Definitive Painter User Manual

Plugin version	1.0.2
Supported UE versions	4.25, 4.26, 4.27
Supported platforms	Windows 64-bit
Latest Manual update	31 Jan 2022

Support	info@soucekmartin.cz
Twitter	https://twitter.com/DefPainter
Marketplace	https://www.unrealengine.com/marketplace/product/definitive-painter
Discord	https://discord.com/invite/aDhVsSSShB
Author	Martin Soucek

Table of Contents

Introduction

Definitive Painter is a 2D vector graphics plug-in based on Skia that enables you to draw various vector-based elements inside Unreal Motion Graphics in Unreal Engine. You can create anything from basic widgets to complex user interfaces without the need for any external tools. Everything can be done right inside Unreal Engine.

Definitive Painter seamlessly integrates with existing UMG system by introducing new Widgets. However, it does not replace UMG.

Why?

Unreal Motion Graphics has been known for its numerous limitations in terms of rendering complex dynamic shapes with antialiasing. Do you need to draw an intricate curve or a simple rectangle with rounded corners without firing up Photoshop? Bad luck! Definitive Painter aims to fill this gap that has been there for years.

How?

Definitive Painter is based on Skia by Google. It's an open source graphics library powering rendering in some web browsers and Android applications. Skia does its magic and saves whatever comes out of it into a buffer. Content of the buffer is then copied into a texture right in Unreal Engine.

Installation

1. Install the plugin from Epic Game Launcher to the engine.
2. Launch Unreal Editor, navigate to Edit > Plugins and enable Definitive Painter plugin and that's it! You're good to go!

Architecture

Everything in Definitive Painter is encapsulated in DP Canvas, a Widget managing rendering. All DP Widgets must be direct or indirect descendants of one of these.

While DP Canvas can contain only one child, other DP Widgets acts exactly like UMG Canvas Panel. Therefore, they can contain any number of arbitrarily placed children. Widget Tree can be constructed in any way you're used to. DP Widgets can contain default UMG Widgets and vice versa.

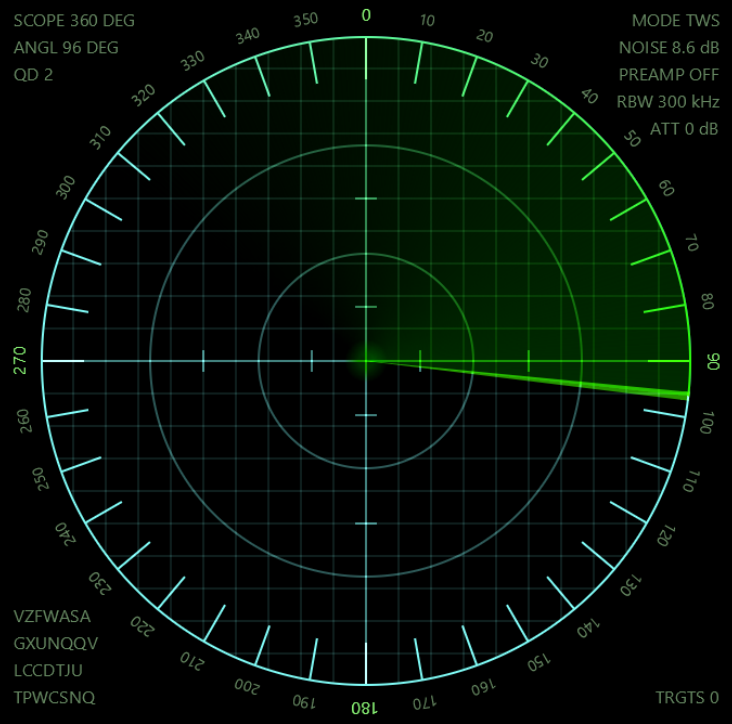
In order to actually draw something, DP Widgets must be provided with a Paint asset. The relationship between the two is similar to the one between meshes and materials in Unreal Engine. While DP Widgets define geometric shape of the Element, Paint defines color, gradient and other effects.

By default, every DP Widget draws one shape (rectangle, circle, text, etc.). DP Widgets can also draw additional shapes in the Quick Draw mode allowing for hundreds of shapes to be drawn without any significant performance impact. However, shapes drawn in this mode

have certain limitations in terms of what kinds of effects can be applied on them and are not subjects to UMG Widget Tree system. Lear more here.

Terminology

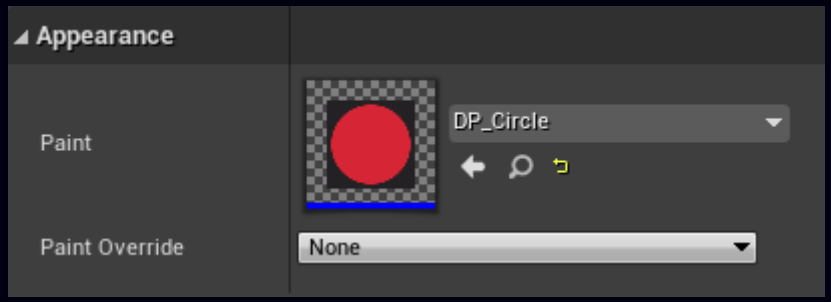
1. Widget
   Widget is the fundamental building block of the UI in Unreal Engine. Widgets are organized within a Widget Tree to form a UI. Definitive Painter extends this system by introducing own Widgets. Each DP Widget draws a single Element.
2. Element
   An Element is a single geometric shape that can be anything from a rectangle to a complex path.
3. Paint
   Paint controls the look of Elements. Paint can be defined as an asset and be reused on multiple Elements, or it can be created on the fly and altered dynamically.


SCOPE 360 DEG
ANGL 96 DEG
QD 2
MODE TWS
NOISE 8.6 dB
PREAMP OFF
RBW 300 kHz
ATT 0 dB
VZFWASA
GXUNQQV
LCCDTJU
TPWCSNQ
TRGTS 0
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350

Widgets

Widget Common parameters

Common parameters for all Widgets except for the DP Canvas.



Paint

The Paint asset used to draw the Element.

Paint Override

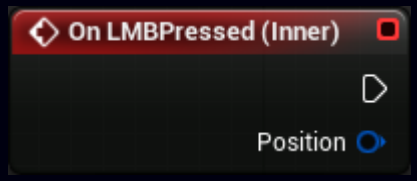
Optional Paint override. Does nothing if set to *None*, otherwise overrides the Paint asset. If the Paint asset is set at the time you set the Paint override, the Paint asset is copied into the Paint override.

On Draw Quick Elements

This event is used to draw so called *Quick Elements* which are basic shapes like rectangles, circles, and even paths, etc. within the bounds of the Widget. These Elements are not managed by the Widget Tree so they are much faster to render. You cannot interact with these Elements in any way while in the game.

A typical usage for *Quick Elements* is to render bars in a bar charts or ticks on a scale. The event is executed right after the Widget itself is rendered. The list of all *Quick Elements* can be found here.

On LMB Pressed
Fires when the left mouse button was just pressed and the Widget is hovered by the mouse.

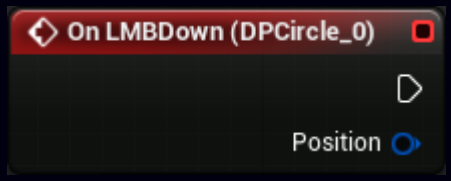


Position FVector2D
Mouse position relative to the top left corner of the Widget.

On LMB Released
Fires when the left mouse button was just released and the Widget is hovered by the mouse.

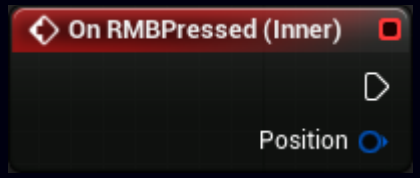
Position FVector2D
Mouse position relative to the top left corner of the Widget.

On LMB Down
Fires when the Widget is hovered by the mouse and the left mouse button is being held down.



Position FVector2D
Mouse position relative to the top left corner of the Widget.

On RMB Pressed
Fires when the right mouse button was just pressed and the Widget is hovered by the mouse.

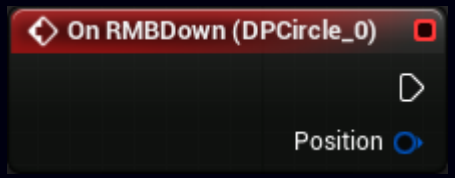


Position FVector2D
Mouse position relative to the top left corner of the Widget.

On RMB Released
Fires when the right mouse button was just released and the Widget is hovered by the mouse.

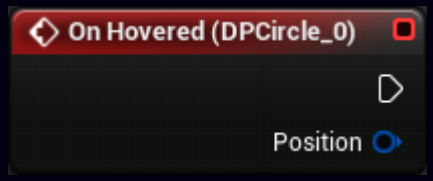
Position FVector2D
Mouse position relative to the top left corner of the Widget.

On RMB Down
Fires when the Widget is hovered by the mouse and the right mouse button is being held down.



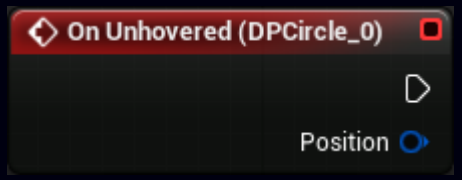
Position FVector2D
Mouse position relative to the top left corner of the Widget.

On Hovered
Fires when the mouse enters the Widget's geometric area.



Position FVector2D
Mouse position relative to the top left corner of the Widget.

On Unhovered
Fires when the mouse leaves the Widget's geometric area.



Position FVector2D
Mouse position relative to the top left corner of the Widget.

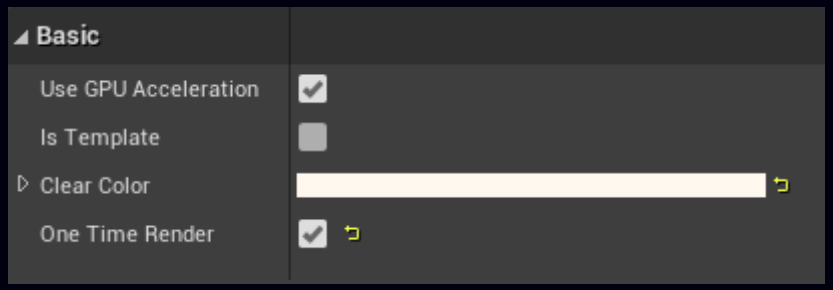
Note that all mouse-related events takes into account the exact geometric shape of the Widget and is not limited to its rectangular bounding box.

DP Canvas

This is the most important Widget of them all. It manages rendering of all descendant DP Widgets. Multiple DP Canvases can be present inside a single Widget Blueprint but you should keep the total number of them as low as possible as every DP Canvas on its own brings additional performance cost.

DP Canvas can contain only one child and doesn't draw anything on its own except for optional background color.

Parameters



Use GPU Acceleration

Enables GPU accelerated rendering of Elements. You will usually benefit from this when using complex effects and/or rendering many (>100) Elements. On the other hand, it's better to disable this option when rendering only a few Elements with a flat color as the GPU acceleration may perform worse in these scenarios.

Is Template

If a Widget Blueprint containing the DP Canvas marked with this flag is inserted as a child into another DP Canvas, all descendant DP Widgets will be rendered only into the topmost DP Canvas saving some performance. This is useful whenever you need to reuse the same Widget Blueprint multiple times - create a template.

Clear Color

This color is used to fill the background of DP Canvas before Elements are rendered.

One Time Render

Content of the DP Canvas will be rendered only once each time the Widget is rebuilt or spawned. This option saves a lot of performance and is useful when you need only a static Widget with no animations or user interaction.

DP Arc

This Widget draws an arc defined by the start and sweep angles. Width and height of the arc are defined by the size of the Widget. Center of the arc lies at the center of the Widget.

Parameters

Start Angle

Start angle of the arc in degrees. Starts at 3 o'clock and goes counterclockwise when increased.

Sweep Angle

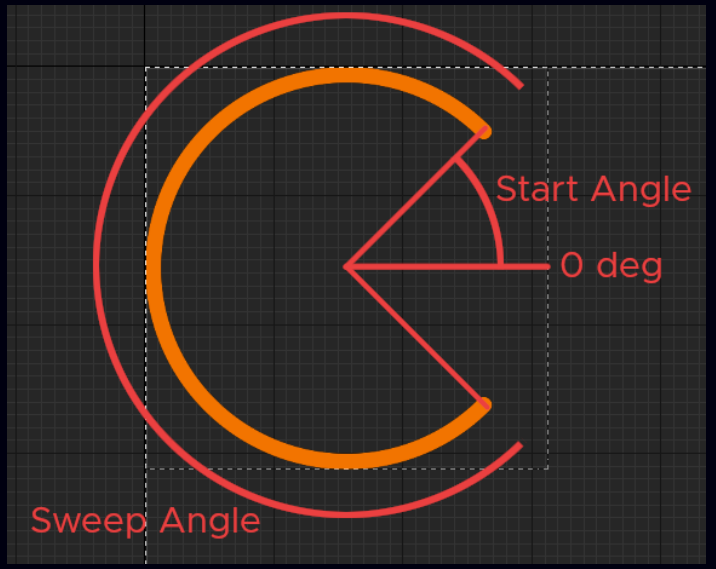
Defines the angle span of the arc in degrees. Starts at the Start Angle and goes counterclockwise.

Use Center

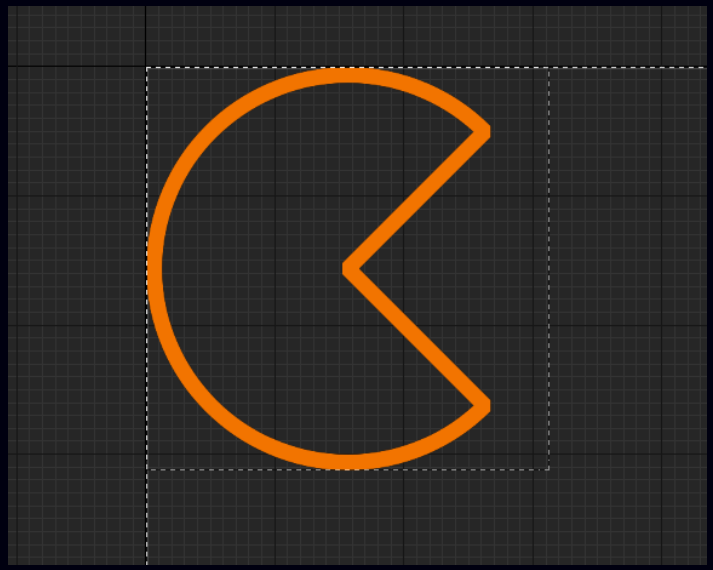
Connect both ends of the arc with its center. Has no effect when used with a filled Paint.

Aggressive Fit

When enabled, the size of the shape will be reduced by the width of the stroke to make it perfectly fit within the bounds. Has no effect when used with a filled Paint.



An arc with Start Angle 45 degrees and Sweep Angle 270 degrees.

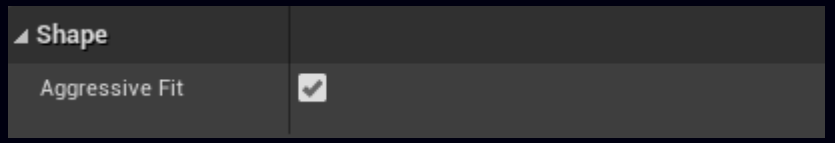


An arc with enabled Center.

DP Circle

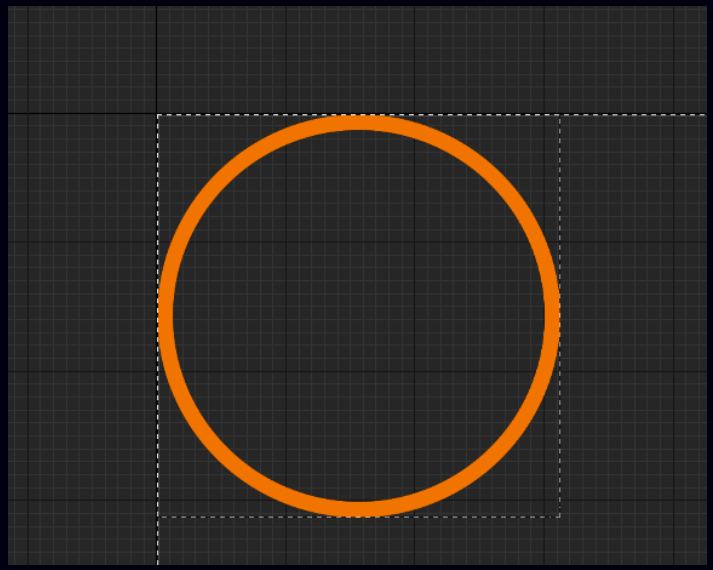
This Widget draws a circle. Radius of the circle is equal to half the smaller side of the Widget. Center of the circle lies at the center of the Widget.

Parameters



Aggressive Fit

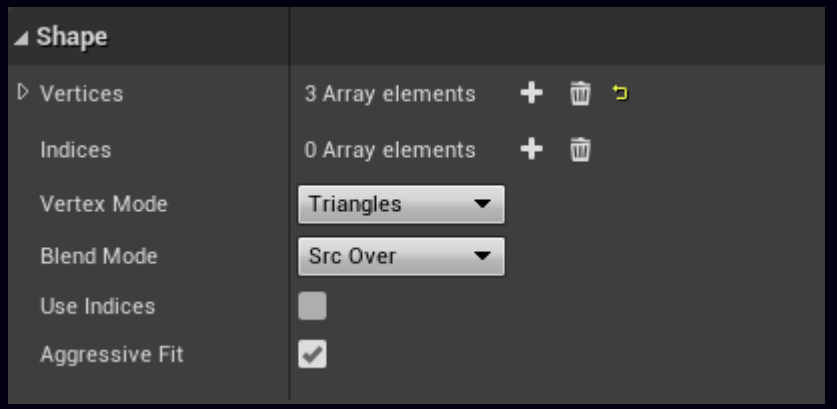
When enabled, the size of the shape will be reduced by the width of the stroke to make it perfectly fit within the bounds. Has no effect when used with a filled Paint.



DP Mesh

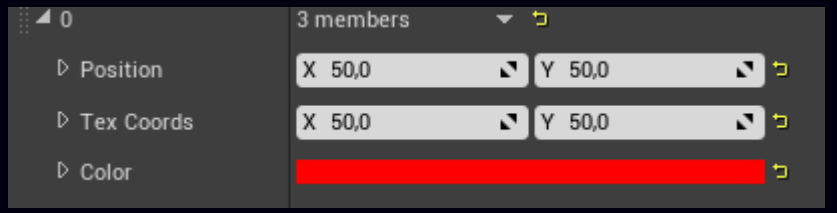
This Widget draws triangles defined by a set of vertices and optional indices. DP Mesh doesn't support stroked Paint. If you need to draw stroked (wireframe) triangles, use DP Path Widget instead.

Parameters



Vertices

An array of vertices defining the mesh. Only the *Position* parameter is required.



Position
Vertex position in screen space.

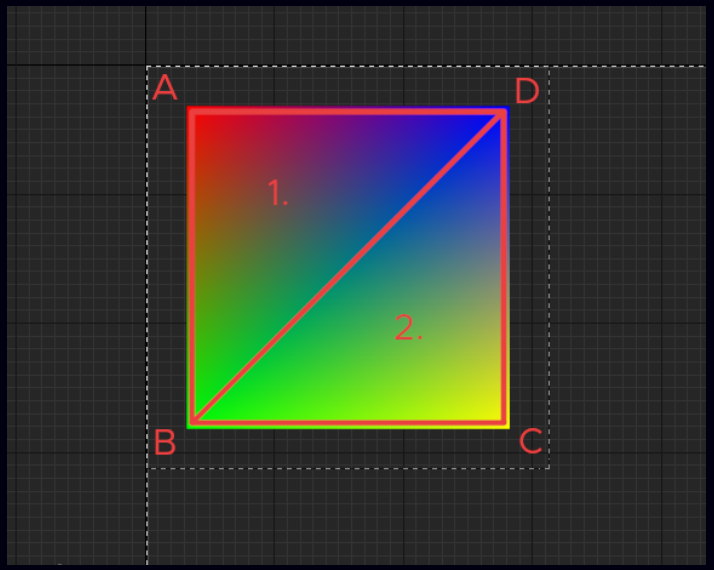
Tex Coords
Texture coordinates used when an effect is applied to the Paint. Value of this parameter is not in the usual range 0...1 but in the range 0...canvas width for the X axis and in the range 0...canvas height for the Y axis.

Color
Vertex color.

Indices

An array of integer indices defining the order in which the vertices are used to construct the mesh. Number of indices must be divisible by 3 as each triangle has 3 vertices thus total number of indices must be equal to the number of triangles times 3.

Each vertex can be used multiple times. Each index is equal to the position of a vertex in the vertex list beginning with 0. Using indices allows you to share vertices between multiple triangles thus saving some memory by defining vertices with same attributes (position, color, etc.) only once.



A rectangle drawn in the Triangles vertex mode using two triangles, four vertices (A = 50,50, B = 50, 450, C = 450, 450, D = 450, 50) and six indices (0, 1, 3, 3, 1, 2). The 1st triangle consists of vertices A, B, D (indices 0, 1, 3). The 2nd triangle consists of vertices D, B, C (3, 1, 2). Note that vertices forming each triangle are defined in so called "counterclockwise winding".

Vertex Mode

Defines the method of constructing the mesh.

A. Triangles
   Draws a list of triangles, one by one. Triangles may or may not share vertices and can be detached from each other.
B. Triangle Strip
   Draws a strip of triangles. The first triangle is defined by 3 vertices. Every additional vertex adds a triangle consisting of the vertex and the 2 previously added ones.

C. Triangle Fan

Draws a fan. The first triangle is defined by 3 vertices. The first vertex forms the center of the fan. Every additional vertex adds a triangle consisting of the vertex, the previously added vertex, and the vertex in the center of the fan.

Blend Mode

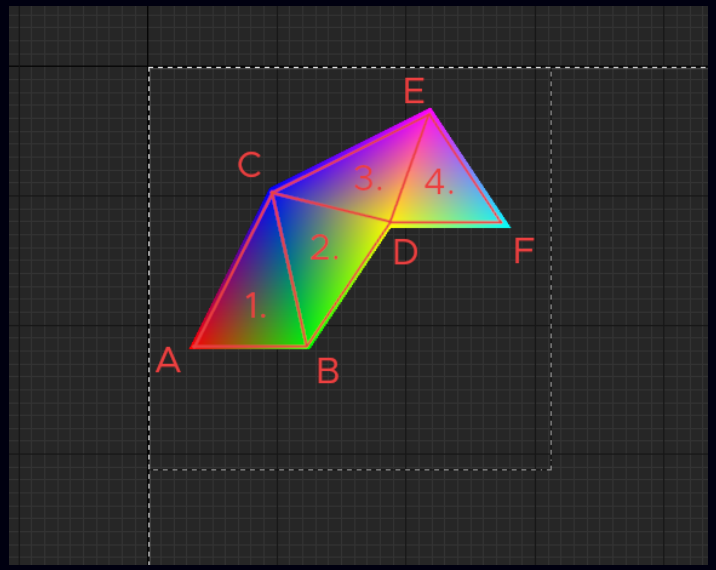
Defines the blending mode of vertex colors and the Widget's Paint.

Use Indices

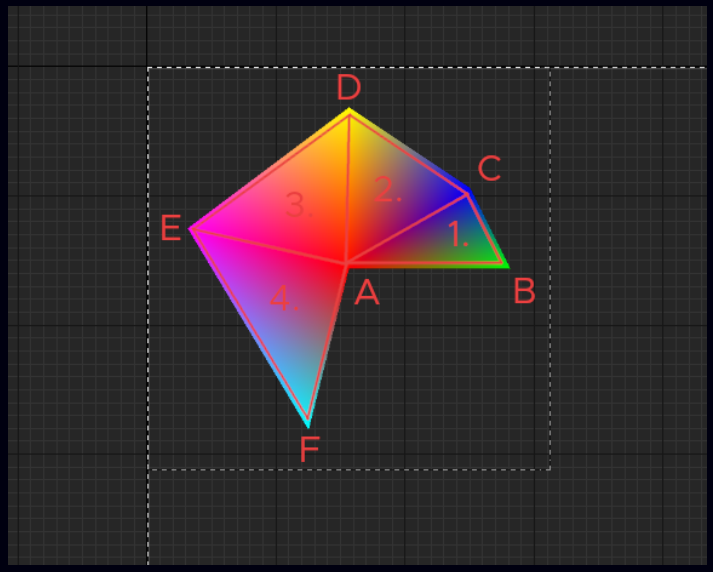
Use indices when constructing the mesh.

Aggressive Fit

Has no effect as DP Mesh can't have a stroked Paint.



A strip drawn using the Triangle strip vertex mode. Only six vertices (A, B, C, D, E, F) are needed to draw four triangles instead of twelve vertices that would be needed to draw the same shape in the Triangles vertex mode. The obvious downside of this mode is that adjacent triangles share an edge and can't be separated.

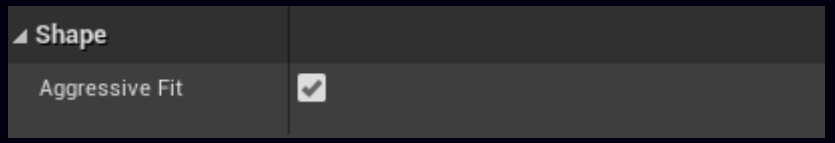


A fan drawn using the Triangle Fan vertex mode. Only six vertices (A, B, C, D, E, F) are needed to draw four triangles instead of twelve vertices that would be needed to draw the same shape in the Triangles vertex mode. The obvious downside of this mode is that adjacent triangles share an edge and can't be separated. Also, all triangles share the vertex forming the center of the fan which is always the first vertex of the mesh.

DP Oval

This Widget draw an oval. The width of the oval is equal to the width of the Widget and the height of the oval is equal to the height of the Widget. The center of the oval lies at the center of the Widget.

Parameters



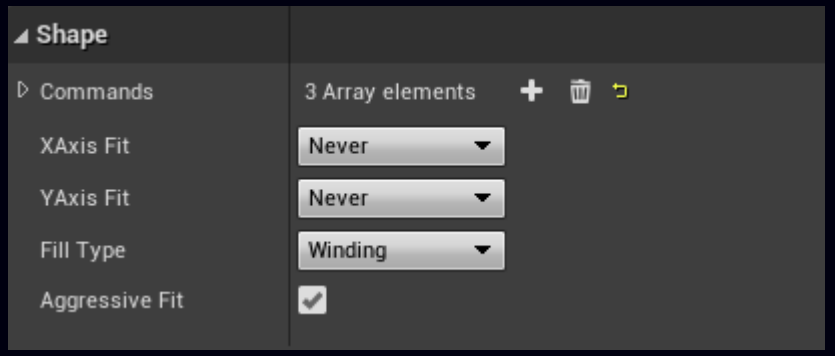
Aggressive Fit

When enabled, the size of the shape will be reduced by the width of the stroke to make it perfectly fit within the bounds. Has no effect when used with a filled Paint.

DP Path

DP Path can draw anything from a simple line to intricate curves and shapes. DP Path comprises of *commands* which defines individual parts of the path like lines, bézier curves, etc.

Parameters



Commands

An array of commands defining the path.

X Axis Fit

Scales the path to fit the Widget's width with the following options:

A. Never
   Disables fitting.
B. If Smaller
   Stretches the path if the path is narrower than the Widget.
C. If Bigger
   Shrinks the path if the path is wider than the Widget.

D. Always
Combines the last two options.

Y Axis Fit

Does the same as the previous parameter but for the Y axis.

Fill Type

Defines the method of filling the path.

A. Winding
Specifies that inner area of the path is computed by a non-zero sum of signed edge crossings. Has no effect for paths with stroked Paint.

B. Even-Odd
Specifies that inner area of the path is computed by an odd number of edge crossings. Has no effect for paths with stroked Paint.

C. Inverse Winding
Is inverted Winding.

D. Inverse Even-Odd
Is Inverted Even-Odd.

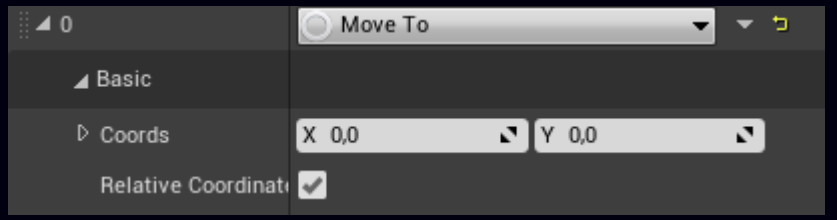
Aggressive Fit

When enabled, the size of the shape will be reduced by the width of the stroke to make it perfectly fit within the bounds. Has no effect when used with a filled Paint.

Commands

Move To

Adds beginning of a new contour at the specified coordinates.



Coords

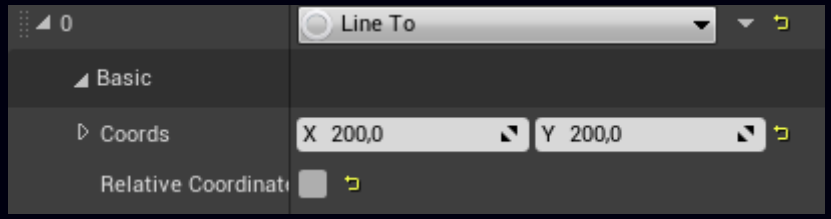
Coordinates at which the new contour should begin.

Relative Coordinates

If enabled, all coordinated specified in this command are relative to the last point of the path.

Line To

Adds a line from the last point of the path ending at the specified coordinates.



Coords

Coordinates at which the line ends.

Relative Coordinates

If enabled, all coordinated specified in this command are relative to the last point of the path.

Quad To

Adds a quad curve going from the last point of the path toward the *Control Point* and ending at the *End Point*.

Control Point

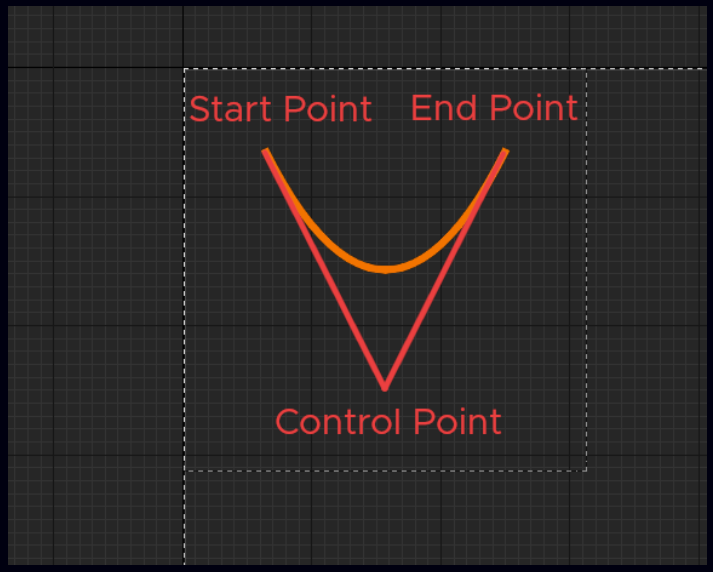
Defines the control point toward which the curve heads.

End Point

Defines the end point of the curve.

Relative Coordinates

If enabled, all coordinated specified in this command are relative to the last point of the path.



A quad curve drawn from the start point (100, 100) toward the control point (250, 400) ending at the end point (400, 100).

Cubic To

Adds a cubic curve going from the last point of the path toward the *1st Control Point*, then toward the *2nd Control Point* and ending at the *End Point*.

Control Point 1

Defines the 1st control point toward which the curve heads.

Control Point 2

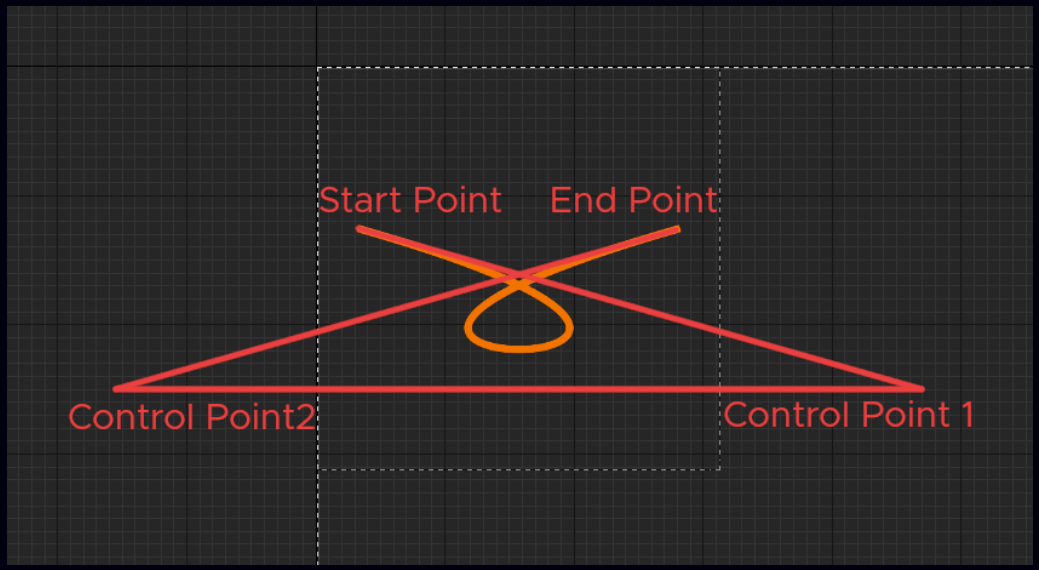
Defines the 2nd control point toward which the curve heads.

End Point

Defines the end point of the curve.

Relative Coordinates

If enabled, all coordinated specified in this command are relative to the last point of the path.



A cubic curve drawn from the start point (50, 200) toward the 1st control point (750, 400, then toward the 2nd control point (-250, 400) ending at the end point (450, 200).

Conic To

Adds a conic curve going from the last point of the path toward the *Control Point* and ending at the *End Point*. The curve is weighted by the parameter *Weight*.

Control Point

Defines the control point toward which the curve heads.

End Point

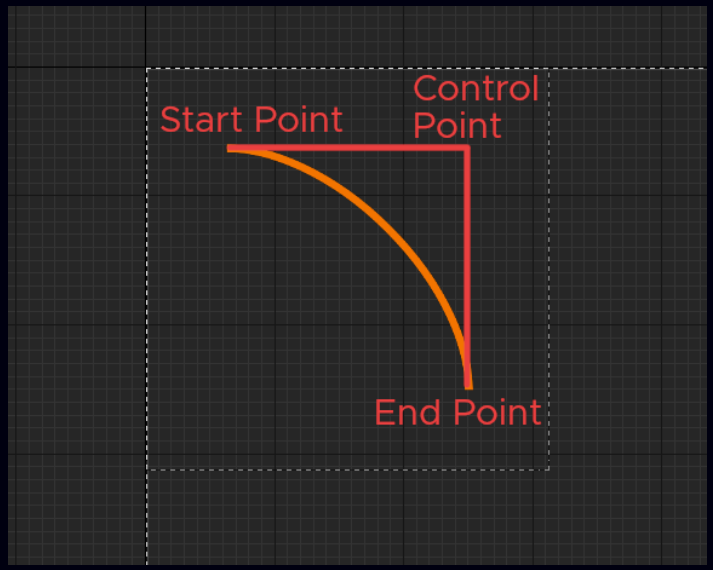
Defines the end point of the curve.

Weight

Defines the weight of the curvature of the curve. The higher the weight, the closer the curve to the control point. 0 results in a straight line between end start and the end points, while high numbers likes 100 and more result in a situation very close to two straight lines connecting the control point with the start point and the end point respectively.

Relative Coordinates

If enabled, all coordinated specified in this command are relative to the last point of the path.



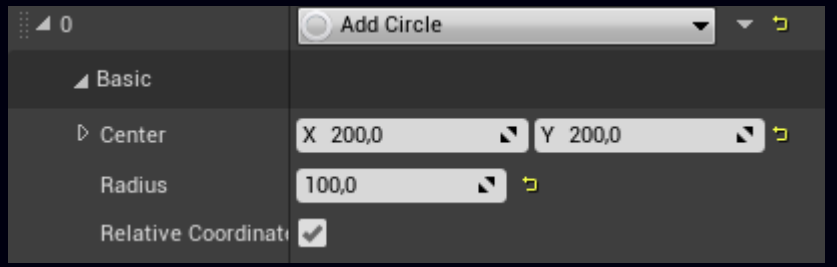
A conic curve drawn from the start point (100, 100) toward the control point (400, 100) ending at the end point (400, 400) with weight 0.5.

Close

Closes the path by connecting the last point with the first point with a line. This command has no parameters.

Add Circle

Adds a circle with the center at the *Center* and radius *Radius*.



Center

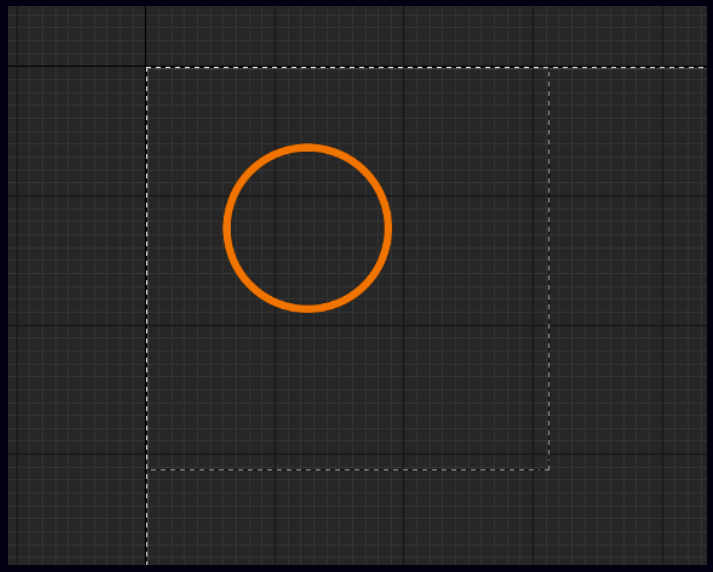
Center of the circle.

Radius

Radius of the circle.

Relative Coordinates

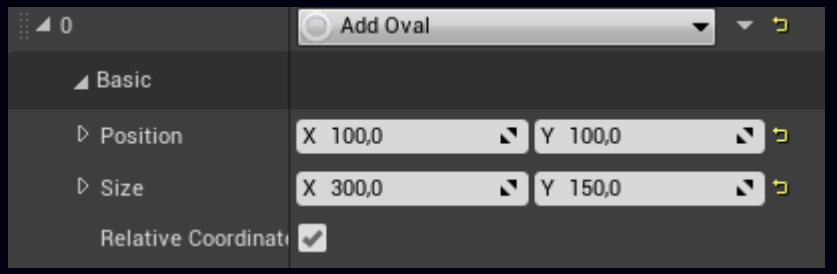
Not used.



A circle with center (200, 200) and radius (100).

Add Oval

Adds an oval drawn within the bounds of a rectangle defined by *Position* and *Size*.



Position

Position of the top left corner of the bounding rectangle.

Size

Size of the bounding rectangle.

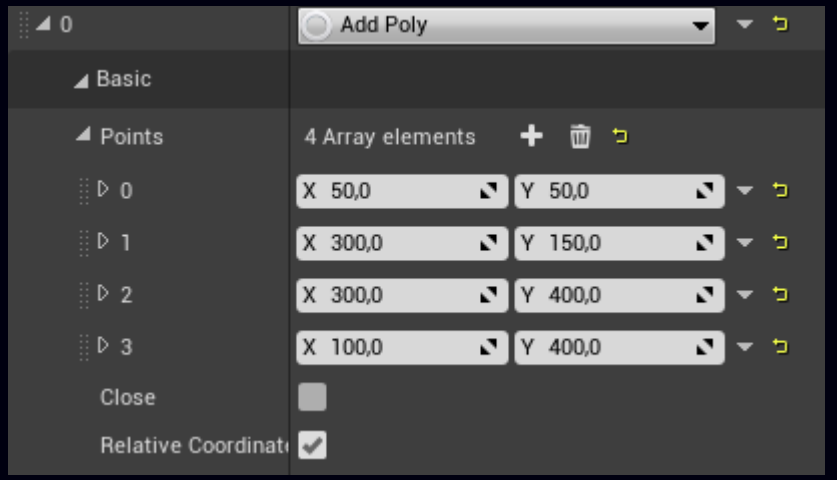
Relative Coordinates

Not used.

An oval drawn within the rectangle with position (100, 100) and size (300, 150).

Add Poly

Adds a polyline defined by a set of points.



Points

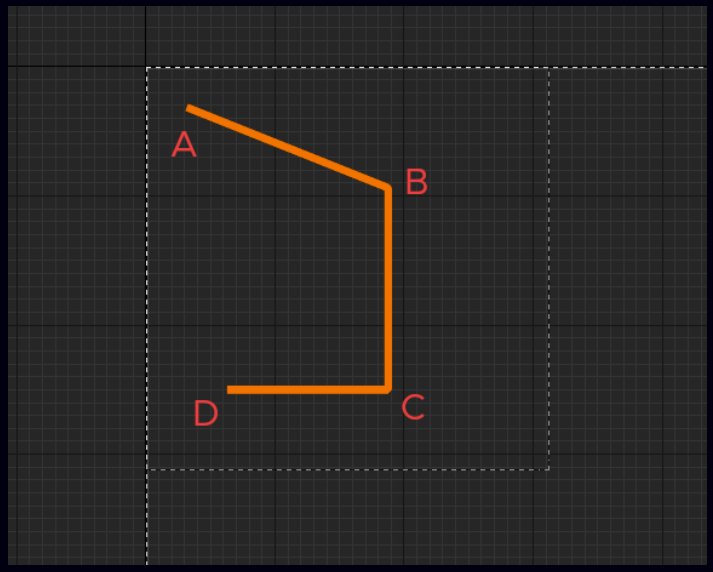
Points defining the polyline.

Close

Close the polyline. If enabled, the polyline is closed by adding a line between the last and the first point.

Relative Coordinates

Not used.



A polyline made of a set of four points; A (50, 50), B (300, 150), C (300,400), and D (100, 400).

Add Rect

Adds a rectangle with position at *Position* and size *Size*.

Position

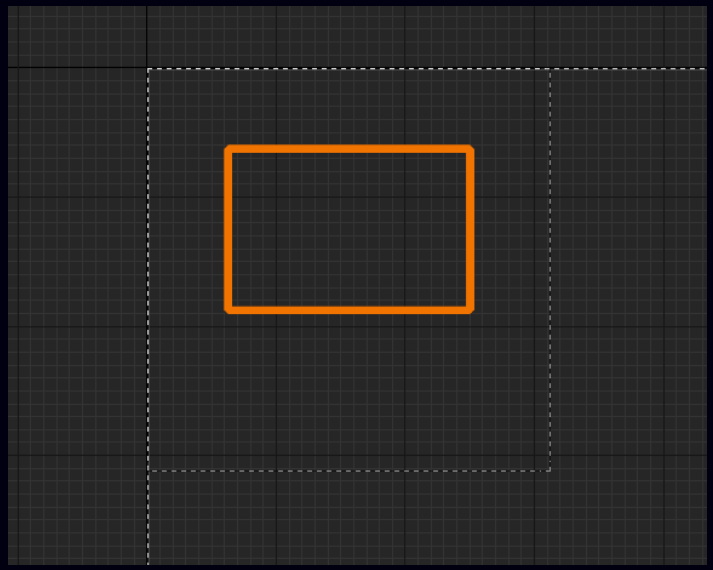
Position of the top left corner of the rectangle.

Size

Size of the rectangle.

Relative Coordinates

Not used.



A rectangle drawn at position (100, 100) with size (300, 200).

Add Round Rect

Adds a rectangle with position at *Position* and size *Size* and optionally rounded corners. Each corner can have different radius.

Position

Position of the top left corner of the rectangle.

Size

Size of the rectangle.

Top Left Radius

Radius of the top left corner.

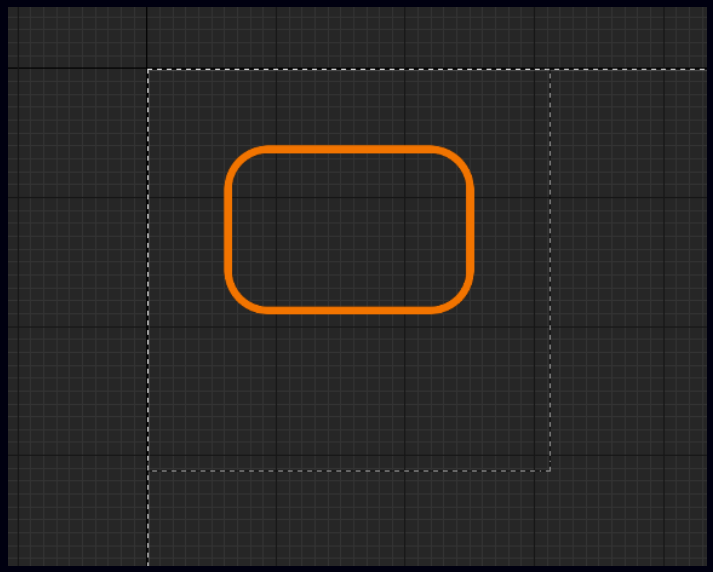
Top Right Radius

Radius of the top right corner.

Bottom Right Radius
Radius of the bottom right corner.

Bottom Left Radius
Radius of the bottom left corner.

Relative Coordinates
Not used.

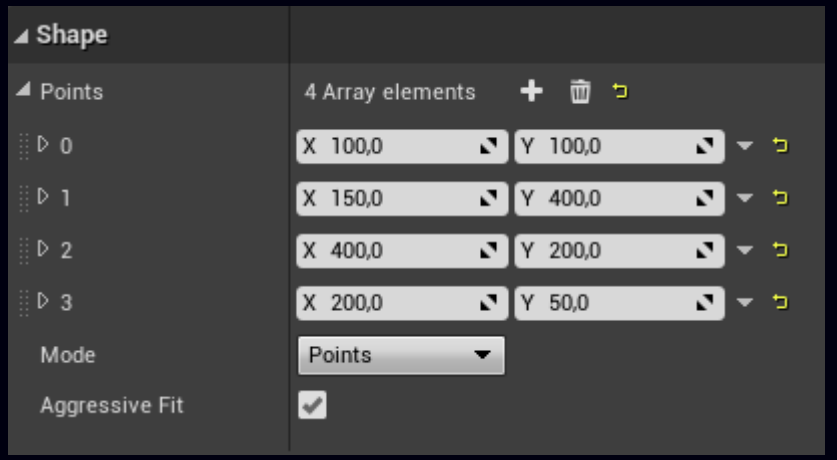


A rectangle drawn at position (100, 100) with size (300, 200) and radius of all 4 corners set to (50).

DP Point Cloud

DP Point Cloud draws a set of points. The points can be drawn individually or connected to form separated lines or a polyline.

Parameters



Points

The point set defining the cloud.

Mode

Defines the mode of drawing the point cloud.

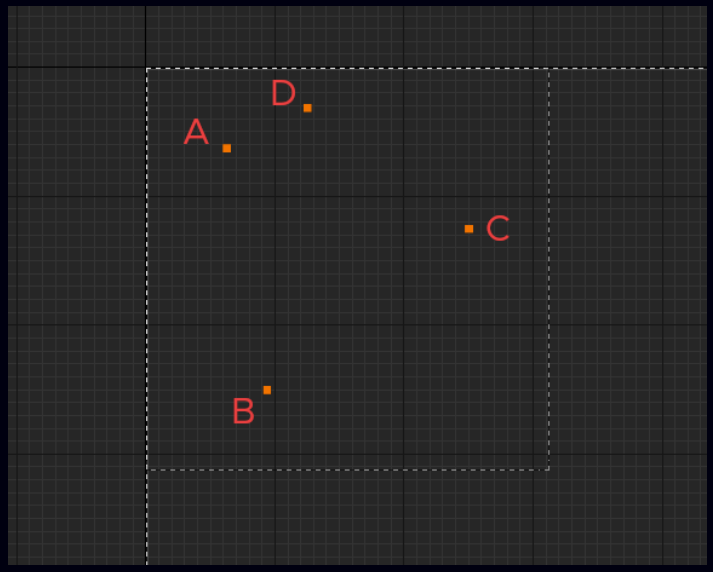
A. Points
   Draw each point separately.
B. Lines
   Draw each pair of odd-even points as a line.

C. Polygon

Draw the points as an open polygon.

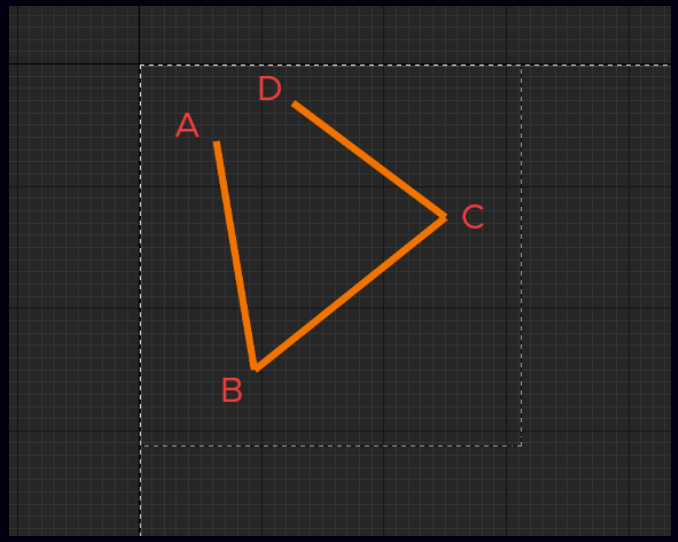
Aggressive Fit

Not used.



A Point Cloud drawn as separated points A (100, 100), B (150, 400), C (400, 200), and D (200, 50).

A Point Cloud drawn as lines connecting points A (100, 100) and B (150, 400); C (400, 200) and D (200, 50).



A Point Cloud drawn as an open polygon connecting points A (100, 100), B (150, 400), C (400, 200), and D (200, 50).

DP Rectangle

DP Rectangle draws a rectangle within the bounds of the Widget with optionally rounded corners.

Parameters

Top Left Radius

Radius of the top left corner.

Top Right Radius

Radius of the top right corner.

Bottom Right Radius

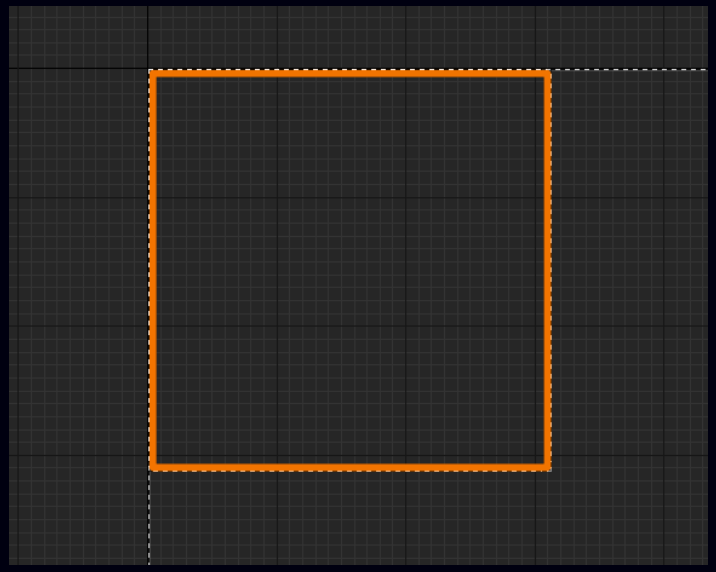
Radius of the bottom right corner.

Bottom Left Radius

Radius of the bottom left corner.

Aggressive Fit

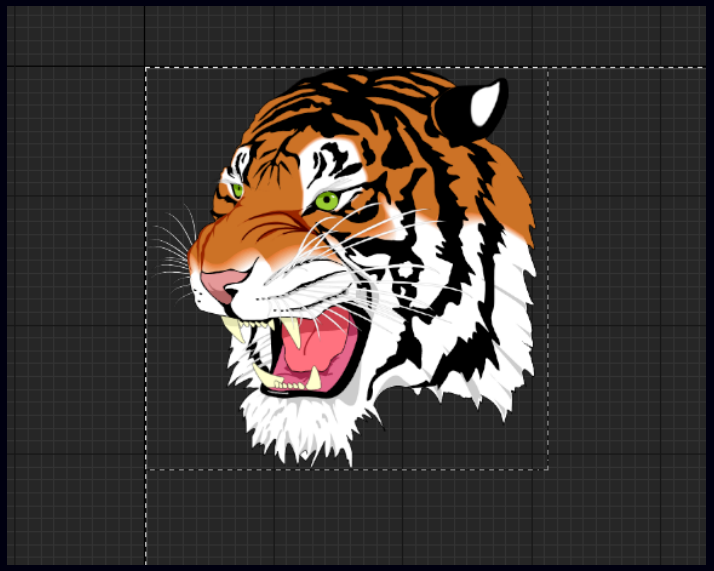
When enabled, the size of the shape will be reduced by the width of the stroke to make it perfectly fit within the bounds. Has no effect when used with a filled Paint.



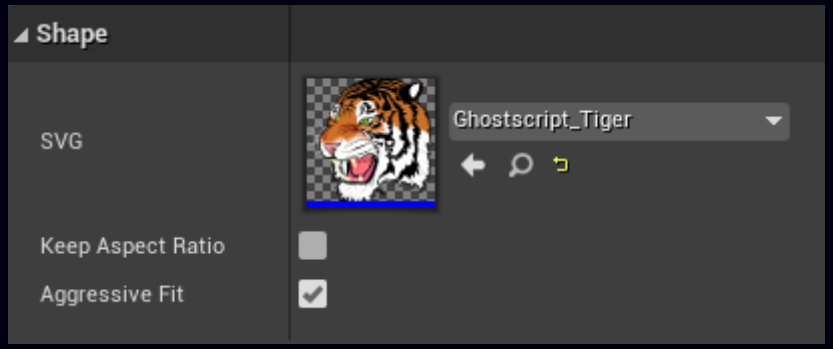
A rectangle drawn with aggressive fit enabled.

DP SVG

DP SVG draws an SVG asset. SVG images can be imported directly into the editor to make an asset. Most features of SVG 1.1 are supported. Advanced features like *scripting* and *animations* aren't supported but this may change in the future.



Parameters



SVG

SVG asset to be drawn.

Keep Aspect Ratio

If enabled, the SVG image will be drawn respecting its original aspect ratio. The image will be stretched to fit within the Widget's bounds otherwise.

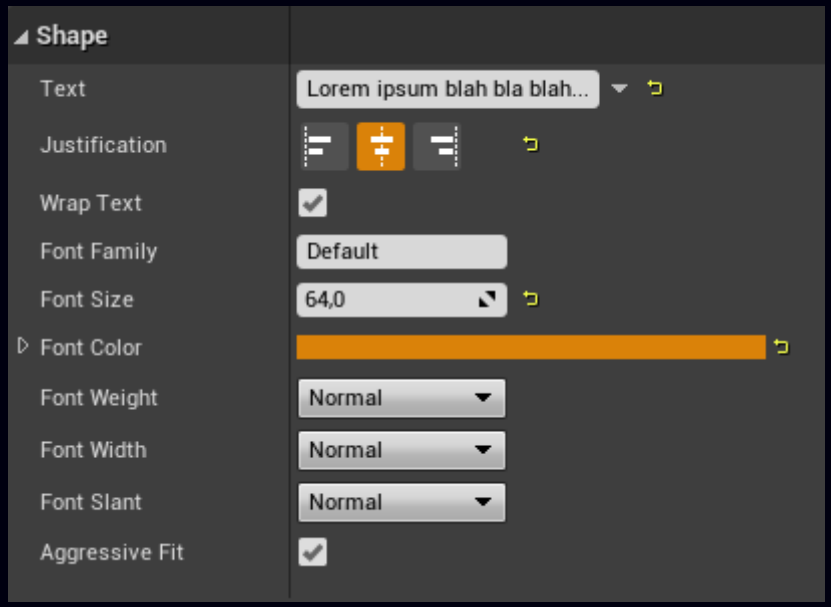
Aggressive Fit

Not used.

DP Text

DP Text draws a text using a defined font style.

Parameters



Text

The text to be drawn.

Justification

Justification of the text.

A. Left
B. Center
C. Right

Wrap Text

Wrap the text to make it fit within the bounds of the Widget.

Font Family

Font family used to draw the text. Note that Definitive Painter uses this parameter to pick a font from fonts installed in the operating system rather then from fonts imported into the Unreal Editor. If a font from this family isn't available, the system default font is used.

Font Size

Size of the font.

Font Color

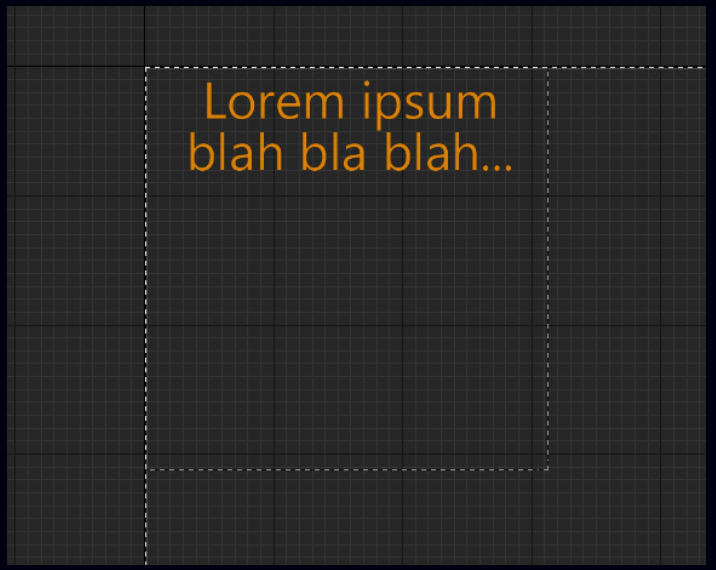
Color of the font.

Font Weight

Weight of the font. Note that if selected font doesn't support the selected weight, the *Normal* weight is used.

A. Invisible (0)
B. Thin (100)
C. Extra Light (200)
D. Light (300)
E. Normal (400)
F. Medium (500)
G. Semi Bold (600)
H. Bold (700)
I. Extra Bold (800)
J. Black (900)

K. Extra Black (1000)



Font Width

Width of the font. Note that if selected font doesn't support the selected width, the *Normal* width is used.

A. Undefined
B. Ultra Condensed
C. Extra Condensed
D. Condensed
E. Semi Condensed
F. Normal
G. Semi Expanded
H. Expanded
I. Extra Expanded
J. Ultra Expanded

Font Slant

Slant of the font. Note that if selected font doesn't support the selected slant, the *Normal* slant is used.

A. Normal
B. Italic
C. Oblique

Aggressive Fit

Not used.

DP Text On Path

DP Text On Path draws a text following a path.

Parameters

Text

The text to be drawn.

Path Commands

An array of commands defining the path.

Warp Text

Warp individual characters to make them perfectly fit the followed path. Note that this feature is slow and might not work properly on paths with sharp edges.

Draw Path

Draw the followed path.

Vertical Offset

Offsets the text vertically.

Horizontal Offset

Offsets the text horizontally.

Justification

Justification of the text along the followed path.

A. Left
B. Center
C. Right

Font Family

Font family used to draw the text. Note that Definitive Painter uses this parameter to pick a font from fonts installed in the operating system rather then from fonts imported into the Unreal Editor. If a font with this family isn't available, the system default font is used.

Font Size

Size of the font.

Font Color

Color of the font.

Font Weight

Weight of the font. Note that if selected font doesn't support the selected weight, the *Normal* weight is used.

A. Invisible (0)
B. Thin (100)
C. Extra Light (200)
D. Light (300)
E. Normal (400)
F. Medium (500)
G. Semi Bold (600)
H. Bold (700)
I. Extra Bold (800)
J. Black (900)
K. Extra Black (1000)

Font Width

Width of the font. Note that if selected font doesn't support the selected width, the *Normal* width is used.

A. Undefined
B. Ultra Condensed
C. Extra Condensed
D. Condensed
E. Semi Condensed
F. Normal
G. Semi Expanded
H. Expanded
I. Extra Expanded
J. Ultra Expanded

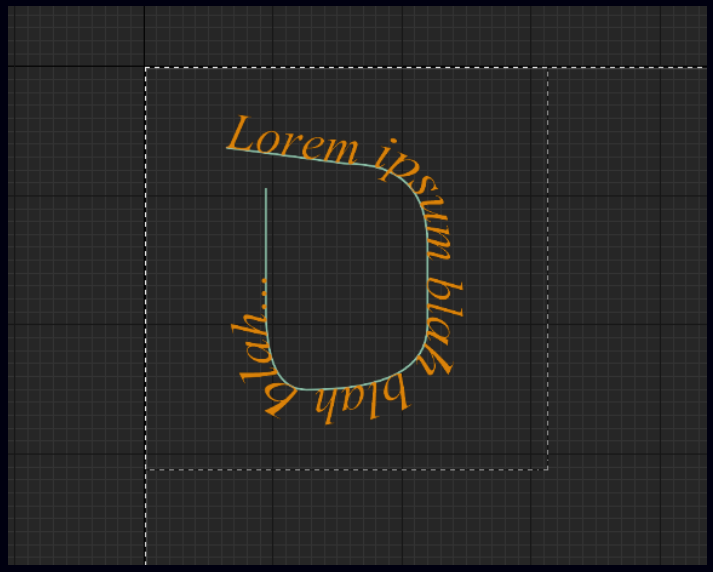
Font Slant

Slant of the font. Note that if selected font doesn't support the selected slant, the *Normal* slant is used.

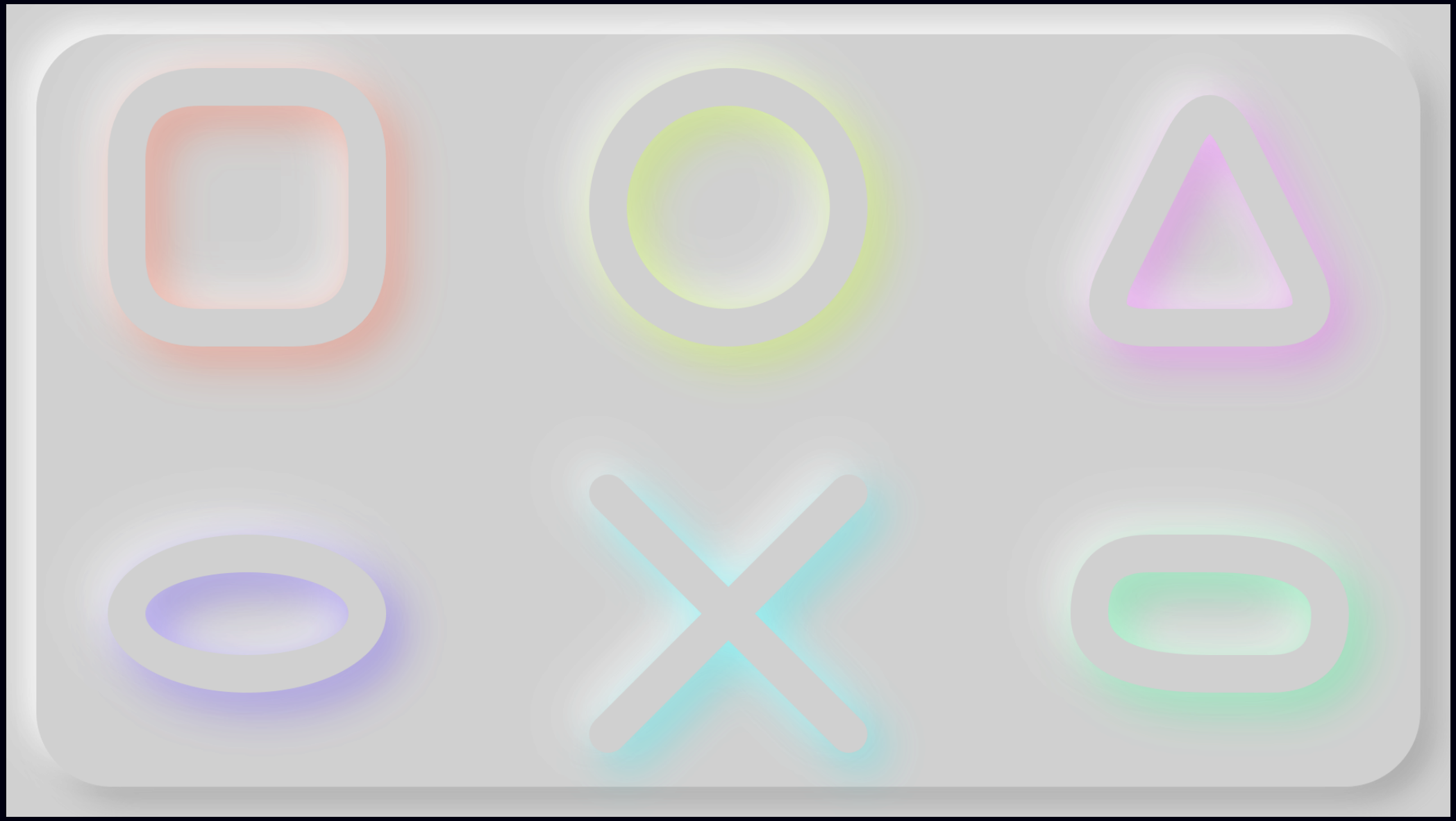
A. Normal
B. Italic
C. Oblique

Aggressive Fit

Not used.



Text following a path. Warp Text and Draw Path options are enabled.



Widget Animation

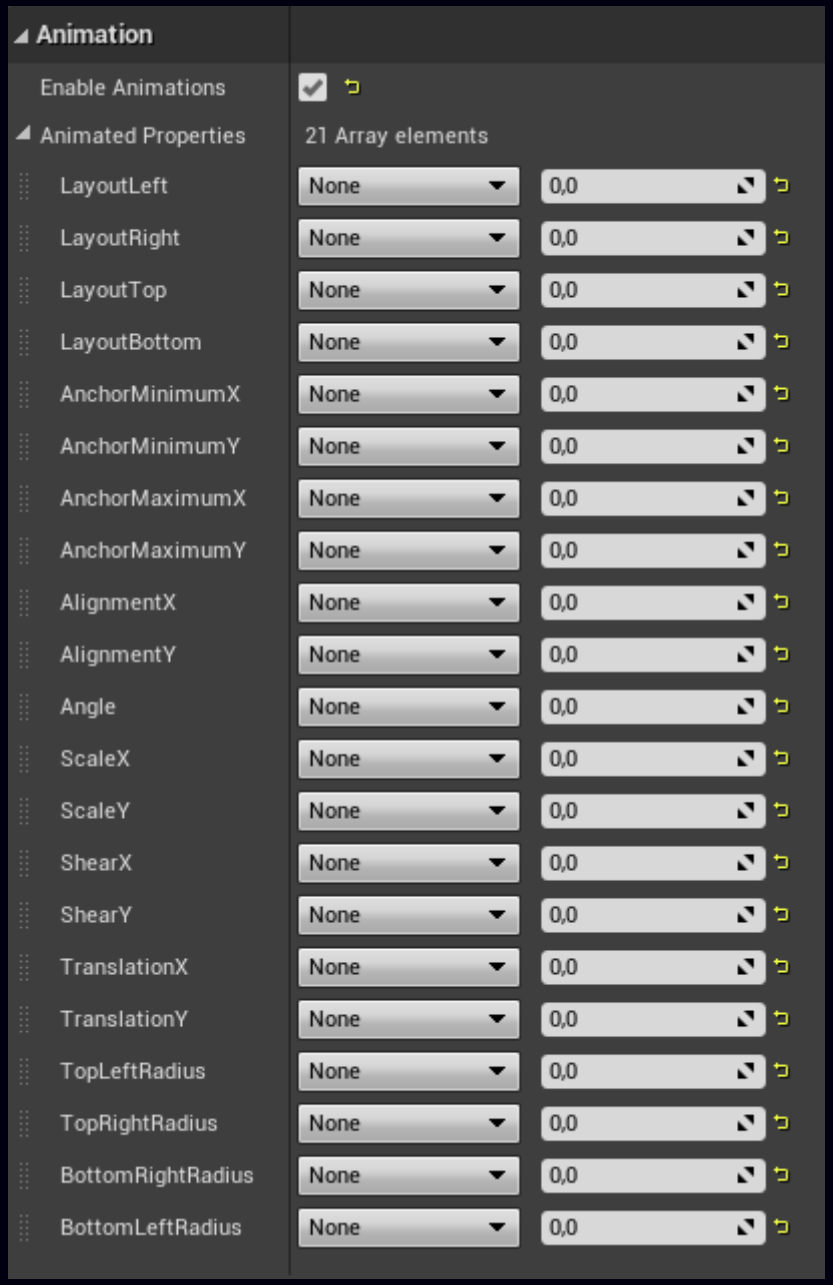
Every Definitive Painter Widget has its own predefined set of parameters which can be animated. All animations start upon the creation of the Widget and last for a specified amount of seconds. You can choose from 32 different types of animation. The animation for each parameter is independent from other animations.

Animation types

A. None
B. Linear
C. In Sine
D. Out Sine
E. In-Out Sine
F. In Quad
G. Out Quad
H. In-Out Quad
I. In Cubic
J. Out Cubic
K. In-Out Cubic
L. In Quart
M. Out Quart
N. In-Out Quart
O. In Quint
P. Out Quint
Q. In-Out Quint
R. In Expo
S. Out Expo
T. In-Out Expo
U. In Circ
V. Out Circ
W. In-Out Circ
X. In Back
Y. Out Back
Z. In-Out Back
AA. In Elastic
AB. Out Elastic
AC. In-Out Elastic
AD. In Bounce
AE. Out Bounce
AF. In-Out Bounce

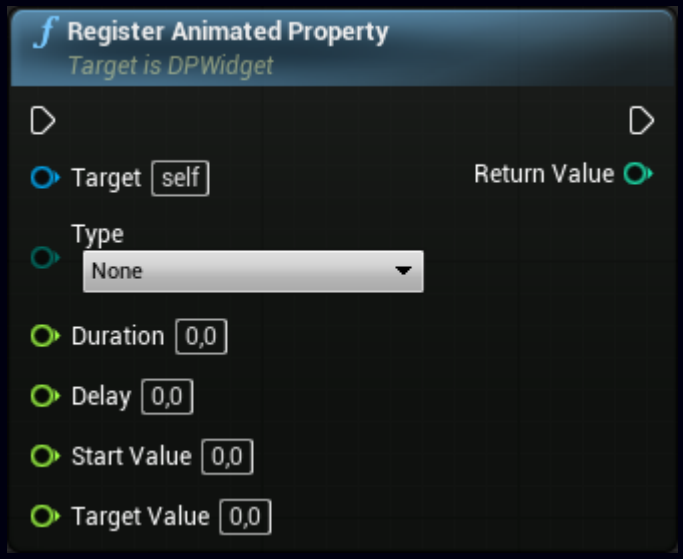
If the animation type is set to *None* and/or the length is set to 0, the animation is disabled for the corresponding parameter. All animations for the Widget are disabled by default.

See this amazing Easing Functions Cheat Sheet to learn more about available animation types.



You can also register a custom Animated Property which is basically a helper that outputs a float value based on animation *Type*, *Duration*, *Delay* and *Start* and *Target Value*. The helper sets the output value to the *Start Value* upon the creation of the Widget, waits for a number of seconds specified in *Delay*, then interpolates the output value toward the *Target Value* for a period of time specified in *Duration* (seconds).

Register Animated Property



Target UDPWidget

A Definitive Painter Widget. The Widget is responsible for managing custom Animated Properties.

Type EDPAnimationType

Animation type.

Duration float

Animation duration.

Delay float

Animation delay.

Start Value float

The output value is set to the *Start Value* upon the creation of the Widget.

Target Value float

The output value is interpolated toward the *Target Value*.

Return Value int

The ID of the custom Animated Property. Store this value in a variable. You need the ID every time you interact with a custom Animated Property (reading its output value, etc.).

Get Animated Property Value

Target UDPWidget

A Definitive Painter Widget. The Widget is responsible for managing custom Animated Properties.

Return Value int

Number of custom Animated Properties registered in the target Widget.

Get Animated Property Value

Target UDPWidget

A Definitive Painter Widget. The Widget is responsible for managing custom Animated Properties.

Animated Property Index int

The ID of the custom Animated Property.

Return Value float

The output value of the custom Animated Property.

Set Animated Property Start Value

Sets a new *Start Value* for a custom Animated Property. This resets the timer of the Animated Property, sets the output value to the new *Start Value* and immediately starts interpolating the output value toward the *Target Value* ignoring the *Delay*.

Target UDPWidget

A Definitive Painter Widget. The Widget is responsible for managing custom Animated Properties.

Animated Property Index int

The ID of the custom Animated Property.

Start Value float

New *Start Value*.

Set Animated Property Target Value

Sets a new *Target Value* for a custom Animated Property. This resets the timer of the Animated Property and immediately starts interpolating the output value toward the new *Target Value* ignoring the *Delay*.

Target UDPWidget

A Definitive Painter Widget. The Widget is responsible for managing custom Animated Properties.

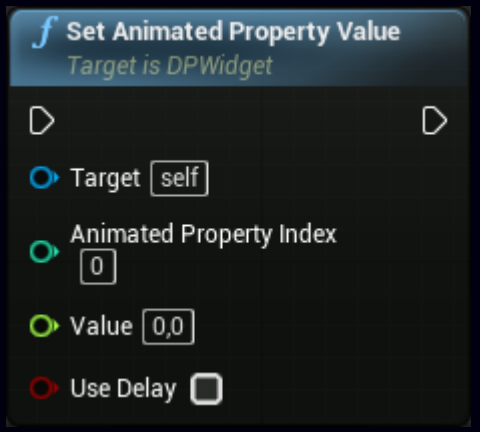
Animated Property Index int

The ID of the custom Animated Property.

Target Value float

New *Target Value*.

Set Animated Property Value



Sets the output value to the *Value* for a custom Animated Property. This resets the timer of the Animated Property and starts interpolating the output value toward the *Target Value* ignoring the *Delay* and the *Start Value*.

Target UDPWidget

A Definitive Painter Widget. The Widget is responsible for managing custom Animated Properties.

Animated Property Index int

The ID of the custom Animated Property.

Value float

New *Value*.

Use Delay bool

Wait for the *Delay* time set when registering the Animated Property before interpolating the output value toward the *Target Value*.

Reset Animated Property

Resets the output value and the timer of the Animated Property.

Target UDPWidget

A Definitive Painter Widget. The Widget is responsible for managing custom Animated Properties.

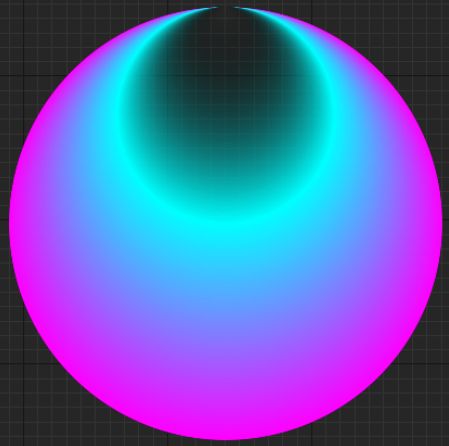
Animated Property Index int

The ID of the custom Animated Property.

Use Delay bool

Wait for the *Delay* time set when registering the Animated Property before interpolating the output value toward the *Target Value*.

CONICAL GRADIENT

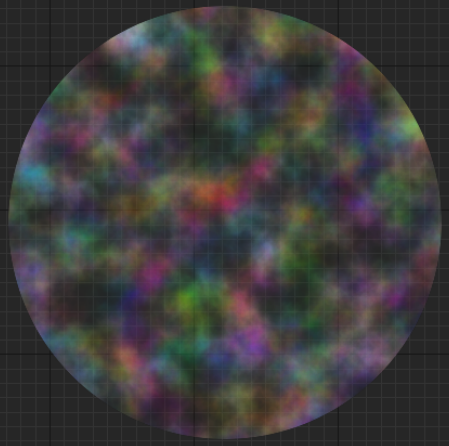


LINEAR GRADIENT

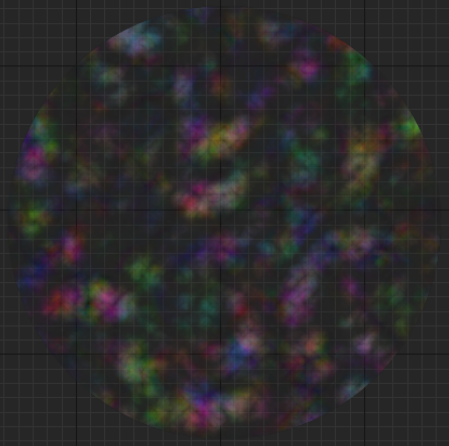
RADIAL GRADIENT

SWEEP GRADIENT

FRACTAL NOISE



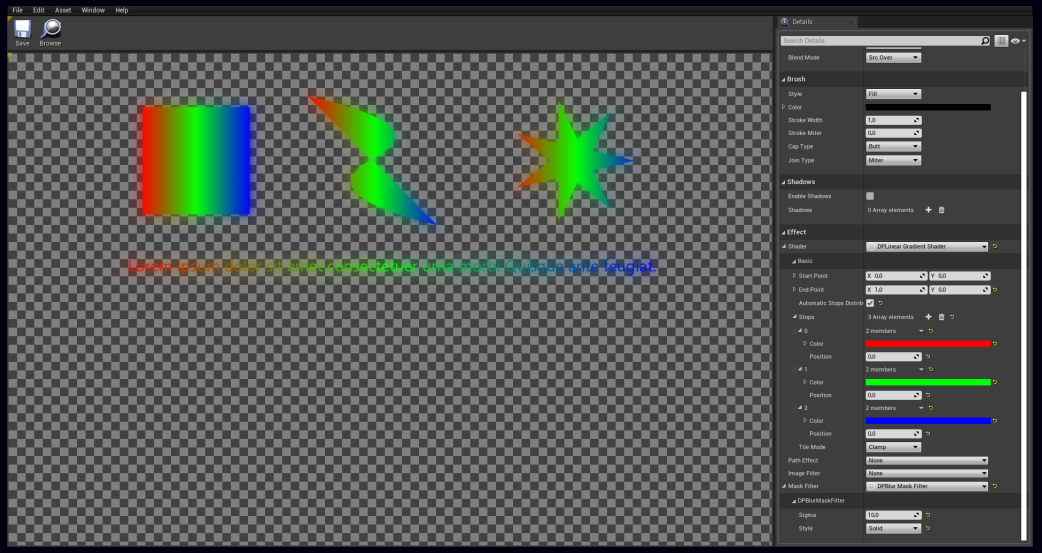
TURBULENT NOISE



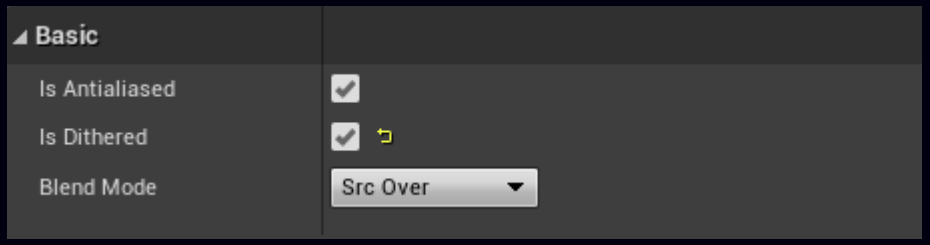
Paint

Paint is an asset defining the appearance of DP Widgets. To create a Paint, right click in the Content Browser → Definitive Painter → Definitive Painter Paint.

Paint has its own editor. The editor window is divided into two panels; the *preview panel* and the *detail panel*. In the *detail panel*, you can change the properties of the Paint and see it applied onto basic shapes drawn in the *preview panel*.



Parameters



Is Antialiased

This parameter is intended for a future optimization. Right now, every piece of geometry is rendered with hardware multisample antialiasing enabled (if supported).

Is Dithered

Enable this to reduce visible bands between similar colors. This is especially useful for Paints with gradients where blending between similar colors over a long distance is performed.

Blend Mode

Defines the blending between the currently rendered Element and the Elements that were rendered before the current one.

A. Clear
B. Src
C. Dst
D. Src Over
E. Dst Over
F. Src In

G. Dst In
H. Src Out
I. Dst Out
J. Src A Top
K. Dst A Top
L. Xor
M. Plus
N. Modulate
O. Screen
P. Overlay
Q. Darken
R. Lighten
S. Color Dodge
T. Color Burn
U. Hard Light
V. Soft Light
W. Difference,
X. Exclusion
Y. Multiply
Z. Hue
AA. Saturation
AB. Color
AC. Luminosity

Default is *Src Over*. You can learn more about the available blend modes here.

Style

Defines whether the Element is filled, stroked or both.

A. Fill
   Fill the area of the Element with the Paint.
B. Stroke
   Draw only the outline of the Element.
C. Fill and Stroke
   The previous two combined.

Color

Color used to draw the Element.

Stroke Width

The width of the outline. Has effect only with the *Fill* and *Fill and Stroke Style*.

Stroke Miter

Defines the limit at which sharp corners are drawn beveled. Has effect only with the *Fill* and *Fill and Stroke Style*.

Cap Type

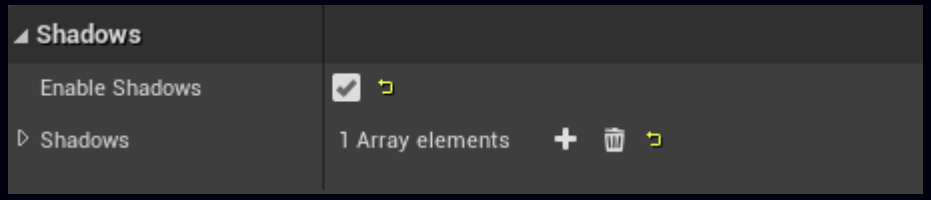
Defines the cap type of strokes. Has effect only with the *Fill* and *Fill and Stroke Style*.

A. Butt
   Default. Strokes have no extension.
B. Round
   Adds a circle at each and of strokes.
C. Square
   Adds a square at each end of strokes.

Join Type

Defines the corner type of strokes. Has effect only with the *Fill* and *Fill and Stroke Style*.

A. Miter
B. Round
C. Bevel



Enable Shadows

Enable shadows for the Element.

Shadows

An array of *Shadows*. Each Paint can hold an unlimited number of *Shadows* the Element can cast shadows with different colors in many directions.

Enable

Enable the shadow.

Color

Shadow color.

Blur

Shadow blur or softness.

Offset

Shadow offset (or shadow length) from the position of the Element.

Style

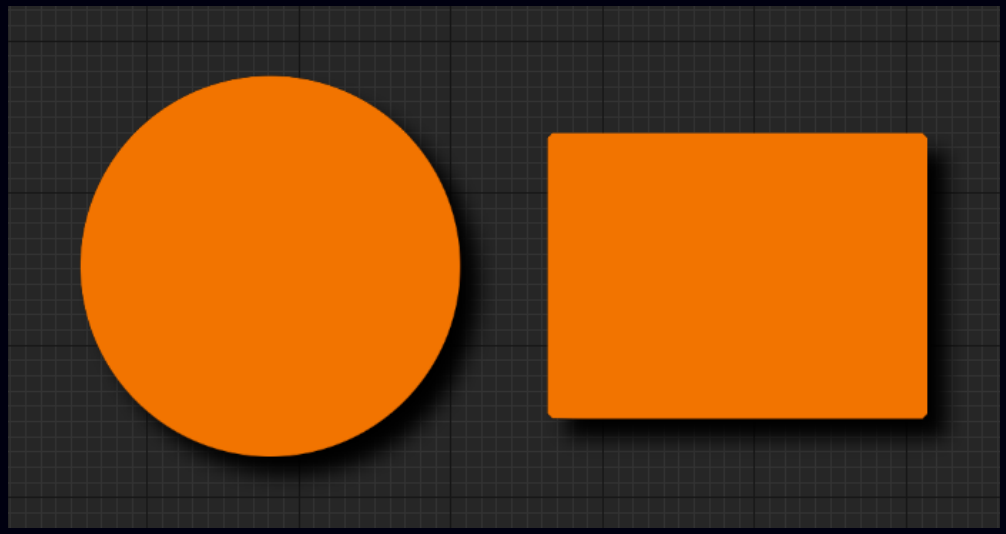
Shadow style.

A. Outset

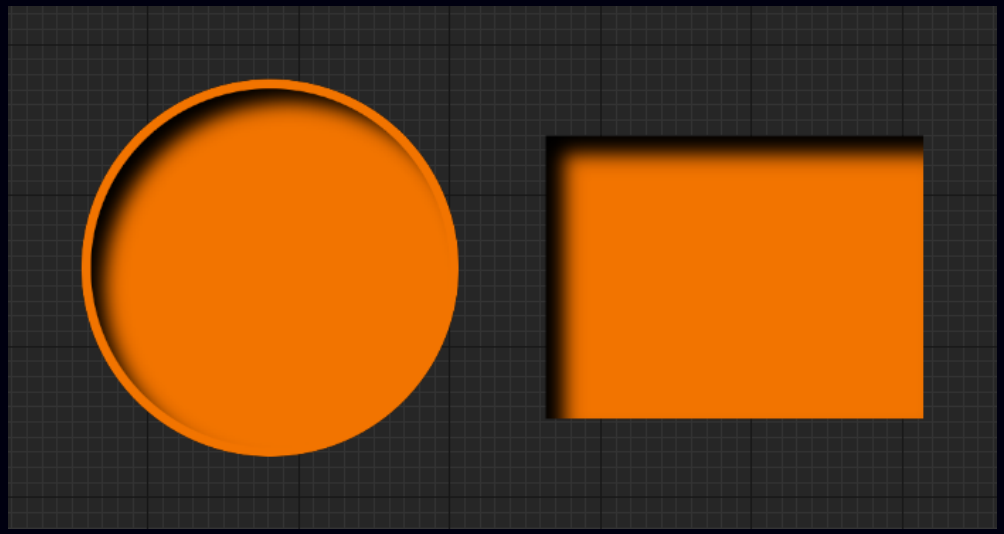
   The Element casts shadow as if it was above the background.

B. Inset

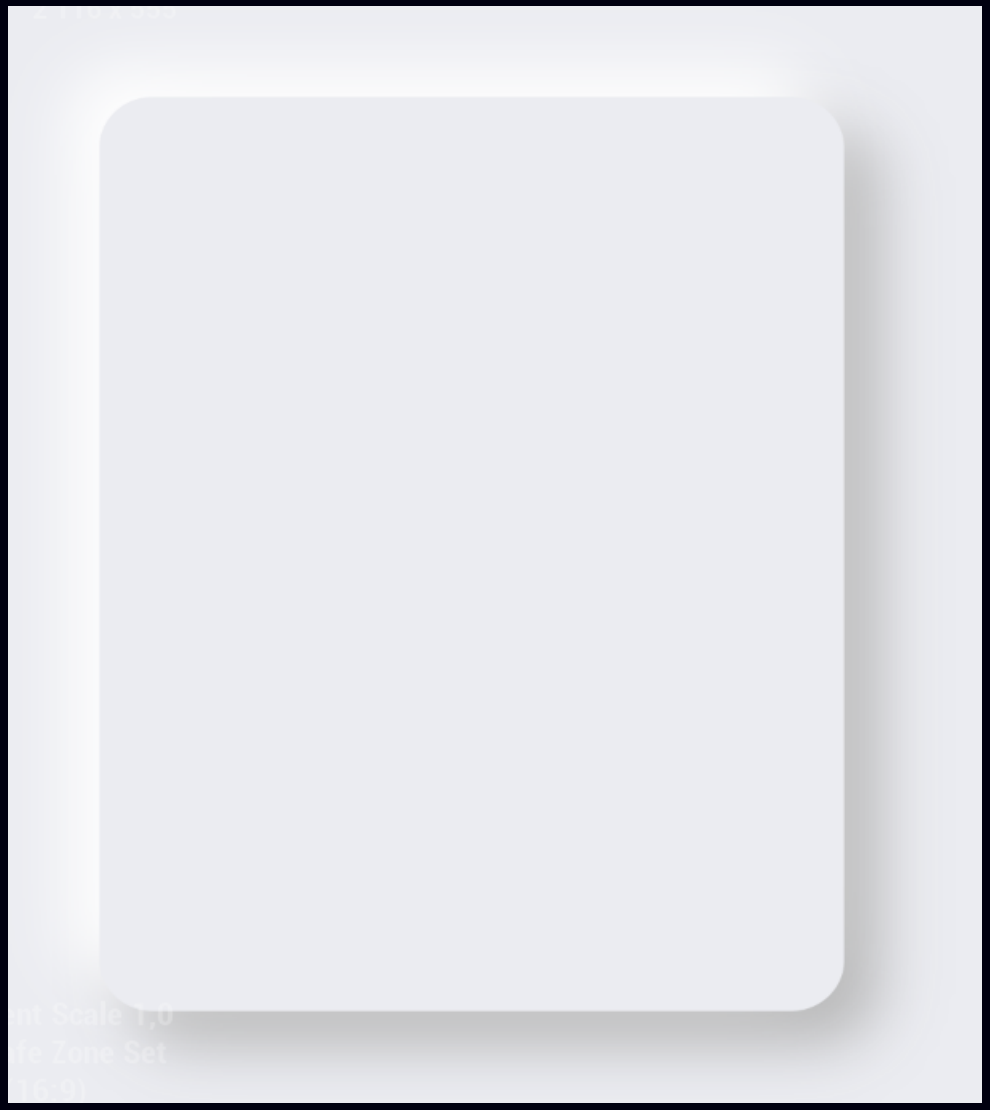
   The background casts shadow on the Element as if the Element formed a hole in the background.



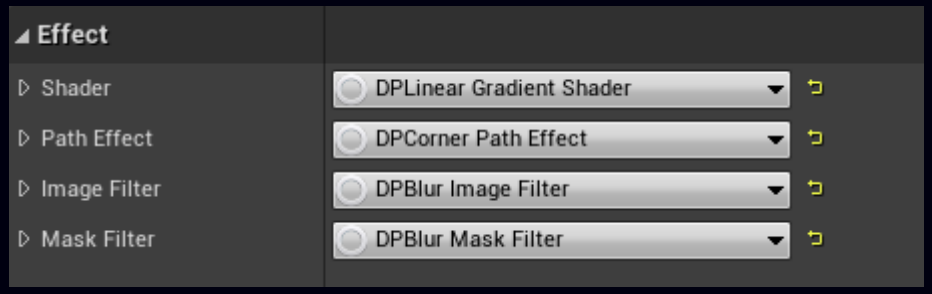
A single outset Shadow with offset (20, 20), blur (10), and black color.



A single inset Shadow with offset (20, 20), blur (10), and black color. Note that the circle is drawn with a filled&stroked Paint, while the rectangle is drawn only with a filled Paint.



Two outset Shadows; one with offset (30, 30), blur (30) and darker color (#B3B3B3FF), and one with offset (-30, -30), blur (30) and lighter color (white).



Shader

An image shader defining the texture of the Element.

Path Effect

An effects altering the geometry of the Element.

Image Filter

A filter altering the appearance of the whole Element.

Mask Filter

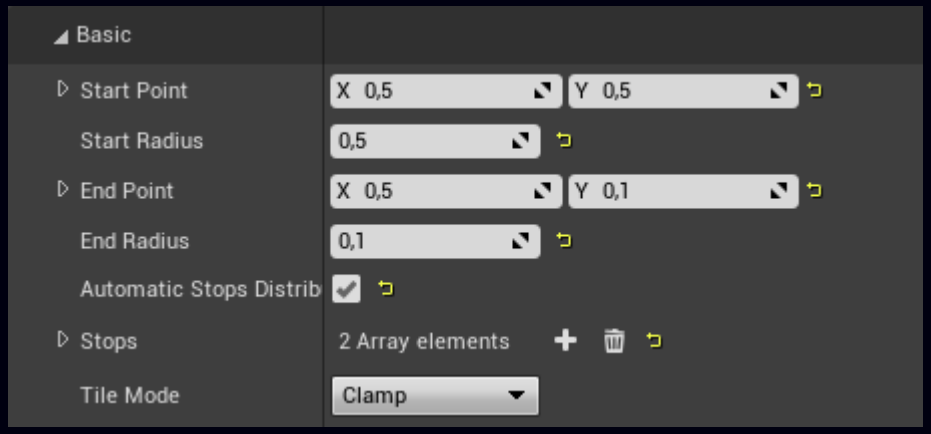
A filter altering the appearance of the Element's border.

Shaders

All positions, centers, radii and stop positions are expressed as fractions of the Element's geometrical size. For an Element of size 400, 400, a Center (0.5, 0.75) is transformed to the position (200, 300) relative to the top left corner of the Element. For the same Element, Radius (0.25) is transformed to (100).

DP Conical Gradient Shader

Creates a conical gradient which is basically a radial gradient with two focal points. Each point has its own position and radius.



Start Point

Start point position (0...1, 0...1).

Start Radius

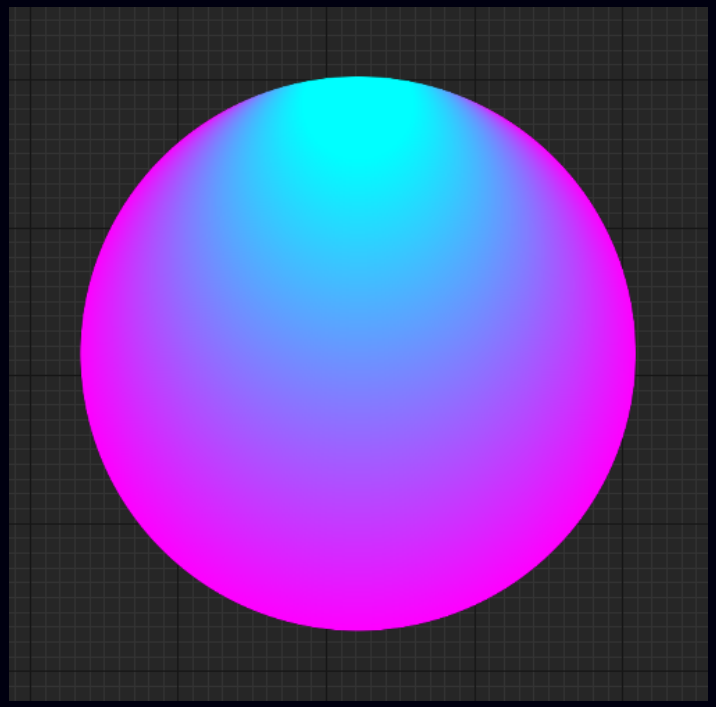
Radius around the *Start Point* (0...1).

End Point

End point position (0...1, 0...1).

End Radius

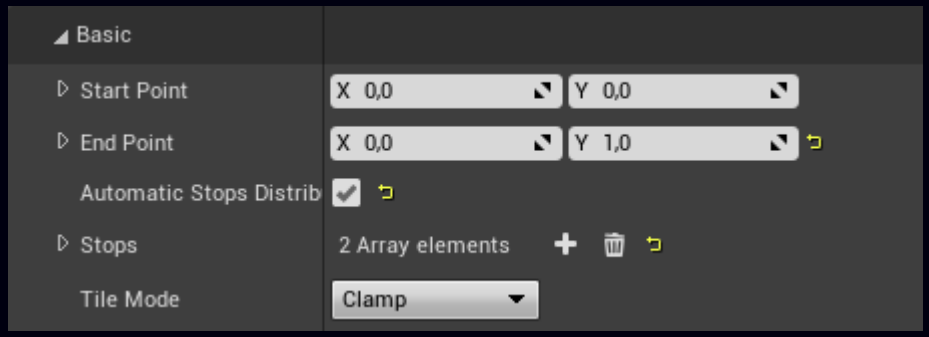
Radius around the *End Point* (0...1).



A Conical Gradient with Start Point (0.5, 0.5), Start Radius 0.5, End Point (0.5, 0.1), End Radius (0.1), and two Stops.

DP Linear Gradient Shader

Creates a linear gradient from the *Start Point* to the *End Point*.



Start Point

Start point position.

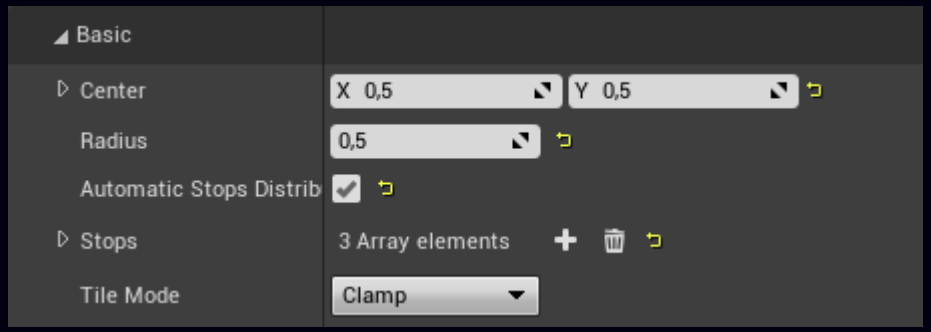
End Point

End point position.

A Linear Gradient with Start Point (0.0, 0.0), End Point (0.0, 1.0), and three Stops.

DP Radial Gradient Shader

Creates a radial gradient from at the *Center* with the radius *Radius*.

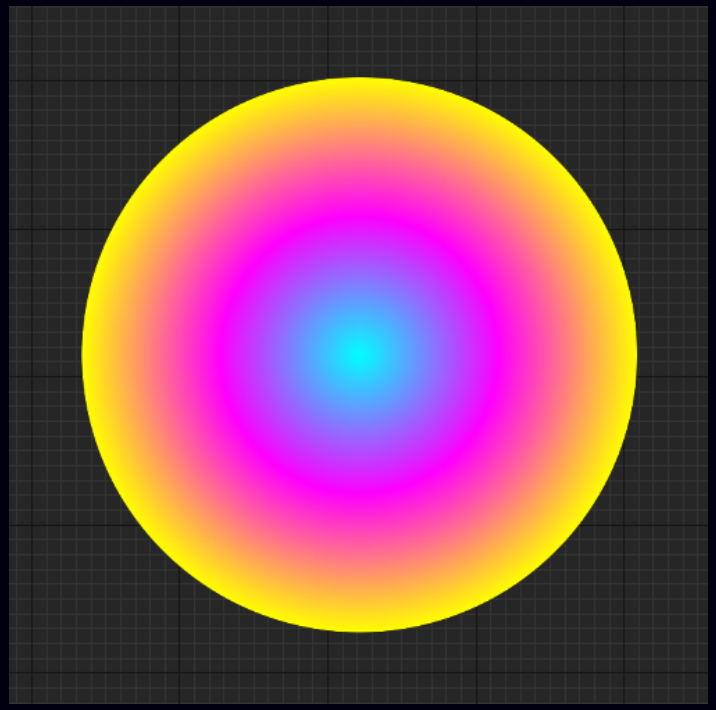


Center

Center position.

Radius

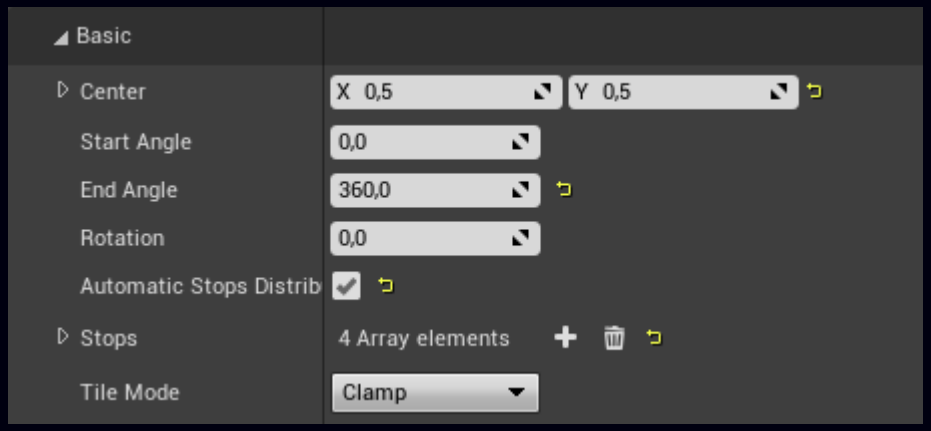
Gradient radius.



A Linear Gradient with Center (0.5, 0.5), Radius (0.5), and three Stops.

DP Sweep Gradient Shader

Creates a Sweep (umbrella-like) gradient at the *Center*, starting at the *Start Angle*, ending at the *End Angle* and with optional *Rotation*.



Center

Center position.

Start Angle

Start angle of the gradient in degrees. Starts at 3 o'clock and goes clockwise when increased.

End Angle

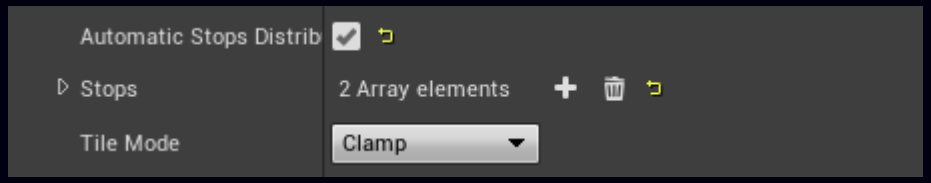
End angle of the gradient in degrees. Starts at 3 o'clock and goes clockwise when increased.

Rotation

Optional rotation of the gradient.

A Sweep Gradient with Center (0.5, 0.5), Start Angle (0.0), End Angle (360.0), Rotation(0.0) and three Stops.

Gradient Common parameters



Automatic Stops Distribution
Center position.

Stops
An array of *Stops*. Each *Stop* defines a color at a certain point along the direction of the gradient.

Color
Stop color.

Position
Stop position (0...1) along the direction of the Gradient.

Tile Mode
Controls drawing outside of the Gradient range.

A. Clamp
   Repeat the edge color.

B. Repeat
   Repeats the texture generated by the shader.

C. Mirror
   Repeatedly mirrors the texture generated by the shader.

D. Decal
   Fills the area outside of the Gradient range with transparent black.

DP Turbulent Noise Shader

Creates a turbulent noise texture useful for simulating organic images like water, lava, clouds, etc.

Base Frequency X

Frequency X.

Base Frequency Y

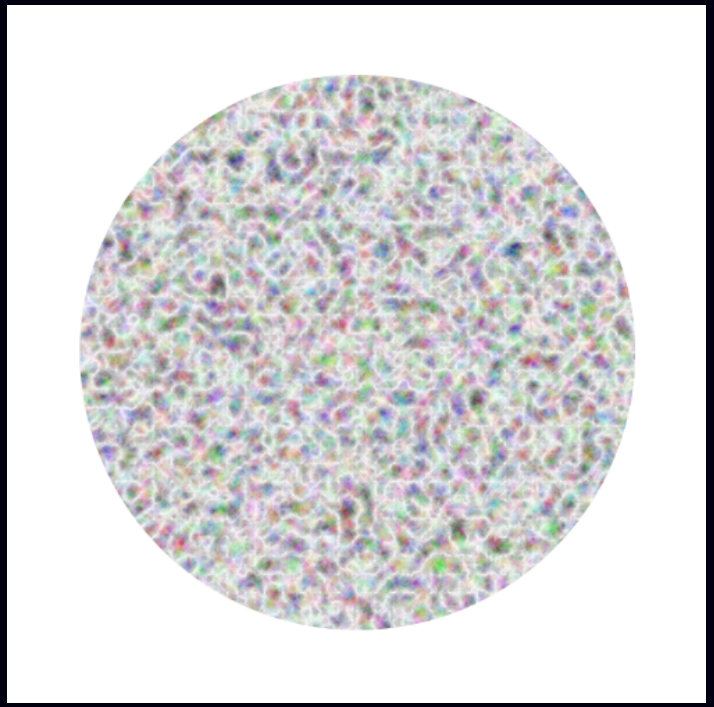
Frequency Y.

Num Octaves

Number of octaves.

Seed

Noise randomization parameter.



Turbulent noise with Base Frequency X (0.05), Base Frequency Y (0.05), Num Octaves (5), and Seed (123).

DP Fractal Noise Shader

Creates a fractal noise texture useful for simulating landscapes, force fields, etc.

Base Frequency X

Frequency X.

Base Frequency Y

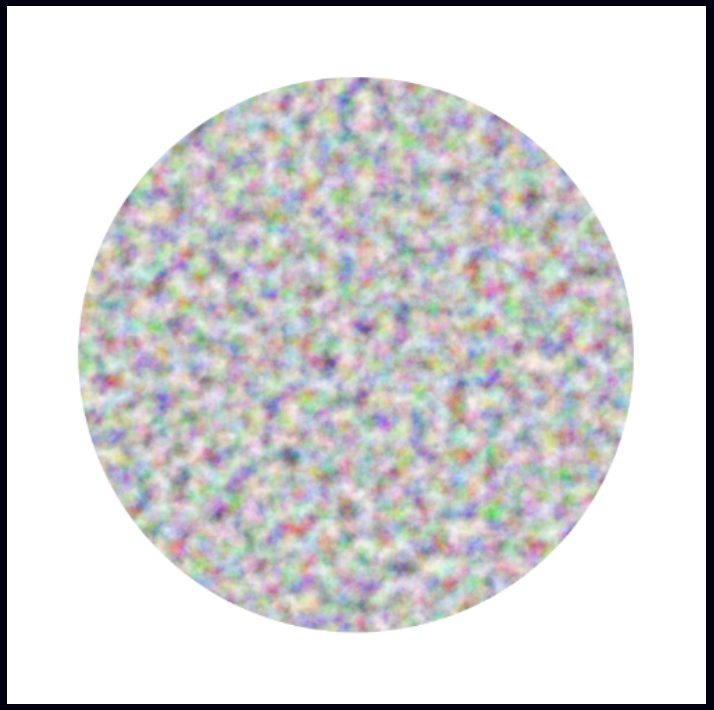
Frequency Y.

Num Octaves

Number of octaves.

Seed

Noise randomization parameter.

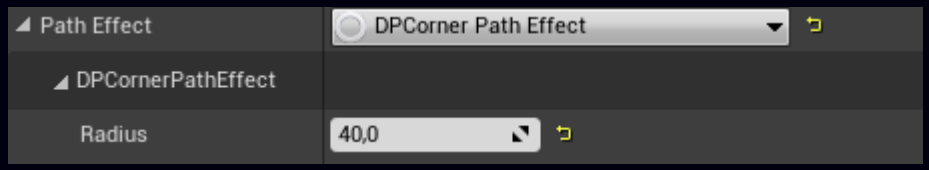


Fractal noise with Base Frequency X (0.05), Base Frequency Y (0.05), Num Octaves (5), and Seed (123).

Path Effects

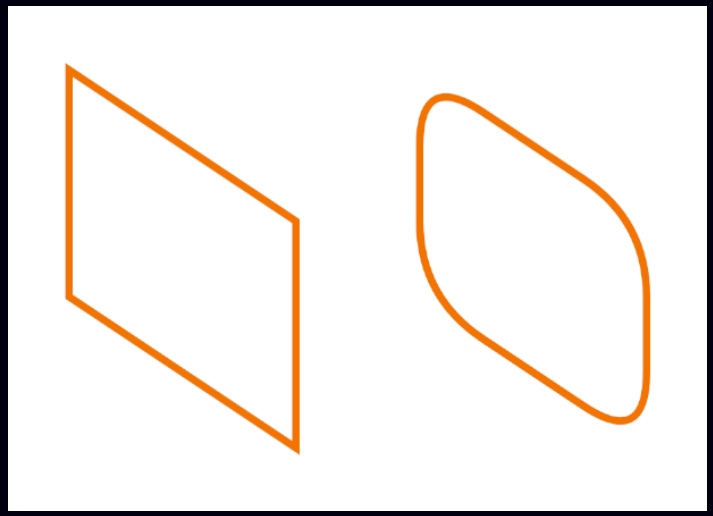
DP Corner Path Effect

Rounds corners of the shape to the specified radius.



Radius

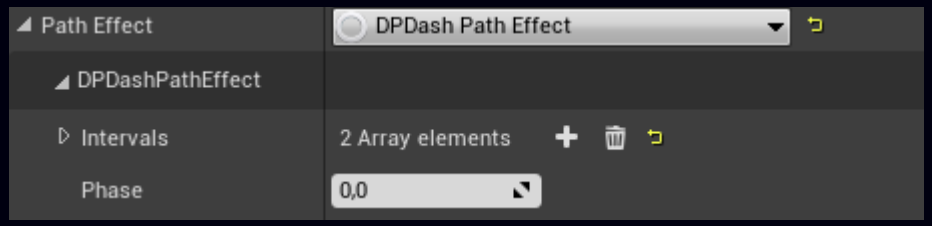
Corner radius.



A path without (left) and with rounded corners (right) with radius (100).

DP Dash Path Effect

Turns a stroked path into a dashed path. This works only with a stroked Paint.

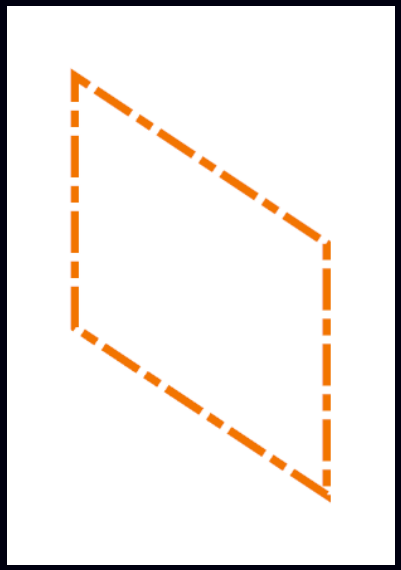


Intervals

An array of floats defining interval at which path is either visible or invisible. Intervals with odd indices define the length of visible parts and intervals with even indices define the length of invisible parts.

Phase

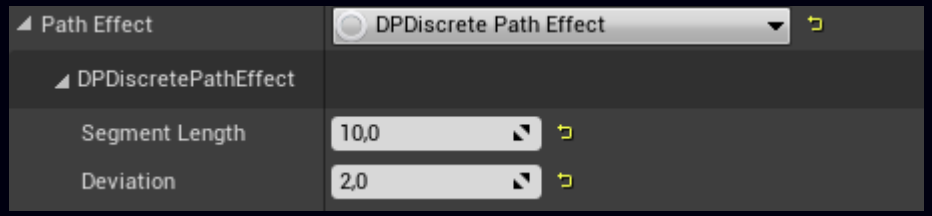
The offset of *Intervals* along the path.



A dashed path with 4 intervals (20, 10, 50, and 10). The effect starts at the top left corner and goes clockwise.

DP Discrete Path Effect

Randomly deforms the shape of the Element.

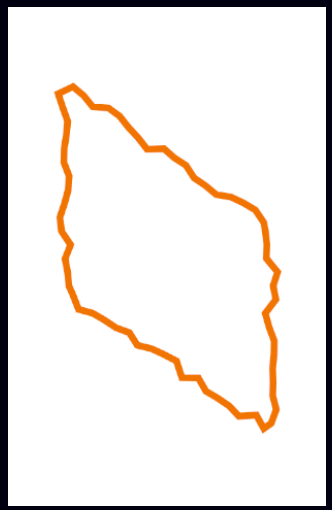


Segment Length

Length of segments the path is broken into.

Deviation

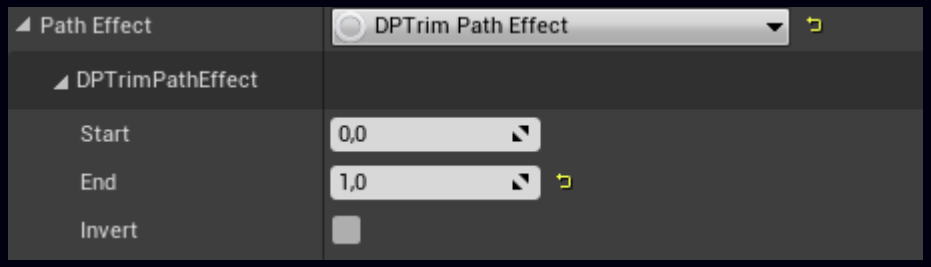
Maximum amount of random offset applied to the forementioned path segments.



A deformed path with segment length (20) and deviation (10).

DP Trim Path Effect

Trims the path.



Start

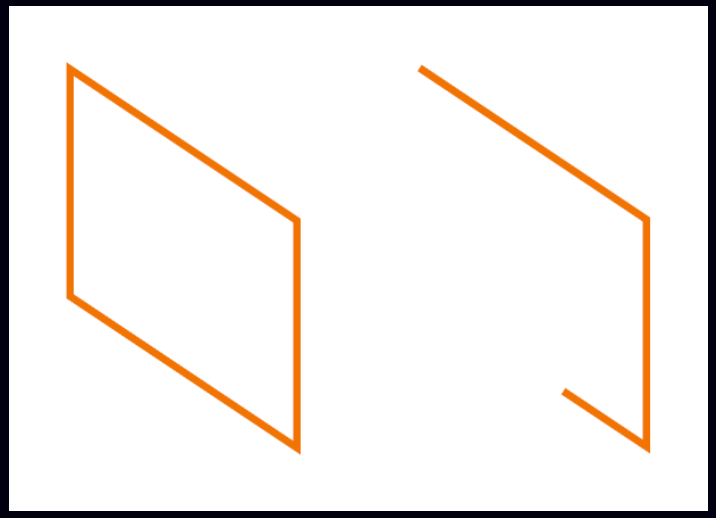
Start of the visible part of the path (0...1).

End

End of the visible part of the path (0....1).

Invert

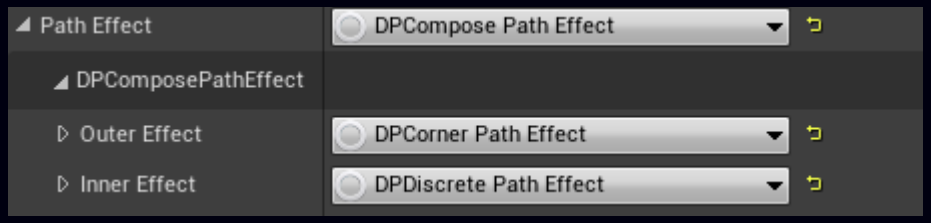
Invert the visible and invisible parts of the path.



A path without (left) and with the trim effect (right) with Start (0), End (0.6) end Inverse (disabled).

DP Compose Path Effect

Composes two Path Effects. It applies the *Inner Effect* to the path and the applies the *Outer Effect* to the result of that operation.



Outer Effect

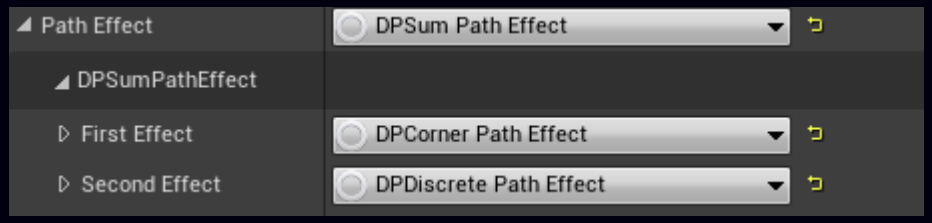
This effect is applied to the result of applying the *Inner Effect* to the path.

Inner Effect

This effect is the first to be applied to the path.

DP Sum Path Effect

Sums two Path Effects. It applies the *First Effect* and the *Second Effect* to the path and sums the results of these operations.



First Effect

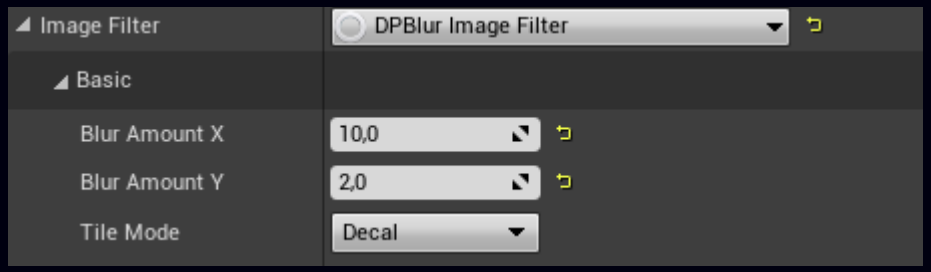
The first effect,

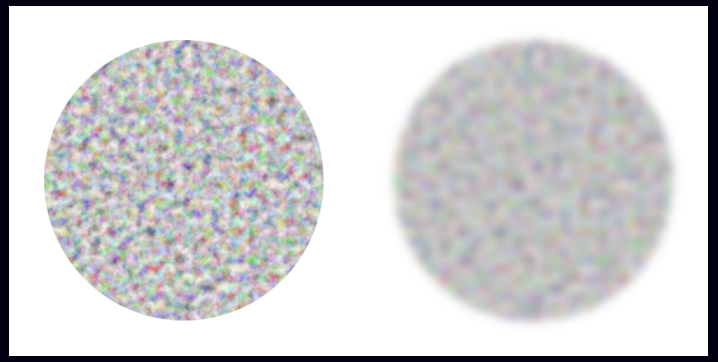
Second Effect

The second effect.

DP Blur Image Filter

Applies a Gaussian blur filter to the whole Element.





A circle without (left) and with the Blur Image Filter (right) with Blur Amount X (10), Blur Amount Y (10), and Tile Mode (Decal).

Blur Amount X

Blur amount in the X direction.

Blur Amount Y

Blur amount in the Y direction.

Tile Mode

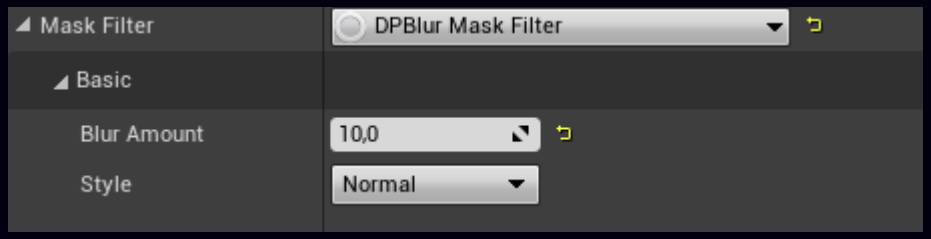
Controls drawing outside of the Element area.

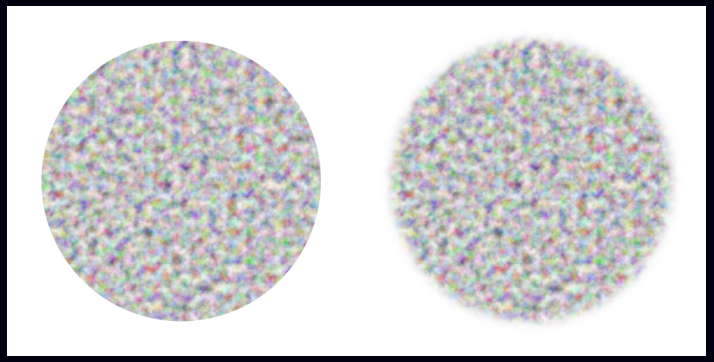
A. Clamp
   Repeat the edge color.
B. Repeat
   Repeats the blurred Element.
C. Mirror
   Repeatedly mirrors the blurred Element.
D. Decal
   Fills the area outside of the Element with transparent black.

Mask Filters

DP Blur Mask Filter

Applies a Gaussian blur filter to the Element's borders.





A circle without (left) and with the Blur Mask Filter (right) with Blur Amount (10) and Style (Normal).

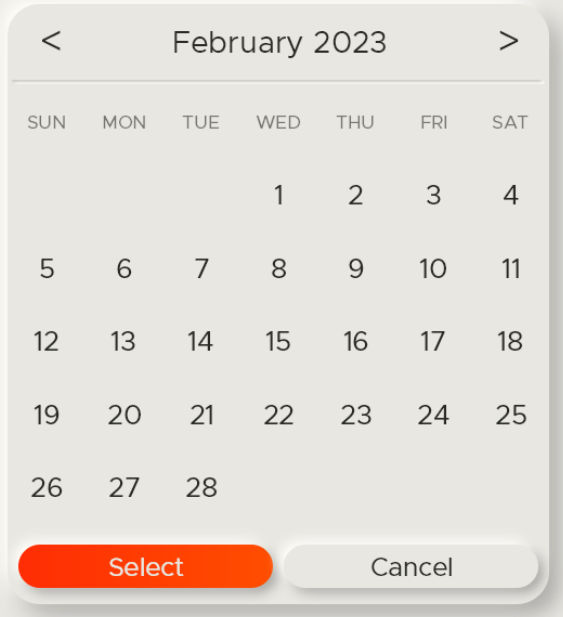
Blur Amount

Blur amount.

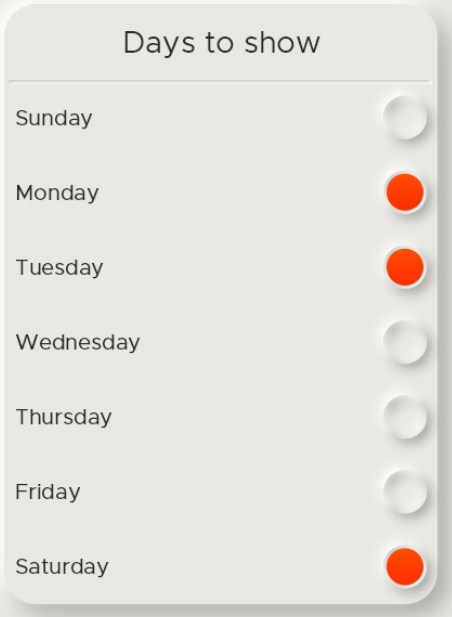
Style

Controls the blur style.

A. Normal
   Blurs the border to both inside and outside of the Element.
B. Solid
   Blurs the border only to the outside of the Element.
C. Outer
   Blurs the border only to the outside of the Element. The inside of the Element is filled with transparent black.
D. Inner
   Blurs the border only to the inside of the Element. The outside of the Element is filled with transparent black.

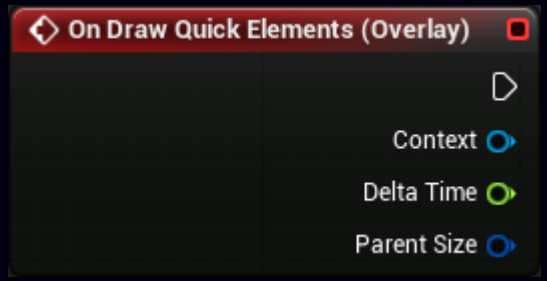

< February 2023 >
SUN MON TUE WED THU FRI SAT
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28
Select
Cancel


SUN MON TUE WED THU FRI SAT
Weekly productivity
All workers
Some workers


Days to show
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

Quick Elements

Quick Elements are basic shapes like rectangles, circles, and even paths, etc. drawn within the bounds of the Widget in the On Draw Quick Elements event. These Elements are not managed by the Widget Tree and ignore user input so they render much faster.



Context UDPContext

A Context that must be passed as an argument to every Quick Element draw call.

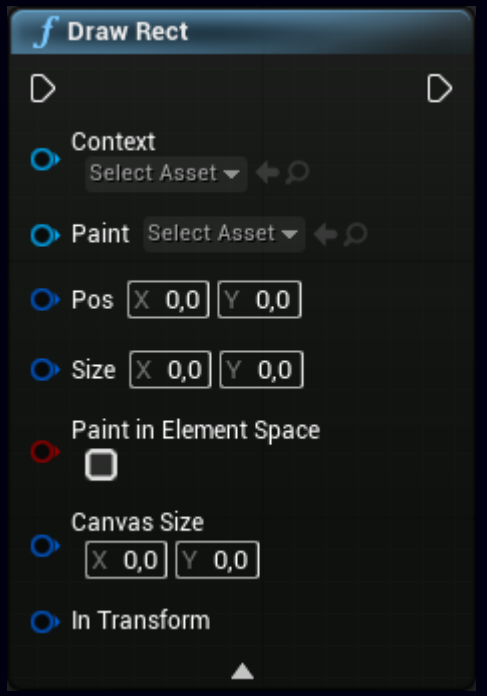
Delta Time float

The time elapsed since the last frame in seconds.

Parent Size FVector2D

The size of the Widget.

Draw Rect



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Pos FVector2D

Rectangle position.

Size FVector2D

Rectangle size.

Paint in Element Space bool

When enabled, the *Canvas Size* will be used as the reference size for generating Paint effects. Otherwise the Element's size will be used for that purpose.

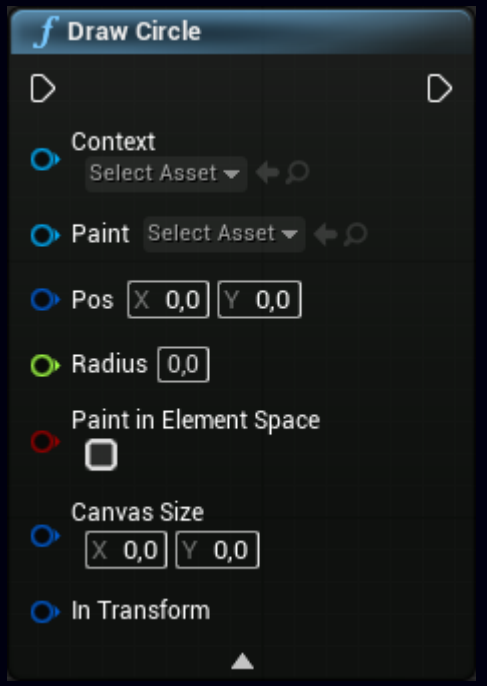
Canvas Size FVector2D

The reference size for generating Paint Effects. Has effect only with enabled *Paint in Element Space*.

In Transform FDPQuickTransform

Element transformation relative to the Widget. Learn more here.

Draw Circle



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Pos FVector2D

Circle center.

Radius float

Circle radius.

Paint in Element Space bool

When enabled, the *Canvas Size* will be used as the reference size for generating Paint effects. Otherwise the Element's size will be used for that purpose.

Canvas Size FVector2D

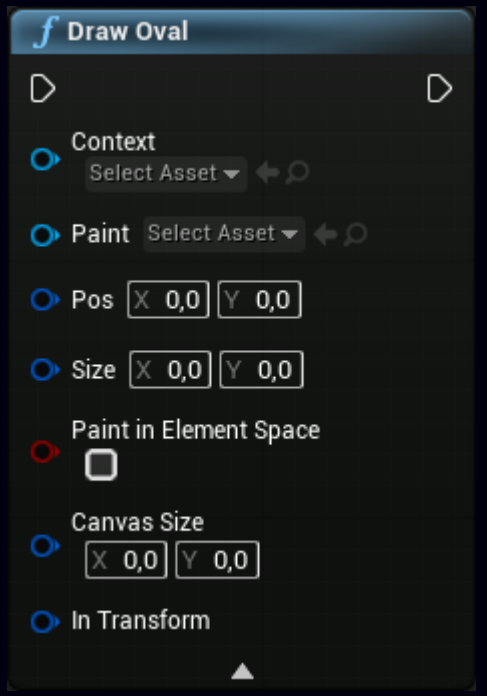
The reference size for generating Paint Effects. Has effect only with enabled *Paint in Element Space*.

In Transform FDPQuickTransform

Element transformation relative to the Widget. Learn more here.

Draw Oval

Learn more about ovals here.



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Pos FVector2D

Bounding rectangle position.

Size FVector2D

Bounding rectangle size.

Paint in Element Space bool

When enabled, the *Canvas Size* will be used as the reference size for generating Paint effects. Otherwise the Element's size will be used for that purpose.

Canvas Size FVector2D

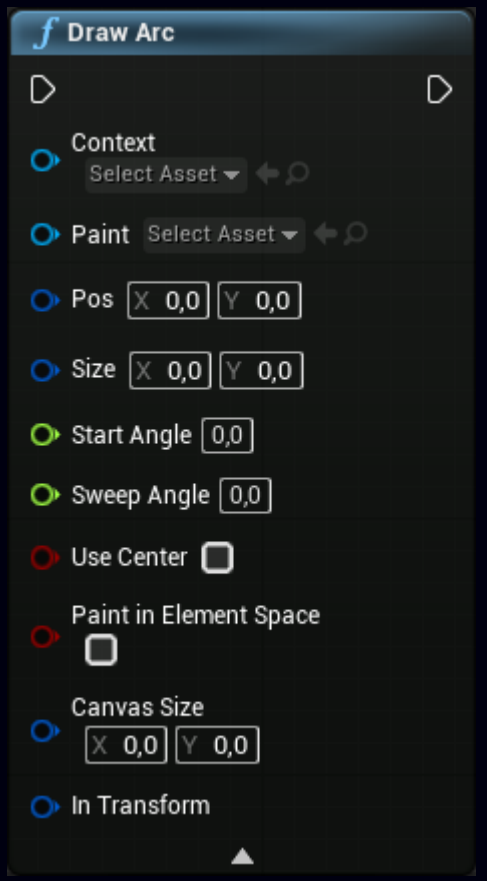
The reference size for generating Paint Effects. Has effect only with enabled *Paint in Element Space*.

In Transform FDPQuickTransform

Element transformation relative to the Widget. Learn more here.

Draw Arc

Learn more about arcs here.



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Pos FVector2D

Bounding rectangle position.

Size FVector2D

Bounding rectangle size.

Start Angle float

Start angle of the arc in degrees. Starts at 3 o'clock and goes counterclockwise when increased.

Sweep Angle float

Defines the angle span of the arc in degrees. Starts at the *Start Angle* and goes counterclockwise.

Use Center bool

Connect both ends of the arc with its center. Has no effect when used with a filled Paint.

Paint in Element Space bool

When enabled, the *Canvas Size* will be used as the reference size for generating Paint effects. Otherwise the Element's size will be used for that purpose.

Canvas Size FVector2D

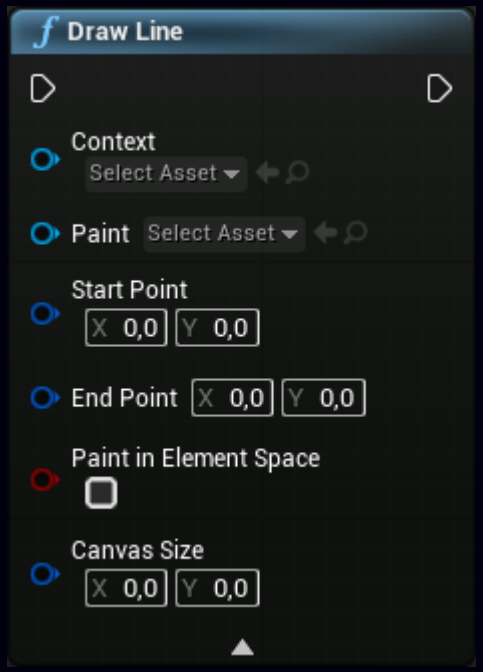
The reference size for generating Paint Effects. Has effect only with enabled *Paint in Element Space*.

In Transform FDPQuickTransform

Element transformation relative to the Widget. Learn more here.

Draw Line

Draws a straight line from the *Start Point* to the *End Point*.



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Start Point FVector2D

Line start point.

End Point FVector2D

Line end point.

Paint in Element Space bool

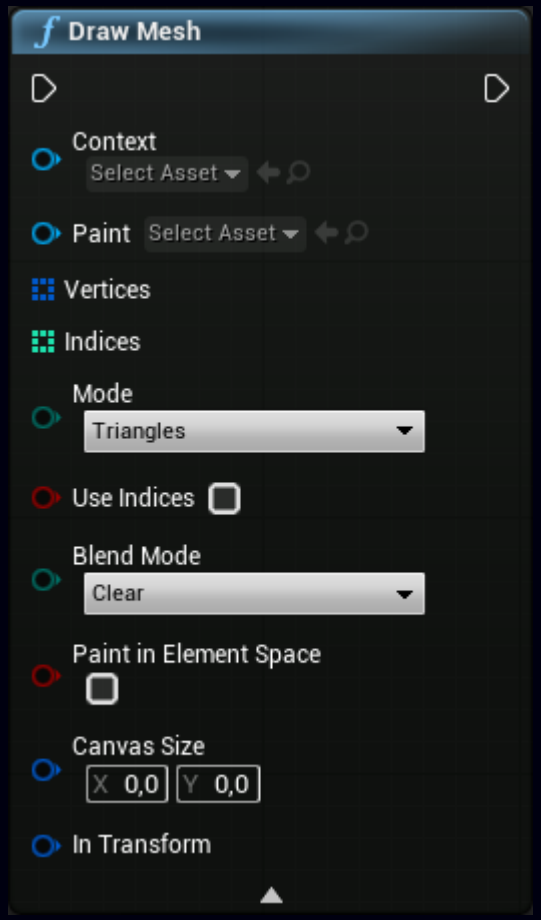
When enabled, the *Canvas Size* will be used as the reference size for generating Paint effects. Otherwise the Element's size will be used for that purpose.

Canvas Size FVector2D

The reference size for generating Paint Effects. Has effect only with enabled *Paint in Element Space*.

Draw Mesh

Learn more about meshes here.



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Vertices FDPVertex (array)

An array of vertices defining the mesh. Learn more here.

Indices int (array)

An array of integer indices defining the order in which the vertices are used to construct the mesh. Learn more here.

Mode EDPVertexMode

Defines the method of constructing the mesh. Learn more here.

Use Indices bool

Use indices when constructing the mesh.

Blend Mode EDPBlendMode

Defines the blending mode of vertex colors and the Widget's Paint.

Paint in Element Space bool

When enabled, the *Canvas Size* will be used as the reference size for generating Paint effects. Otherwise the Element's size will be used for that purpose.

Canvas Size FVector2D

The reference size for generating Paint Effects. Has effect only with enabled *Paint in Element Space*.

In Transform FDPQuickTransform

Element transformation relative to the Widget. Learn more here.

Make Path

Makes a Quick Path object that can be filled with commands and used to draw a path.

Path FDPQuickPath

A Quick Path object.

Path Move To

Adds beginning of a new contour at the specified coordinates. Learn more about this command here.

Path FDPQuickPath

A Quick Path object.

Coords FVector2D

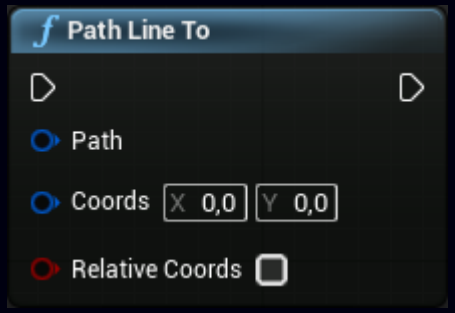
Coordinates at which the new contour should begin.

Relative Coordinates bool

If enabled, all coordinated specified in this command are relative to the last point of the path.

Path Line To

Adds a line from the last point of the path ending at the specified coordinates. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Coords FVector2D

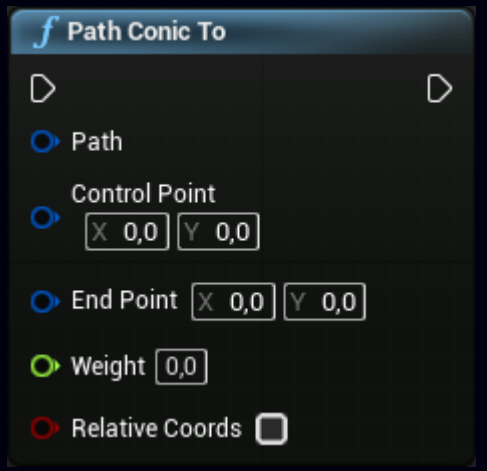
Coordinates at which the line ends.

Relative Coordinates bool

If enabled, all coordinated specified in this command are relative to the last point of the path.

Path Conic To

Adds a conic curve going from the last point of the path toward the *Control Point* and ending at the *End Point*. The curve is weighted by the parameter *Weight*. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Control Point FVector2D

Defines the control point toward which the curve heads.

End Point FVector2D

Defines the end point of the curve.

Weight float

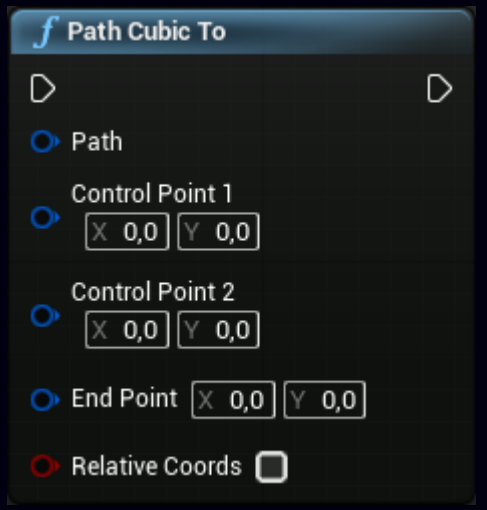
Defines the weight of the curvature of the curve.

Relative Coordinates bool

If enabled, all coordinated specified in this command are relative to the last point of the path.

Path Cubic To

Adds a cubic curve going from the last point of the path toward the *1st Control Point*, then toward the *2nd Control Point* and ending at the *End Point*. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Control Point 1 FVector2D

Defines the 1st control point toward which the curve heads.

Control Point 2 FVector2D

Defines the 2nd control point toward which the curve heads.

End Point FVector2D

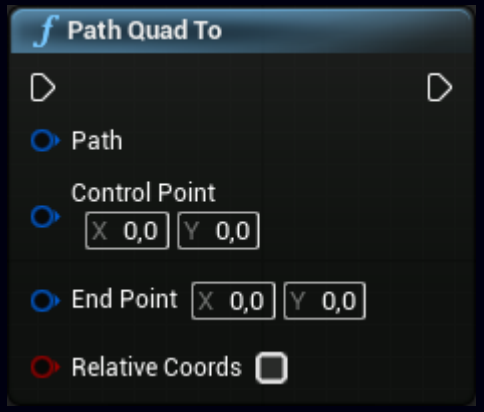
Defines the end point of the curve.

Relative Coordinates bool

If enabled, all coordinated specified in this command are relative to the last point of the path.

Path Quad To

Adds a quad curve going from the last point of the path toward the *Control Point* and ending at the *End Point*. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Control Point FVector2D

Defines the control point toward which the curve heads.

End Point FVector2D

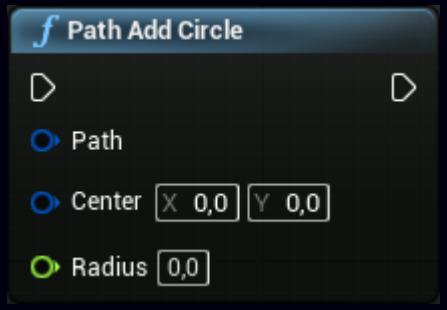
Defines the end point of the curve.

Relative Coordinates bool

If enabled, all coordinated specified in this command are relative to the last point of the path.

Path Add Circle

Adds a circle with the center at the *Center* and radius *Radius*. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Center FVector2D

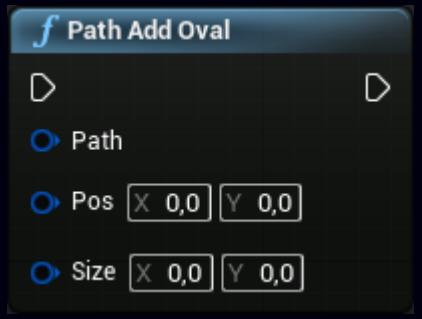
Center of the circle.

Radius float

Radius of the circle.

Path Add Oval

Adds an oval drawn within the bounds of a rectangle defined by *Position* and *Size*. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Pos FVector2D

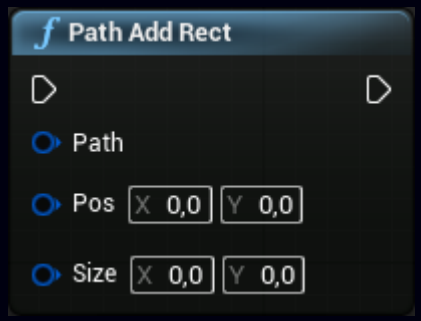
Position of the top left corner of the bounding rectangle.

Size FVector2D

Size of the bounding rectangle.

Path Add Rect

Adds a rectangle with position at *Position* and size *Size*. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Pos FVector2D

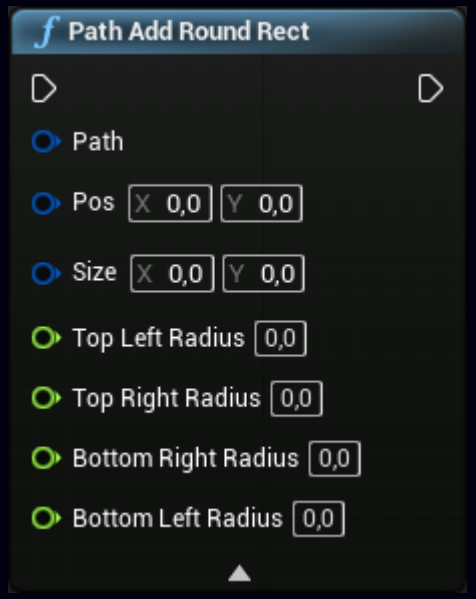
Position of the top left corner of the rectangle.

Size FVector2D

Size of the rectangle.

Path Add Round Rect

Adds a rectangle with position at *Position* and size *Size* and optionally rounded corners. Each corner can have different radius. Learn more about this command here.



Path FDPQuickPath

A Quick Path object.

Pos FVector2D

Position of the top left corner of the rectangle.

Size FVector2D

Size of the rectangle.

Top Left Radius float

Radius of the top left corner.

Top Right Radius float

Radius of the top right corner.

Bottom Right Radius float

Radius of the bottom right corner.

Bottom Left Radius float

Radius of the bottom left corner.

Path Add Poly

Adds a polyline defined by a set of points. Learn more about this command here.

Path FDPQuickPath

A Quick Path object.

Points FVector2D (array)

Points defining the polyline.

Close bool

Close the polyline. If enabled, the polyline is closed by adding a line between the last and the first point.

Path Close

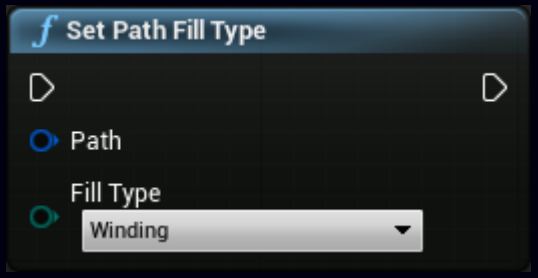
Closes the path by connecting the last point with the first point with a line. Learn more about this command here.

Path FDPQuickPath

A Quick Path object.

Set Path Fill Type

Defines the method of filling the path. Learn more about this command here.



Path FDPQuickPath

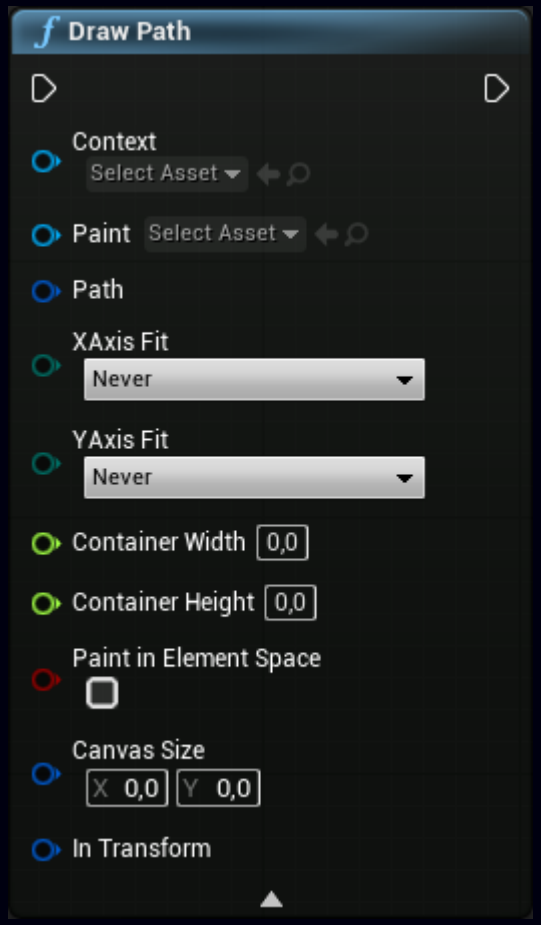
A Quick Path object.

Fill Type EDPPathFillType

Fill type.

Draw Path

Learn more about paths here.



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Path FDPQuickPath

A Quick Path to draw.

X Axis Fit EDPFitType

Scales the path to fit the *Container Width* parameter with the following options:

A. Never
   Disables fitting.
B. If Smaller
   Stretches the path if the path is narrower than the *Container Width*.
C. If Bigger
   Shrinks the path if the path is wider than the *Container Width*.
D. Always
   Combines the last two options.

Y Axis Fit EDPFitType

Does the same as the previous parameter but for the Y axis and the *Container Height* parameter.

Container Width float

Width of the container to fit the path into.

Container Height float

Height of the container to fit the path into.

Paint in Element Space bool

When enabled, the *Canvas Size* will be used as the reference size for generating Paint effects. Otherwise the Element's size will be used for that purpose.

Canvas Size FVector2D

The reference size for generating Paint Effects. Has effect only with enabled *Paint in Element Space*.

In Transform FDPQuickTransform

Element transformation relative to the Widget. Learn more here.

Get Path Length

Returns Path length.

Path FDPQuickPath

A Quick Path to measure.

Return Value float

Path length.

Get Position on Path

Returns the position on the path at the given distance from the beginning.

Path FDPQuickPath

A Quick Path to measure.

Distance float

Distance from the beginning.

Return Value FVector2D

Position on the path.

Get Path Tangent

Returns the tangent vector at the given distance from the beginning.

Path FDPQuickPath

A Quick Path to measure.

Distance float

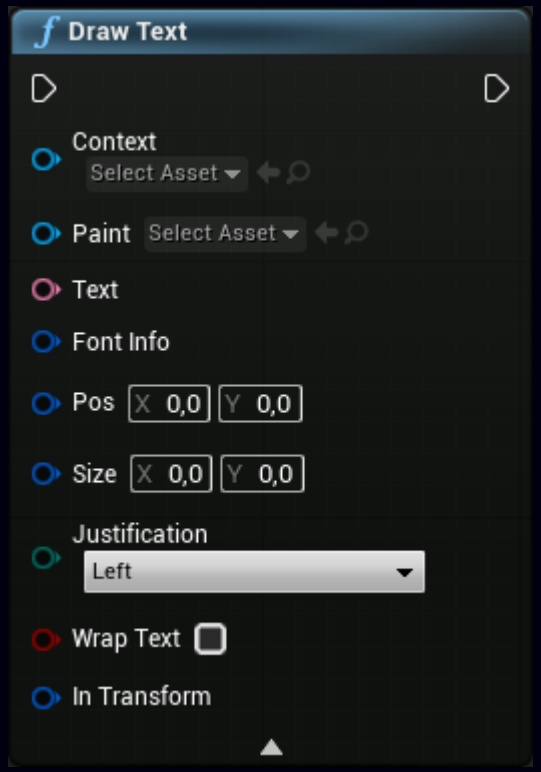
Distance from the beginning.

Return Value FVector2D

Tangent vector.

Draw Text

Draws a block of text.



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Text FText

Text to be drawn.

Font Info FDPFontInfo

Structure of parameters defining font properties. Learn more here.

Pos FVector2D

Text block position.

Size FVector2D

Text block size.

Justification ETextJustify

Text justification.

A. Left
B. Center
C. Right

Wrap Text bool

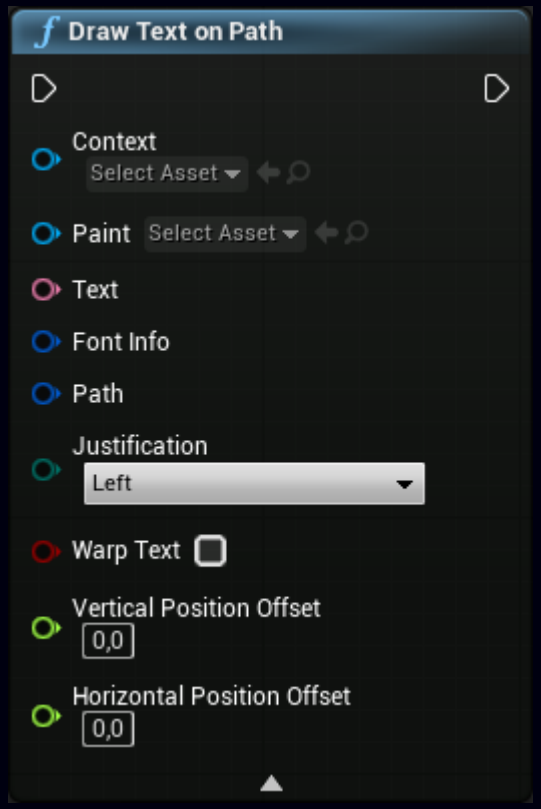
Wrap the text to prevent overflowing of the block.

In Transform FDPQuickTransform

Element transformation relative to the Widget. Learn more here.

Draw Text on Path

Draw a line of text following a Path.



Context UDPContext

A Context that came with the On Draw Quick Elements event.

Paint UDPPaint

The Paint asset used to draw the Element.

Text FText

Text to be drawn.

Font Info FDPFontInfo

Structure of parameters defining font properties. Learn more here.

Path FDPQuickPath

A Quick Path to follow.

Justification ETextJustify

Text justification along the path.

D. Left
A. Center
B. Right

Warp Text bool

Warp individual characters to make them perfectly fit the followed path. Note that this feature is slow and might not work properly on paths with sharp edges.

Vertical Position Offset float

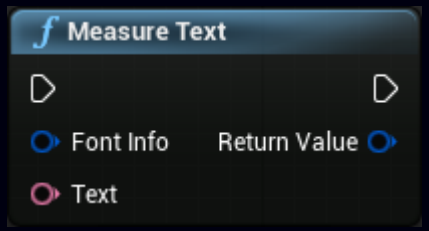
Move the text above or below the path.

Horizontal Position Offset float

Text offset along the path.

Measure Text

Returns the size of a line of text.



Font Info FDPFontInfo

Structure of parameters defining font properties. Learn more here.

Text FText

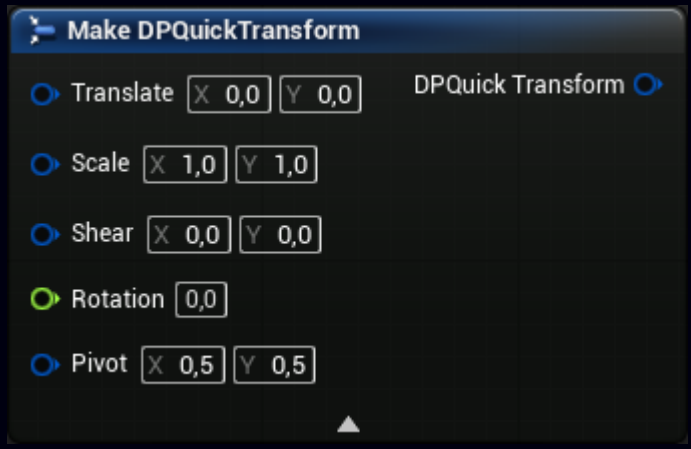
Text to measure.

Return Value FVector2D

Size of the text line.

FDPQuickTransform

DP Quick Transform is a structure holding attributes controlling the transformation of *Quick Elements*.



Translate FVector2D

Element position.

Scale FVector2D

Element scale.

Shear FVector2D

Element shear (skew).

Rotation float

Element rotation in degrees.

Pivot FVector2D

Transformation pivot or center. Default is (0.5, 0.5) at the center of the Element. (0.0, 0.0) would be at the top left corner and (1.0, 1.0) at the bottom right corner. The transformation is performed around this point.

DP Quick Transform FDPQuickTransform

Created transformation.

Performance considerations

The same rule applies here as anywhere else; use only what you really need when you need it. Every Element drawn has its cost. So has every effect applied to a Paint.

Geometry

1. Complexity
   The more complex the geometry, the more expensive it is to draw. Avoid paths whenever possible and try to substitute them with basic shapes like rectangles, circles, arcs, etc.
2. Dynamic
   While generating basic shapes at runtime is fast, generating paths at runtime is quite expensive. Whenever possible, construct paths at startup and reuse them.
3. Size
   The bigger the geometry, the more expensive it is to draw. Always try to keep the screen area occupied by the Element as small as possible.

Paint

1. Opacity
   Set the opacity to anything lower than 1, i.e. make the Element transparent, only when necessary. The same applies to setting the Blend Mode to anything other than *Src Over*.
2. Shadows
   Shadows bring significant performance cost. If you really need to use them (they look cool so I believe you do!), avoid using very high *Blur* amount (>10) and keep them from overlapping each other.
3. Shaders
   Shaders in general are relatively cheap but avoid using a large number (>20) of Stops for Gradients.
4. Path Effects
   The performance cost of Path Effects is heavily dependent on the complexity of the geometry. The more complex the geometry, the more expensive Path Effects are.
5. Filters
   The performance cost of Filters is determined by the *blur amount* and by the screen area covered by the geometry affected by the Filter.

Animations

Animations themselves are very cheap but frequent changes to the transformation of Elements and/or shape of Elements and/or Paint properties might result in a significant impact on performance.

Known limitations and issues

1. Drawing lots of slant path segments may result in a significant performance degradation in the Editor. This issue isn't present in projects packaged for shipping.
2. Mouse hit test and text path follow ignore shape alterations made using Path Effects.
3. Some changes to DP Widgets made in the Widget Blueprint Editor might not take effect until the Widget Blueprint is recompiled.
4. The Editor might occasionally freeze when more than one Widget Blueprint Editors containing at least one DP Canvas Widget and/or DP Paint Editors and/or DP SVG Editors are opened at the same time. This issue isn't present in the PIE mode or in packaged projects.

Changelogs

Changelog 1.0.2

Fixed

- Animations aren't updating in the Editor
- DP Widgets overflow their Scroll Box parent
- Path X&Y fit options are missing from the Quick Draw mode.

Changelog 1.0.1

Fixed

- DP Widgets in the Editor don't redraw when zooming and/or panning the Editor viewport
- DP Path not properly fitting into its Widget bounds
- DP Arc mouse hit test not working for Arcs with enabled center