```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[RequireComponent(typeof(MeshFilter),typeof(MeshRender))]
public class Generator : MonoBehaviour
{
    private GameObject _feet;
    public float widthMesh = 50f;
    public float heightMesh = 50f;
    public static Mesh Triangle(Vector3 vertex0, Vector3 vertex1, Vector3
vertex2)//Òðèêóòíèê
    {
        var normal = Vector3.Cross((vertex1 - vertex0), (vertex2 -
vertex0)).normalized;
        var mesh = new Mesh
        {
            vertices = new[] { vertex0, vertex1, vertex2 },
            normals = new[] { normal, normal, normal },
            uv = new[] { new Vector2(0, 0), new Vector2(0, 1), new
Vector2(1, 1) },
            triangles = new[] { 0, 1, 2 }
        };
        return mesh;
    }
    public static Mesh Quad(Vector3 origin, Vector3 width, Vector3
length)//Êâàäðàò
    {
        var normal = Vector3.Cross(length, width).normalized;
        var mesh = new Mesh
        {
            vertices = new[] { origin, origin + length, origin + length +
width, origin + width },
            normals = new[] { normal, normal, normal, normal },
            uv = new[] { new Vector2(0, 0), new Vector2(0, 1), new
Vector2(1, 1), new Vector2(1, 0) },
            triangles = new[] { 0, 1, 2, 0, 2, 3 }
        };
        return mesh;
    }
    public static Mesh Plane(Vector3 origin, Vector3 width, Vector3
length, int widthCount, int lengthCount)//Ïåâíà ïëîùèíà
    {
        var combine = new CombineInstance[widthCount * lengthCount];

        var i = 0;
        for (var x = 0; x < widthCount; x++)
        {
            for (var y = 0; y < lengthCount; y++)
            {
                combine[i].mesh = Quad(origin + width * x + length * y,
width, length);
                i++;
            }
        }

        var mesh = new Mesh();
        mesh.CombineMeshes(combine, true, false);
        return mesh;
    }
```

```
    public static Mesh Cube(Vector3 width, Vector3 length, Vector3
height)//Ïàðàëëåëåïèïåä
    {
        var corner0 = -width / 2 - length / 2 - height / 2;
        var corner1 = width / 2 + length / 2 + height / 2;

        var combine = new CombineInstance[6];
        combine[0].mesh = Quad(corner0, length, width);
        combine[1].mesh = Quad(corner0, width, height);
        combine[2].mesh = Quad(corner0, height, length);
        combine[3].mesh = Quad(corner1, -width, -length);
        combine[4].mesh = Quad(corner1, -height, -width);
        combine[5].mesh = Quad(corner1, -length, -height);

        var mesh = new Mesh();
        mesh.CombineMeshes(combine, true, false);
        return mesh;
    }
    public static Mesh Octahedron(float radius)//îêòàºäð
    {
        var v = new Vector3[6];
        v[0] = new Vector3(0, -radius, 0);
        v[1] = new Vector3(-radius, 0, 0);
        v[2] = new Vector3(0, 0, -radius);
        v[3] = new Vector3(+radius, 0, 0);
        v[4] = new Vector3(0, 0, +radius);
        v[5] = new Vector3(0, radius, 0);

        var mesh = new Mesh
        {
            vertices = v,
            triangles = new[] { 0, 1, 2,
                                0, 2, 3,
                                0, 3, 4,
                                0, 4, 1,
                                5, 2, 1,
                                5, 3, 2,
                                5, 4, 3,
                                5, 1, 4}
        };
        mesh.RecalculateNormals();
        return mesh;
    }
    public static Mesh Tetrahedron(float radius)//òåòðàºäð
    {
        var v0 = new Vector3(0, radius, 0);
        var v1 = new Vector3(0, -radius * 0.333f, radius * 0.943f);
        var v2 = new Vector3(radius * 0.816f, -radius * 0.333f, -radius *
0.471f);
        var v3 = new Vector3(-radius * 0.816f, -radius * 0.333f, -radius
* 0.471f);

        var combine = new CombineInstance[4];
        combine[0].mesh = Triangle(v0, v1, v2);
        combine[1].mesh = Triangle(v1, v3, v2);
        combine[2].mesh = Triangle(v0, v2, v3);
        combine[3].mesh = Triangle(v0, v3, v1);

        var mesh = new Mesh();
        mesh.CombineMeshes(combine, true, false);
        return mesh;
```

```
    }
    public static Mesh Icosahedron(float radius)//²êîñà°äð
    {
        var magicAngle = Mathf.PI * 26.565f / 180;
        var segmentAngle = Mathf.PI * 72 / 180;
        var currentAngle = 0f;

        var v = new Vector3[12];
        v[0] = new Vector3(0, radius, 0);
        v[11] = new Vector3(0, -radius, 0);

        for (var i = 1; i < 6; i++)
        {
            v[i] = new Vector3(radius * Mathf.Sin(currentAngle) *
Mathf.Cos(magicAngle),
                radius * Mathf.Sin(magicAngle),
                radius * Mathf.Cos(currentAngle) *
Mathf.Cos(magicAngle));
            currentAngle += segmentAngle;
        }
        currentAngle = Mathf.PI * 36 / 180;
        for (var i = 6; i < 11; i++)
        {
            v[i] = new Vector3(radius * Mathf.Sin(currentAngle) *
Mathf.Cos(-magicAngle),
                radius * Mathf.Sin(-magicAngle),
                radius * Mathf.Cos(currentAngle) * Mathf.Cos(-
magicAngle));
            currentAngle += segmentAngle;
        }

        var combine = new CombineInstance[20];
        combine[0].mesh = Triangle(v[0], v[1], v[2]);
        combine[1].mesh = Triangle(v[0], v[2], v[3]);
        combine[2].mesh = Triangle(v[0], v[3], v[4]);
        combine[3].mesh = Triangle(v[0], v[4], v[5]);
        combine[4].mesh = Triangle(v[0], v[5], v[1]);

        combine[5].mesh = Triangle(v[11], v[7], v[6]);
        combine[6].mesh = Triangle(v[11], v[8], v[7]);
        combine[7].mesh = Triangle(v[11], v[9], v[8]);
        combine[8].mesh = Triangle(v[11], v[10], v[9]);
        combine[9].mesh = Triangle(v[11], v[6], v[10]);

        combine[10].mesh = Triangle(v[2], v[1], v[6]);
        combine[11].mesh = Triangle(v[3], v[2], v[7]);
        combine[12].mesh = Triangle(v[4], v[3], v[8]);
        combine[13].mesh = Triangle(v[5], v[4], v[9]);
        combine[14].mesh = Triangle(v[1], v[5], v[10]);

        combine[15].mesh = Triangle(v[6], v[7], v[2]);
        combine[16].mesh = Triangle(v[7], v[8], v[3]);
        combine[17].mesh = Triangle(v[8], v[9], v[4]);
        combine[18].mesh = Triangle(v[9], v[10], v[5]);
        combine[19].mesh = Triangle(v[10], v[6], v[1]);

        var mesh = new Mesh();
        mesh.CombineMeshes(combine, true, false);
        return mesh;
    }
    void Start()
```

```
    {
        Generator gn = new Generator();
        _feet = GameObject.Find("TTBFeetB") as GameObject;
        MeshFilter m_f = GetComponent<MeshFilter>();


        Vector3 L1 = new Vector3(widthMesh, 0f, 0f);
        Vector3 L2 = new Vector3(0f, heightMesh, 0f);
        Vector3 L3 = new Vector3(widthMesh, heightMesh, 0f);

        //m_f.mesh = Triangle(L1, L2, L3);
        //m_f.mesh = Quad(L1, L2, L3);
        //m_f.mesh = Plane(L1, L2, L3, 4, 4);
        //m_f.mesh = Cube(L1, L2, L3);//bad
        //m_f.mesh = Octahedron(20f);//Good
        //m_f.mesh = Tetrahedron(10f);//Good
        //m_f.mesh = Icosahedron(5f);//Good
    }

    // Update is called once per frame
    void Update()
    {

    }
}
```