Session 24

|  |
| --- |
| **Objectives**  The goals of this session |
| 1. Introduction to Perlin Noise (Recap) 2. Introduction to World Generation (Recap) 3. Focus Activity 2 Introduction |

|  |
| --- |
| **Method Focuses**  New methods or functions that you will learn during this session |
| **Mathf**   * float PerlinNoise(float x, float y);   **GameObject**   * GameObject CreatePrimitive(PrimitiveType type); |
| **Property Focuses**  Properties (variables) of classes you will learn during this session |
| GameObject plane = GameObject.CreatePrimitive(PrimitiveType.Plane);  GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);  GameObject sphere = GameObject.CreatePrimitive(PrimitiveType.Sphere);  GameObject capsule = GameObject.CreatePrimitive(PrimitiveType.Capsule);  GameObject cylinder = GameObject.CreatePrimitive(PrimitiveType.Cylinder); |
| **Documentation Links** |
| Perlin Noise  <https://docs.unity3d.com/ScriptReference/Mathf.PerlinNoise.html>  Create Primitive  <https://docs.unity3d.com/ScriptReference/GameObject.CreatePrimitive.html> |

## Focus Activity 2

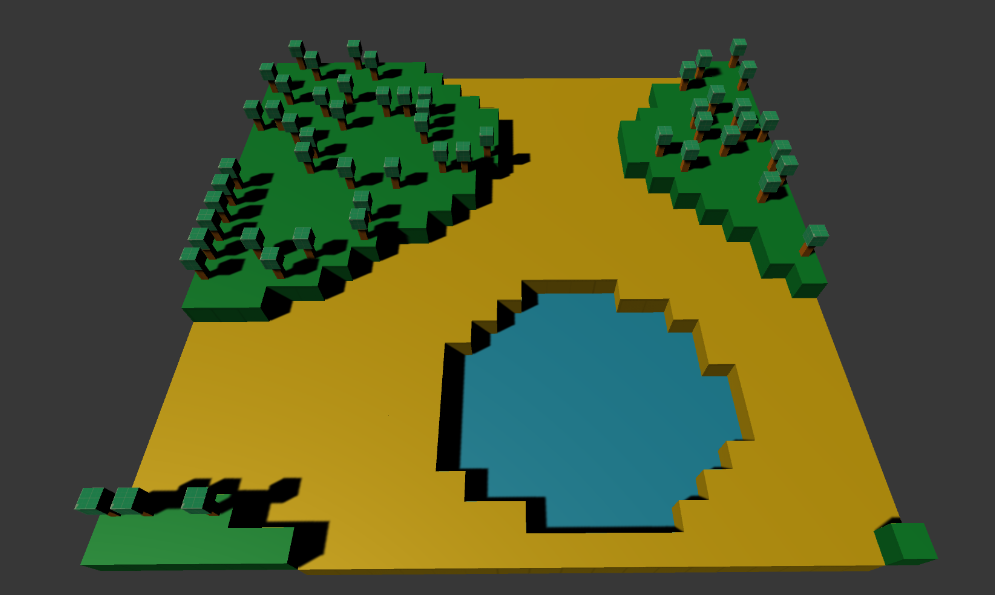
Based on Focus Activity 1.

Prompt: Create a basic forest that has trees in certain areas and has a basic cubic terrain with varying heights.

Requirements:

1. Terrain created based on Perlin Noise
2. Trees are placed based on Perlin Noise and elevation (must be on grass)
3. Low elevation is blue, slightly higher is yellow and above is green as shown below.
4. Variable size

|  |
| --- |
| Grass (green) |
| Sand (yellow) |
| Water (blue) |



\*Note in the image above the noise value of the tree was adjusted to make it more random by using the formula:

PerlinNoise = Random.Range(0,5)

For your benefit examples from session 23 has been provided below.

## Examples:

Example 23\_1

|  |
| --- |
| ///////////////////////////////////////////////////////////  // Perlin Noise and 2D Grid Worlds  // Many worlds in games are based on a 2D grid.  // Each 'cell' on the world is represented by its x and y value  // These 'cells' contain the data about it like what colour it is, how high it is et cetera.  // Generating a flat plane of cubes  public float seed = 423423432;  public float scale = 1;  public Vector2 offset = new Vector2(0, 0);  public void GenerateCubes()  {  // Setting how big the plane is going to be  int xSize = 10; // How wide  int ySize = 10; // How long  // It is a 2D plane so we loop through X and Y  for (int x = 0; x < xSize; x++)  {  for (int y = 0; y < ySize; y++)  {  // Note:  // You can spawn primitives by using GameObject.CreatePrimitive  // PrimitiveType.Cube;  // PrimitiveType.Capsule;  // PrimitiveType.Cylinder;  // Instantiating a new primitive in the world  GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);  // Change the position based what cell we are on  cube.transform.position= new Vector3(x, 0, y);  // Changing the name of the cube  cube.name = "Cube (" + x + "," + y + ")";  }  }  }  public void GenerateCubesPerlin()  {  // Setting how big the plane is going to be  int xSize = 25; // How wide  int ySize = 25; // How long  // It is a 2D plane so we loop through X and Y  for (int x = 0; x < xSize; x++)  {  for (int y = 0; y < ySize; y++)  {  // Note:  // You can spawn primitives by using GameObject.CreatePrimitive  // PrimitiveType.Cube;  // PrimitiveType.Capsule;  // PrimitiveType.Cylinder;  // Instantiating a new primitive in the world  GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);  // Get perlin noise value to determine position  float xNoise = (offset.x + x) / xSize \* scale + 0.01f;  float yNoise = (offset.y + y) / ySize \* scale + 0.01f;  float perlinNoise = Mathf.PerlinNoise(xNoise, yNoise);  float height = 0;  if (perlinNoise <= 0.25f) height = 0;  else if (perlinNoise <= 0.40f) height = 1;  else if (perlinNoise <= 0.6f) height = 2;  else if (perlinNoise <= 1f) height = 3;  // Change the position based what cell we are on  cube.transform.position = new Vector3(x, height, y);  // Changing the name of the cuve  cube.name = "Cube (" + x + "," + y + ")";  }  }  }  // Start is called before the first frame update  void Start()  {  GenerateCubesPerlin();  } |

Example 23\_2

|  |
| --- |
| // Width and height of the texture in pixels.  public int pixWidth;  public int pixHeight;  // The origin of the sampled area in the plane.  public float xOrg;  public float yOrg;  // The number of cycles of the basic noise pattern that are repeated  // over the width and height of the texture.  public float scale = 1.0F;  private Texture2D noiseTex;  private Color[] pix;  private Renderer rend;  void Start()  {  rend = GetComponent<Renderer>();  // Set up the texture and a Color array to hold pixels during processing.  noiseTex = new Texture2D(pixWidth, pixHeight);  pix = new Color[noiseTex.width \* noiseTex.height];  rend.material.mainTexture = noiseTex;  }  void CalcNoise()  {  // For each pixel in the texture...  float y = 0.0F;  while (y < noiseTex.height)  {  float x = 0.0F;  while (x < noiseTex.width)  {  float xCoord = xOrg + x / noiseTex.width \* scale;  float yCoord = yOrg + y / noiseTex.height \* scale;  float sample = Mathf.PerlinNoise(xCoord, yCoord);  pix[(int)y \* noiseTex.width + (int)x] = new Color(sample, sample, sample);  x++;  }  y++;  }  // Copy the pixel data to the texture and load it into the GPU.  noiseTex.SetPixels(pix);  noiseTex.Apply();  }  void Update()  {  CalcNoise();  } |

From: <https://docs.unity3d.com/ScriptReference/Mathf.PerlinNoise.html>

|  |  |
| --- | --- |
| **Session Information**  You can ignore this | |
| **Planned Information** | |
| Session Time Session Date | 5:30PM – 6:30PM AWST 23 March 2021 |
| **Real Information** | |
| Session Time Session Date |  |
| Github Name | Session23Examples |
| Session Length | 1 Hour 3 Minutes |
| Activities Completed | |  |  | | --- | --- | | □ | Recap Perlin Noise | | □ | Recap World Generation | | □ | Focus Activity 2 Started | |
| Signature | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *By signing this you confirm that the session has occurred and the subjects that have been taught have been allocated on this document.* |