Procedural library by cogobyte

Support Email: cogobyte@gmail.com

September 22, 2018

Contents

1 Introduction

Procedural library is the base package for procedural generation. It is used by procedural indicator package. It contains basic shapes (2D meshes, 3D meshes) and path arrays.

2 Mesh Object

To view any 2D or 3D object in procedural library, first create an empty object. Drag and drop the MeshObject script on it. Mesh object has a Generated Mesh property. This property accepts any ProceduralMesh (Primitive and SolidMesh are derived from this class). These are asset objects that can be dragged and dropped to MeshObject Generated Mesh property. Either create your own assets using menu path:

Assets » Create » Cogobyte » Procedural library » SolidMeshes or Primitives

or use one of the assets provided in the examples. Upon running the game Mesh Object will generate a new mesh each frame and feed it to the mesh filter.

3 Procedural Mesh

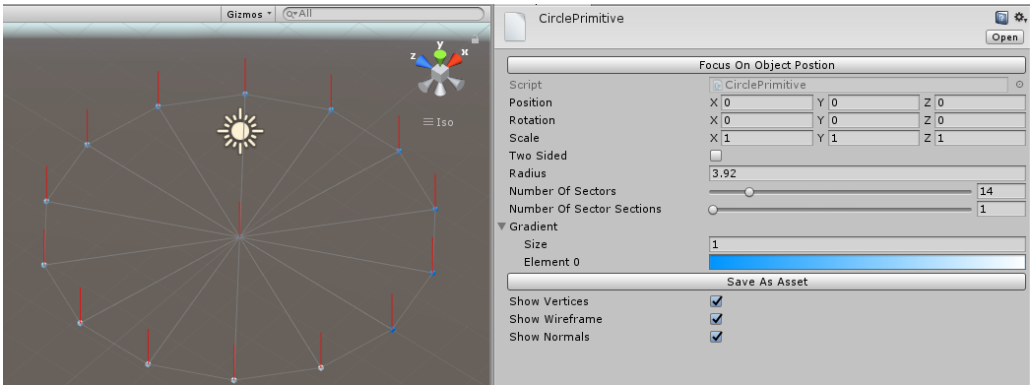

Figure 1: Mesh Object Editor

Each procedural mesh has code that generates its mesh. When selected, every procedural mesh will draw generated vertices, vertice colors, normals and wireframe if these options are enabled. Save as asset button will ask you to chose a folder where generated mesh will be saved. Upon saving, generated mesh will now be a regular unity mesh that can be used with mesh filter (saving is enabled to use this mesh without the procedural library, if you want a permanent mesh asset). Button "Focus On Object Position" will focus camera in scene view on the origin position of the object. Each

procedural mesh has its position, rotation and scale. These are relative to the transform of the object that will contain the mesh filter.

4 Primitives

To create a 2D Primitive, use the Assets > Create > Cogobyte > ProceduralLibrary > Primitives menu. Each Primitive has an outline. Outline defines the outline border of a 2D shape which is mostly used for extruding. If a class extends a 2D primitive it must generate its outline and length of that outline (or keep the default method) if it is to be used for extruding.

4.1 Circle Primitive

Circle primitive has parameters: numberOfSectors (3 will create a triangle, 6 a hexagon, a lot will be a circle), numberOfSectorSections (each sector is split by number of section sectors from center to radius), radius (radius of circle), and gradient array for coloring. Gradient index determines the color position for each sector section vertex (as a cyclical list), and each gradient determines the colors for each sector vertex in a counter clockwise manner.

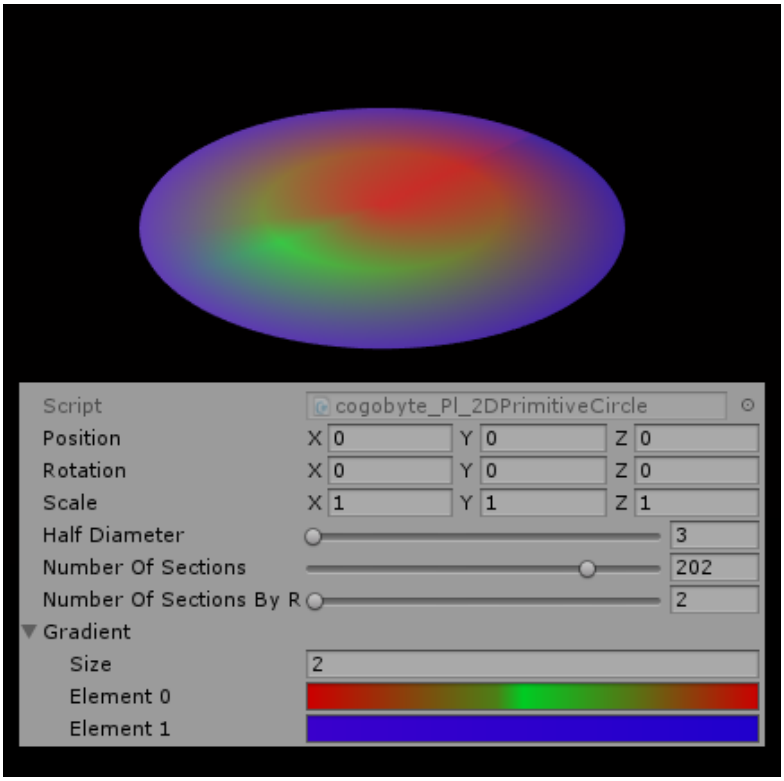

Figure 2: Circle Primitive

4.2 Plane Primitive

Plane primitive has paramers: width, length, numberOfSectionsByWidth, numberOfSectionsByLength and gradient. Number of sections determines the level of detail by length and width. Gradient index is color by length and gradient determines color by width. Gradient array is circular.

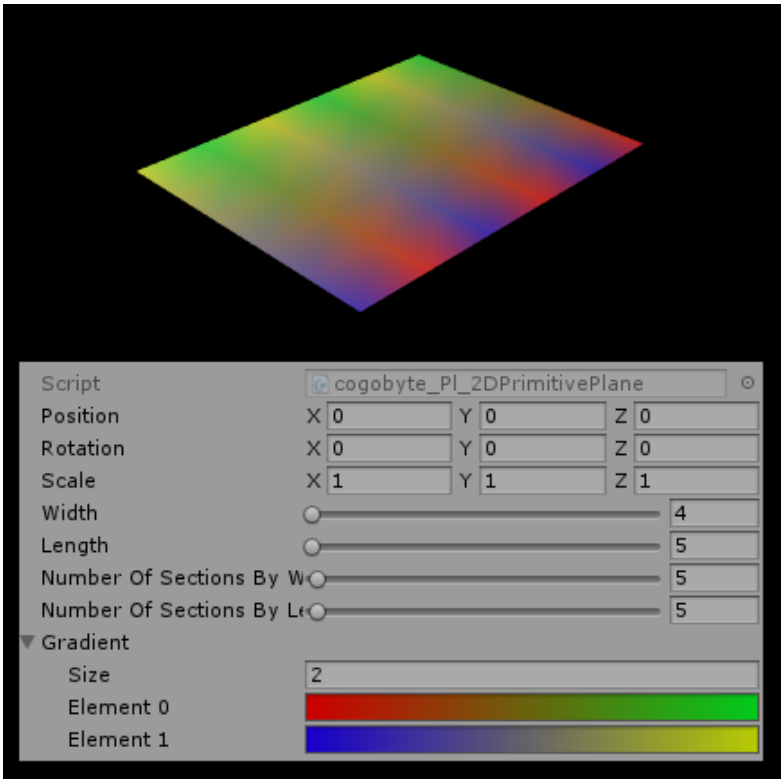

Figure 3: Plane primitive

4.3 Custom Primitive

Custom primitive enables using an existing mesh asset as a 2D primitive. Outline will be calculated automatically.

4.4 Custom Poly Primitive

Custom poly primitive will make a polygon using a list of points. It uses an ear clipping algorithm. It will color the vertices using the gradient parameter (it uses only the first gradient in array). It has a custom editor that enables manipulation of points. It must have minimum 3 points. To add a point click on the line that connects any two points. To delete a point, shift click it. Each point has a text that shows its position and a handle to drag it around. Parameter handleRadius determines the radius of that handle and

show position text is a toggle to show/hide text position. Never set a point in a way that makes lines intersect each other.

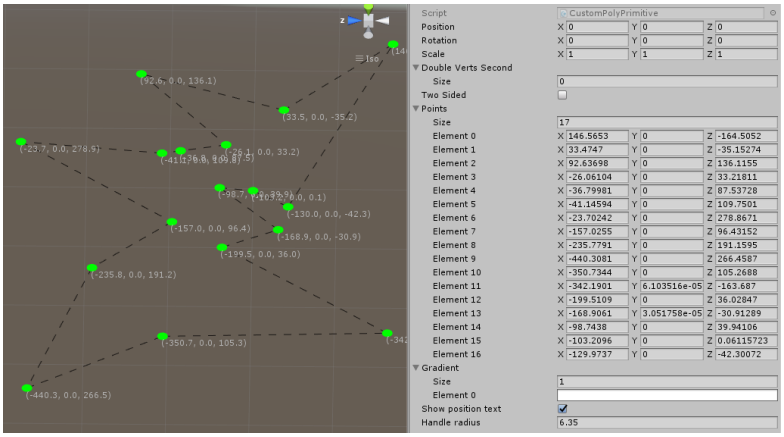

Figure 4: Custom Poly Primitive

4.5 Primitive List

A primitive list is just a list of primitives. It is used for more complex shapes (like the extrude that can change shape along path). Just create a list and drag and drop primitives that you want to use.

5 Solid Meshes

A solid mesh is just a classification of a procedural mesh. It is a full 3d mesh.

5.1 Sphere Mesh

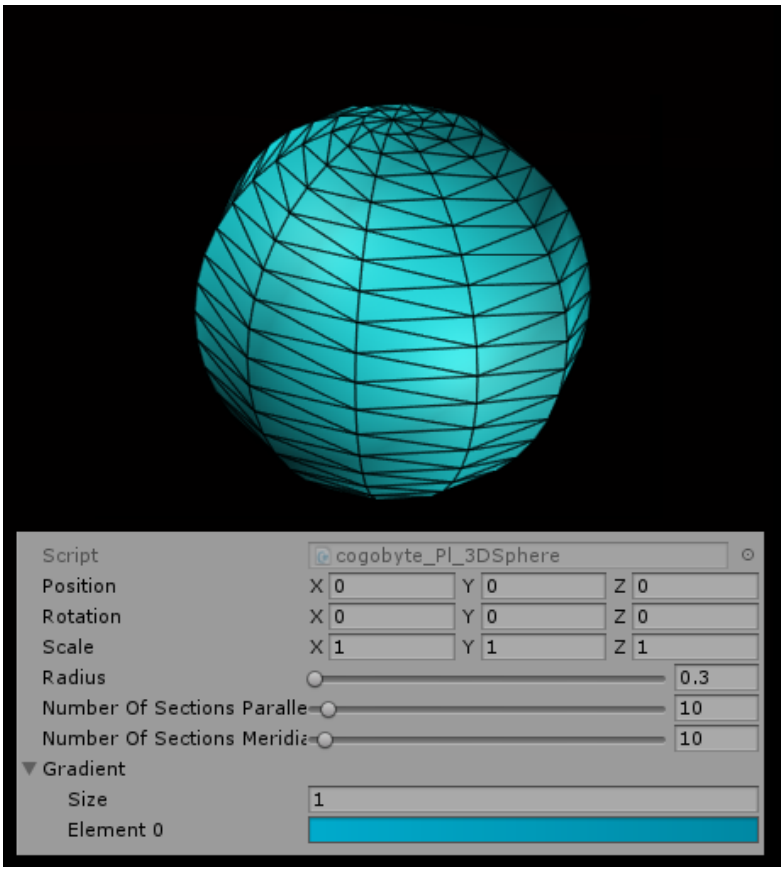

Figure 5: Sphere Mesh

5.2 Custom Mesh

Custom primitive enables using an existing mesh asset as a Solid Mesh.

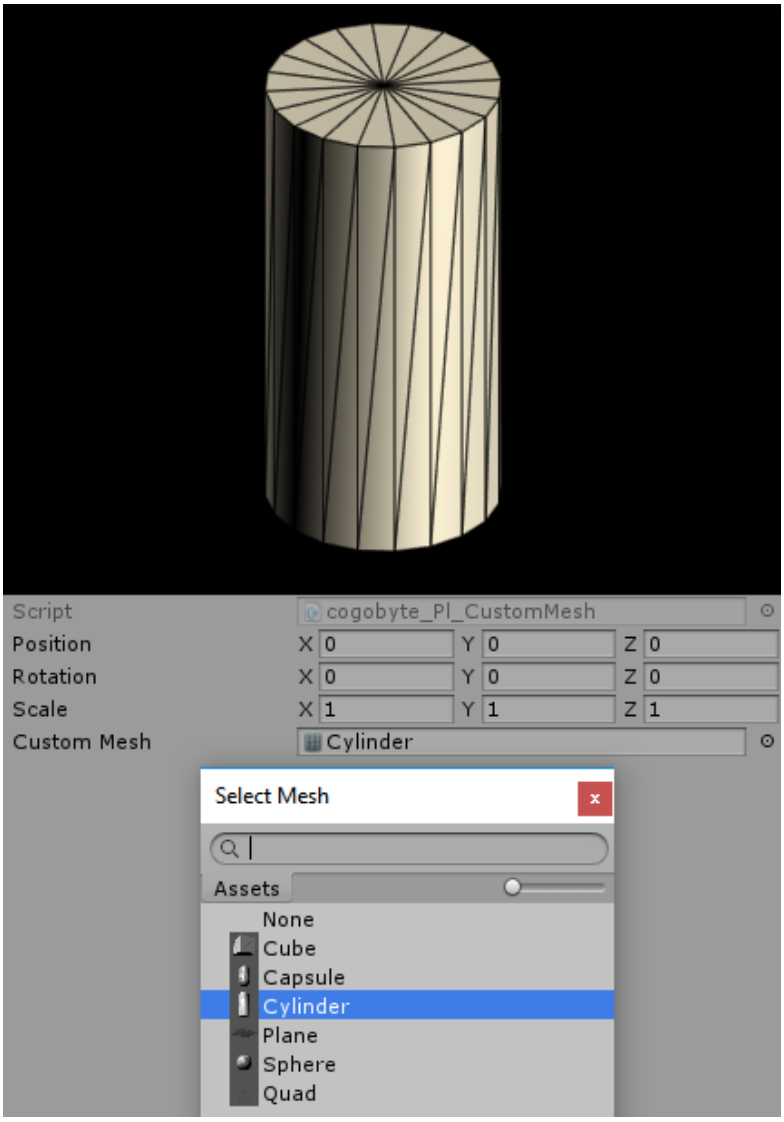

Figure 6: Custom Mesh

5.3 Winged Arrow Mesh

Winged arrow mesh creates an sharp arrow. It has options for the positions of left, middle and right vertices for front side and back side.

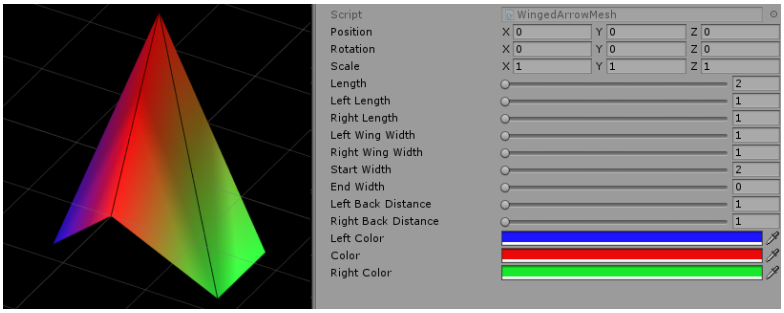

Figure 7: Winged Arrow Mesh

5.4 Meshes List

A list of procedural meshes for more complex shapes. Just drag and drop meshes that you want to use.

6 Path Arrays

All path arrays are derived from the PathArray class. PathArray class contains properties common for all path arrays: Basic transform properties (translation,rotation and scale) and obstacle projection properties. Closed parameter will add a final point at the start of the path.

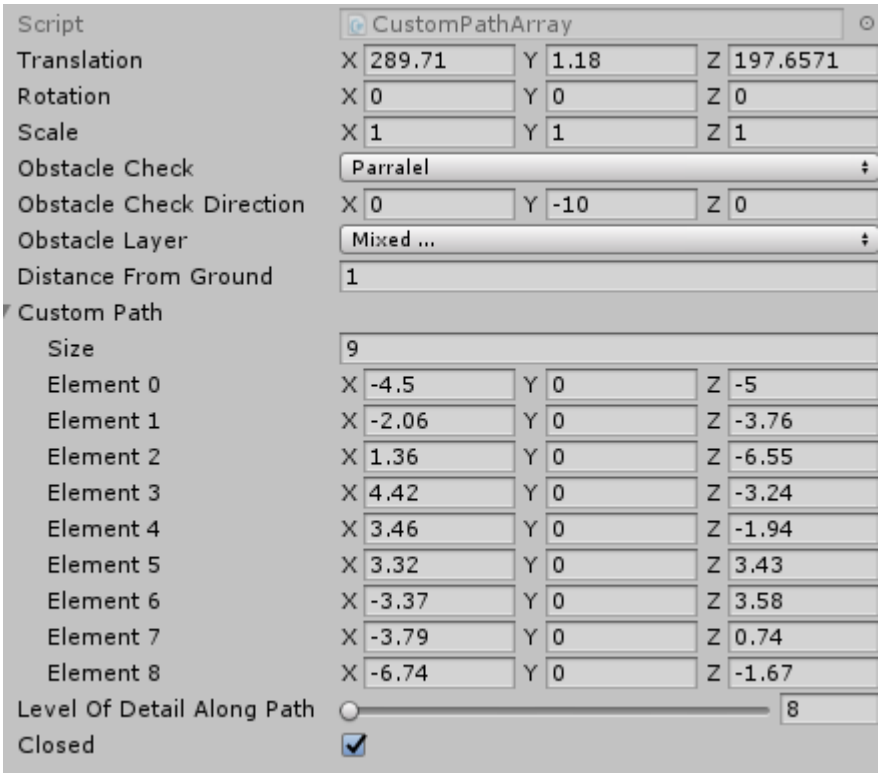

Figure 8: Path Array Options

Obstacle check options define projections of path onto objects. Parallel obstacle check option will project parallel rays from path array points while projection obstacle check option will project from the position of the path array. Obstacle check direction determines the direction and distance of the projection origin. Obstacle layer will project only to objects with specified layer. Distance from ground will make path array hover above projected points for a given distance.

6.1 Circular path array

Circular path array has radius and levelOfDetail. Radius determines the radius of the circle and levelOfDetail determines number of path points along circle.

6.2 Custom path array

Custom path array is an array of path points. Path will be generated as lines between each consecutive point and level of detail along path determines number of points that will be generated for each line. Points can be edited in 3d space by dragging them arround. To make an additional point on the center of line, select the start point of the line and click on the button split line. To delete a point select it and click the delete button. Points can also be directly edited using the inspector point list like any other list.

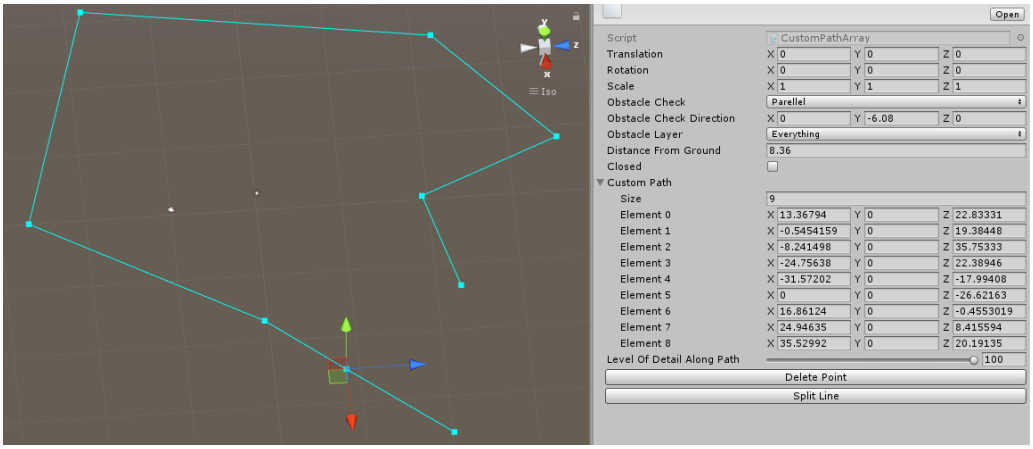

Figure 9: Custom Path Array

6.3 Bezier Path array

Bezier path array generates points using the provided bezier spline. Level of detail determines number of generated points along whole spline. Bezier spline is a list of bezier points (each two consecutive points make a bezier curve with 2 control points). Each point has two control points: one for the end of the previous curve and one for the start of the next bezier curve. To make an additional point on the center of curve, select the start point of the curve and click on the button split curve. To delete a point select it and click the delete point button. Each point can be moved around using the gizmo and has 2 additional gizmos for control points. There are 3 modes for control points. Free mode enables editing control points independently. Mirror mode mirrors the position of other control point around the main point, aligned just keeps the curve smooth by aligning only the angle between two control

points around the main point. Button add point will add a new point close to the last point in the spline.

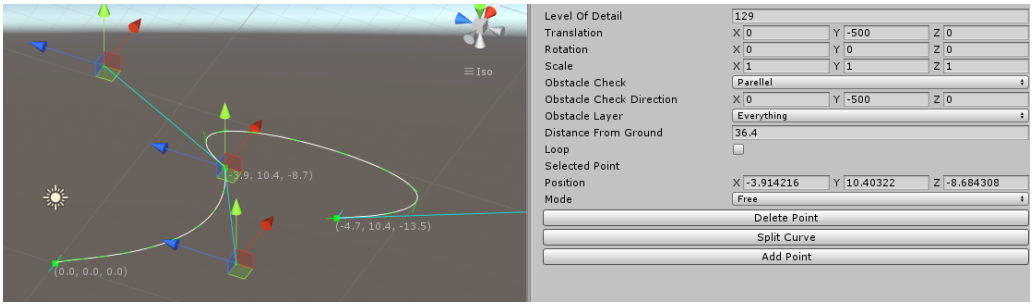

Figure 10: Custom Path Array