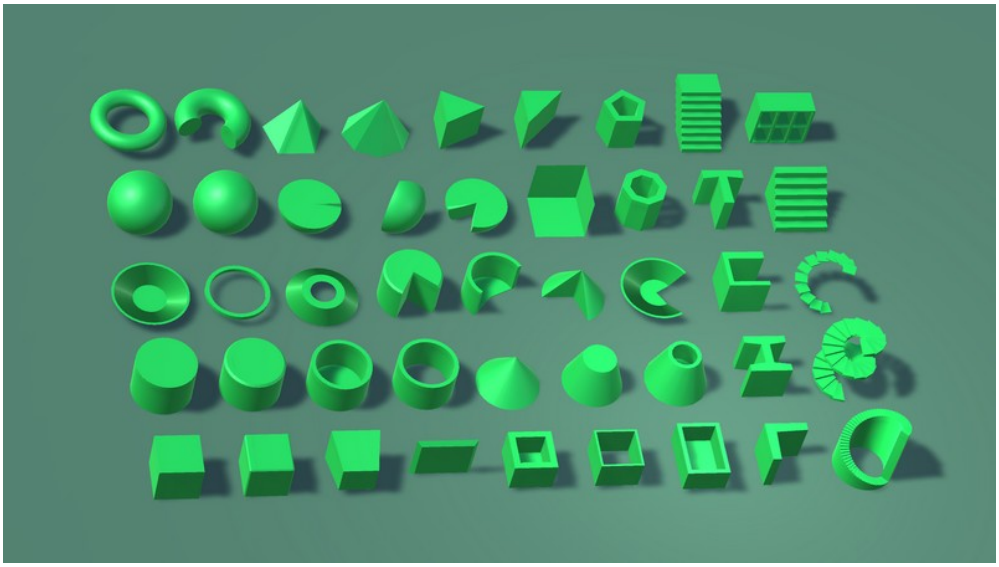
Quick Primitives

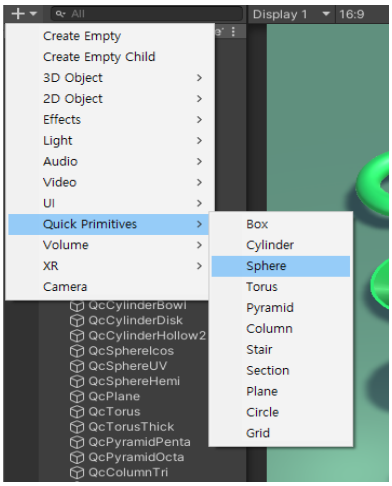
Version 1.60

Quick Primitives is a collection of basic geometric shapes that give more flexibility than the default shapes available in Unity. Using Quick Primitives, 3D shapes can be added to Unity and edited very easily and quickly. The shapes are generated by codes procedurally; no fixed models are. They can be changed by setting their parameter values in the inspector interactively. To add Quick Primitives shapes, just select Create >> Quick Primitives >> [Object].

In addition, UV texture coordinates and colliders can be generated automatically.

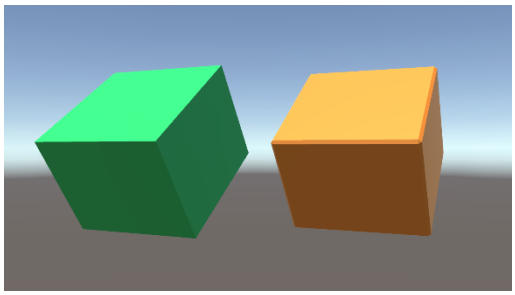
All source codes in c# are included.

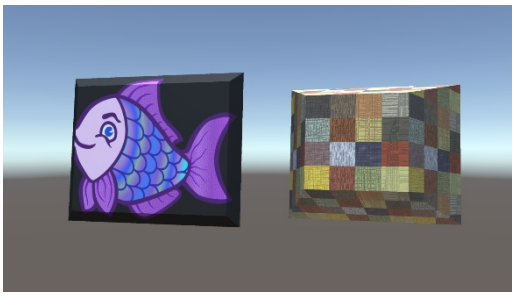




Box Primitive

As an asset for Unity, Quick Boxes allows creating 3D box shape objects easily with a few mouse clicks. Supported features include bevels on the edges, hollow boxes, slanted sides, offset position and texture wrapping.





How to Use

1. To create a box object, select Create >> Quick Primitives >> Box. Or you may create an empty object and add *QcBoxMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a box collider in addition to the script itself to the editor.

2. In the script section labeled 'Qc Box Mesh', you may select options or set values to change the shape of the box.

    - *Width, Height, and Depth*

      To set the dimension of the box, change the values: *Width, Height, and Depth.*

    - *Width, Depth, Height Segments*

      Sets the number of divisions along each axis of the object.

    - *Offset*

      Allows moving the object position by *offset*. It can be used to set pivot point.

    - *Option*

      Selects one of the three features below: Slanted Sides, Beveled Edge, or Hollow

    - *Slanted Sides*

      Makes side faces trapezoids. This can be used to create a 3D panel

    - *Beveled Edge*

      Add bevels on edges.

    - *Hollow*

      Makes a hollow box. If the height is the same as the box height, a rectangular tube is created.

    - *Gen Texture Coords*

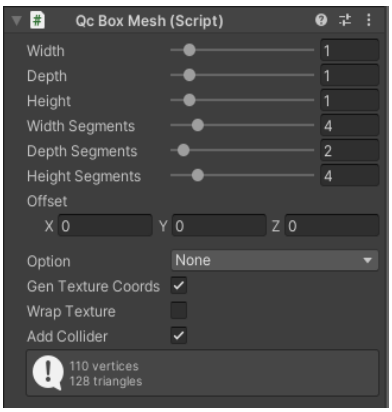
      Generates coordinates for applying texture mapping

    - *Texture Wrapped*

      Selecting *Texture Wrapped* makes the effect of the box wrapped around with a texture.

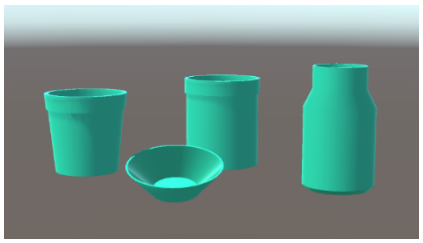
    - *Add Collider*

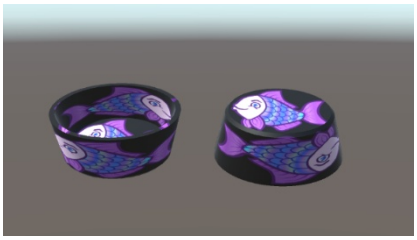
      Add a box collider to the object



Cylinder Primitive

Quick Cylinders allows creating 3D cylinder shape objects easily with a few mouse clicks. Supported features include bevels on the edges, hollow cylinders, different top and bottom radius and offset position. In addition, texture coordinates are automatically generated.





How to Use

- To create a cylinder object, select Create >> Quick Primitives >> Cylinder. Or you may create an empty object and add *QcCylinderMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a capsule collider in addition to the script itself to the inspector.
- In the script section labeled 'Qc Cylinder Mesh', you may select options or set values to change the shape of the cylinder.
  - *Radius, Height*

    Set the dimension of the cylinder by *Radius* and *Height.*
  - *Top Radius*

    Allows creating cylinders with different top and bottom radius.
  - *Offset*

    Allows moving the object position by *offset*. It can be used to set pivot point.
  - *Slice On*

    Enables the Slice function.
  - *Slice From, Slice To*

    Sets the number of degrees around the local y.
  - *Sides*

    The number of segment to form a circle around a cylinder. The larger value makes the circular shape more smoothly rounded, but adds more triangles.
  - *Option*

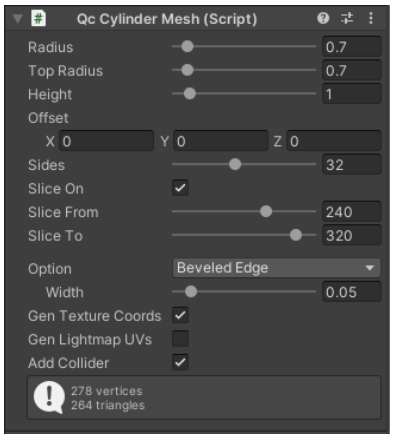
    Selects one of the two features below: *Beveled Edge* or *Hollow*
  - *Beveled edge*

    Add bevels on all edges.
  - *Hollow*

    Makes a hollow cylinder. If the height is smaller than the cylinder height, a cylinder with the bottom is created.
  - *Gen Texture Coords*

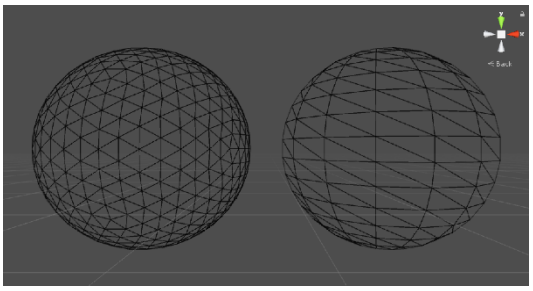
    Generates texture coordinates
  - *Add Collider*

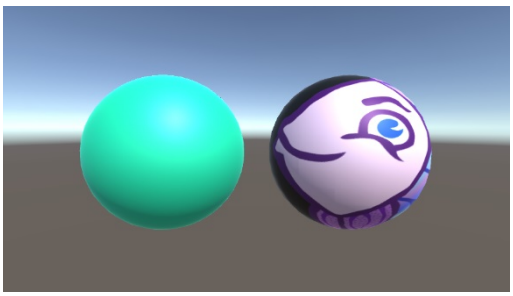
    Add a capsule collider to the object



Sphere Primitive

Quick Spheres allows creating sphere objects easily with a few mouse clicks. Icosahedron sphere or UV sphere can be selected to create a sphere object. In addition, texture coordinates are automatically generated.

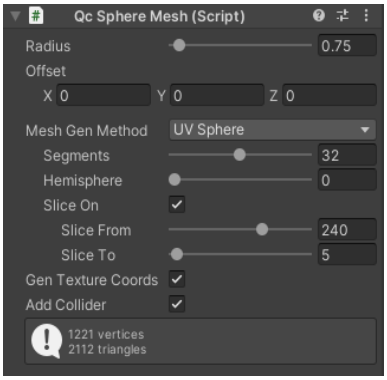




How to Use

1. To create a sphere object, select Create >> Quick Primitives >> Sphere. Or you may create an empty object and add *QcSphereMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a sphere collider in addition to the script itself to the inspector.

2. In the script section labeled 'Qc Sphere Mesh', you may select options or set values to change the shape of the sphere.



- *Radius*

  To set the dimension of the sphere, change the values: *Radius* and *Height.*

- *Offset*

  Allows moving the object position by *offset*. It can be used to set pivot point.

- *Mesh Gen Method*

  Selects one of the two features below: *Icosphere* or *UV Sphere*

- *Icosphere*

  Selects the method for even distribution of vertices, especially around the poles. However, texture mapping does not work well on the vertical seam. A Higher number for *Subdivisions* results in a smoother surface, but more triangles.

- *UV Sphere*

  This method generates faces along longitude and longitude lines divided by *Segments.*

  Hemisphere allows creating a half sphere shape. The range of Hemisphere value is between 0 and 0.9. The value 0 gives a full sphere while 0.9 creates the smallest hemisphere.

  Slice On enables the Slice function.

  *Slice From and Slice To s*ets the number of degrees around the local y.

- *Gen Texture Coords*

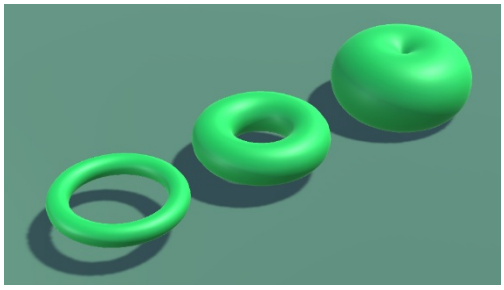
  Generates texture coordinates

- *Add Collider*

  Add a sphere collider to the object

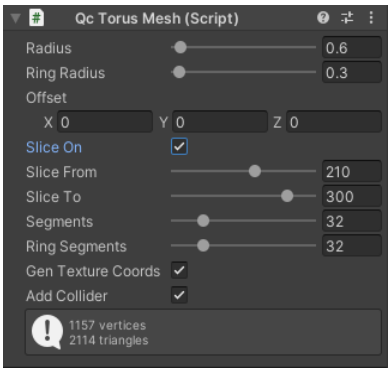
Torus Primitive

Ring or donut shape objects can be created by selecting Torus primitive.



How to Use

1. To create a torus object, select Create >> Quick Primitives >> Torus. Or you may create an empty object and add *QcTorusMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a capsule collider in addition to the script itself to the inspector.
2. In the script section labeled 'Qc Torus Mesh', you may select options or set values to change the shape of the torus.



- *Radius*

  Sets the distance from the center of the torus to the center of the cross-sectional circle.

- *Offset*

  Allows moving the object position by *offset*. It can be used to set pivot point.

- *Slice On*

  Enables the Slice function.

- *Slice From, Slice To*

  Sets the number of degrees around the local y.

- *Ring Radius*

  Sets the radius of the cross-sectional circle.

- *Segments*

  Sets the number of radial divisions around the torus.

- *Ring Segments*

  Sets the number of sides along the cross-sectional circle of the tours.

- *Gen Texture Coords*

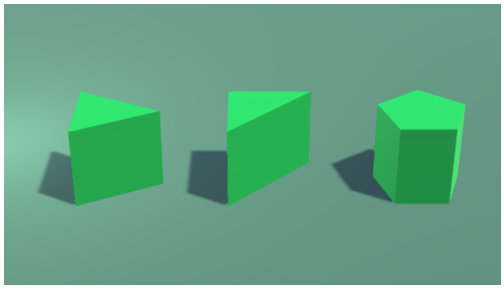
  Generates texture coordinates

- *Add Collider*

  Add a box collider to the object

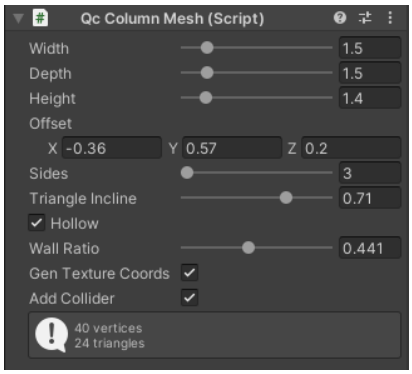
Column Primitive

Creates columns with a varying number of sides.



How to Use

1. To create a column object, select Create >> Quick Primitives >> Column. Or you may create an empty object and add *QcColumnMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a box collider in addition to the script itself to the inspector.
2. In the script section labeled 'Qc Column Mesh', you may select options or set values to change the shape of the column.



- *Width, Depth, Height*

  To set the dimension of the column, change the values: *Width, Height, and Depth.*

- *Offset*

  Allows moving the object position by *offset*. It can be used to set pivot point.

- *Sides*

  Sets the number of sides. Three is the minimum number allowed.

- *Triangle Incline*

  When the value of *Sides* is three, resulting in a triangular prism, the value of *Triangle Incline* sets the position of a vertex relative to the other two vertices.

- *Hollow*

  Enables hollow columns

- *Wall Ratio*

  Wall Ratio value set the ratio of the wall of the hollow column to the width and depth of the column.

- *Gen Texture Coords*

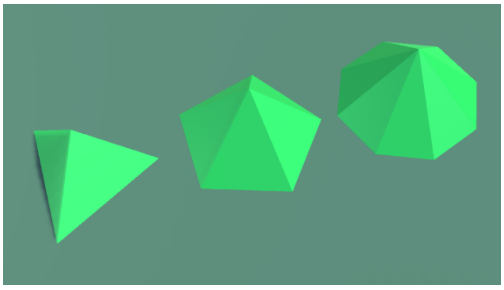
  Generates texture coordinates

- *Add Collider*

  Add a box collider to the object

Pyramid Primitive

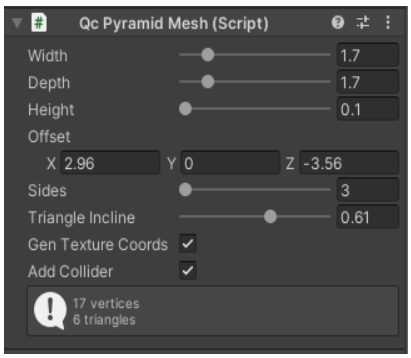
Creates pyramids with a varying number of sides.



How to Use

1. To create a sphere object, select Create >> Quick Primitives >> sphere. Or you may create an empty object and add *QcSphereMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a box collider in addition to the script itself to the inspector.

2. In the script section labeled 'Qc Sphere Mesh', you may select options or set values to change the shape of the pyramid.



- *Width, Depth, Height*

  Sets the dimension of the pyramid.

- *Offset*

  Allows moving the object position by *offset*. It can be used to set pivot point.

- *Sides*

  Sets the number of sides.

- *Triangle Incline*

  When the value of *Sides* is three, resulting in a triangular pyramid, the value of *Triangle Incline* sets the position of a vertex relative to the other two vertices.

- *Gen Texture Coords*

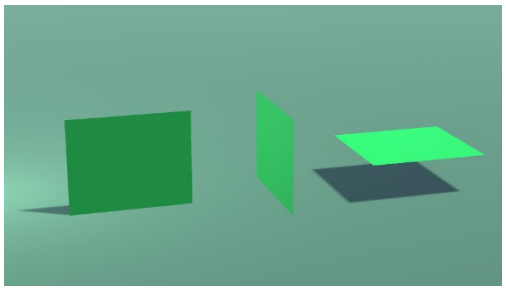
  Generates texture coordinates

- *Add Collider*

  Add a box collider to the object

Plane Primitive

Creates a rectangular plane.



How to Use

1. To create a sphere object, select Create >> Quick Primitives >> Plane. Or you may create an empty object and add *QcPlaneMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a box collider in addition to the script itself to the inspector.
2. In the script section labeled 'Qc Plane Mesh', you may select options or set values to change the shape of the plane.

- *Width, Height*

  Sets the width and height of the plane object.

- *Offset*

  Allows moving the object position by *offset*. It can be used to set pivot point.

- *Width Segments, Height Segments*

  Sets the number of divisions along each axis of the object.

- *Double sided*

  When selected, create two identical planes facing opposite to each other.

- *Direction*

  Sets the direction the plane object along the x, y, and z axes.

- *Gen Texture Coords*

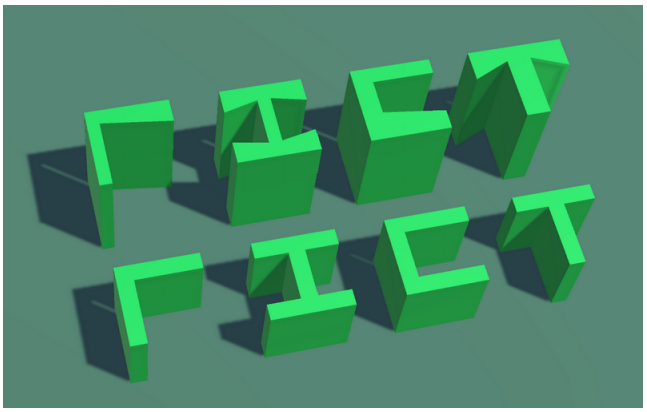
  Generates texture coordinates

- *Add Collider*

  Add a box collider to the object

Section Primitive

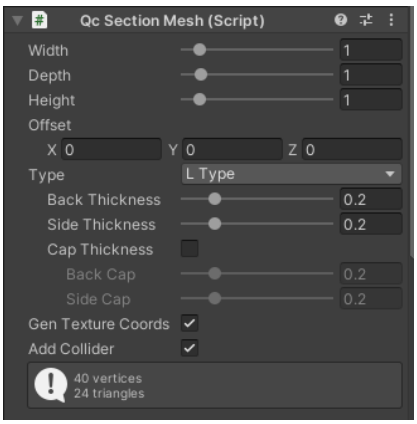
Creates an extruded L, I, C, or T shaped object or beam.



How to Use

1. To create a sphere object, select Create >> Quick Primitives >> Section. Or you may create an empty object and add *QcSectionMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a box collider in addition to the script itself to the inspector.

2. In the script section labeled 'Qc Section Mesh', you may select options or set values to change the shape of the plane.



   - *Width, Depth, Height*

     Sets the width, depth, and height of the plane object, effectively setting the bounds of the shape.

   - *Offset*

     Allows moving the object position by *offset*.
     It can be used to set pivot point.

   - *Type*

     Selects one of L, I, C, and T shapes.

   - *Front, Back, Side Thickness*

     According to the type selected above, two or three thickness sliders appear to set the thicknesses.

   - *Cap Thickness*

     When selected, the end thickness can be set in addition to the above thicknesses.

   - *Gen Texture Coords*

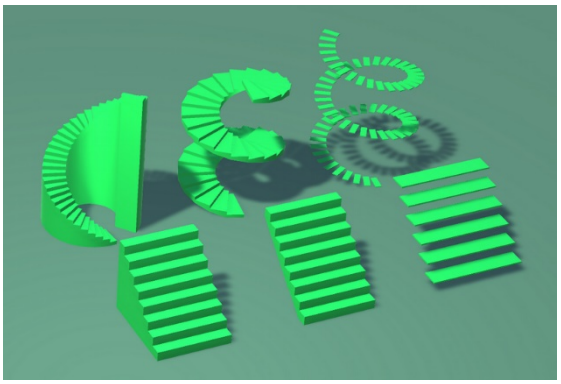
     Generates texture coordinates

   - *Add Collider*

     Add a box collider to the object

Stair Primitive

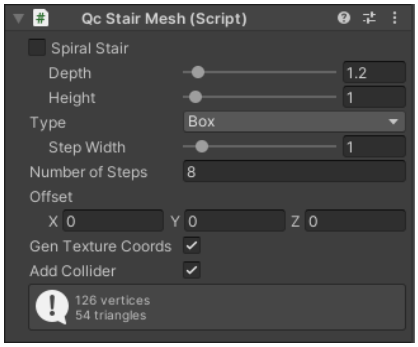
Creates a rectangular plane.



How to Use

1. To create a stair object, select Create >> Quick Primitives >> Stair. Or you may create an empty object and add *QcStairMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a box collider in addition to the script itself to the inspector.

2. In the script section labeled 'Qc Stair Mesh', you may select options or set values to change the shape of the plane.



- *Spiral Stair*

  A spiral shape stair or a straight stair can be selected. According to the selection, a different set of parameters is shown below.

- *Depth, Height*

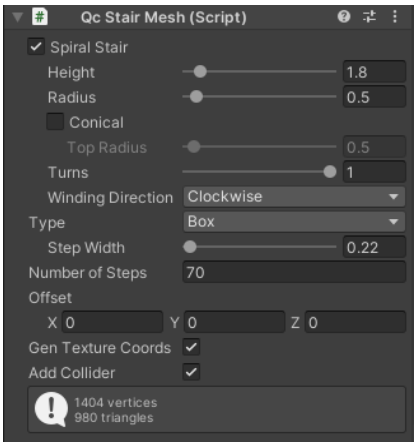
  Sets the depth and height of a straight stair.

- *Height, Radius*

  Sets the height and inner radius of a spiral stair.

- *Conical*

  Enables to set Top Radius so that bottom radius and top radius become different.



- *Revolutions*

  Sets the number of revolutions of a spiral stair winds around.

- *Winding Direction*

  Selects either clockwise or counterclockwise direction.

- *Type*

  Selects one of Box, Closed, and Closed types. Please refer to the picture on top for the resulting shapes.

- *Step Width*

  Sets the width of a step.

- *Number of Steps*

  Sets the number of steps in the stair.

- *Offset*

  Allows moving the object position by *offset*. It can be used to set pivot point.

- *Gen Texture Coords*

  Generates texture coordinates

- *Add Collider*

  Add a box collider to the object.

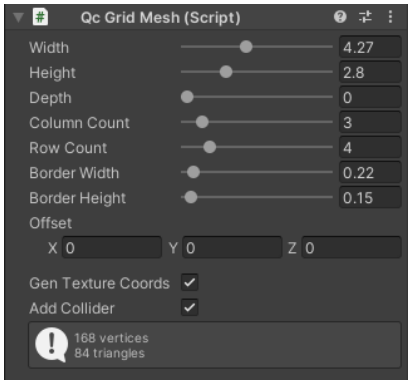
Grid Primitive

Creates a grid.

How to Use

1. To create a grid object, select Create >> Quick Primitives >> Grid. Or you may create an empty object and add *QcGridMesh.cs* script under *Assets/QcPrimitives/Scripts* folder. This effectively adds a mesh renderer, a mesh filter, and a box collider in addition to the script itself to the inspector.

2. In the script section labeled 'Qc Grid Mesh', you may select options or set values to change the shape of the plane.



- *Width, Height, Depth*

  Sets the width, height and depth of a grid.

- *Column Count*

  Number of columns in a grid

- *Row Count*

  Number of rows in a grid

- *Border Width, Border Height*

  Sets the width and the height of grid borders

- *Offset*

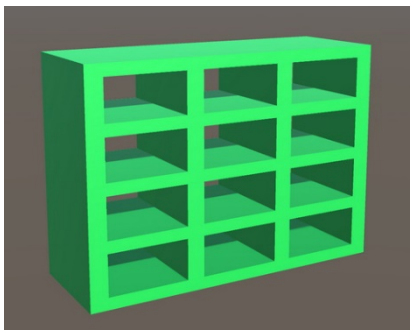
  Allows moving the object position by *offset*. It can be used to set pivot point.

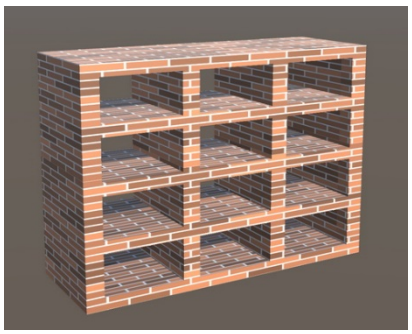
- *Gen Texture Coords*

  Generates texture coordinates

- *Add Collider*

  Add a box collider to the object.





Creating from Scripts

Create GameObjects using your own scripts and set values of public variables to change the shape like the sample code below:

```
using UnityEngine;
using QuickPrimitives;

public class mainControl : MonoBehaviour
{
    void Update ()
    {
        if (Input.GetKeyDown(KeyCode.A))
        {
            GameObject box = new GameObject("QcBoxMesh");
            box.AddComponent<QcBoxMesh>();
            box.GetComponent<QcBoxMesh>().properties.width = 1.0f;
            box.GetComponent<QcBoxMesh>().properties.depth = 0.8f;
            box.GetComponent<QcBoxMesh>().properties.option =
                QcBoxMesh.QcBoxProperties.Options.BeveledEdge;
            box.GetComponent<QcBoxMesh>().properties.beveledEdge.width = 0.1f;

            GameObject cylinder = new GameObject("QcCylinderMesh");
            cylinder.AddComponent<QcBoxMesh>();
            cylinder.GetComponent<QcCylinderMesh>().properties.option =
                QcCylinderMesh.QcCylinderProperties.Option.Hollow;
            cylinder.GetComponent<QcCylinderMesh>().properties.hollow.thickness = 0.1f;
            cylinder.GetComponent<QcCylinderMesh>().properties.hollow.height = 0.8f;
        }
    }
}
```

Updating value changes

After changing the properties in your script, RebuildGeometry() method needs to be called to make the changes effective. The sample code below shows how to use RebuildGeometry() method.

```
GameObject torus = GameObject.Find("MyQcTorus");
QcTorusMesh torusMesh = torus.GetComponent<QcTorusMesh>();
torusMesh.properties.radius = 0.4f;
torusMesh.properties.ringRadius = 0.2f;
torusMesh.RebuildGeometry();
```

Reports and Suggestions

Error reports or any suggestions to new features are welcome. Please send your email to chorocks8@gmail.com