Bohemia Interactive

Spojitý šum v praxi

Jan Gehr

Úvod

Za použití **Unity** si ukážeme následující příklady:

1. Jednoduchý příklad z praxe (náhodný spojitý pohyb terče)
2. Ukázka implementace generátoru terénu (podobně jako ve hře "They are billions")

Jednoduchý příklad z praxe

-Náhodně pohybující se terč

Jednoduchý příklad z praxe

-Náhodně pohybující se terč

```
/// <summary>
/// Přesune objekt na náhodnou pozici x každý frame
/// </summary>
public class RandomMove : MonoBehaviour
{
    // Update se volá jendou za herní frame
    public void Update()
    {
        transform.position = new Vector3(Random.Range(-6f,6f), 0, 0);
    }
}
```

Jednoduchý příklad z praxe

-Náhodně pohybující se terč

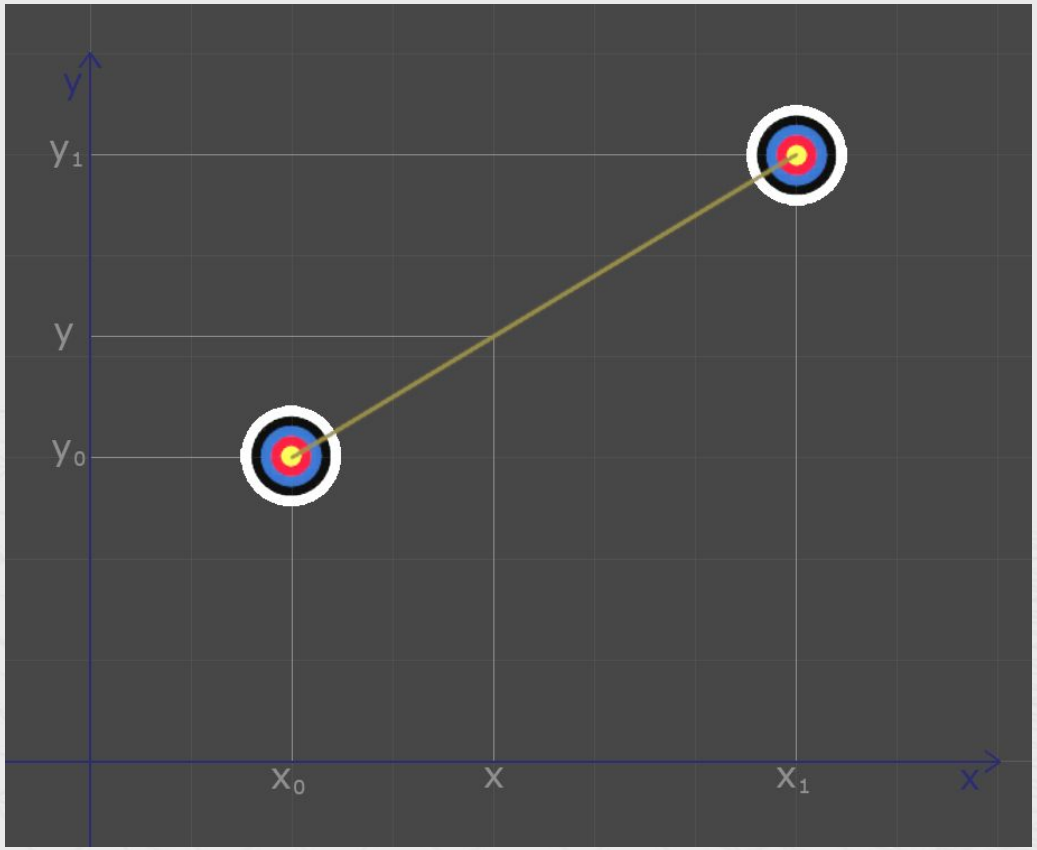
Lineární interpolace

-Proložení dvou bodů přímkou

-Obecný vzorec pro interpolaci

mezi body A(x0,y0) a B(x1,y1)

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$$

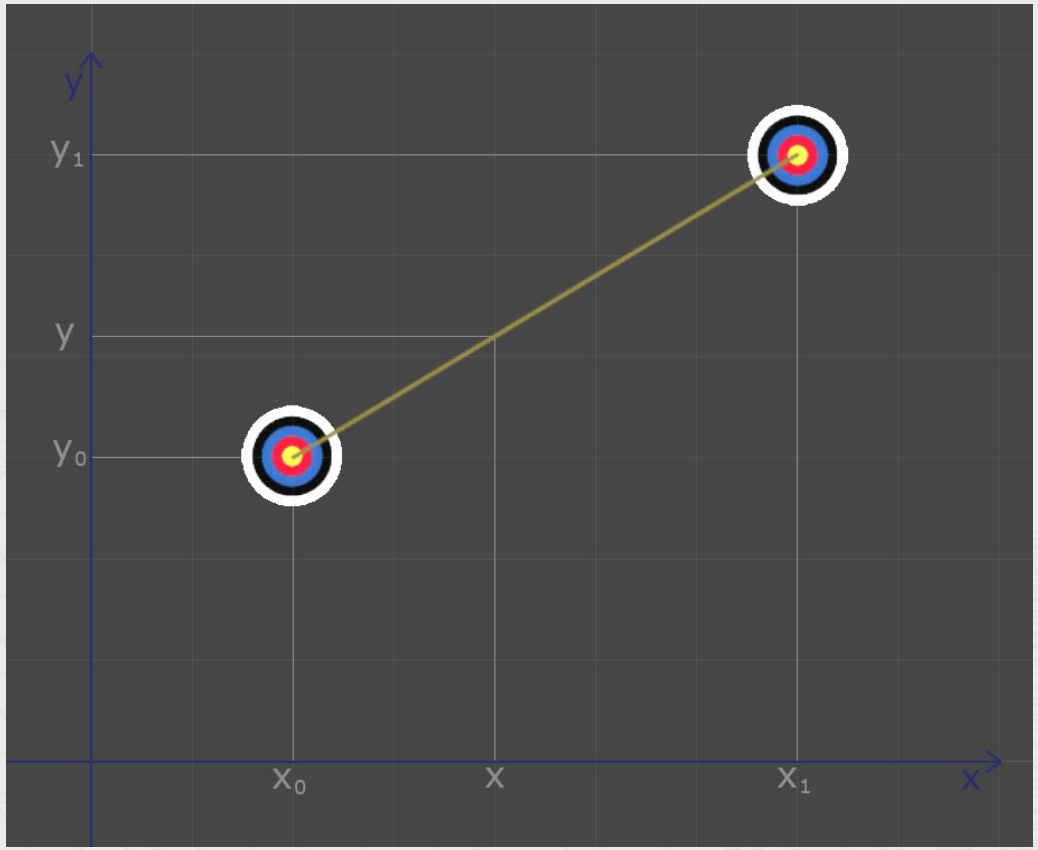


Lineární interpolace

-Proložení dvou bodů přímkou

-Obecný vzorec pro interpolaci mezi body A(x0,y0) a B(x1,y1)

$$y = y_0 + x(y_1 - y_0)$$



Lineární interpolace

-Proložení dvou bodů přímkou

-Obecný vzorec pro interpolaci mezi body A(x0,y0) a B(x1,y1)

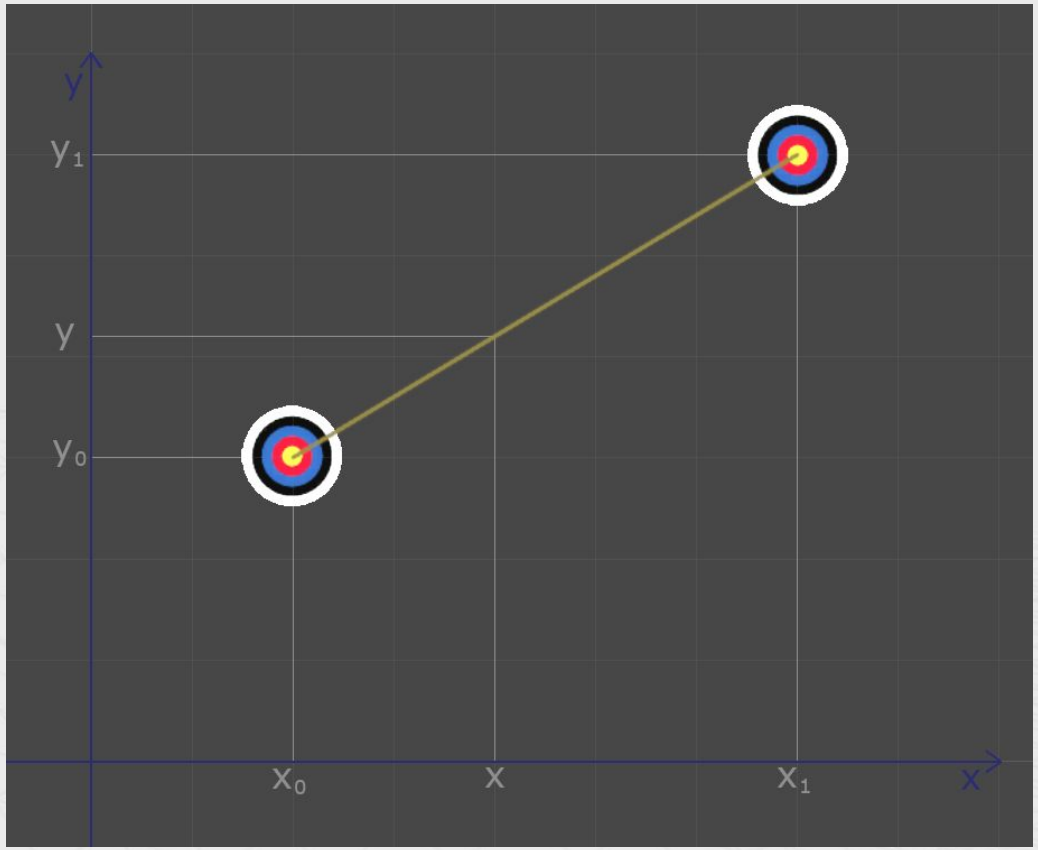
$$y = y_0 + x(y_1 - y_0)$$

A=(0,0); B=(1,5)

x = 0,25

y = 0 + 0,25 * (5-0)

y = 1,25



Lineární interpolace

-Proložení dvou bodů přímkou

```
/// <summary>
/// Lineárně interpoluje mezi hodnotami A a B podle t
/// </summary>
/// <param name="a"></param>
/// <param name="b"></param>
/// <param name="t"></param>
/// <returns></returns>
public float LinearInterpolate(float a, float b, float t)
{
    return a + t * (b - a);
}
```

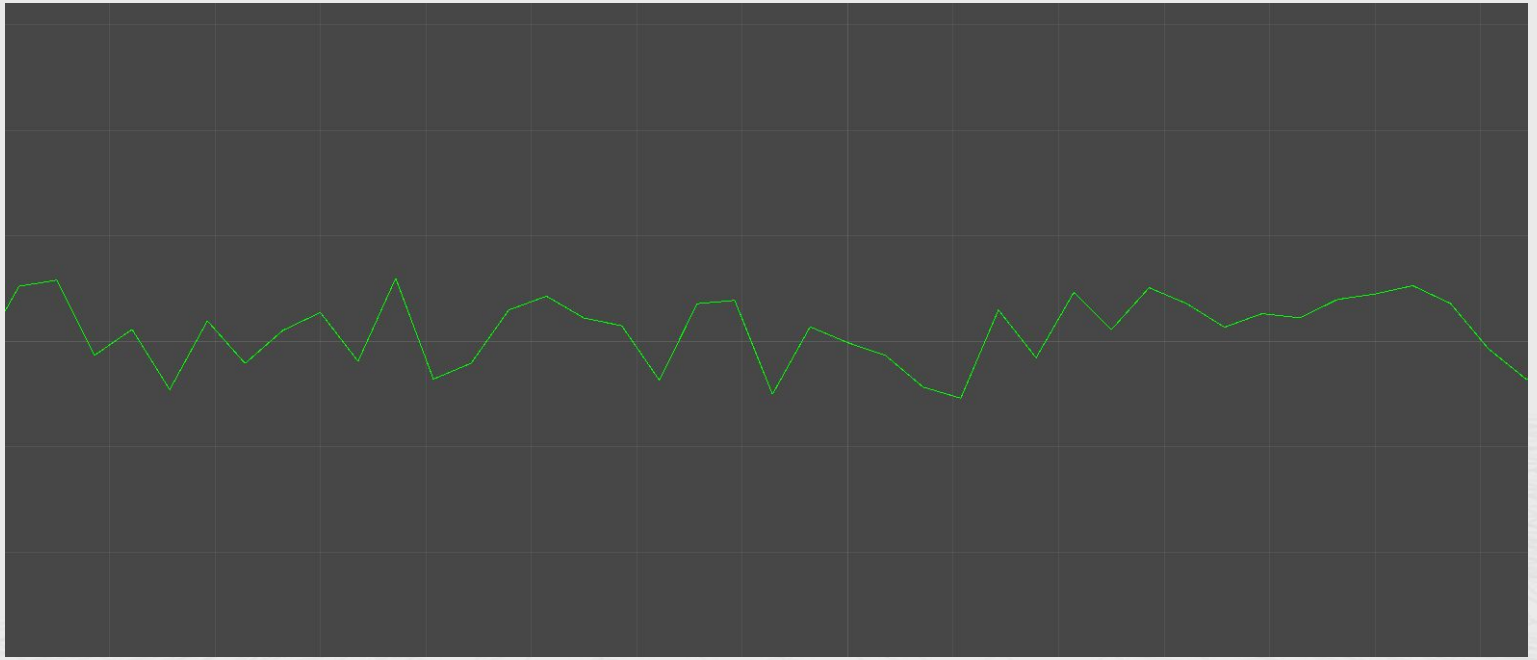
Spojitý šum (Lineárně interpolovaný)

-Lineární interpolace mezi náhodně vybranými hodnotami v určitém intervalu ⟨-6,6⟩

Spojitý šum (Lineárně interpolovaný)

-Lineární interpolace mezi náhodně vybranými hodnotami v určitém intervalu ⟨-6,6⟩

Spojitý šum (Lineárně interpolovaný)



-Lineární interpolace mezi náhodně vybranými hodnotami v určitém intervalu ⟨-6,6⟩

Jednoduchý příklad z praxe

-Náhodně pohybující se terč

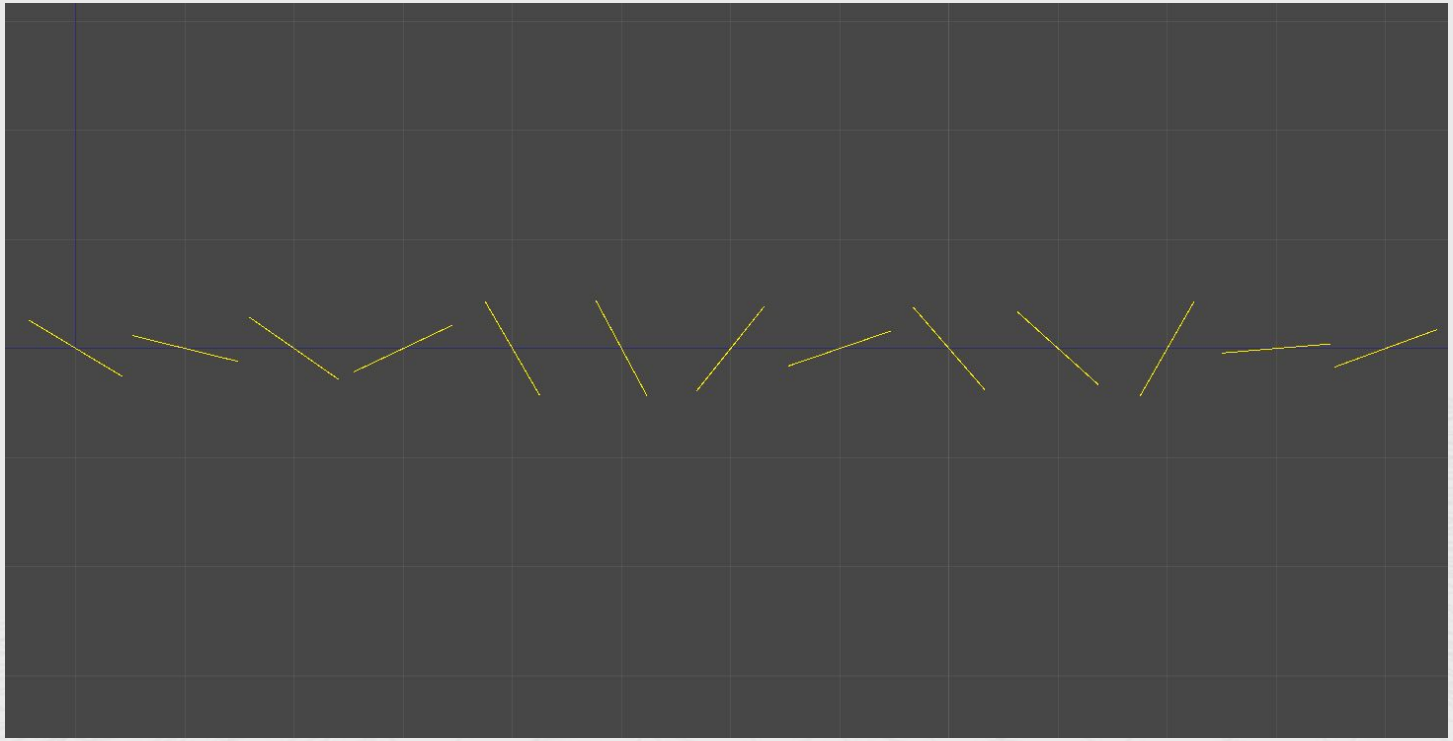
-Aby byl pohyb přirozenější / hezčí / hladký

Jednoduchý příklad z praxe

-Náhodně pohybující se terč

-Aby byl pohyb přirozenější / hezčí / hladký - (Gradientový spojitý šum)

Gradientový spojitý šum



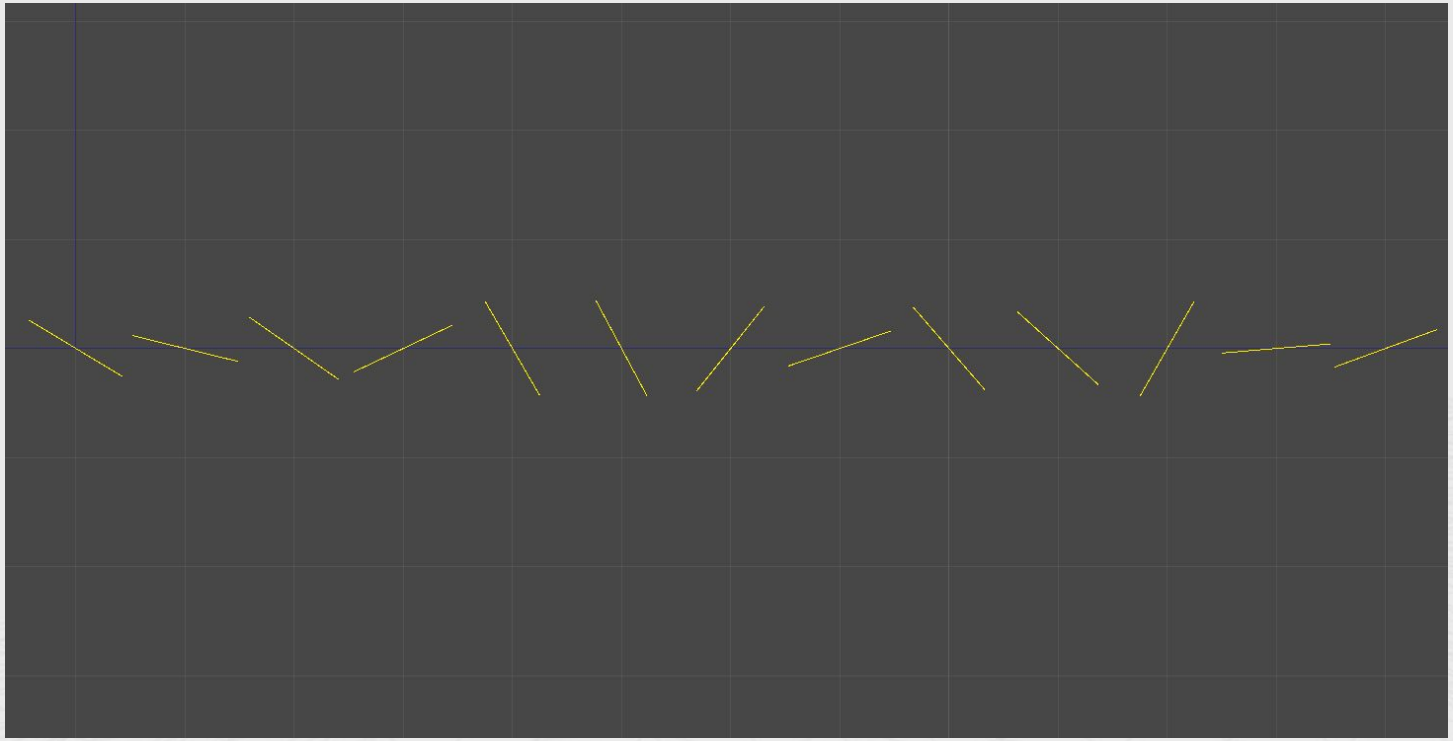
-Náhodně zvolené gradienty v intervalu ⟨-2,2⟩

Gradientový spojitý šum

```
// Array náhodných gradientů
private float[] _gradients = new float[kNumberOfGradients];

private const int kNumberOfGradients = 256;

// Awake se volá jen jednou
public void Awake()
{
    //Generování náhodných gradientů
    for (int i = 0; i < kNumberOfGradients; i++)
    {
        _gradients[i] = Random.Range(-2f, 2f);
    }
}
```

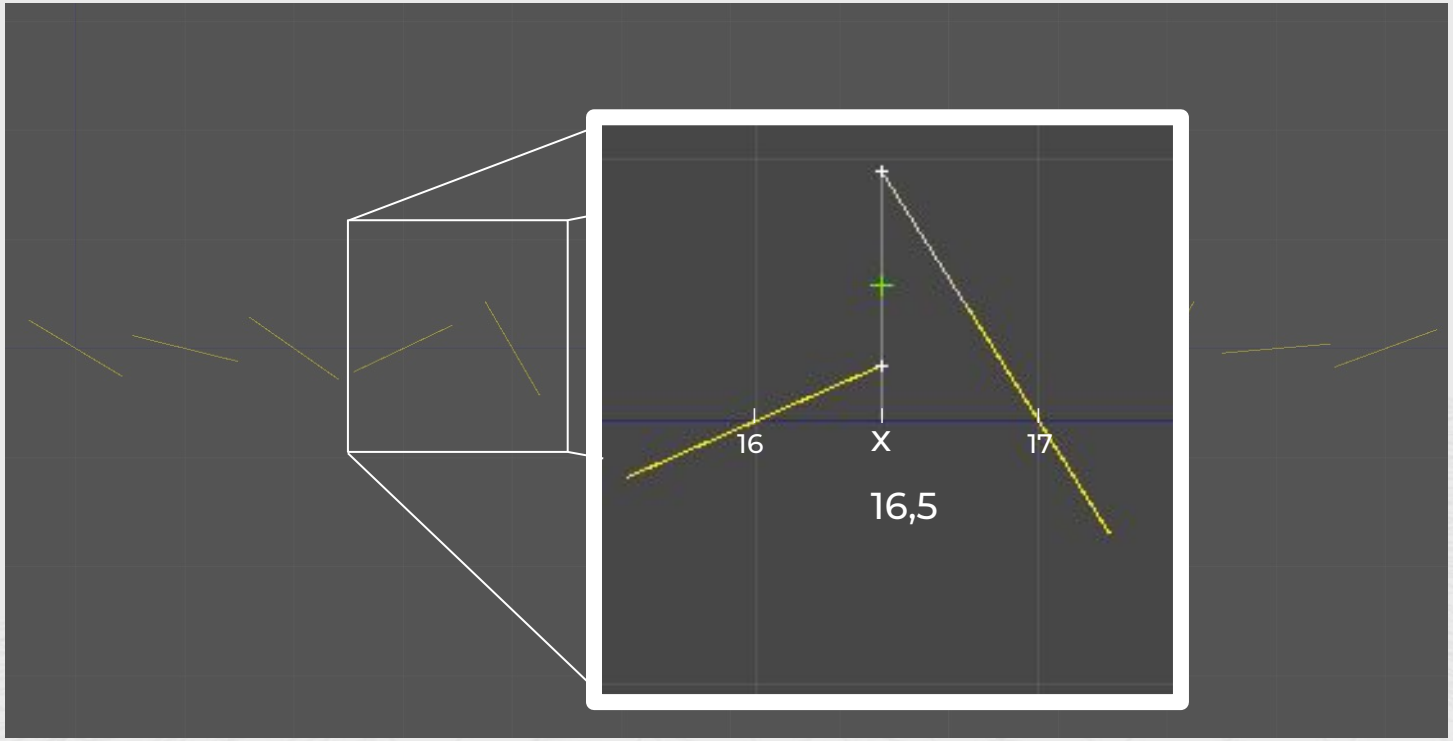
-Náhodné gradienty

Gradientový spojitý šum



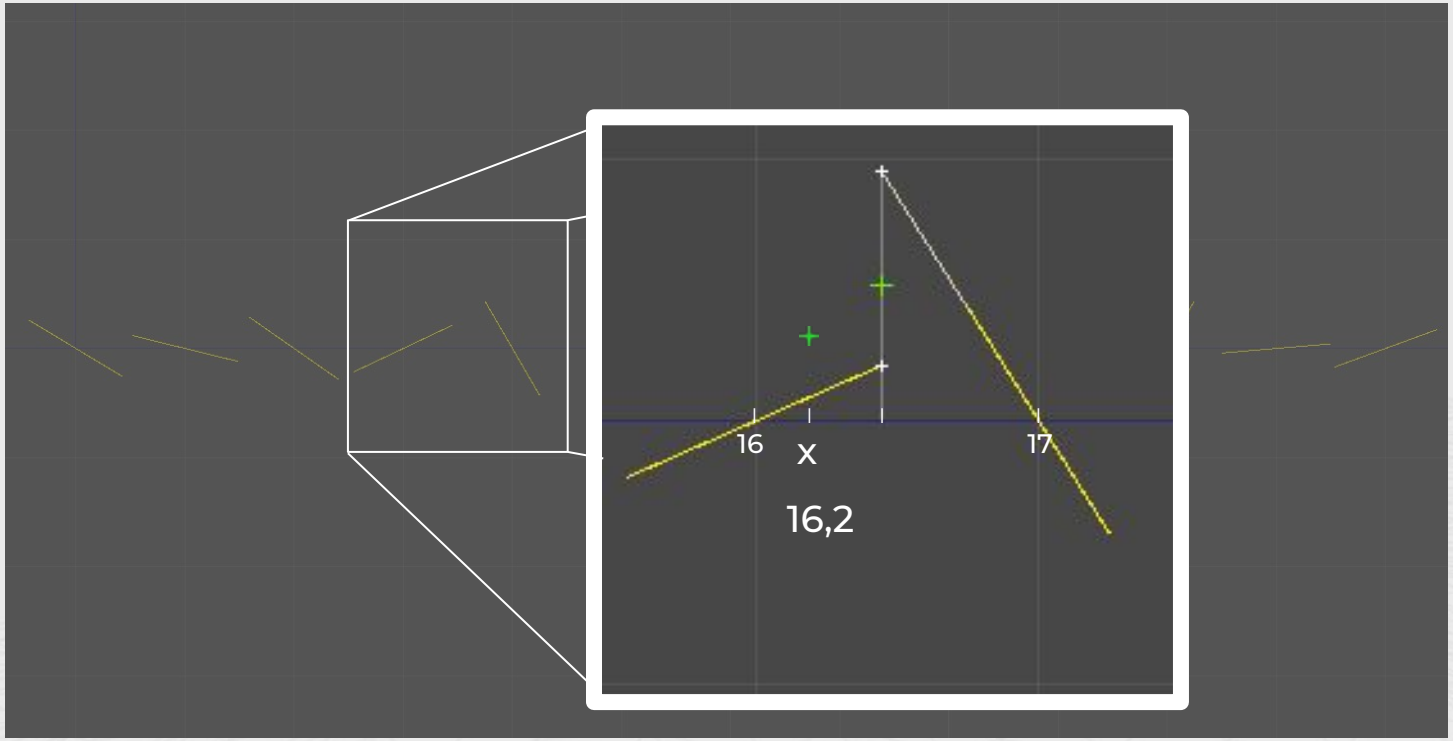
-Náhodně zvolené gradienty v intervalu ⟨-2,2⟩

Gradientový spojitý šum



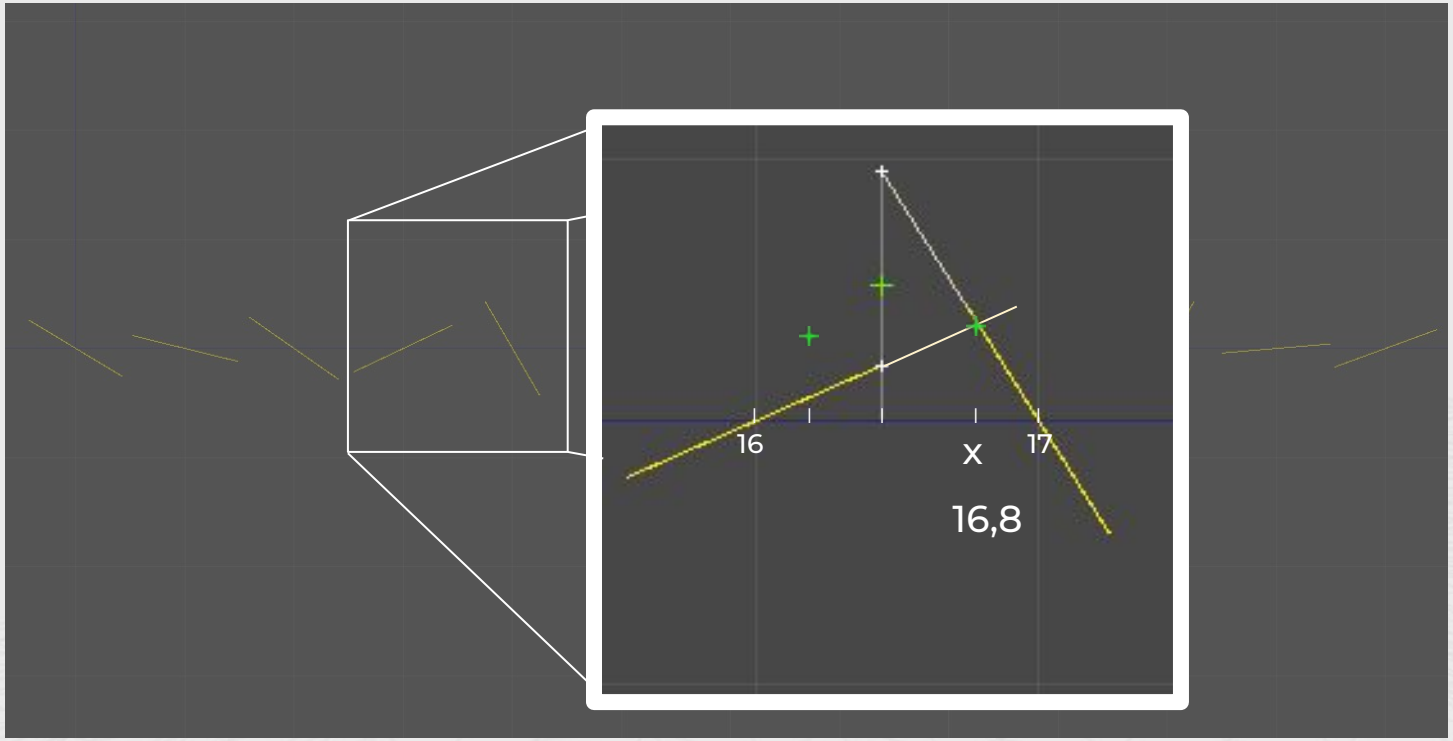
-Náhodně zvolené gradienty v intervalu ⟨-2,2⟩

Gradientový spojitý šum



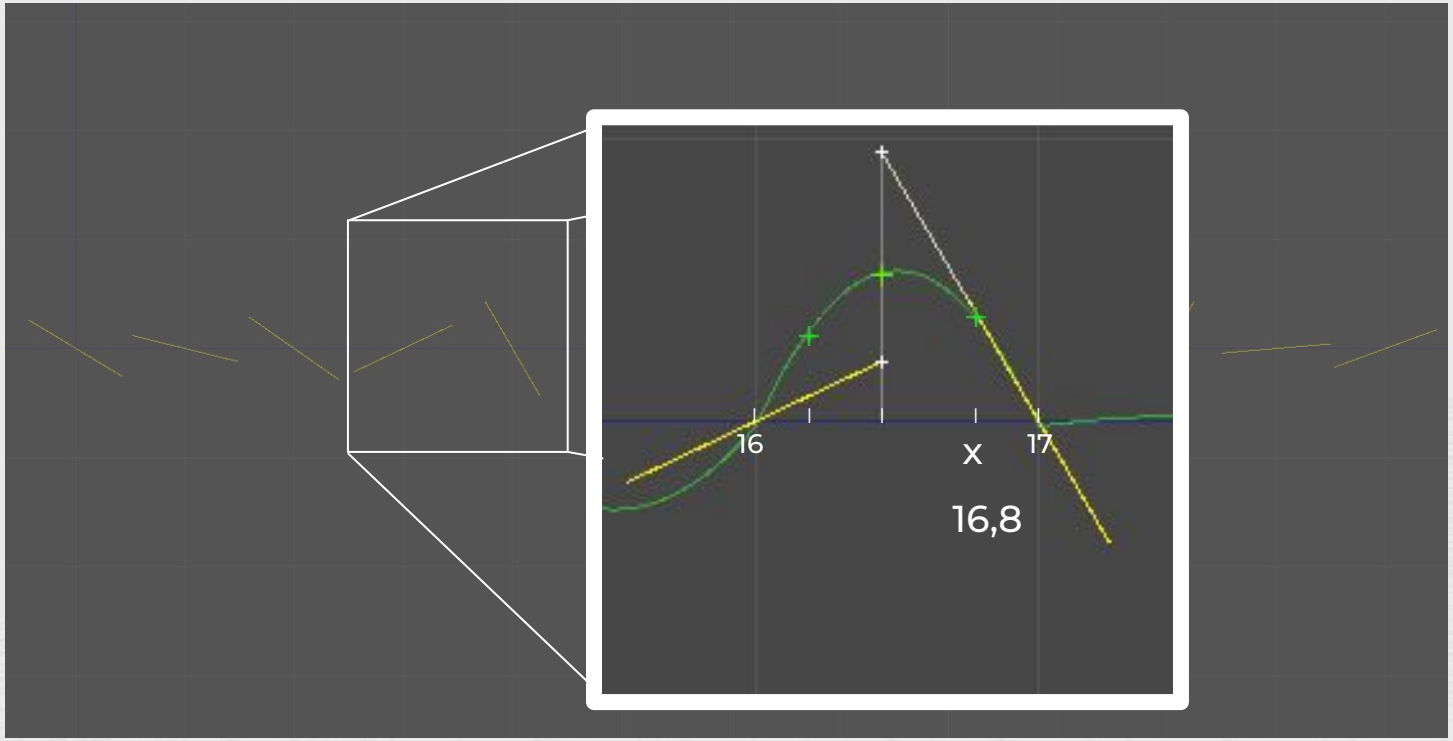
-Náhodně zvolené gradienty v intervalu ⟨-2,2⟩

Gradientový spojitý šum



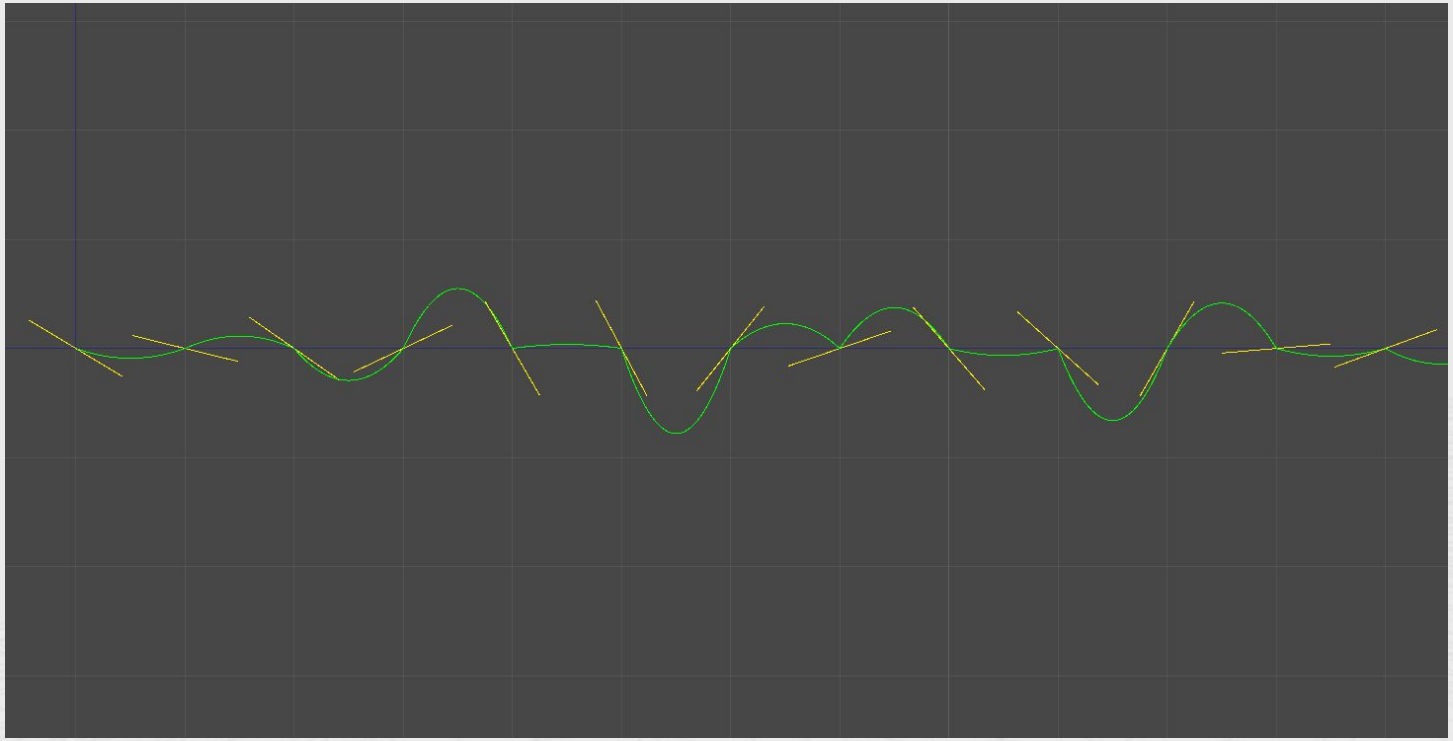
-Náhodně zvolené gradienty v intervalu ⟨-2,2⟩

Gradientový spojitý šum



-Náhodně zvolené gradienty v intervalu ⟨-2,2⟩

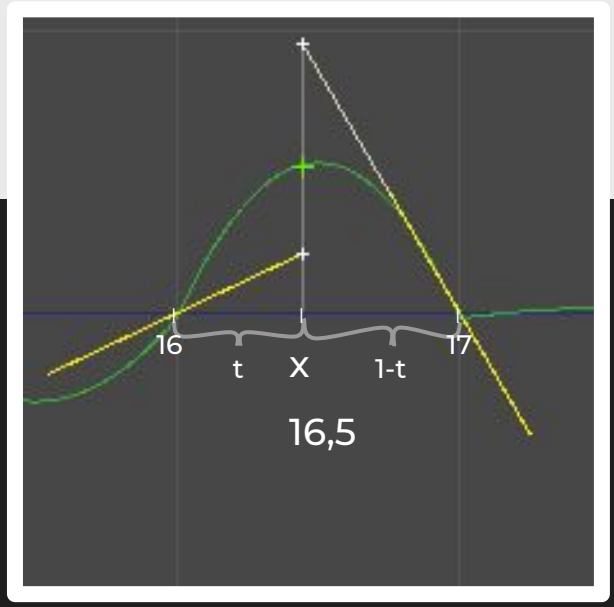
Gradientový spojitý šum



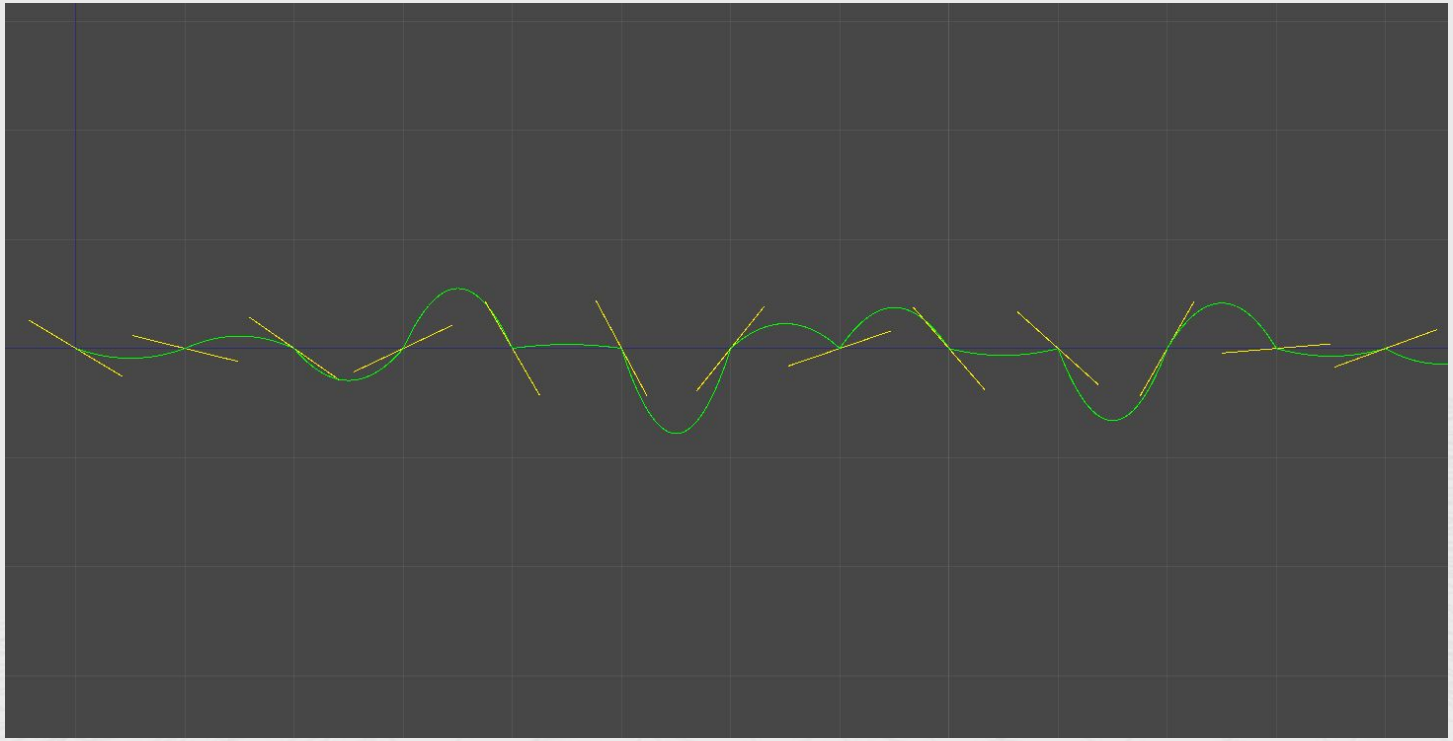
-Lineární interpolace mezi hodnotami přímek

Gradientový spojitý šum

```
/// <summary>
/// Vrací hodnotu šumu v čase
/// </summary>
/// <param name="time">čas</param>
/// <returns></returns>
public float GetNoiseAt(float time)
{
    int firstIndex = (int)time;
    float t = (time - firstIndex);

    float f0 = _gradients[firstIndex % kNumberOfGradients] * t;
    float f1 = -_gradients[(firstIndex + 1) % kNumberOfGradients] * (1 - t);

    return LinearInterpolate(f0, f1, t);
}
```



Gradientový spojitý šum



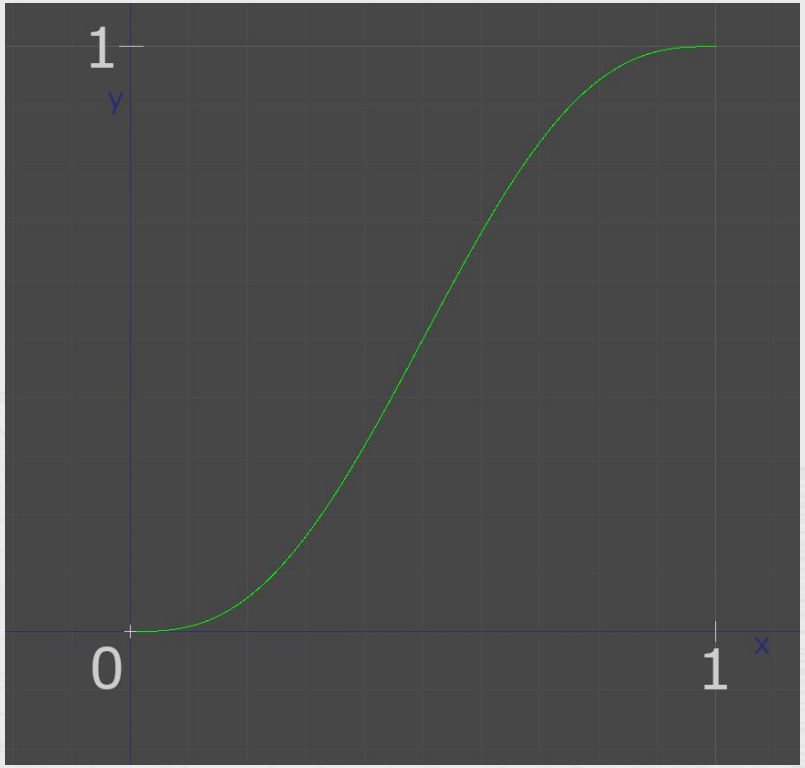
-Lineární interpolace mezi hodnotami přímek

Gradientový spojitý šum

$$6x^5 - 15x^4 + 10x^3$$

-První derivace v bodech x=0 a x=1 je rovna 0

(Body podezřelé z extrému)



Gradientový spojitý šum

```
/// <summary>
/// Vrací hodnoty polynomu 5. stupne v čase t
/// </summary>
/// <param name="t"></param>
/// <returns></returns>
public float SmoothInterpolate(float t)
{
    return t * t * t *(t * (t * 6 - 15) + 10);
}
```

$$6x^5 - 15x^4 + 10x^3$$

Gradientový spojitý šum

```
/// <summary>
/// Vrací hodnotu šumu v čase
/// </summary>
/// <param name="time">čas</param>
/// <returns></returns>
public float GetNoiseAt(float time)
{
    int firstIndex = (int)time;
    float t = (time - firstIndex);

    float f0 = _gradients[firstIndex % kNumberOfGradients] * t;
    float f1 = -_gradients[(firstIndex + 1) % kNumberOfGradients] * (1 - t);

    return LinearInterpolate(f0, f1, t);
}
```

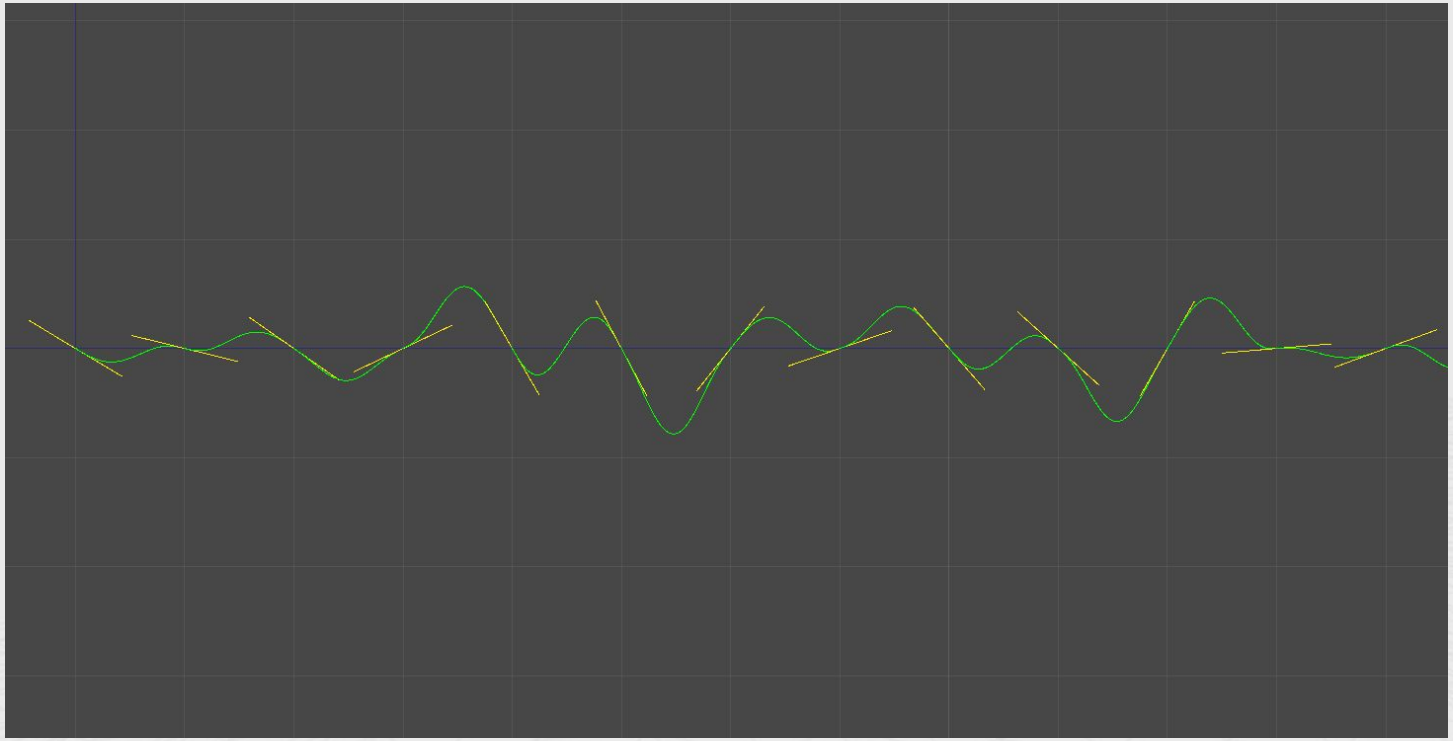
-Lineární interpolace mezi hodnotami přímek

Gradientový spojitý šum

```
/// <summary>
/// Vrací hodnotu šumu v čase
/// </summary>
/// <param name="time">čas</param>
/// <returns></returns>
public float GetNoiseAt(float time)
{
    int firstIndex = (int)time;
    float t = (time - firstIndex);

    float f0 = _gradients[firstIndex % kNumberOfGradients] * t;
    float f1 = -_gradients[(firstIndex + 1) % kNumberOfGradients] * (1 - t);

    return LinearInterpolate(f0, f1, SmoothInterpolate(t));
}
```

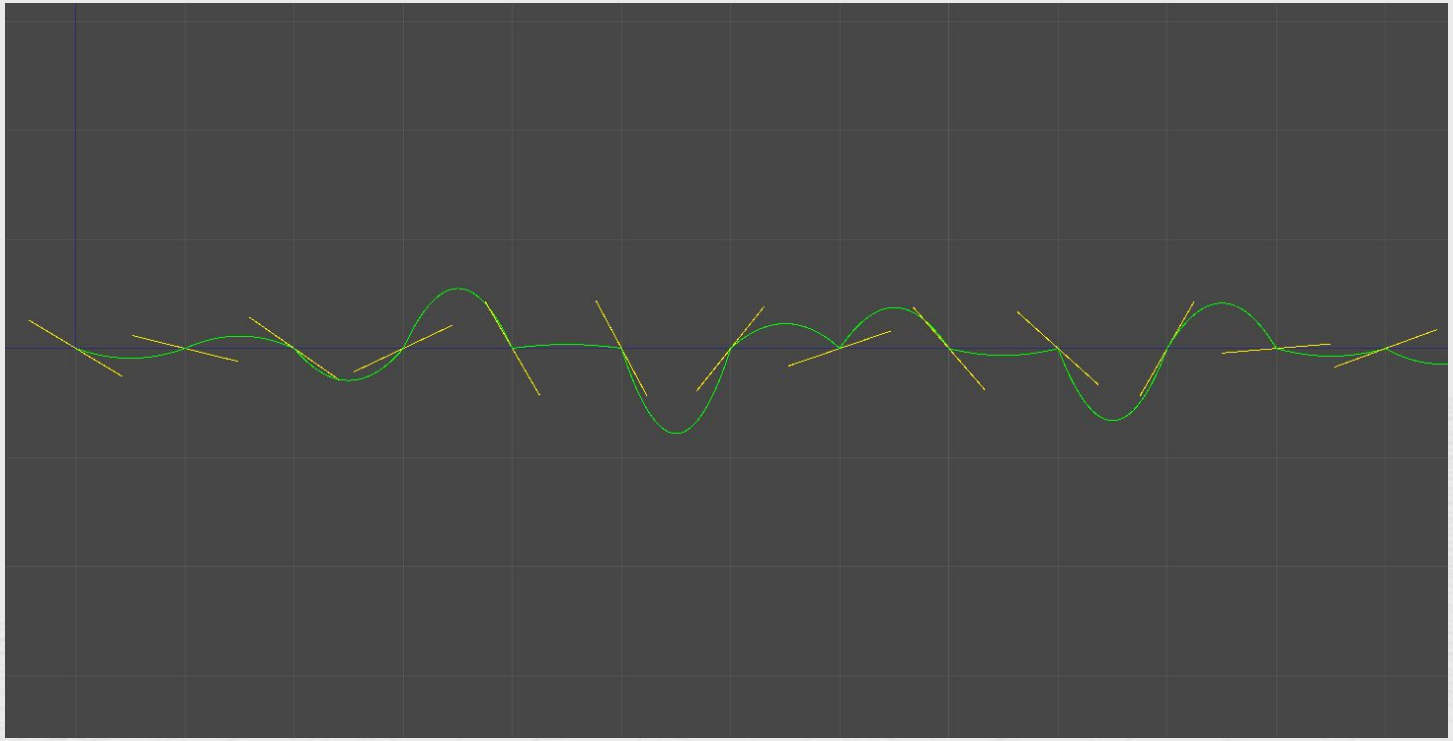
-Interpolace polynomem $6x^5 - 15x^4 + 10x^3$

Gradientový spojitý šum



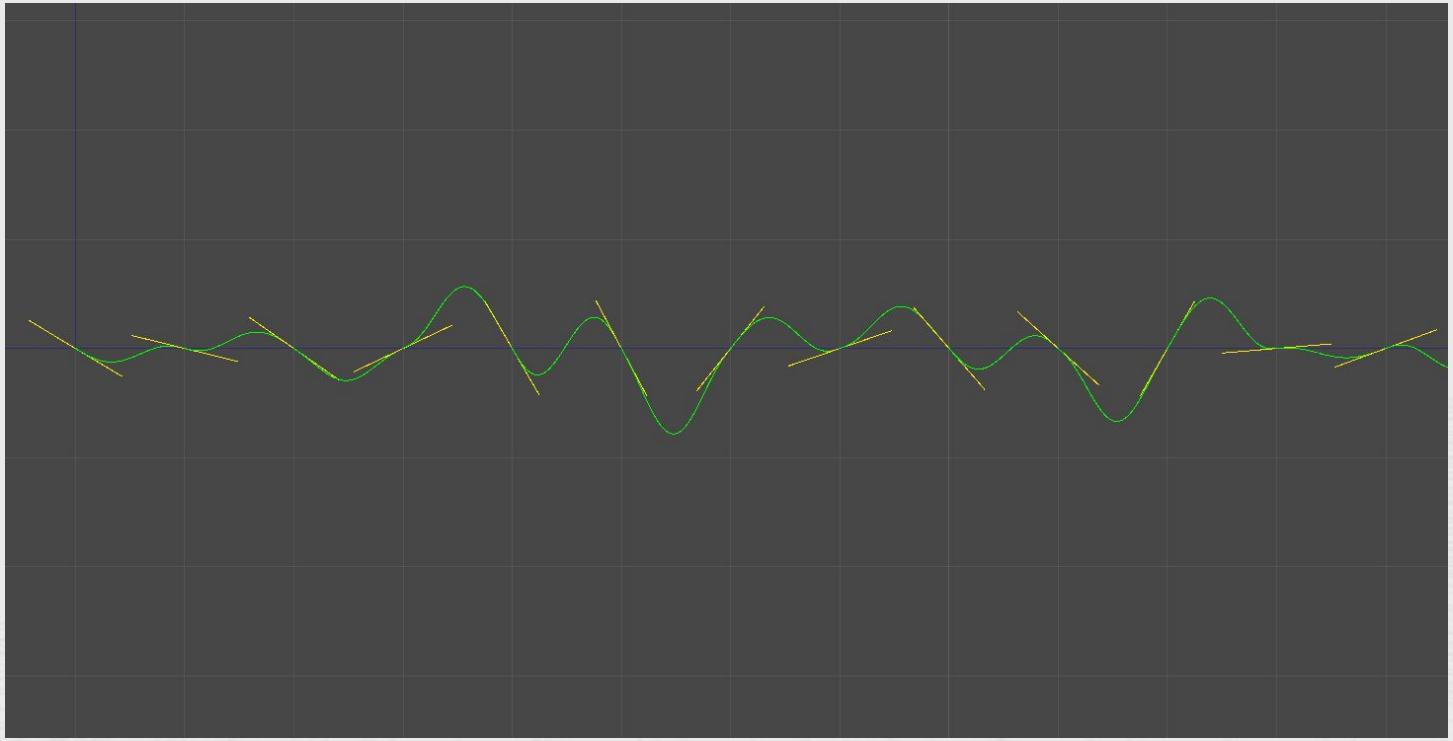
-Interpolace polynomem $6x^5 - 15x^4 + 10x^3$

Gradientový spojitý šum



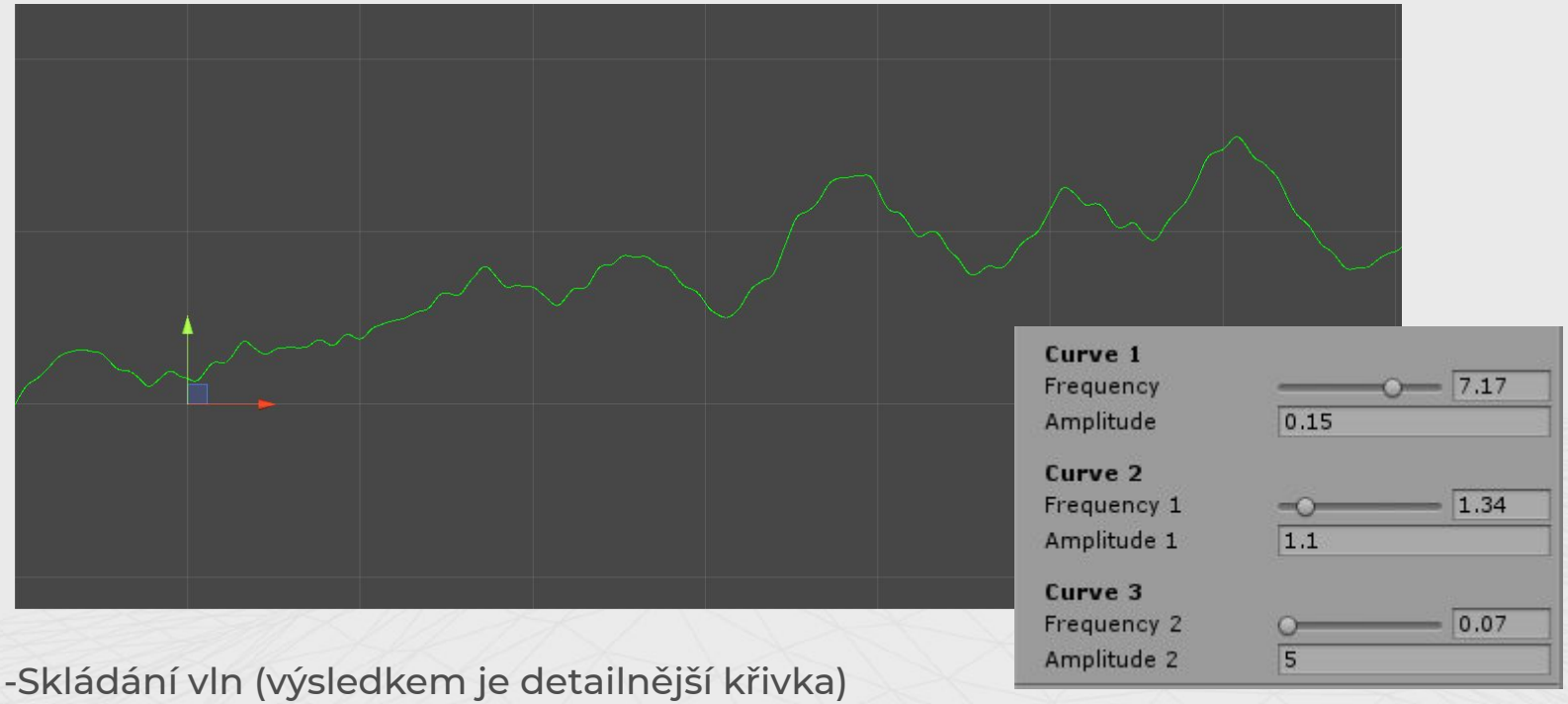
-Lineární interpolace mezi hodnotami přímek

Gradientový spojitý šum



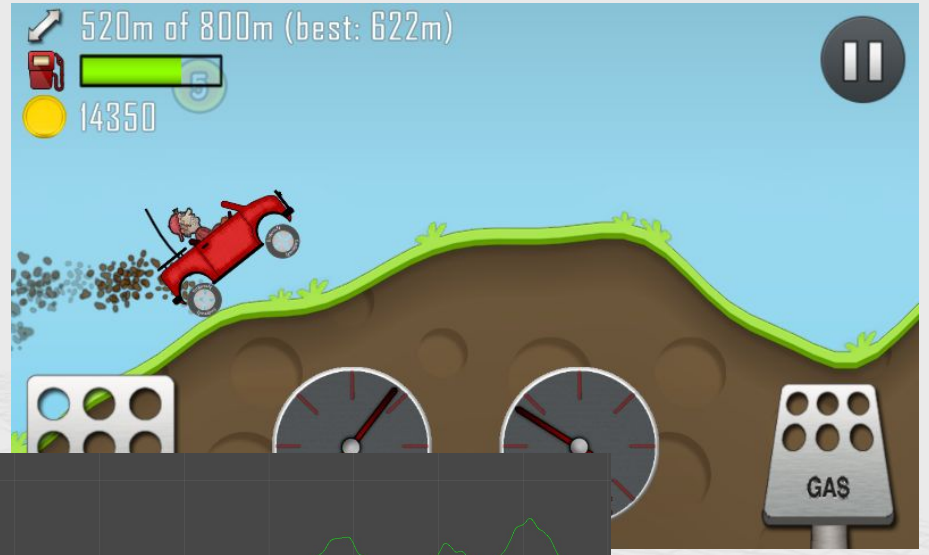
-Interpolace polynomem $6x^5 - 15x^4 + 10x^3$

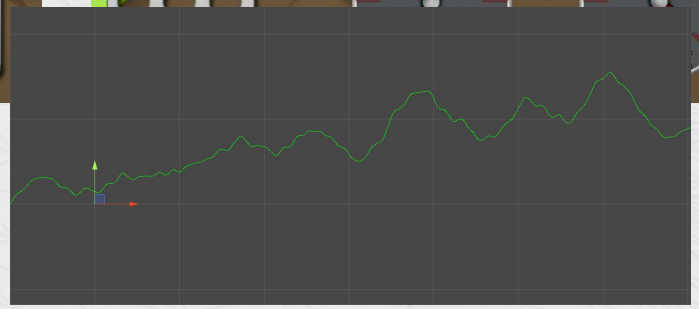
Gradientový spojitý šum



-Skládání vln (výsledkem je detailnější křivka)

Gradientový spojitý šum

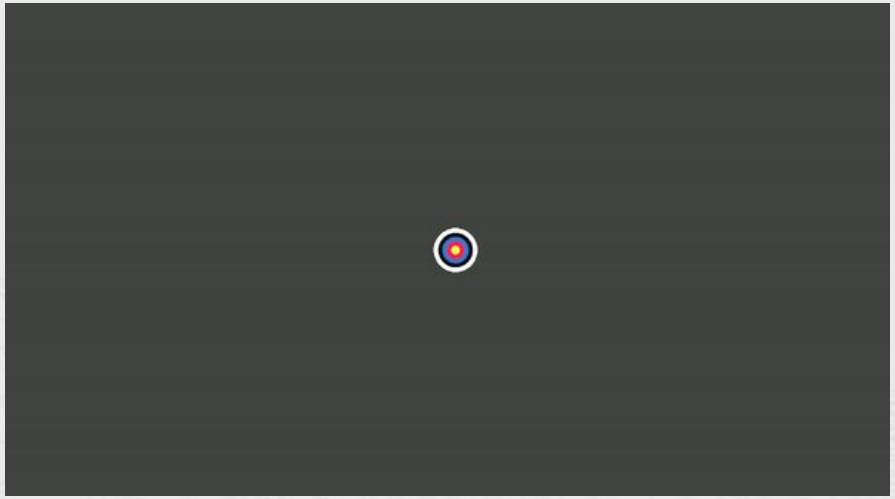




-Skládání vln (výsledkem je detailnější křivka)

Gradientový spojitý šum

-Spojitý gradientový šum



-Lineární interpolace (náhodných hodnot)

Generátor terénu

-Ukázka implementace generátoru terénu (podobně jako ve hře "They are billions")

Generátor terénu

-Jednoduchý generátor terénu (mřížka 100x100)

-Každé políčko mřížky může být jeden z následujících typů terénu:

Hlína, Skála, Voda nebo Les

Perlinův šum

-Ken Perlin při práci na filmu Tron (1982)

-Motivace: Přirozeně vypadající textury

-Implementován může být ve více dimenzích

Perlinův šum (Unity)

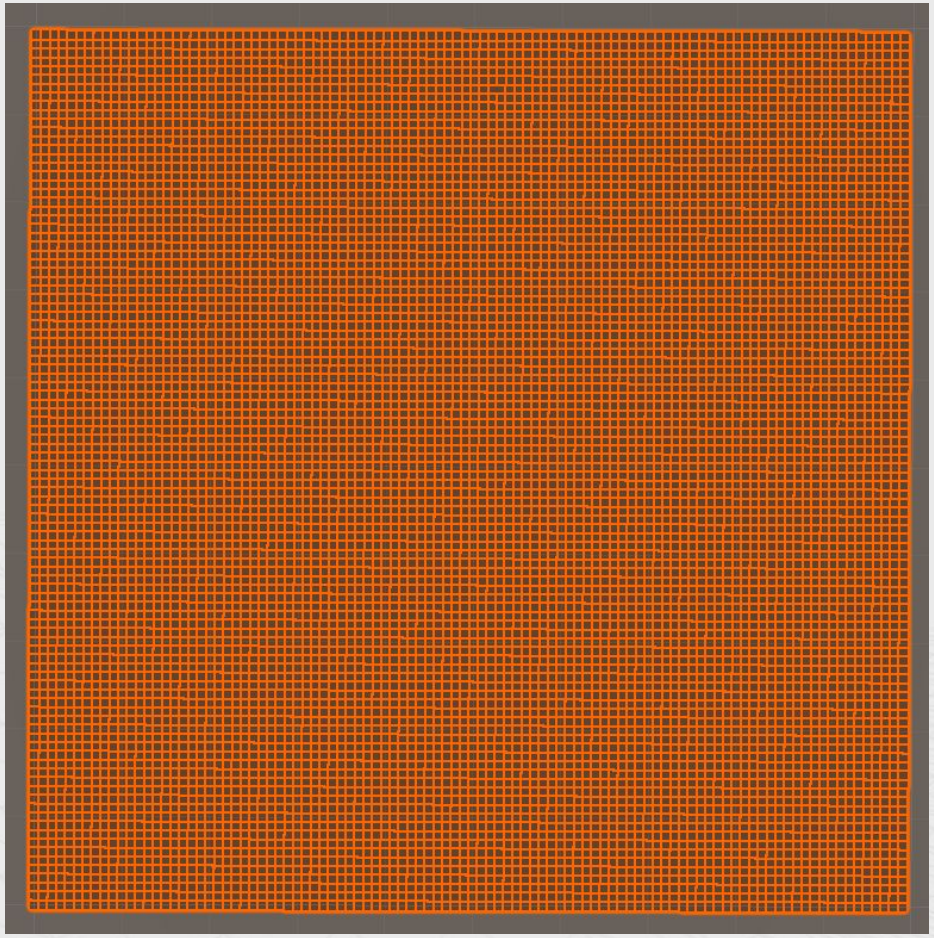
-Definován ve dvou dimenzích

-Stejný pro všechny hry v Unity

Mřížka mapy

Zadani:

Mřížka 100x100



Perlinův šum (Unity)

```
public class MapGenerator : MonoBehaviour
{
    /// <summary>
    /// Vektory pozic na kterých se na naší mapě nachází hlína
    /// </summary>
    private List<Vector2> GroundPositions = new List<Vector2>(kMapXSize * kMapZSize);
    /// <summary>
    /// Třída, která se stará o vizualizaci bodů na mapě
    /// </summary>
    public VisualsGenerator _visualsGenerator = default;

    //Velikost mapy na ose x
    public const int kMapXSize = 100;
    //Velikost mapy na ose z (Generujeme mapu do prostoru na osách X a Z)
    public const int kMapZSize = 100;
```

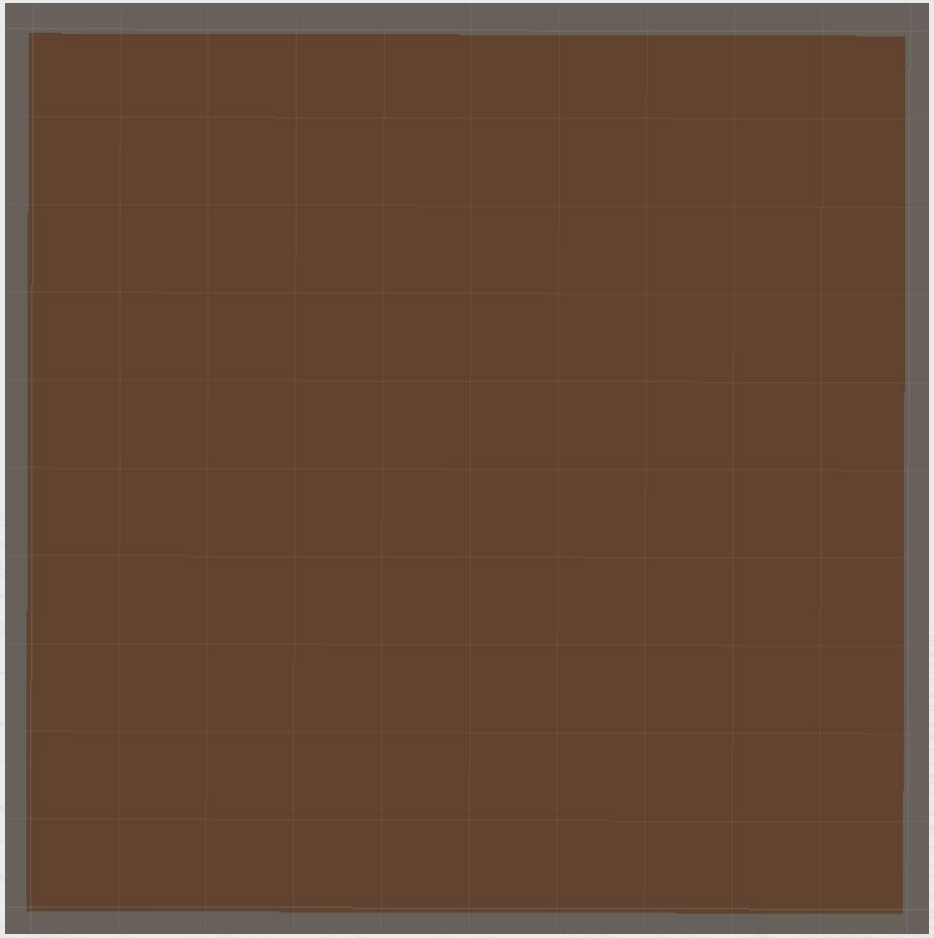
Perlinův šum (Unity)

```
/// <summary>
/// Vytvoří vektory, které reprezentují jednotlivé pozice v mřížce naší mapy
/// </summary>
private void CreateMapPositionGrid()
{
    for (int x = 0; x < kMapXSize; x++)
    {
        for (int y = 0; y < kMapZSize; y++)
        {
            GroundPositions.Add(new Vector2(x, y));
        }
    }

}
```

Perlinův šum (Unity)

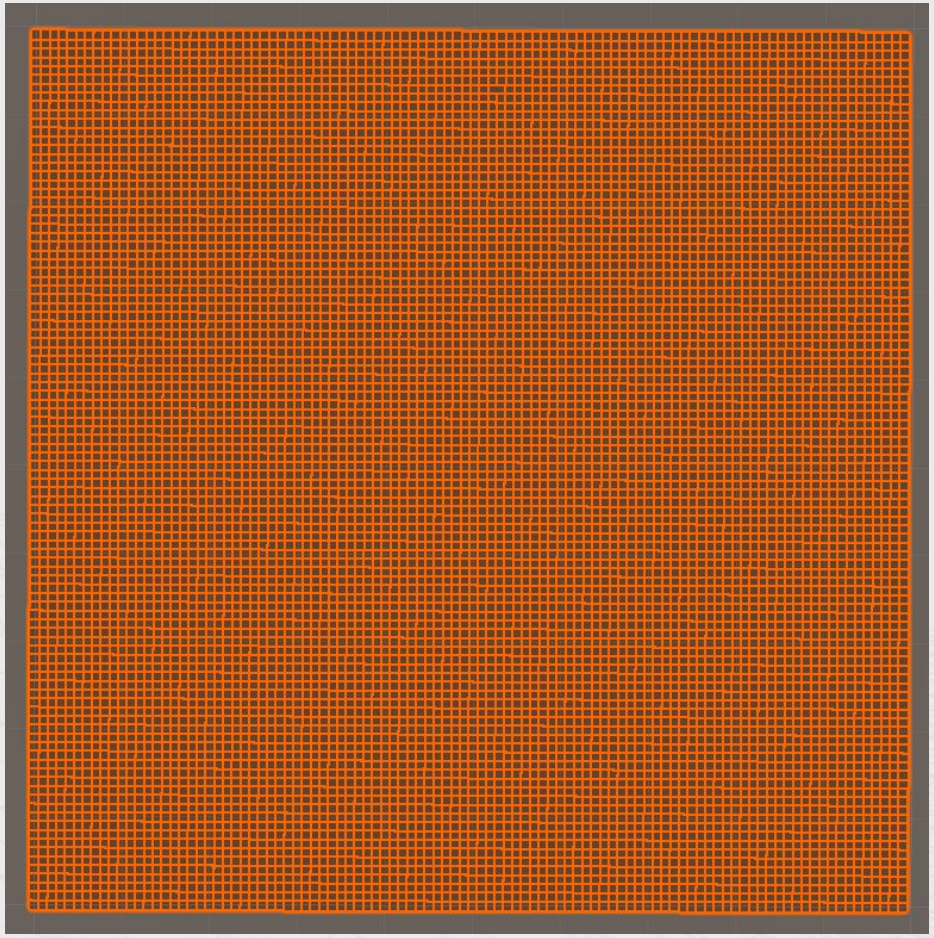
```
private void Awake()
{
    GenerateMap();
}

/// <summary>
/// Vytvoří mapu
/// </summary>
public void GenerateMap()
{
    CreateMapPositionGrid();

    _visualsGenerator.GenerateVisuals(GroundPositions, VisualsGenerator.ETerrainType.Ground);
}
```

Mřížka mapy



Mřížka mapy

100x100



Perlinův šum (Unity)

-Definován ve dvou dimenzích

-Stejný pro všechny hry v Unity

Mathf.PerlinNoise(x, y)

-Vrací hodnoty od 0 do 1

Generátor terénu

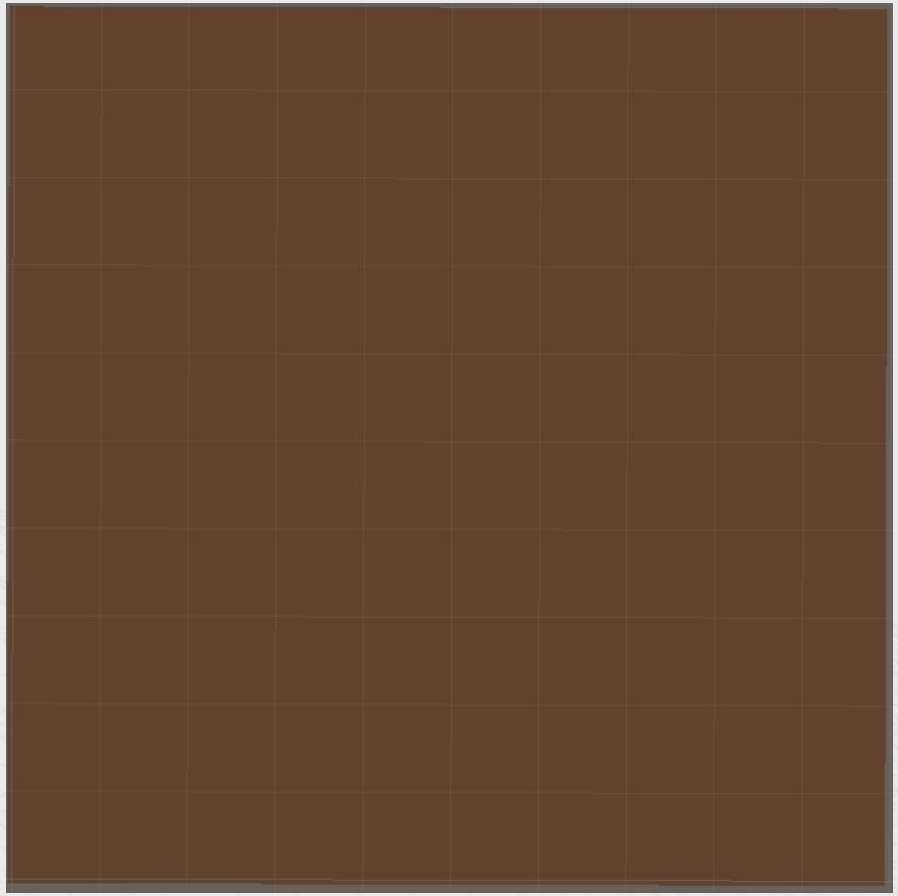
```
private void FillList(List<Vector2> list)
{
    for (int x = 0; x < kMapXSize; x++)
    {
        for (int y = 0; y < kMapZSize; y++)
        {
            float value = Mathf.PerlinNoise(x, y);
            Vector2 currentPos = new Vector3(x, y);

            if (value >= 0.5f)
            {
                list.Add(currentPos);
                GroundPositions.Remove(currentPos);
            }
        }
    }
}
```

Generátor terénu

```
/// <summary>
/// Vytvoří mapu
/// </summary>
public void GenerateMap()
{
    CreateMapPosiitonGrid();

    FillList(MountainPositions);

    _visualsGenerator.GenerateVisuals(MountainPositions, VisualsGenerator.ETerrainType.Mountain);
    _visualsGenerator.GenerateVisuals(GroundPositions, VisualsGenerator.ETerrainType.Ground);
}
```

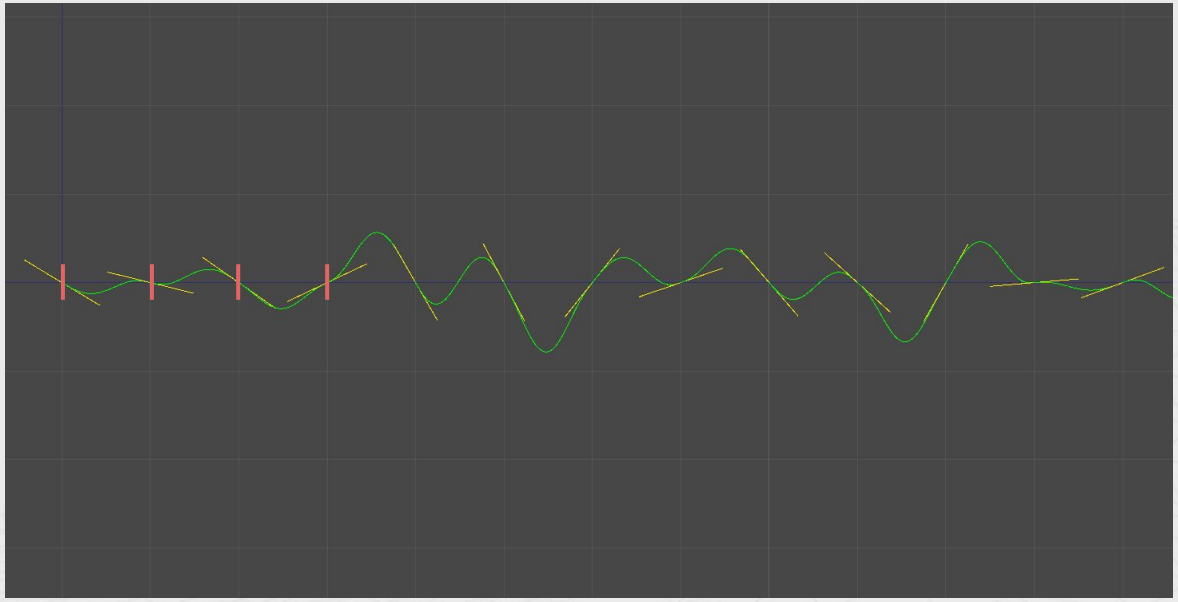
Generátor terénu

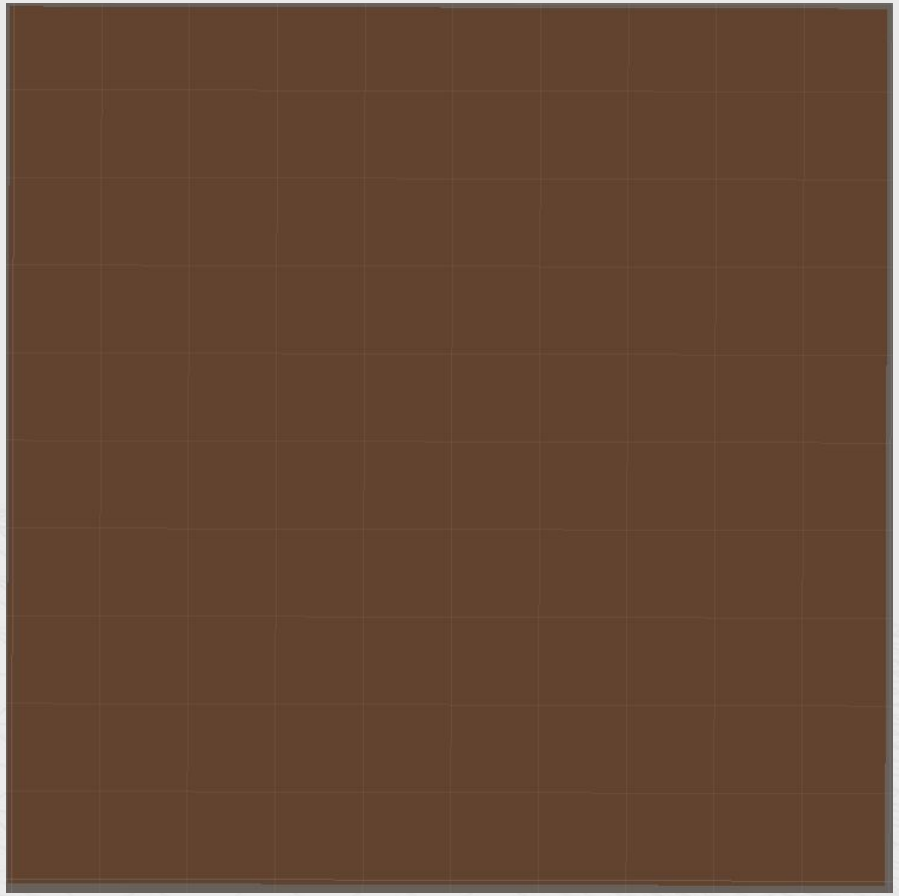
-Nic?



Generátor terénu

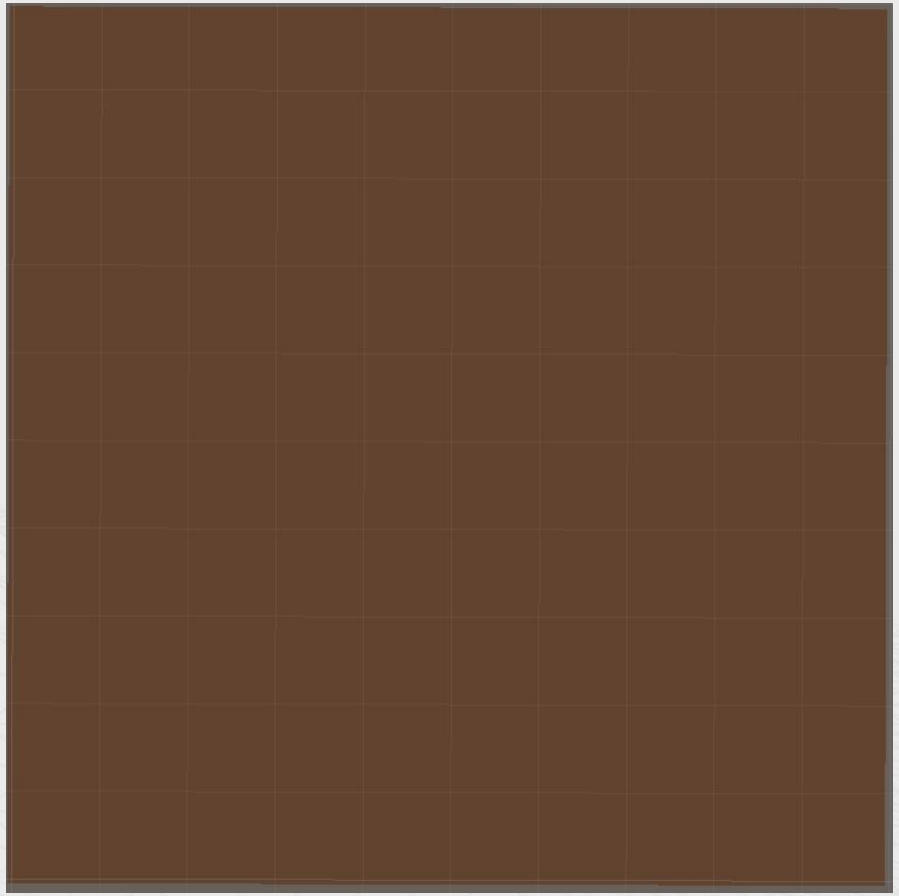
-Vždy stejná hodnota





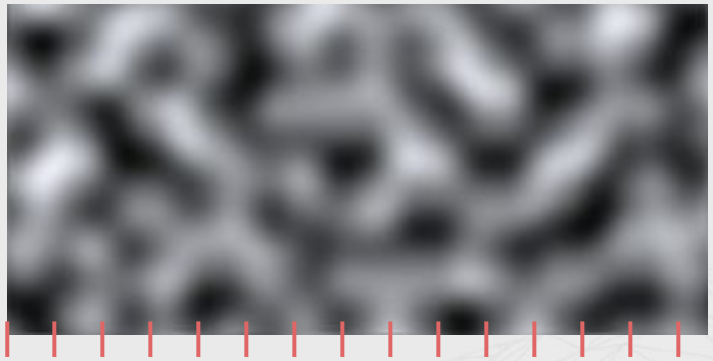
Generátor terénu

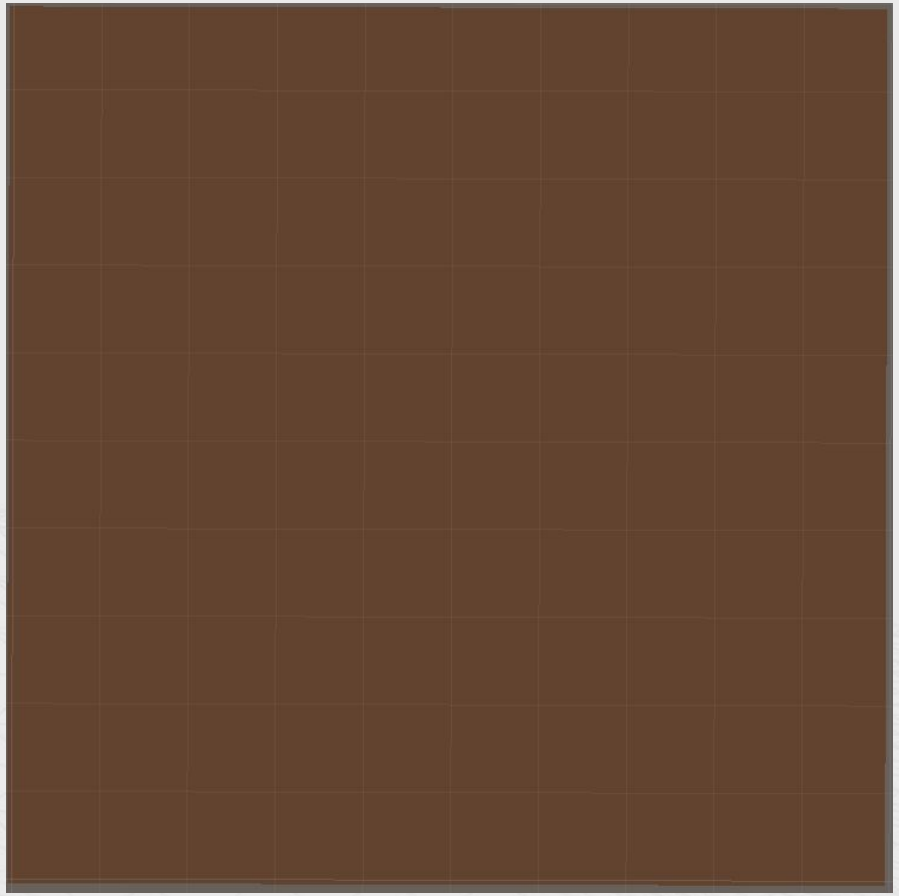
-Vybrané hodnoty jsou příliš daleko od sebe



Generátor terénu

-Vybrané hodnoty jsou příliš daleko od sebe



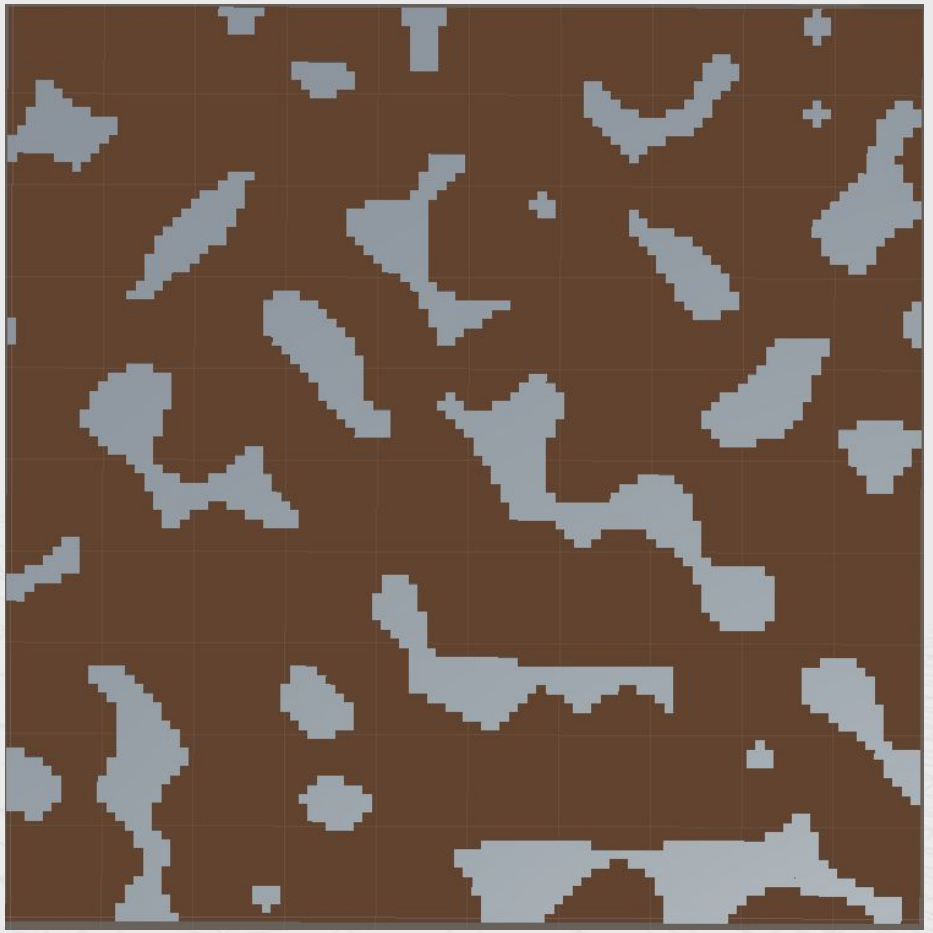


Generátor terénu

```
/// <param name="swapXandY"></param>
private void FillList(List<Vector2> list, float threshold)
{
    for (int x = 0; x < kMapXSize; x++)
    {
        for (int y = 0; y < kMapZSize; y++)
        {
            float value = Mathf.PerlinNoise(x / 10f, y / 10f);
            Vector2 currentPos = new Vector3(x, y);

            if (value >= threshold)
            {
                list.Add(currentPos);
                GroundPositions.Remove(currentPos);
            }
        }
    }
}
```

Generátor terénu

-Vybrané hodnoty blíž u sebe

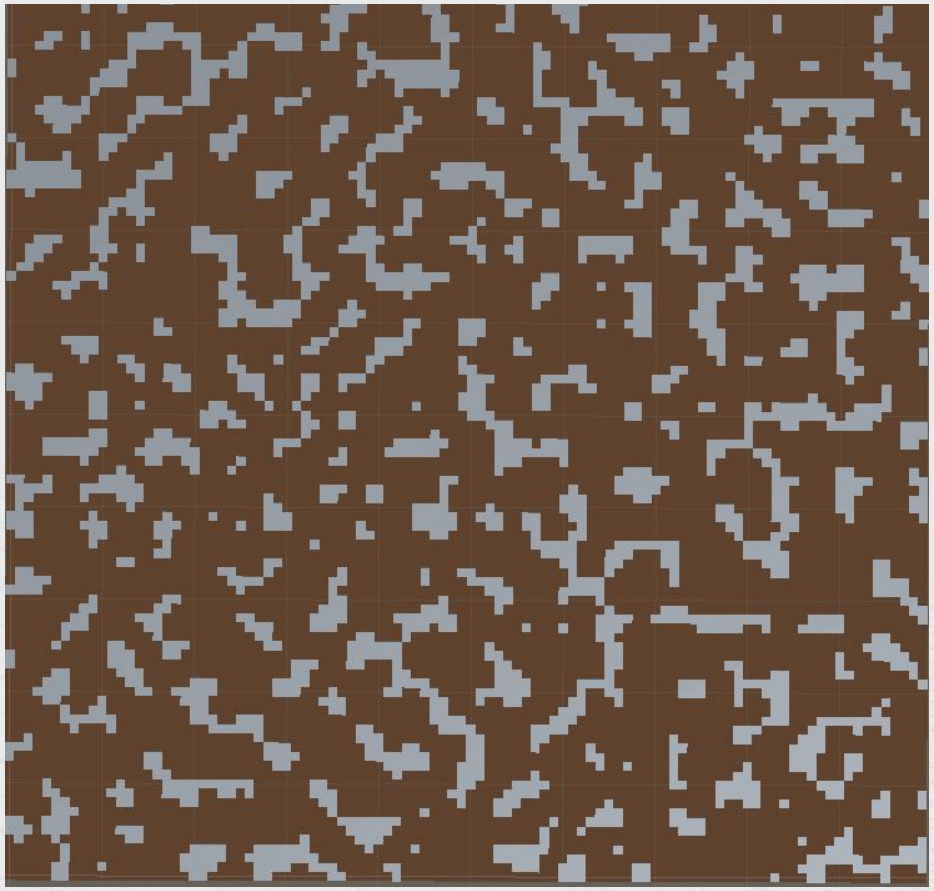


Generátor terénu

```
private void FillList(List<Vector2> list,float sampleRate, float threshold)
{
    for (int x = 0; x < kMapXSize; x++)
    {
        for (int y = 0; y < kMapZSize; y++)
        {
            float value = Mathf.PerlinNoise(x / sampleRate, y / sampleRate);
            Vector2 currentPos = new Vector3(x, y);

            if (value >= threshold)
            {
                list.Add(currentPos);
                GroundPositions.Remove(currentPos);
            }
        }
    }
}
```

Generátor terénu

-Efekt přibližování a oddalování



Generátor terénu

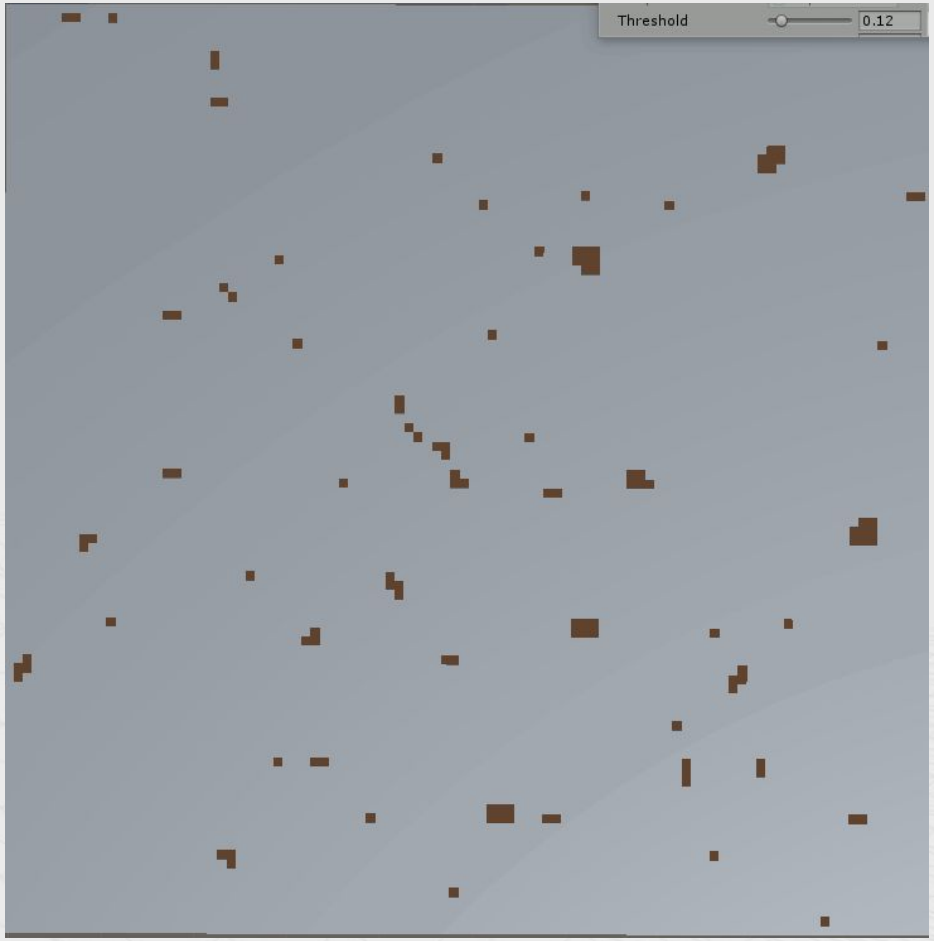
```
private void FillList(List<Vector2> list, float sampleRate, float threshold)
{
    for (int x = 0; x < kMapXSize; x++)
    {
        for (int y = 0; y < kMapZSize; y++)
        {
            float value = Mathf.PerlinNoise(x / sampleRate, y / sampleRate);
            Vector2 currentPos = new Vector3(x, y);

            if (value >= threshold)
            {
                list.Add(currentPos);
                GroundPositions.Remove(currentPos);
            }
        }
    }
}
```

Generátor terénu

PRÁH

-Vysoké hodnoty: méně terénu

-Nízké hodnoty: více terénu

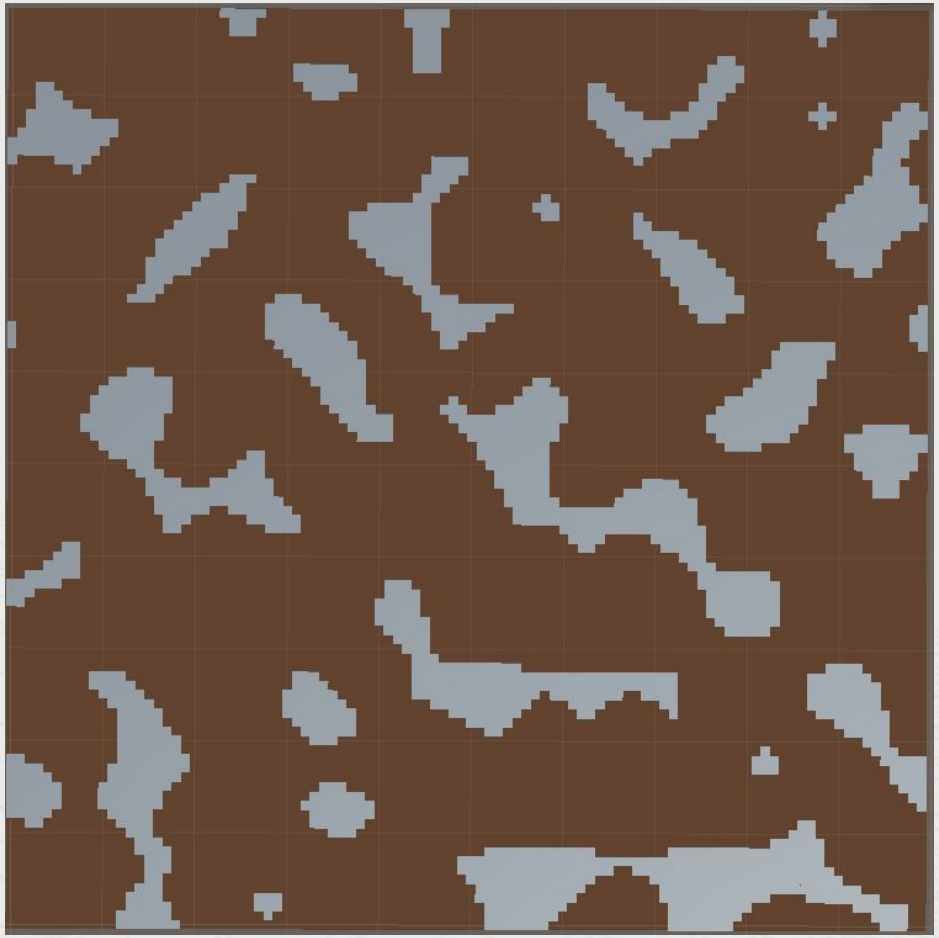


Generátor terénu

PRÁH

-Rozumné hodnoty

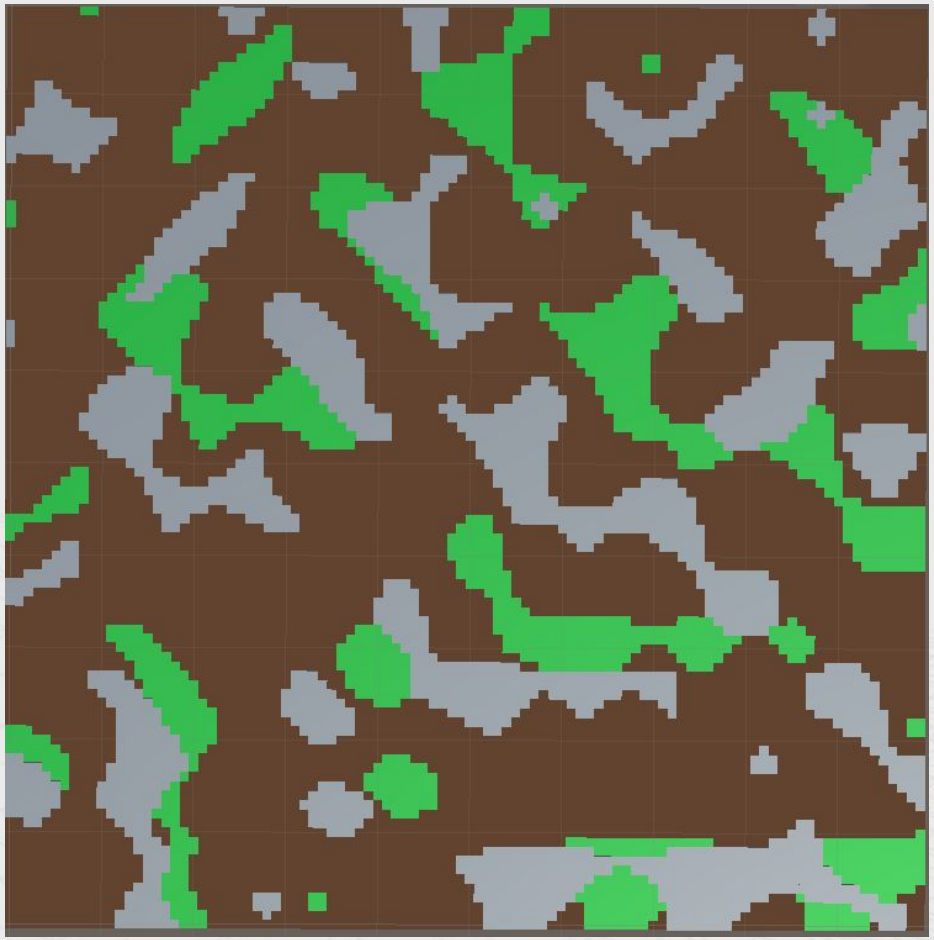
-Záleží na hře samotné



Generátor terénu

-Příliš podobné

-V některých případech i identické

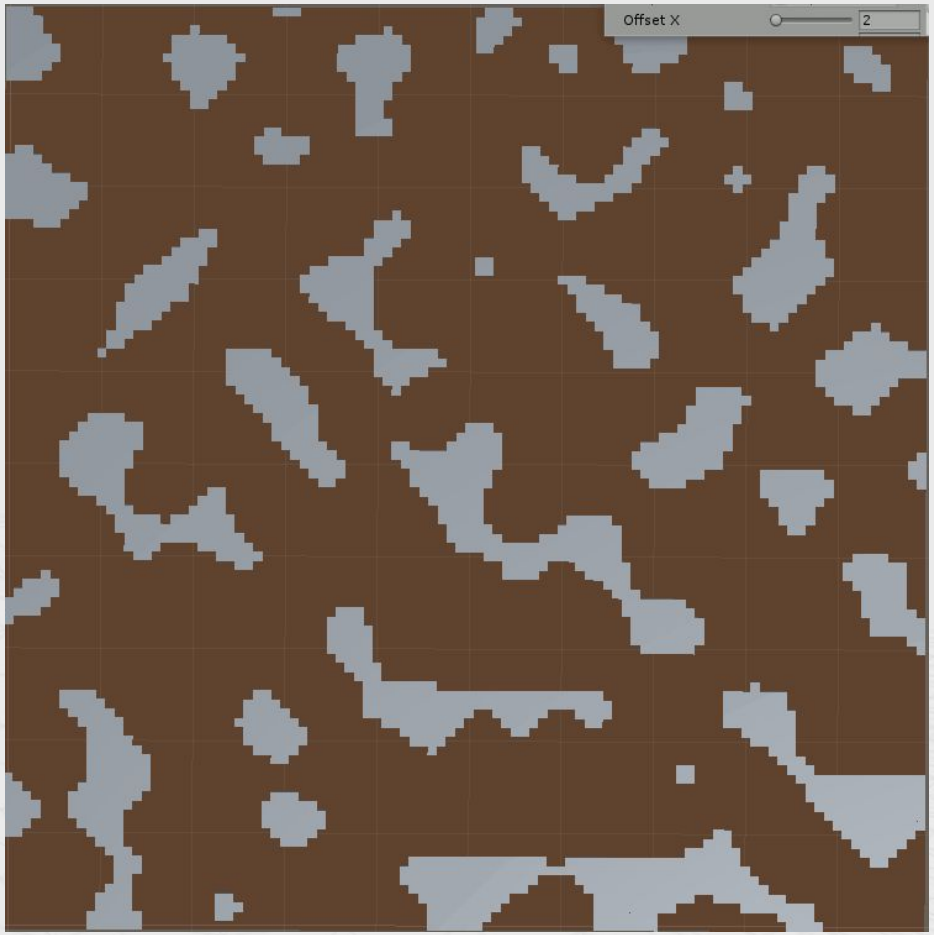


Generátor terénu

```
private void FillList(List<Vector2> list,float xOffset, float yOffset, float sampleRate, float threshold)
{
    for (int x = 0; x < kMapXSize; x++)
    {
        for (int y = 0; y < kMapZSize; y++)
        {
            float value = Mathf.PerlinNoise((x + xOffset) / sampleRate, (y + yOffset) / sampleRate);
            Vector2 currentPos = new Vector3(x, y);

            if (value >= threshold)
            {
                list.Add(currentPos);
                GroundPositions.Remove(currentPos);
            }
        }
    }
}
```

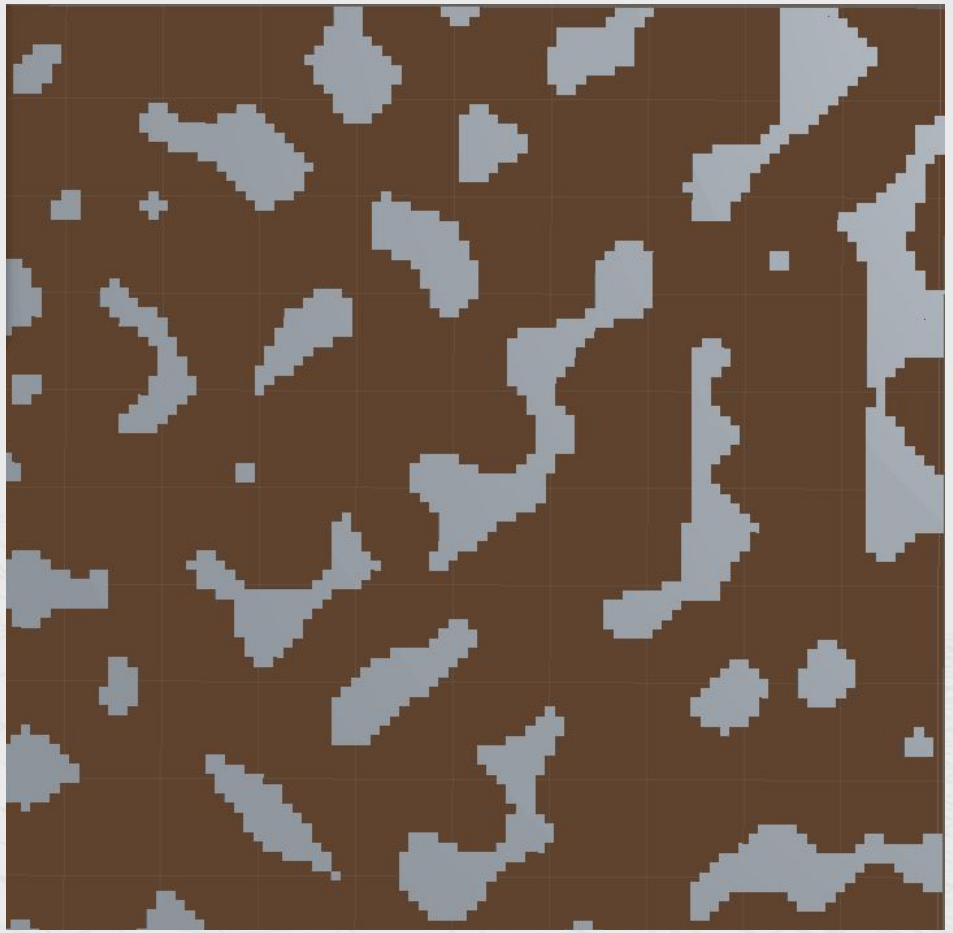
Generátor terénu

Posouvání po ose x



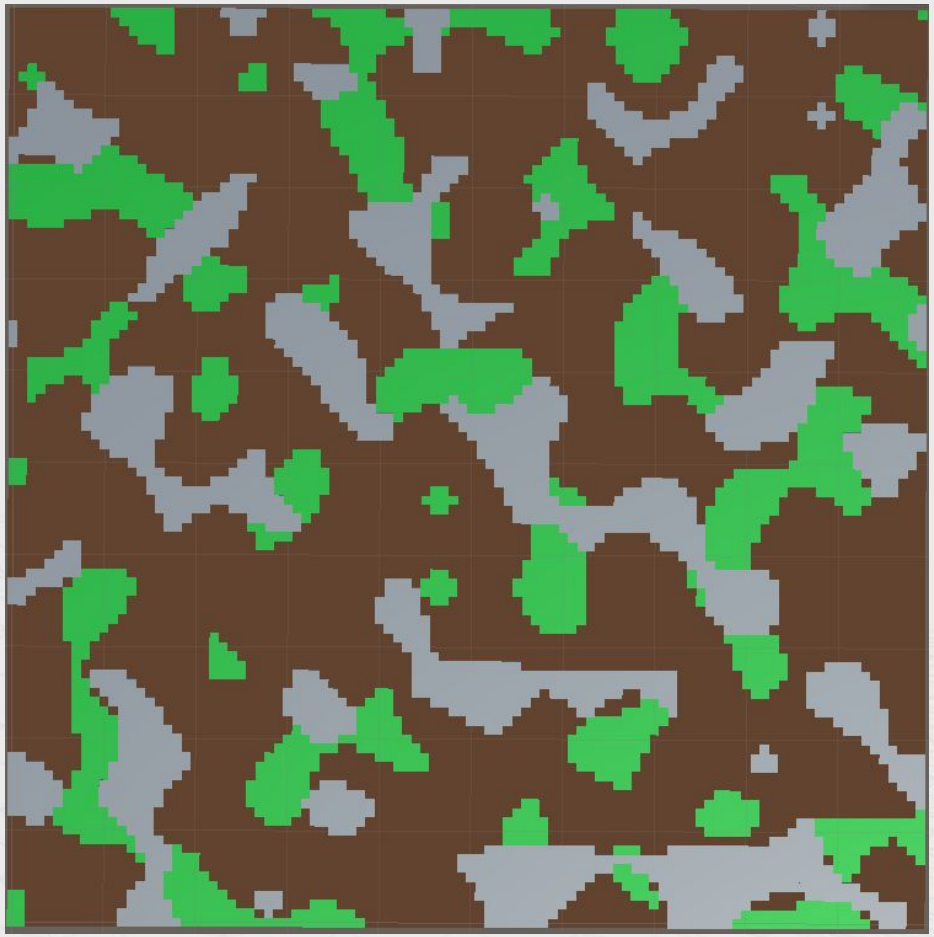
Perlinův šum (Unity)

-Posouvání po ose y



Generátor terénu

-Posun po ose x a ose y

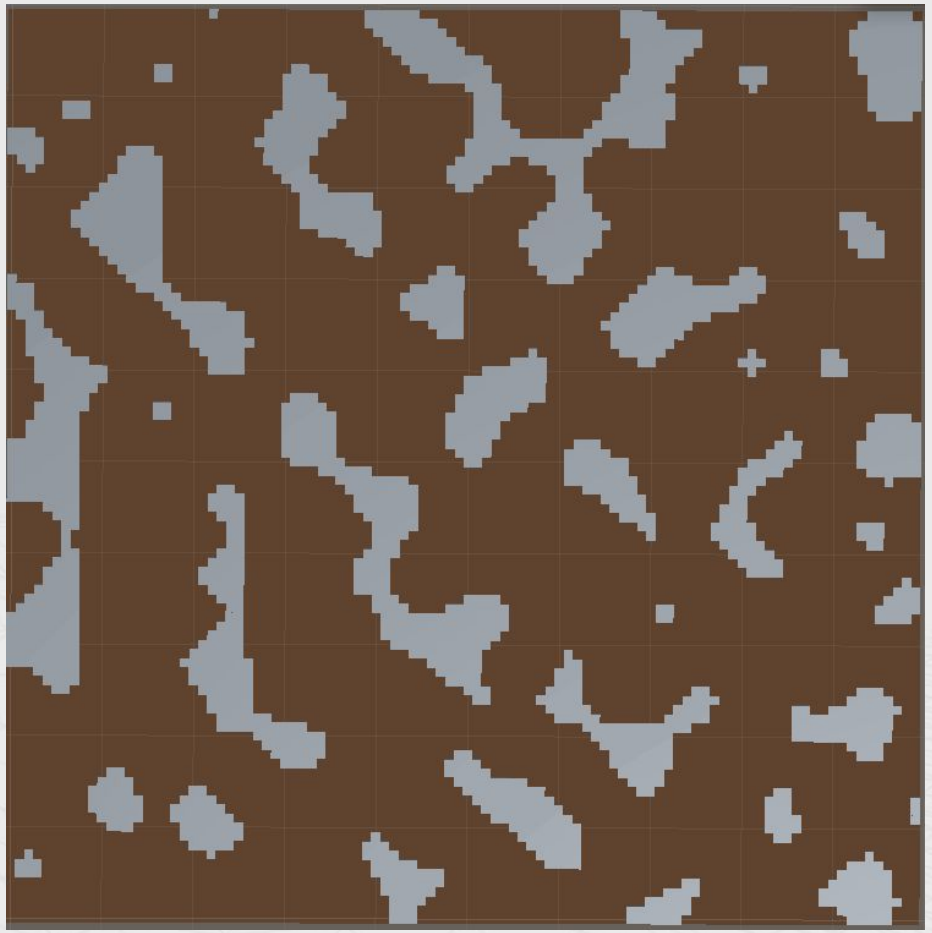


Generátor terénu

```
private void FillList(List<Vector2> list,float xOffset, float yOffset, float sampleRate, float threshold, bool swapXandY = false)
{
    for (int x = 0; x < kMapXSize; x++)
    {
        for (int y = 0; y < kMapZSize; y++)
        {
            float value = Mathf.PerlinNoise((x + xOffset) / sampleRate, (y + yOffset) / sampleRate);
            Vector2 currentPos = swapXandY ? new Vector2(y, x) : new Vector2(x, y);

            if (value >= threshold)
            {
                list.Add(currentPos);
                GroundPositions.Remove(currentPos);
            }
        }
    }
}
```

Generátor terénu

-Prohození osy x a osy y



Generátor terénu

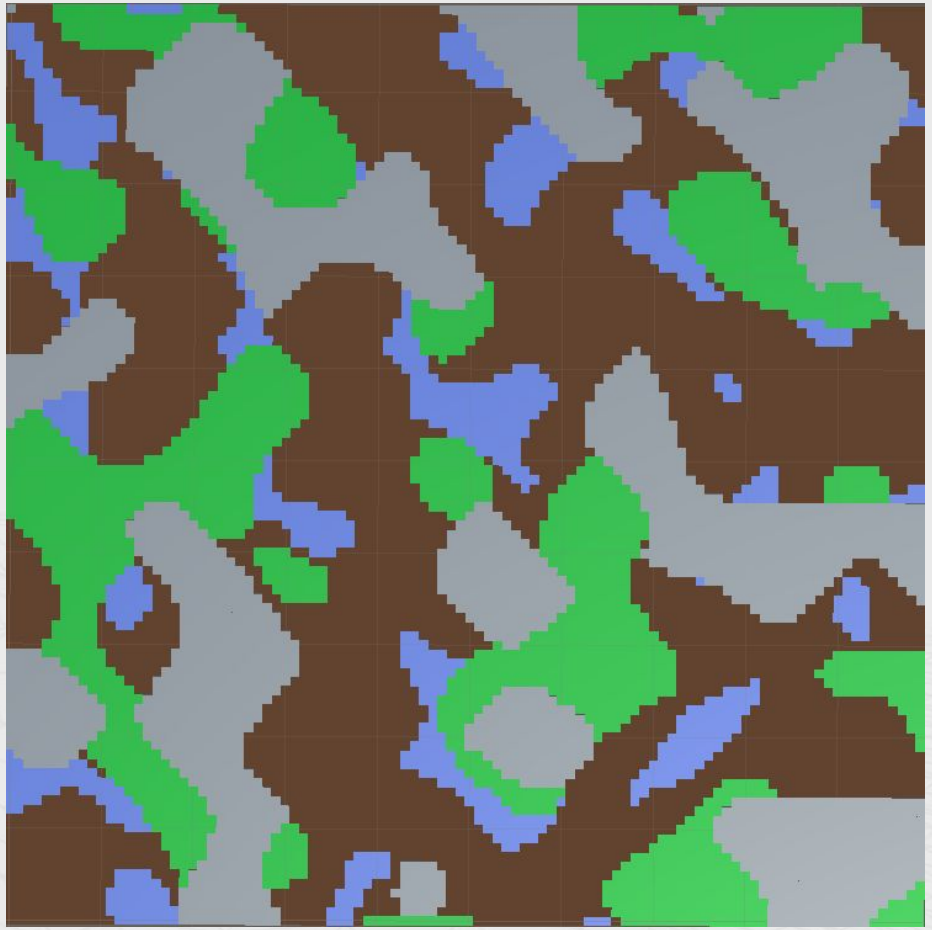
```
/// <summary>
/// Vytvoří mapu
/// </summary>
public void GenerateMap()
{
    CreateMapPosiitonGrid();

    FillList(MountainPositions, 0, 0, 5, 0.6f);

    _visualsGenerator.GenerateVisuals(MountainPositions, VisualsGenerator.ETerrainType.Mountain);
    _visualsGenerator.GenerateVisuals(GroundPositions, VisualsGenerator.ETerrainType.Ground);
}
```

Generátor terénu

```
/// <summary>
/// Vytvoří mapu
/// </summary>
public void GenerateMap()
{
    CreateMapPosiitonGrid();

    FillList(MountainPositions, 0, 0, Random.Range(5, 25), Random.Range(0.5f, 0.6f), false);
    FillList(TreePositions, 128, 128, Random.Range(5, 25), Random.Range(0.5f, 0.6f));
    FillList(WaterPositions, 0, 0, Random.Range(5, 25), Random.Range(0.5f, 0.6f), true);

    _visualsGenerator.GenerateVisuals(MountainPositions, VisualsGenerator.ETerrainType.Mountain);
    _visualsGenerator.GenerateVisuals(TreePositions, VisualsGenerator.ETerrainType.Tree);
    _visualsGenerator.GenerateVisuals(WaterPositions, VisualsGenerator.ETerrainType.Water);
    _visualsGenerator.GenerateVisuals(GroundPositions, VisualsGenerator.ETerrainType.Ground);
}
```

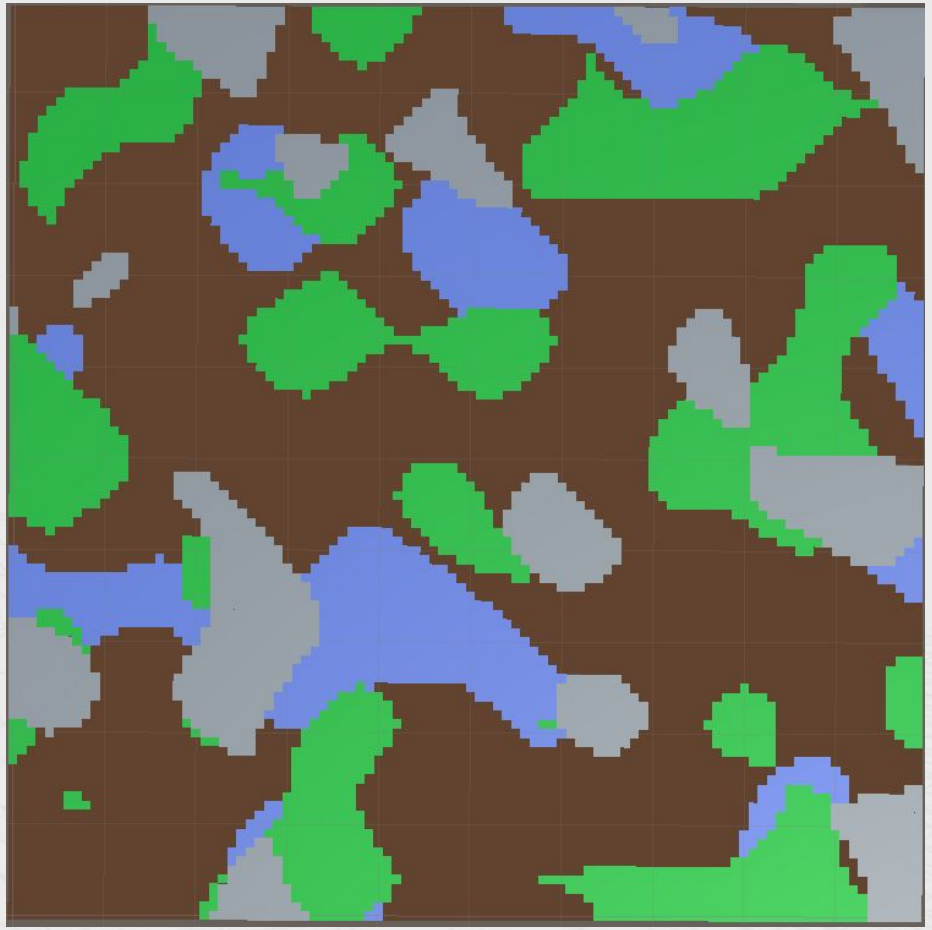
Generátor terénu

-Výsledek čtyř druhů terénu



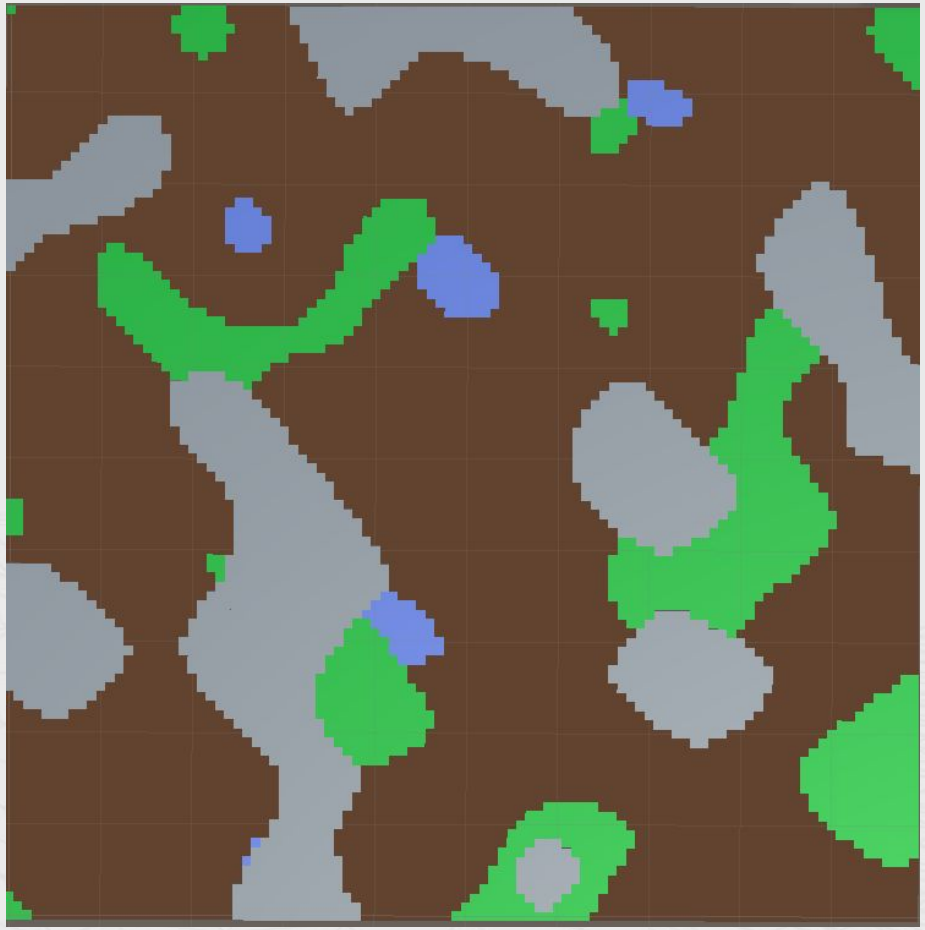
Generátor terénu

-Výsledek čtyř druhů terénu



Generátor terénu

-Výsledek čtyř druhů terénu



Reference

-**Simplex noise demystified** Stefan Gustavson, Linköping University, Sweden, 2005-03-22

DĚKUJI ZA POZORNOST

OTÁZKY?

Bohemia Interactive

@bohemiainteract

facebook.com/BohemiaInteractive/

linkedin.com/company/bohemia-interactive/

Jan Gehr

jan.gehr@bistudio.com