вимірів



РЕАЛІЗАЦІЯ

$$n_i = t_i^4 * (\overrightarrow{g_i} * \overrightarrow{x_i})$$

- $t_i = 0.5 - ||\overrightarrow{x_i}||$ - функція згасання;
- $\overrightarrow{g_i}$ – градієнтний вектор у вершині;
- $\overrightarrow{x_i}$ - вектор від вершини до точки.



- Таблиця перестановок perm[]
- Градієнтні вектори задаються масивом grad3[].
- Визначення симплексу базується на сумі координат із коефіцієнтом F2:

$$F2 = \frac{\sqrt{3} - 1}{2}$$

- Компенсація відбувається за допомогою G2:

$$G2 = \frac{3 - \sqrt{3}}{6}$$

- Кожен із трьох трикутників симплексу вносить свій вклад у фінальне значення.

Висновки

- Було реалізовано 2D-гру з автоматичною генерацією рівнів на рушії Unity.
- Досліджено основні підходи до процедурної генерації: шумові алгоритми (Perlin, Simplex), синусоїдальні функції та їх комбінації.
- Розроблено прототип, Hill Climb Racing, але з автоматичним створенням трас.
- Завдяки використанню Sprite Shape Controller та Edge Collider 2D, реалізовано гнучке і реалістичне формування рельєфу.
- Реалізовано декілька методів створення траси.
- Розроблено Simplex Noise для Unity.

ДЯКУЮ ЗА УВАГУ