# Cartesian Coordinate Node

Documentation for the Cartesian Coordinate node available in the Unity Asset Store.  This implementation is based on the example in How to Create a Custom Shader Graph Node (https://gamedevbill.com/custom-shader-graph-nodes/)

Online docs: https://gamedevbill.com/cartesian-coordinate-shader-graph-node/

## Description

This node for the Unity Shader Graph converts a 2D coordinates from Polar Coordinate form into standard UV Cartesian Coordinates.  It is the reverse of the Unity-provided Polar Coordinate node <link: https://github.com/Unity-Technologies/ScriptableRenderPipeline/blob/master/com.unity.shadergraph/Documentation~/Polar-Coordinates-Node.md >

Cartesian coordinates, also known as rectangular coordinates, involve representing a two dimensional point by an x and y offset. UV coordinates used for texture sampling are in this form.  In Polar form, this point is instead represented as a distance from the center, and angle of rotation.

## Ports

| Name | Direction | Type | Binding | Description |
|---|---|---|---|---|
| Polar Coordinates | Input | Vector 2 | UV | Input coordinates in Polar form |
| Center | Input | Vector 2 | None | Center reference point |
| Radial Scale | Input | Vector 1 | None | Scale of distance value |

| | | | | |
|---|---|---|---|---|
| Length Scale | Input | Vector 1 | None | Scale of angle value |
| Out | Output | Vector 2 | None | Output coordinates in Cartesian (or Rectangular) form |

## Full Code

```
#ifndef CARTESIAN_INCLUDE
#define CARTESIAN_INCLUDE

//Code to reverse the effects of Unity's Polar Coordinate node:
//
https://github.com/Unity-Technologies/ScriptableRenderPipeline/blob/master/com.unity.shadergraph/Documentation~/Polar-Coordinates-Node.md
// more info at gamedevbill.com
void CartesianCoords_float(float2 PolarCoords, float2 Center, float RadialScale, float LengthScale, out float2 UV)
{
	// reverse the magic number division that occurs inside Unity's node
	float2 adjustedCoord = PolarCoords * float2(0.5, 6.28);

	// reverse the scaling factors (why is one called "LengthScale"?  just copying Unity's name, not sure why they called it that)
	adjustedCoord = adjustedCoord / float2(RadialScale, LengthScale);

	// standard polar to cartesian math
	float2 result;
	result.x = sin(adjustedCoord.y) * adjustedCoord.x;
	result.y = cos(adjustedCoord.y) * adjustedCoord.x;

	//our polar coords had set 0,0 to be at "Center", and we need it to be at the corner instead.
	UV = result + Center;
}

#endif //CARTESIAN_INCLUDE
```