# Plot Scripting Reference

0.0.14

## Classes


- Plot
- Plot.Polygon
- Plot.Polyline


## class Plot

**static void DrawCircle( float x, float y, float diameter )**
**static void DrawCircle( Vector2 position, float diameter )**
Draw a circle using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void DrawRing( float x, float y, float innerDiameter, float outerDiameter )**
**static void DrawRing( Vector2 position, float innerDiameter, float OuterDiameter )**
Draw a ring using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void DrawPie( float x, float y, float diameter, float angleBegin, float angleEnd, float cutOff, float roundness )**
**static void DrawPie( Vector2 position, float diameter, float angleBegin, float angleEnd, float cutOff, float roundness )**
Draw a pie using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void DrawArc( float x, float y, float innerDiameter, float outerDiameter, float beginAngle, float endAngle, float cutOff, float roundness, bool useGeometricRoundness, bool constrainAngleSpanToRoundness )**
**static void DrawArc( Vector2 position, float innerDiameter, float outerDiameter, float beginAngle, float endAngle, float cutOff, float roundness, bool useGeometricRoundness, bool constrainAngleSpanToRoundness )**
Draw an arc using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting. Angles in degrees. AngleBegin must be smaller than AngleEnd.

**static void DrawRect( float x, float y, float width, float height )**
**static void DrawRect( float x, float y, float width, float height, float roundness )**
**static void DrawRect( float x, float y, float width, float height, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawRect( Vector2 position, float width, float height, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawRect( Vector2 position, float width, float height, float roundness )**
Draw a rectangle using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void DrawSquare( float x, float y, float size )**
**static void DrawSquare( float x, float y, float size, float roundness )**
**static void DrawSquare( float x, float y, float size, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawSquare( Vector2 position, float size, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawSquare( Vector2 position, float size, float roundness )**
Draw an arc using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void DrawLine( float ax, float ay, float bx, float by )**
**static void DrawLine( float ax, float ay, float bx, float by, StrokeCap caps )**
**static void DrawLine( float ax, float ay, float bx, float by, StrokeCap beginCap, StrokeCap endCap )**
**static void DrawLine( Vector2 positionA, Vector2 positionB, StrokeCap beginCap, StrokeCap endCap )**
**static void DrawLine( Vector2 positionA, Vector2 positionB, StrokeCap caps )**
**static void DrawLine( Vector2 positionA, Vector2 positionB )**
Draw a line using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void DrawPolygon( Polygon polygon )**
Draw a polygon using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void DrawPolyline( Polyline polyline, StrokeCap beginCap, StrokeCap endCap )**
**static void DrawPolyline( Polyline polyline, StrokeCap caps )**
Draw a polygon using Graphics.DrawMesh. This supports Unity's instancing, culling, and sorting.

**static void SetFillColor( float brightness )**
**static void SetFillColor( float brightness, float alpha )**
**static void SetFillColor( float red, float green, float blue )**
**static void SetFillColor( float red, float green, float blue, float alpha )**
**static void SetFillColor( Color color, float alphaOverride )**
**static void SetFillColor( Color color )**
Set the fill color to be used for subsequently drawn shapes.

**static void SetNoFillColor()**
Set no fill for subsequently drawn shapes. This will effectively set the fill color to (0,0,0,0) and forget the fill texture;

**static void SetStrokeColor( float brightness )**
**static void SetStrokeColor( float brightness, float alpha )**
**static void SetStrokeColor( float red, float green, float blue )**
**static void SetStrokeColor( float red, float green, float blue, float alpha )**
**static void SetStrokeColor( Color color, float alphaOverride )**
**static void SetStrokeColor( Color color )**
Set the stroke color to be used for subsequently drawn shapes.

**static void SetNoStrokeColor()**
Set no stroke for subsequently drawn shapes.

**static void SetStrokeWidth( float width )**
Set the stroke width (thickness) to be used for subsequently drawn shapes.

**static void SetStrokeAlignement( StrokeAlignment alignment )**
Set the stroke alignment to be used for subsequently drawn shapes.

**static void SetStrokeCornerProfile( StrokeCornerProfile cornerStyle )**
Set the stroke corner profile to be used for subsequently drawn shapes.

**static void SetAntiAliasing( bool isOn )**
Enable or disable pixel shader SDF based antialisaing for all subsequently drawn shapes. Note that edge alignment between shapes will not be seamless when anti-alisation is enabled.

**static void SetBlend( Blend blend )**
Set the blend mode used for subsequently drawn shapes.

**static void SetLayer( int layer )**
Set the layer used for subsequently drawn shapes. Does not work for DrawNow methods, just like Graphics.DrawMeshNow not regarding layers.

**static void SetPivot( Pivot pivot )**
Set the point from which Circle will be drawn. Default is Pivot.Center.

**static void PushStyle()**
Push (save) the current style to the stack.

**static void PopStyle()**
Pop (load) the last pushed style from the stack.

**static Style GetStyle()**
Copy and return the current style.

**static void SetStyle( Style style )**
Overwrite the current style.

**static void SetFillTexture( Texture texture )**
Set the fill texture to be used for subsequently drawn shapes. See also SetFillTextureUVRect, SetFillTextureBlend and SetFillTextureTint.

**static void SetNoFillTexture()**
Disable fill texture for subsequently drawn shapes.

**static void SetFillTextureUVRect( Rect uvRect )**
**static void SetFillTextureUVRect( float x, float y, float width, float height )**
Set the uv rect to be used for subsequently drawn shapes that has a fill texture.

**static void SetFillTextureBlend( FillTextureBlend blend )**
Set the texture blend mode to be used for subsequently drawn shapes that has a fill texture.

**static void SetFillTextureTint( float brightness )**
**static void SetFillTextureTint( float brightness, float alpha )**
**static void SetFillTextureTint( float red, float green, float blue )**
**static void SetFillTextureTint( float red, float green, float blue, float alpha )**
**static void SetFillTextureTint( Color color, float alphaOverride )**
**static void SetFillTextureTint( Color tint )**
Set the texture tint to be used for subsequently drawn shapes that has a fill texture.

**static void PushCanvas()**
Push (save) the current canvas transformation matrix to the stack.

**static void PopCanvas()**
Pop (load) the last pushed canvas transformation matrix from the stack.

**static Matrix4x4 GetCanvas()**
Get the current canvas transformation matrix.

**static void SetCanvas( Matrix4x4 matrix )**
**static void SetCanvas( Transform transform )**
Overwrite the current canvas transformation matrix.

**static void TranslateCanvas( float x, float y )**
**static void TranslateCanvas( float x, float y, float z )**
**static void TranslateCanvas( Vector2 translation )**
**static void TranslateCanvas( Vector3 translation )**
Translate the current canvas transformation matrix.

**static void RotateCanvas( float angleZ )**
**static void RotateCanvas( float angleX, float angleY, float angleZ )**
**static void RotateCanvas( Quaternion rotation )**
Rotate the current canvas transformation matrix by angle (in degrees).

**static void ScaleCanvas( float scaleXYZ )**
**static void ScaleCanvas( float scaleX, float scaleY )**
**static void ScaleCanvas( float scaleX, float scaleY, float scaleZ )**
**static void ScaleCanvas( Vector2 scale )**
**static void ScaleCanvas( Vector3 scale )**
Scale the current canvas transformation matrix.

**static void BeginDrawNowToTexture( RenderTexture rt )**
**static void BeginDrawNowToTexture( RenderTexture rt, Space space )**
**static void BeginDrawNowToTexture( RenderTexture rt, Color clearColor )**
**static void BeginDrawNowToTexture( RenderTexture rt, Space space, Color clearColor )**
Begin a DrawNowToTexture session. Call DrawXNow subsequently (for example DrawCircleNow) and don't forget to call EndDrawNowToTexture when you are done. For Space.Normalized 0,0 is center. Left, right, top and bottom is ( -aspect, aspect, -1, 1 ). For Space.Pixels 0,0 is in upper left corner.

**static void EndDrawNowToTexture()**
End a DrawNowToTexture session

**static void DrawCircleNow( float x, float y, float diameter )**
**static void DrawCircleNow( Vector2 position, float diameter )**
Draw a circle immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

**static void DrawRingNow( float x, float y, float innerDiameter, float OuterDiameter )**
**static void DrawRingNow( Vector2 position, float innerDiameter, float OuterDiameter )**
Draw a ring immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

**static void DrawPieNow( float x, float y, float diameter, float angleBegin, float angleEnd, float cutOff, float roundness )**
**static void DrawPieNow( Vector2 position, float diameter, float angleBegin, float angleEnd, float cutOff, float roundness )**
Draw a pie immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

**static void DrawArcNow( float x, float y, float innerDiameter, float outerDiameter, float beginAngle, float endAngle, float cutOff, float roundness, bool useGeometricRoundness, bool constrainAngleSpanToRoundness )**
**static void DrawArcNow( Vector2 position, float innerDiameter, float outerDiameter, float beginAngle, float endAngle, float cutOff, float roundness, bool useGeometricRoundness, bool constrainAngleSpanToRoundness )**
Draw a pie immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture. Angles in degrees. AngleBegin must be smaller than AngleEnd.

**static void DrawRectNow( float x, float y, float width, float height )**
**static void DrawRectNow( float x, float y, float width, float height, float roundness )**
**static void DrawRectNow( float x, float y, float width, float height, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawRectNow( Vector2 position, float width, float height, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawRectNow( Vector2 position, float width, float height, float roundness )**
Draw a rectangle immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

**static void DrawSquareNow( float x, float y, float size )**

**static void DrawSquareNow( float x, float y, float size, float roundness )**
**static void DrawSquareNow( float x, float y, float size, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawSquareNow( Vector2 position, float size, float lowerLeftRoundness, float upperLeftRoundness, float upperRightRoundness, float lowerRightRoundness )**
**static void DrawSquareNow( Vector2 position, float size, float roundness )**
Draw a square immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

**static void DrawLineNow( float ax, float ay, float bx, float by )**
**static void DrawLineNow( float ax, float ay, float bx, float by, StrokeCap caps )**
**static void DrawLineNow( float ax, float ay, float bx, float by, StrokeCap beginCap, StrokeCap endCap )**
**static void DrawLineNow( Vector2 positionA, Vector2 positionB, StrokeCap beginCap, StrokeCap endCap )**
**static void DrawLineNow( Vector2 positionA, Vector2 positionB, StrokeCap caps )**
**static void DrawLineNow( Vector2 positionA, Vector2 positionB )**
Draw a line immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

**static void DrawPolygonNow( Polygon polygon )**
Draw a polygon immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

**static void DrawPolylineNow( Polyline polyline, StrokeCap beginCap, StrokeCap endCap )**
Draw a polygon immediately using Graphics.DrawMeshNow. Call this from OnPostRender or after calling BeginDrawNowToTexture.

# class Plot.Polygon

**Polygon()**
**Polygon( int pointCount )**
**Polygon( Vector2[] points )**
**Polygon( List`1 points )**
Creates a new Polygon to be drawn using Plot.DrawPolygon(). Points must be provided in clockwise order.

**Polygon SetAsNGon( float diameter, int sideCount )**
Fill this polygon with a N-gon shape.

**Polygon SetAsStar( float innerDiameter, float outerDiameter, int armCount )**
Fill this polygon with a star shape.

# class Plot.Polyline

**Polyline()**
**Polyline( int pointCount )**
**Polyline( Vector2[] points )**
**Polyline( List`1 points )**
Creates a new Polyline to be drawn using Plot.DrawPolyline(). Points must be provided in clockwise order.

**Polyline SetAsBezierCurve( Vector2 anchorA, Vector2 controlA, Vector2 controlB, Vector2 anchorB, int resolution )**
Fill this polyline with bezier curve points.

# struct struct Plot.JChColor

**static JChColor Slerp( JChColor c1, JChColor c2, float t )**
Circular interpolation along the hue angle the cylendrical JCh color model.

**static JChColor Lerp( JChColor c1, JChColor c2, float t )**
Liniear interpolation through the cylendrical JCh color model.

**static Color[] LerpCreatePalette( Color colorA, Color colorB, int stepCount )**
Create a palette of colors by linear interpolation through the JCh color space.

**static Color[] SlerpCreatePalette( Color colorA, Color colorB, int stepCount )**
Create a palette of colors by circular interpolation along the hue angle through the JCh color space.