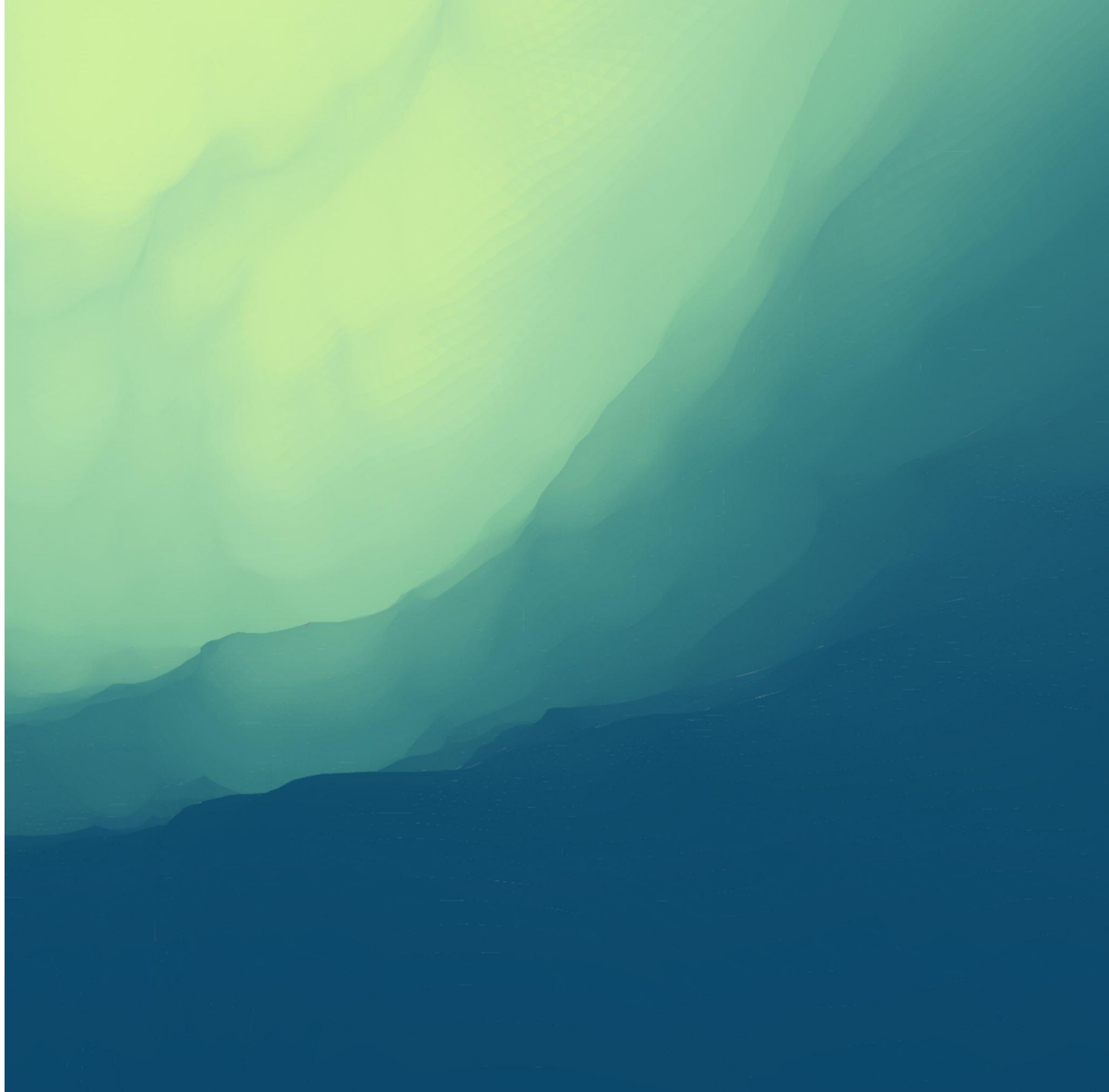
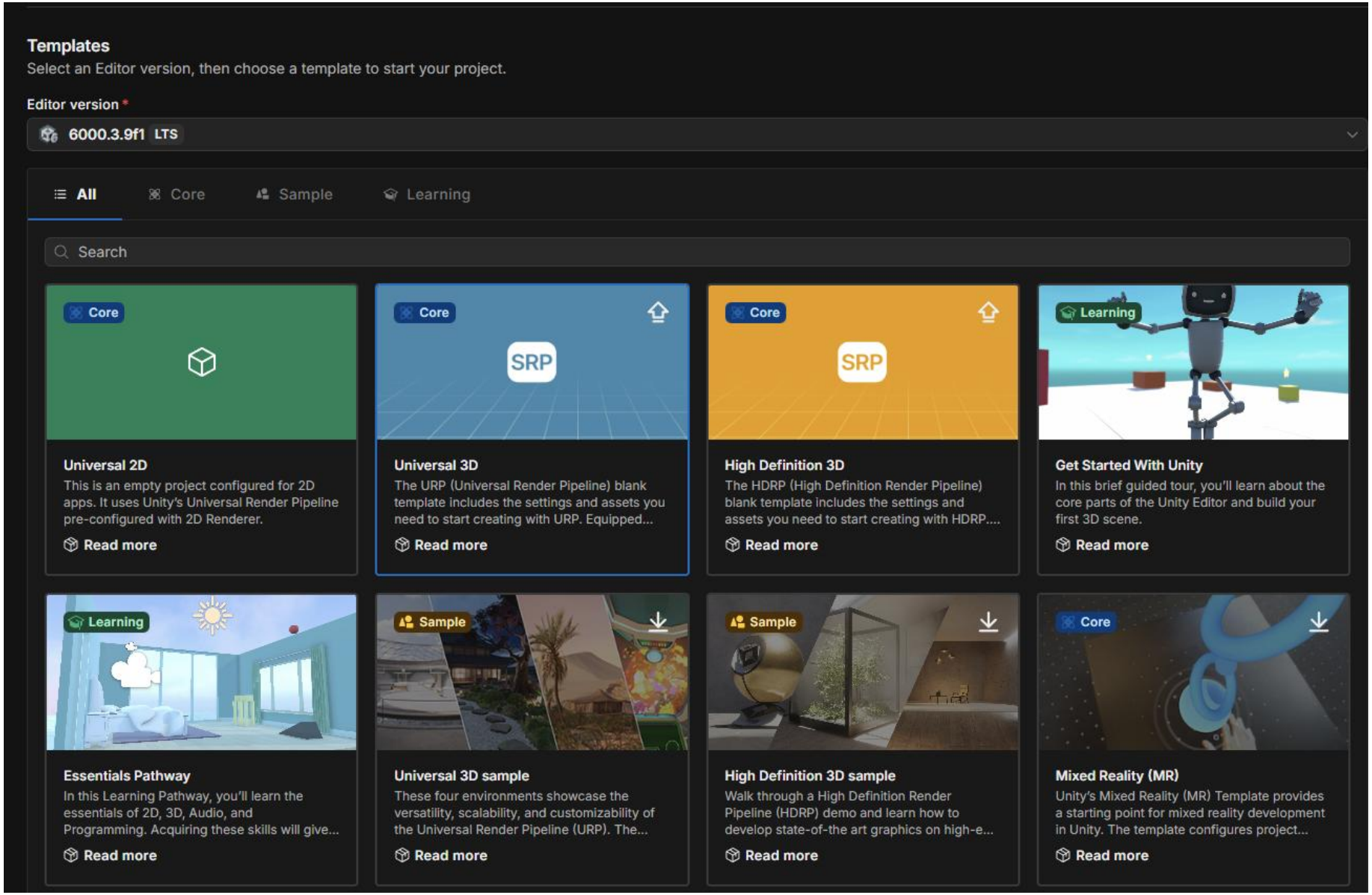
Dynamic procedural ocean & terrain in Unity

ROMAN RAPCO

SGR



Choosing template



https://docs.unity3d.com/2019.1/Documentation/Manual/ProjectTemplates.html

Universal 3D using URP (Universal Render Pipeline).

- optimized for good performance
- works on PC, mobile, console and VR
- uses modern graphics tools, but not extremely realistic
- mostly used for development of indie games



Universal 2D v6.1.2

Info | **Packages**

com.unity.collab-proxy	2.6.0
com.unity.2d.animation	10.1.4
com.unity.2d.psdimporter	10.0.0
com.unity.2d.sprite	1.0.0
com.unity.2d.spriteshape	10.0.7
com.unity.2d.tilemap	1.0.0
com.unity.2d.tilemap.extras	4.2.1
com.unity.2d.aseprite	1.2.1
com.unity.2d.tooling	1.0.0
com.unity.ide.rider	3.0.34
com.unity.ide.visualstudio	2.0.22
com.unity.inputsystem	1.12.0
com.unity.render-pipelines.universal	17.0.3
com.unity.test-framework	1.4.5
com.unity.timeline	1.8.7
com.unity.ugui	2.0.0
com.unity.visualscripting	1.9.5

Universal 3D v17.1.0

Info | **Packages**

Unity UI	2.0.0
Timeline	1.8.9
Rider Editor	3.0.38
Input System	1.14.2
com.unity.collab-proxy	2.10.0
com.unity.ai.navigation	2.0.9
Test Framework	1.6.0
com.unity.visualscripting	1.9.7
Visual Studio Editor	2.0.25
Universal RP	17.0.4

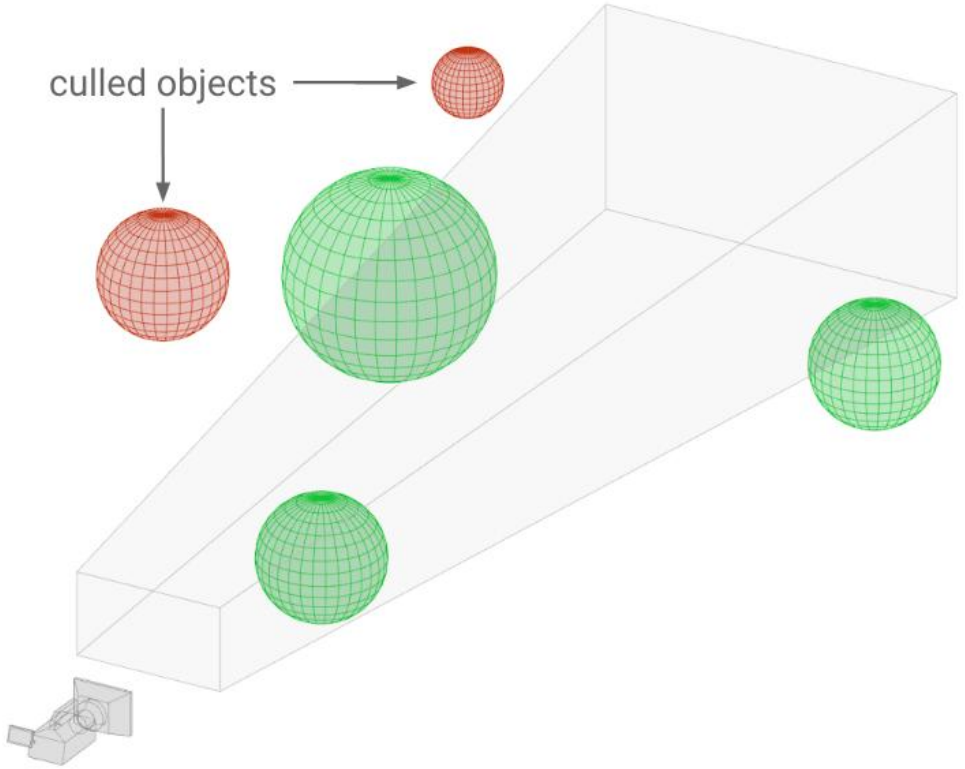
Render pipeline

Spotlight On

Render pipeline stages

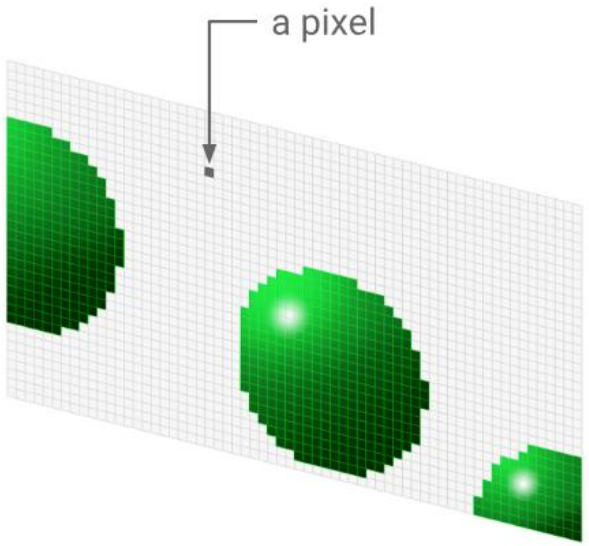
1. Culling

List objects to render



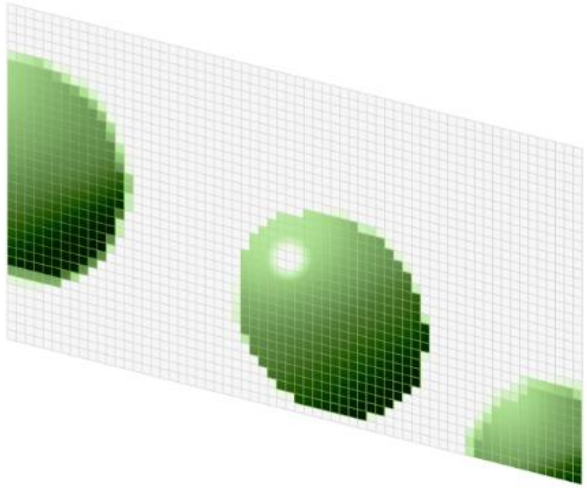
2. Rendering

Draw objects



3. Post-processing

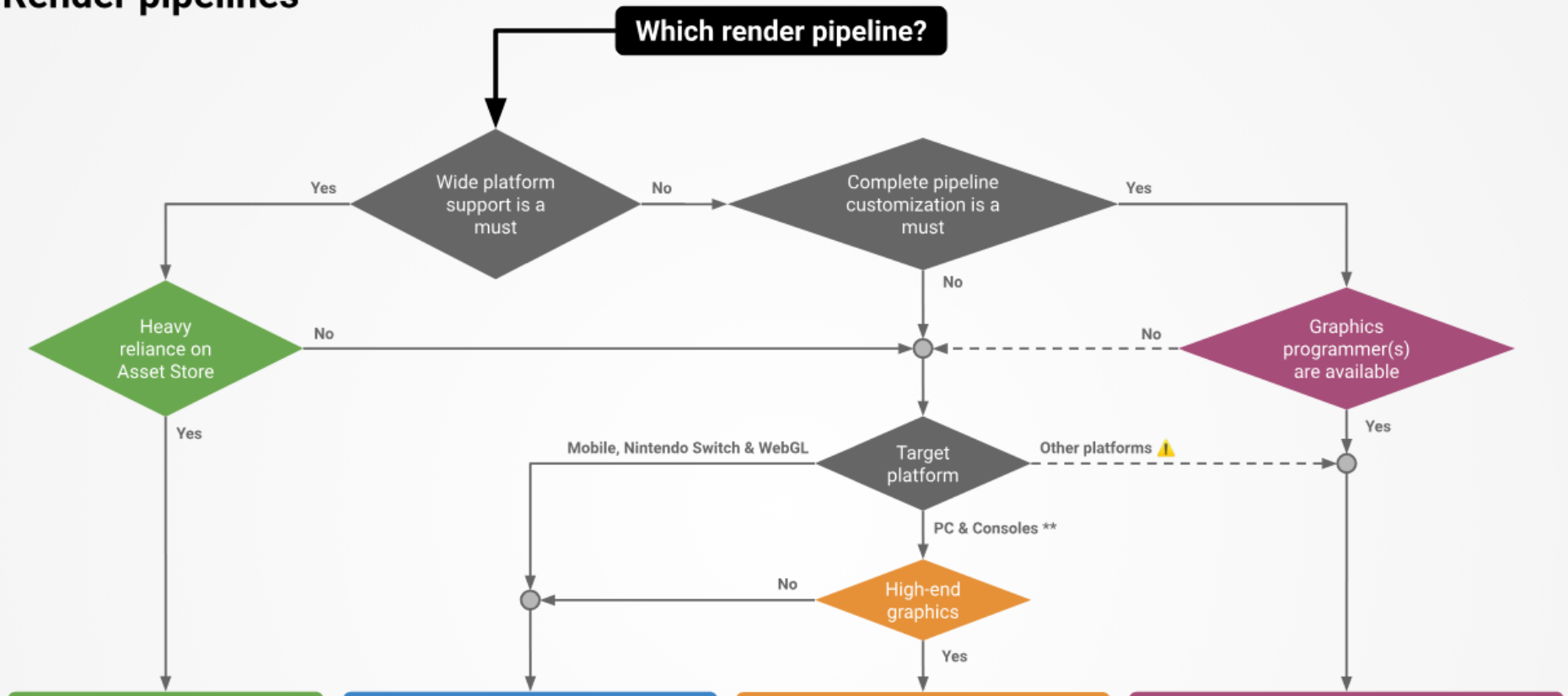
Apply additional image effects



* These images are simplified representations. The actual number of pixels is much higher.

Spotlight on

Render pipelines



Built-In Render Pipeline (Default)

The good old full-featured renderer for a wide range of platforms.

+ Full-featured & stable pipeline
+ Support for ALL platforms
+ Full Asset Store compatibility
+ Large knowledge base
- More difficult to extend
- Older renderer with less focus on optimum performance
- No new features

Universal Render Pipeline (URP)

A high-performance renderer focused on untethered devices.

+ Scalable graphics from mobile to PC
+ Performance-oriented
+ Unity Shader and VFX Graph support
+ Very easy to extend
- Lack advanced real-time shading capabilities from HDRP *
- Limited Asset Store compatibility *

High-Definition Render Pipeline (HDRP)

The go-to renderer for AAA graphics on powerful & modern hardware.

+ Tailored for higher-end platforms (Compute-capable hardware only)
+ Performance-oriented
+ Unity Shader and VFX Graph support
+ Advanced real-time shading capabilities
- Limited Asset Store compatibility *

Custom Scriptable Render Pipeline

Maximum freedom. Maximum optimization. Maximum programming.

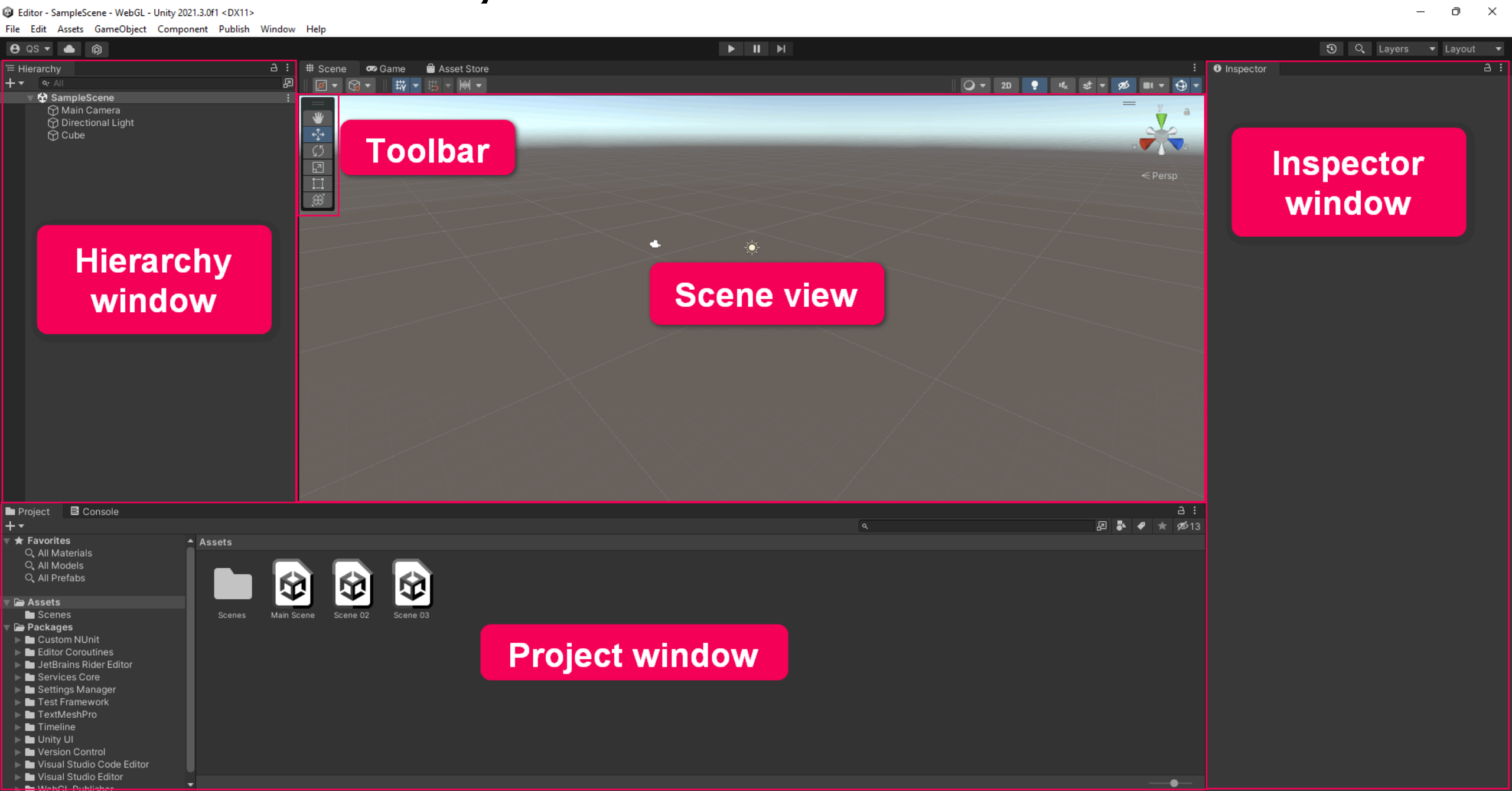
+ Total customization
- Graphics programming required
- Limited Asset Store compatibility*
- Complex support for other platforms than the ones already supported by URP or HDRP

Scriptable Render Pipelines (SRPs)

* To be improved over time

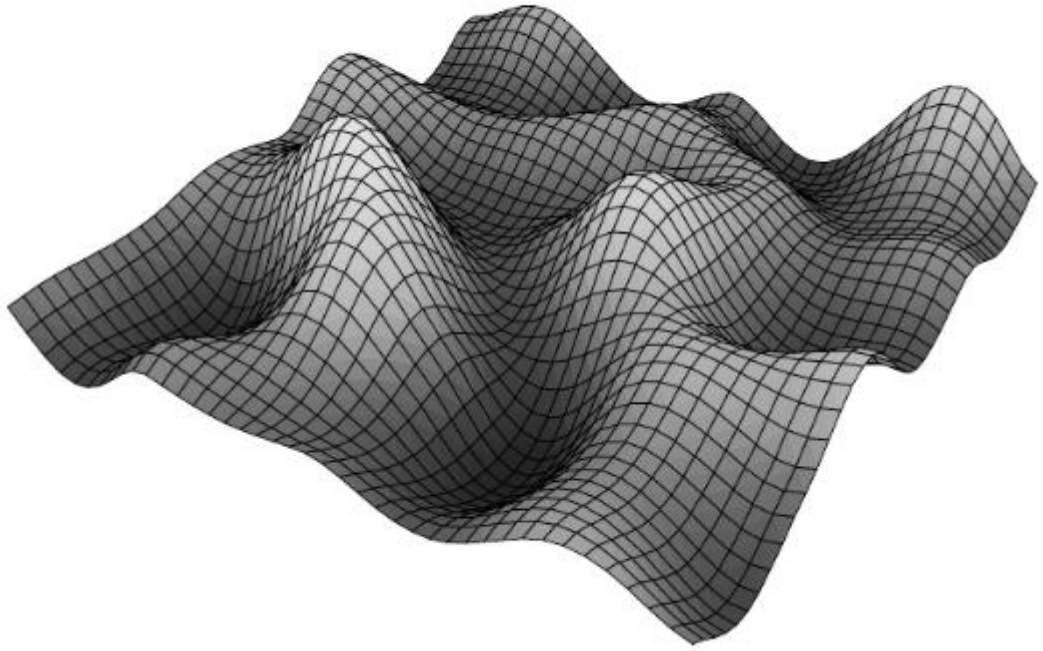
** PC (DX11/DX12, Vulkan), Mac (Metal), Linux (Vulkan), PlayStation 4, Xbox One & Stadia

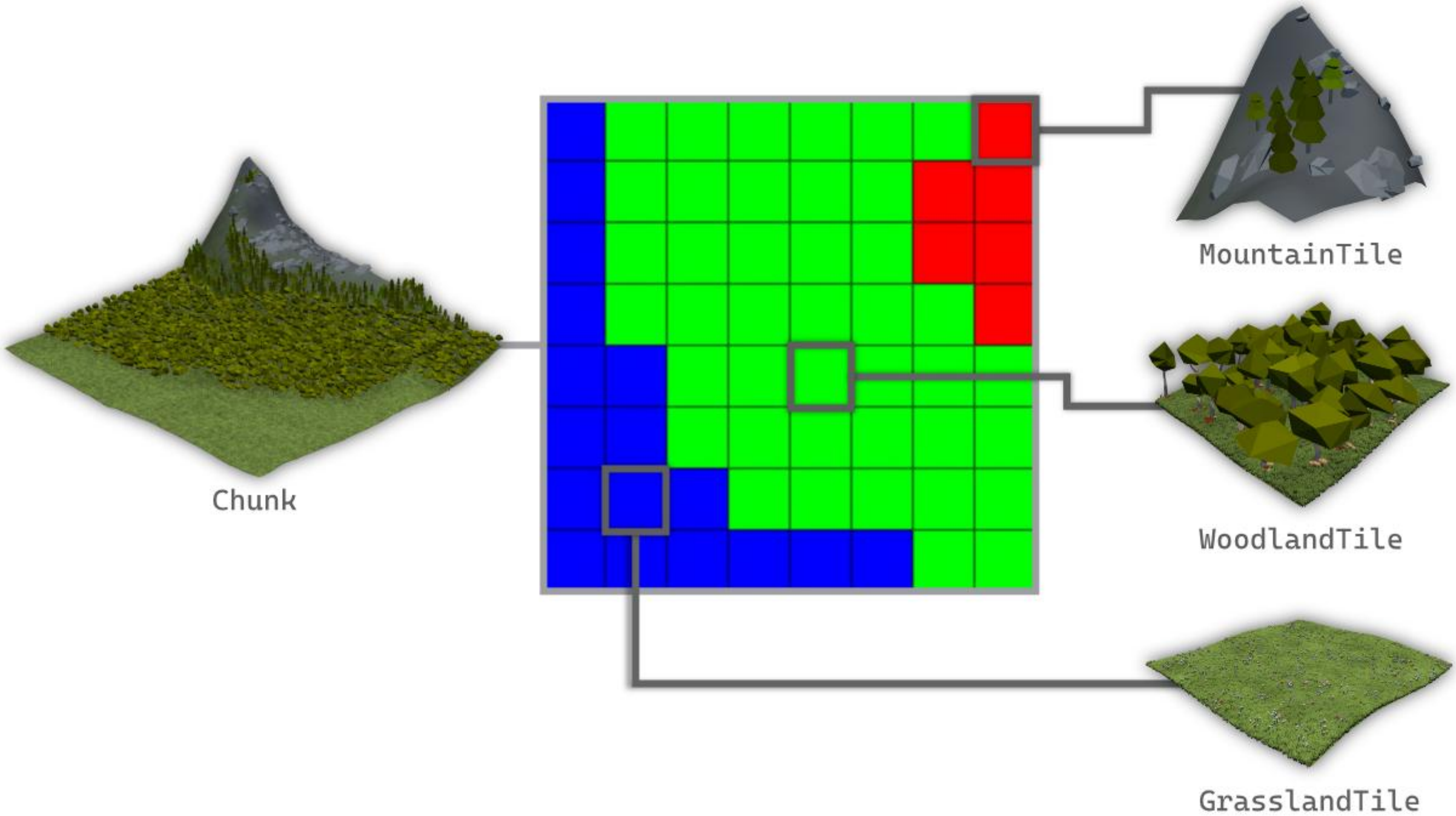
Basics of Unity



Procedural Generation

- generated by algorithm instead of manual creation
- terrains, dungeons, oceans/water simulations
- math function Perlin noise
- Minecraft, No Man's Sky




Chunk
MountainTile
WoodlandTile
GrasslandTile

```
void GenerateVertices()
{
    minTerrainHeight = float.MaxValue;
    maxTerrainHeight = float.MinValue;

    vertices = new List<Vector3>((xSize + 1) * (zSize + 1));
    triangles = new List<int>(xSize * zSize * 6);

    for (int z = 0; z <= zSize; z++)
    {
        for (int x = 0; x <= xSize; x++)
        {
            float localX = x * vertexScale;
            float localZ = z * vertexScale;

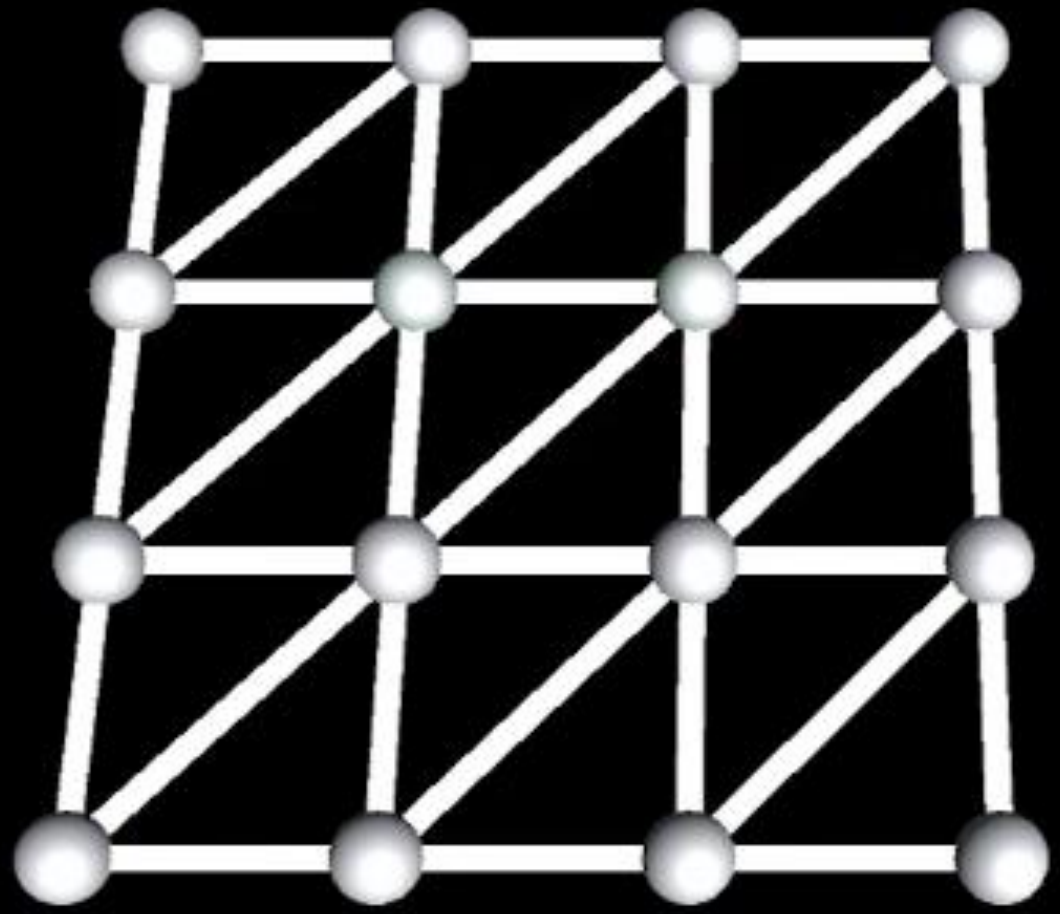
            float worldX = transform.position.x + localX;
            float worldZ = transform.position.z + localZ;

            float sampleX = (worldX + noiseOffsetX) * noiseScale;
            float sampleZ = (worldZ + noiseOffsetZ) * noiseScale;

            float noise = Mathf.PerlinNoise(sampleX, sampleZ);
            float y = noise * heightMultiplier;

            Vector3 vertex = new Vector3(localX - ChunkWorldSizeX/2f, y, localZ - ChunkWorldSizeZ/2f);
            vertices.Add(vertex);

            if (y < minTerrainHeight) minTerrainHeight = y;
            if (y > maxTerrainHeight) maxTerrainHeight = y;
        }
    }
}
```



```
1 reference
void GenerateTriangles()
{
    for (int row = 0; row < zSize; row++)
    {
        for (int column = 0; column < xSize; column++)
        {
            int i = row * (xSize + 1) + column;

            triangles.Add(i);
            triangles.Add(i + xSize + 1);
            triangles.Add(i + xSize + 2);

            triangles.Add(i);
            triangles.Add(i + xSize + 2);
            triangles.Add(i + 1);
        }
    }
}
```

```
1 reference
void GenerateColors()
{
    colors = new Color[vertices.Count];

    for (int i = 0; i < vertices.Count; i++)
    {
        float height = Mathf.InverseLerp(minHeightForColor, maxHeightForColor, vertices[i].y);

        if (gradient != null)
            colors[i] = gradient.Evaluate(height);
        else
            colors[i] = Color.white;
    }
}
```

TerrainManager

```
    2 references
    void UpdateTerrain(bool forceUpdate)
    {
        int gridX = Mathf.FloorToInt(player.position.x / updateStep);
        int gridZ = Mathf.FloorToInt(player.position.z / updateStep);

        Vector2Int currentGridPos = new Vector2Int(gridX, gridZ);

        if (forceUpdate || currentGridPos != lastGridPos)
        {
            lastGridPos = currentGridPos;

            float snappedX = gridX * updateStep;
            float snappedZ = gridZ * updateStep;

            transform.position = new Vector3(snappedX, 0f, snappedZ);
            chunk.GenerateChunk();
        }
    }
}
```

Úloha

1. Pridajte ďalšie dve vlny s inou frekvenciou, rýchlosťou a skombinujte ich (Octaves).
2. Zmeňte smer vĺn pomocou vektora. Pridajte premennú waveDirection typu Vector2 so smerom (1,0),(0,1),(1,1),(2,1).
   float dir = worldX * waveDirection.x + worldZ * waveDirection.y;
   zmeňte premennú worldX v premennej wave na dir
3. Vyskúšajte pridať Perlin noise namiesto sin.

Next steps?

- LOD
- Wind parameter for ocean
- Gerstner waves, Fourier waves
- Bouyancy based on mass and depth submerged – object (boat)
- Better shaders


y
x
z
Persp

Zdroje

- https://unity.com/
- https://docs.unity3d.com/2020.1/Documentation/Manual/BestPracticeLightingPipelines.html
- https://img-c.udemycdn.com/course/480x270/1173872_520c_2.jpg
- https://connect-mediagw.unity.com/h1/20220606/learn/images/7fabb375-5282-4852-9ecf-d8acc254052b_EditorExplore.png
- https://gpuopen.com/learn/work_graphs_mesh_nodes/work_graphs_mesh_nodes-procedural_generation
- https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/perlin-noise-part-2/perlin-noise-terrain-mesh.html
- https://www.noveltech.dev/img/posts/water-shader/depth-section-smoothness-showoff.JPG
- https://www.youtube.com/watch?v=-3ekimUWb9I
- https://www.youtube.com/watch?v=64NblGkAabk&list=PLrMEhC9sAD1zprGu_lphl3cQSS3uFIXA9&index=5
- https://www.youtube.com/watch?v=_j7Rh27ccqk