WirePrimitives

v.1.0

Wire primitives package allows you procedurally create wireframe meshes. Generated meshes are native Unity meshes that is created with flag "**`MeshTopology.Lines`**". This meshes are created in 3d space.

Important!

Due to specific rendering of meshes with **`MeshTopology.Lines`** type there is no way to change thickness (stroke) of generated lines – line always renders with thickness of one pixel.

Getting started

To create a primitive, you can select a GameObject on which you want place a primitive and add one of the WirePrimitive component either by selecting from menu "`Component/WirePrimitives`" or dragging script from "Scripts" folder in package folder. In order to see generated mesh, you must add MeshRenderer component to this GameObject and set some size parameters to non-zero values.

You can save generated mesh using script "`SaveMeshButton.cs`" – just add this script to GameObject with generated mesh and press button "Save Mesh" in script inspector.

Texturing

For some primitives, you can generate UV coordinates for creating different line types such as dashed or center lines. The material for such lines must have a texture with transparent parts (see for example "Dash.png" in "Textures" folder in package). UV coordinates generated only for "U" direction, "V" coordinate always have value zero. The length of ticks can be changed through material texture property "`Tiling X`".

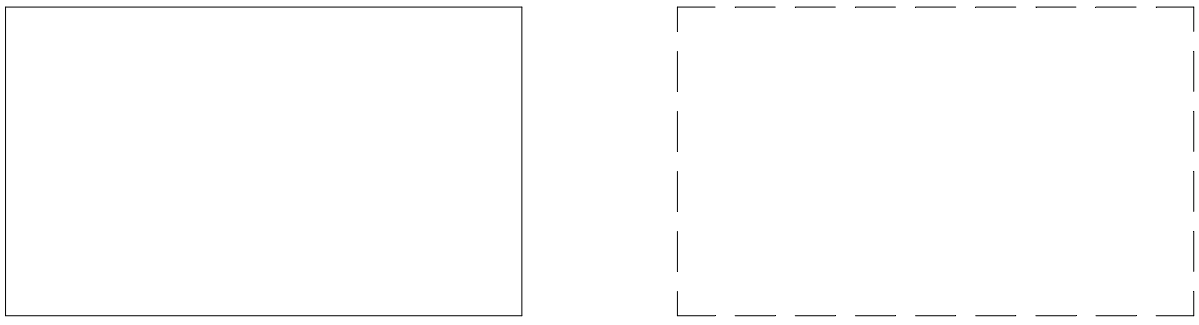


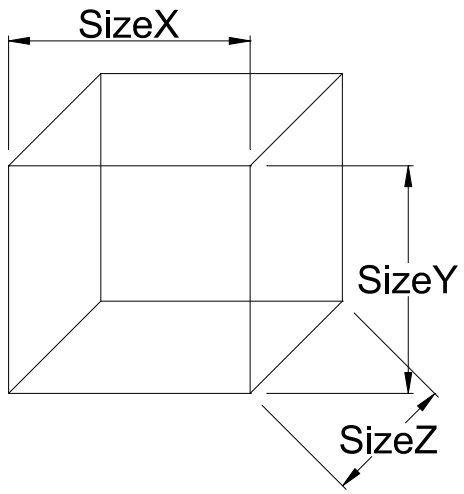
Fig.1 Left rectangle without UV coordinates, right rectangle has UV coordinates and material with "Dash.png" texture.

Primitives

1 3D Primitives

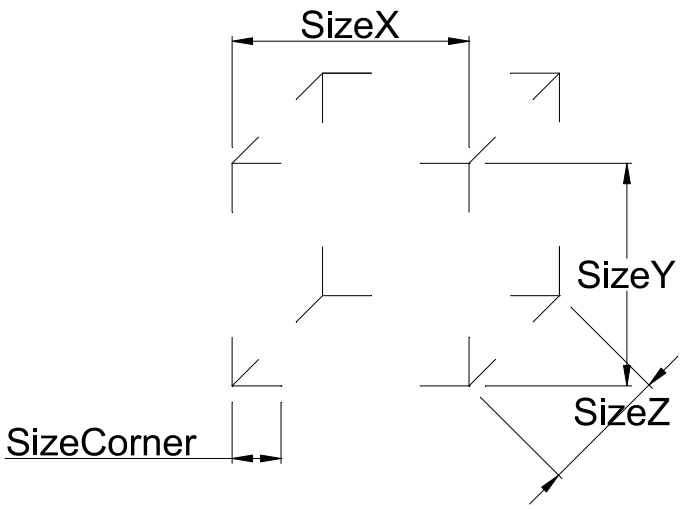
1.1 WireBox

Box have 3 sizes along each axis. You can change pivot position for this primitive by selecting alignment along X, Y, Z axes (X: `left, center, right; Y: bottom, center, top; Z: front, center, back).` Box can be textured.



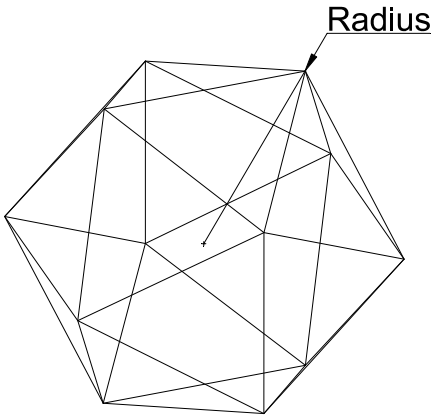
1.2 WireCornerBox

Same as WireBox, but have an additional parameter SizeCorner that responsible for how much of sides are drawn.



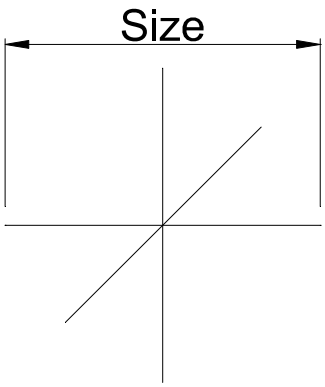
1.3 WireIcoSphere

IcoSphere have two parameters: Radius and RecursionLevel. RecursionLevel controls how dense would be the sphere mesh.



1.4 WireCross3D

Has only one parameter – size. For example, this primitive can be used as center mark for sphere.



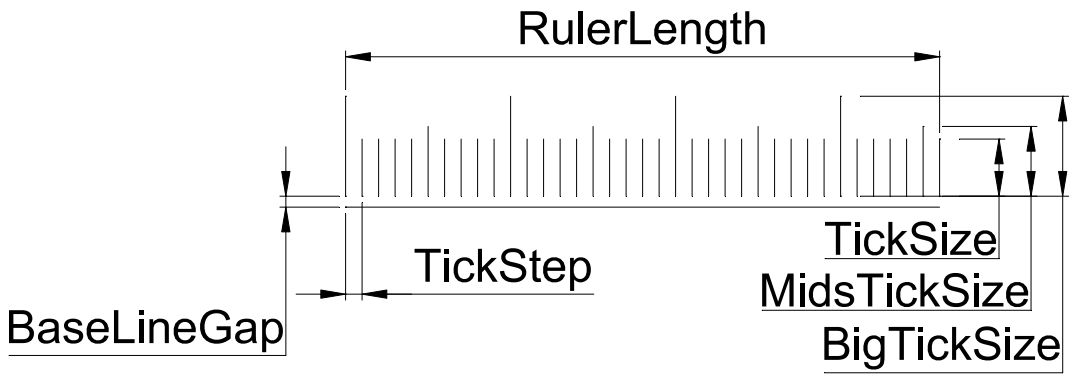
2 2D Primitives

In 2d Primitives “WireRuler”, “WireProtractor” and “WireRectangleGrid” generates with 3 submeshes and therefore requires 3 materials in MeshRenderer component.

2.1 WireRuler

parameters:

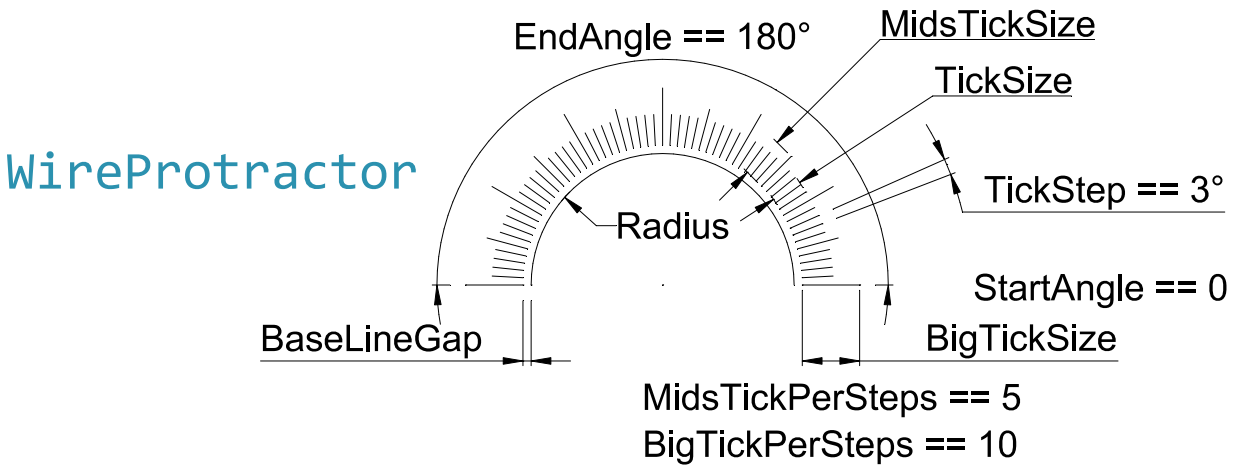
`Direction` – ruler can be horizontal or vertical
`Align` – controls where ruler pivot is placed
`rulerLength` – Ruler length in scene units
`tickStep` – Distance between two adjucent ticks
`tickSize` – Length of ordinary (short) ticks
`midsTickSize` – Length of mid size ticks
`midsTickPerSteps` – Number of steps between mid ticks. Changing this parameter controls how far middle size ticks spaced from each other.
`bigTickSize` – Length of big size ticks
`bigTickPerSteps` – Number of steps between big ticks. Changing this parameter controls how far longest ticks spaced from each other.
`bigTickPerSteps` – Number of steps between big ticks. Changing this parameter controls how far longest ticks spaced from each other.
`baseLine` – bottom line of ruler
`baseLineGap` – gap between bottom line and ticks



2.2 WireProtractor

parameters:

direction - Direction of ticks relative to arc. Tick can points inside circle or outside.
radius - Protractor radius (base arc radius)
startAngle - Start angle of arc in degrees (0 degrees at 3 o'clock)
endAngle - End angle of arc in degrees
tickStep - Ticks step in degrees
tickSize - Length of ordinary ticks
midsTickSize - Length of mid size ticks
midsTickPerSteps - Number of steps between mid ticks
bigTickSize - Length of big ticks
bigTickPerSteps - Number of steps between big ticks
baseArc - Does we need to draw arc for protractor
baseArcGap - Distance from arc to ticks



ACADCursor, WireArc, WireCircle, WireCorner2D, WireCross2D, WireRectangleGrid, WireRectangle, WireChamferRectangle, WireRectangleCorners – these primitives can be generated for different planes (XY, XZ or YZ). You can select desired plane in dropdown named **Axes**.

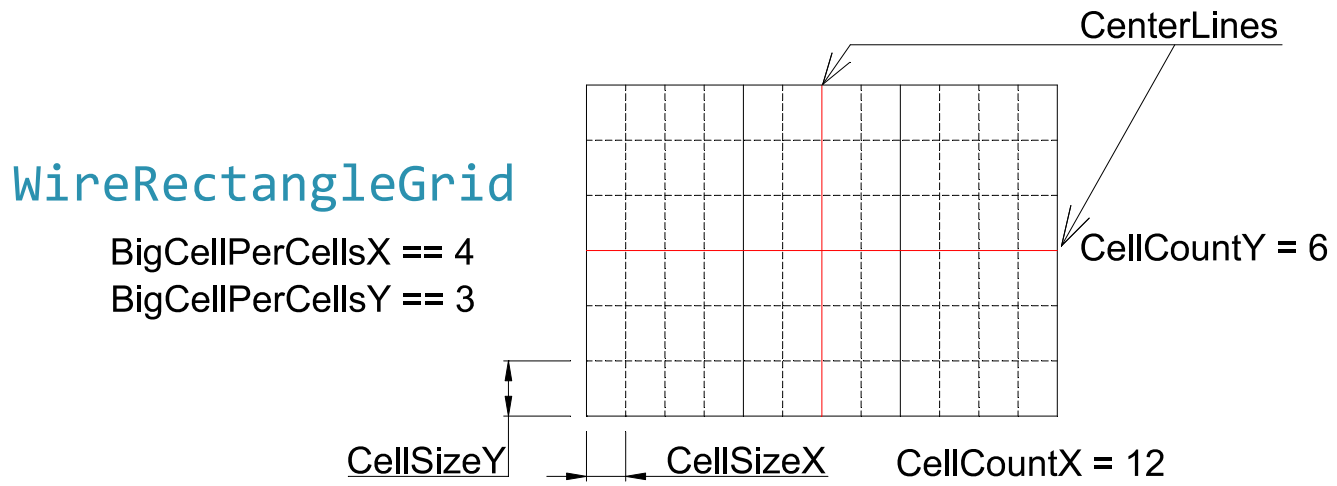
WireRectangleGrid, WireRectangle, WireChamferRectangle, WireRectangleCorners – is a 2d primitives that can have different pivots by setting **XAlign** (left, center, right) and **YAlign** (bottom, center, top) properties which produces pivots at corners, middle points of sides and center of primitive.

2.3 WireRectangleGrid

WireRectangleGrid is a rectangular grid with 3 submeshes (materials) for different line types.

parameters:

cellSizeX - Cell size along first axis (usually X)
cellSizeY– Cell size along second axis (usually Y)
cellCountX – Number of cells in first direction
cellCountY – Number of cells in second direction
bigCellPerCellsX– Number of cells between major lines in first direction
bigCellPerCellsY – Number of cells between major lines in second direction. Major lines are drawn by second material from MeshRenderer materials array. See picture below where major lines are drawn by continuous lines.
drawCenterLines– If you have even number of cells then grid center line can be drawn by dedicated material by switching this to "true"

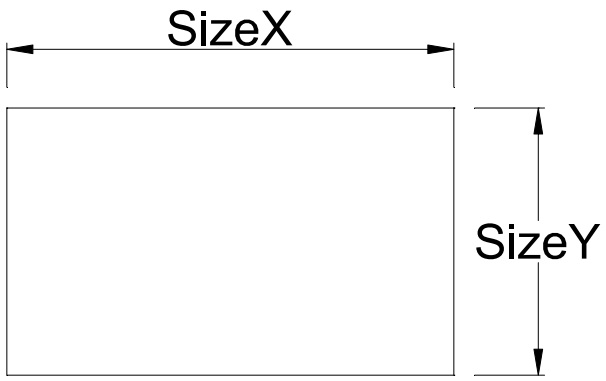


2.4 WireRectangle

parameters:

sizeX – Width of rectangle
sizeY – Height of rectangle

WireRectangle



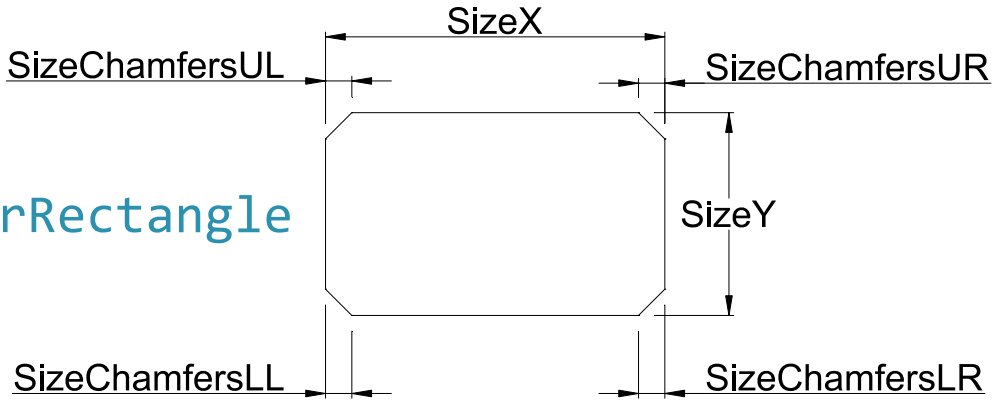
2.5 WireChamferRectangle

Rectangle with cutted corners

parameters:

sizeX – Width of rectangle (if it lay in XY plane)
sizeY – Height of rectangle (if it lay in XY plane)
sizeChamferLL – Size of cutting along rectangle sides at lower left corner
sizeChamferUL – Size of cutting along rectangle sides at upper left corner
sizeChamferUR – Size of cutting along rectangle sides at upper right corner
sizeChamferLR – Size of cutting along rectangle sides at lower right corner

WireChamferRectangle



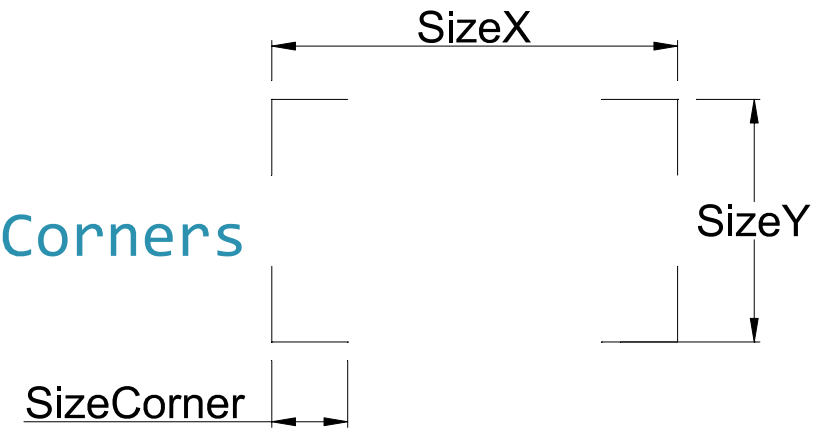
2.6 WireRectangleCorners

Four "L" shapes that form a rectangle

parameters:

`sizeX` – Width of rectangle
`sizeY` – Height of rectangle
`sizeCorner` – Size of "L" segments

`WireRectangleCorners`

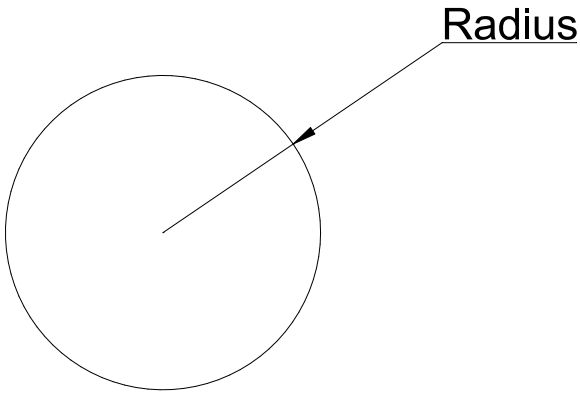


2.7 WireCircle

parameters:

`radius` – radius of circle

`WireCircle`

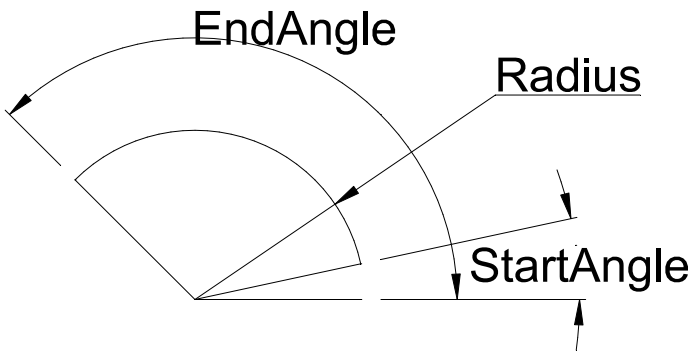


2.8 WireArc

parameters:

radius – Arc radius
startAngle – Start angle of arc in degrees (0 degrees at 3 o'clock)
endAngle – End angle of arc in degrees

WireArc



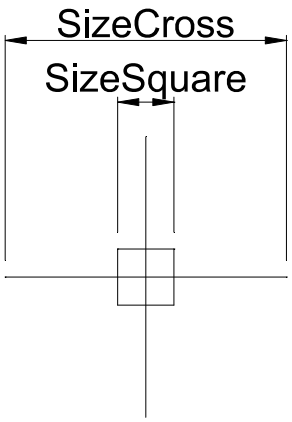
2.9 ACADCursor

Cross with square in center like AutoCAD cursor

parameters:

sizeCross – Length of lines for cross
sizeSquare – Length of square side

ACADCursor



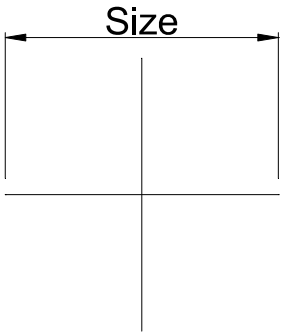
2.10 WireCross2D

Wire cross like "+" sign

parameters:

`size` – Length of lines for cross

WireCross2D



2.11 WireCorner2D

Wire mesh "L" shape

parameters:

`size` – Size of each "L" segment

WireCorner2D