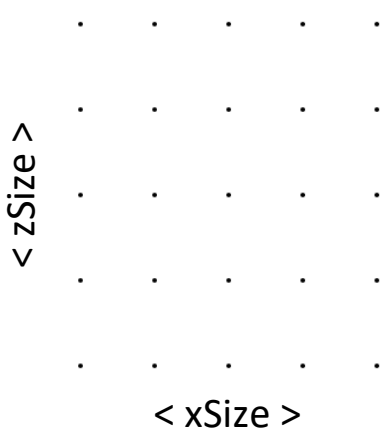
Generating a New Plane Mesh at Runtime

First we need to define the total number of vertices in our mesh, to do this we need to make an array of length (xSize + 1) * (zSize + 1) where our size variables used are defined in the inspector for the length of the mesh on each axis.

Each vertex requires point data in 3D space so the data type used needs to be Vector3

```
vertices = new Vector3[(xSize + 1) * (zSize + 1)];
```

Now, given the length of the mesh on the X and Z axis and the total number of vertices to create, we can create a grid of points in space by looping through two for loops, one for each axis



We will also use two new public floats to define how much space to put between each vertex.

```
for (int i = 0, z = 0; z <= zSize; z++)
{
    for (int x = 0; x <= xSize; x++)
    {
        vertices[i] = new Vector3(x * xSpacing, 0, z *zSpacing);
        i++;
    }
}
```

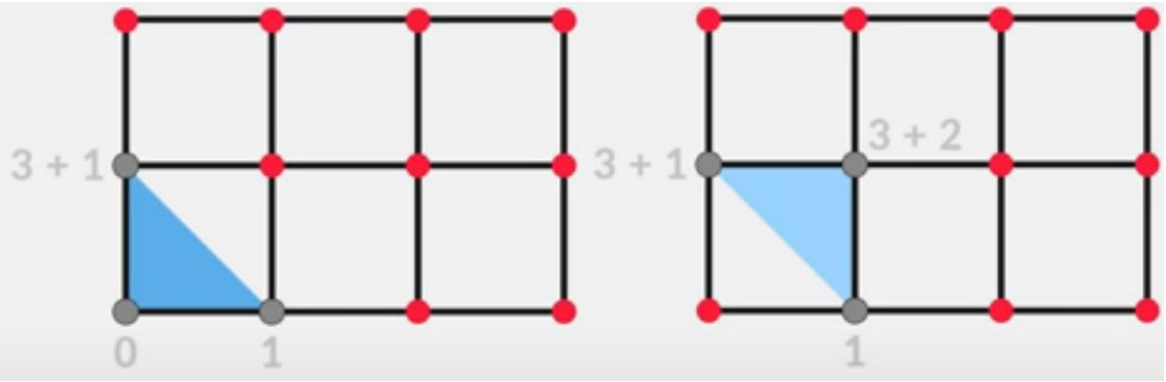
Here we will start to draw our triangles using the grid of vertices we have made. To do this we need two integer variables, one to reference the current vertex we are looking at and another for the index of each point of a triangle in the mesh we draw.

Then we need an integer array to store the index of each vertex we have drawn to draw each triangle in the correct order. The length will be equal to xSize * zSize * 6, the *6 is because in each grid square, two triangles need to be drawn which have 3 points each (2 *3) hence 6.

```
int vert = 0;
int tris = 0;
triangles = new int[xSize * zSize * 6];
```

Now to assign each vertex index to the array of points in triangles[]. Once again we will need a double for loop. Below shows the correct order for how to draw a triangle, this is important as this determines the way that each triangle is drawn as each is only one-sided.

Example: let xSize = 4, zSize = 3



These 6 values are stored in triangles[] before progressing to the next point, when our local variables are increased. We progress one vertex along in our array and increment tris by 6 as that is how many triangle points we have plotted.

```
for (int z = 0; z < zSize; z++)
{
    for (int x = 0; x < xSize; x++)
    {
        //triangle1
        triangles[tris + 0] = vert;
        triangles[tris + 1] = vert + xSize + 1;
        triangles[tris + 2] = vert + 1;
        //triangle2
        triangles[tris + 3] = vert + 1;
        triangles[tris + 4] = vert + xSize + 1;
        triangles[tris + 5] = vert + xSize + 2;
        //Quad Generated

        //Move start position of next vertex
        vert++;
        tris += 6;
    }
    // Skips triangle gen at end of row
    vert++;
}
```

Now that we have the new data for our plane. We have to clear the current data in our mesh component that is generated on load and replace it with the new data that we have now generated.

```
2 references
void UpdateMesh()
{
    mesh.Clear();
    mesh.vertices = vertices;
    mesh.triangles = triangles;
    mesh.RecalculateNormals();
```

Finally, we should be almost ready to generate a plane in unity. For reference this is what your global variables should look like.

Using required component generates the components listed on each object if they do not already have them.

```
[RequireComponent(typeof(MeshFilter))]
[RequireComponent(typeof(MeshRenderer))]

Unity Script | 0 references
public class GenerateMesh : MonoBehaviour
{
    private MeshRenderer MS;
    private Material Mat;

    Mesh mesh;

    Vector3[] vertices;
    int[] triangles;
```

To assign our mesh renderer, material and mesh in code, do the following in void start().

```
Unity Message | 0 references
void Start()
{
    MS = GetComponent<MeshRenderer>();
    Mat = MS.material;
    mesh = new Mesh();
```

Your result should look something like this: