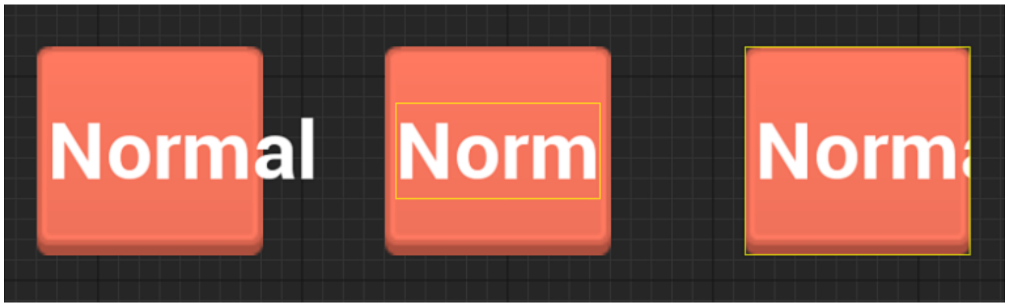
Clipping

An overview of using the Clipping properties within the UMG UI Designer.

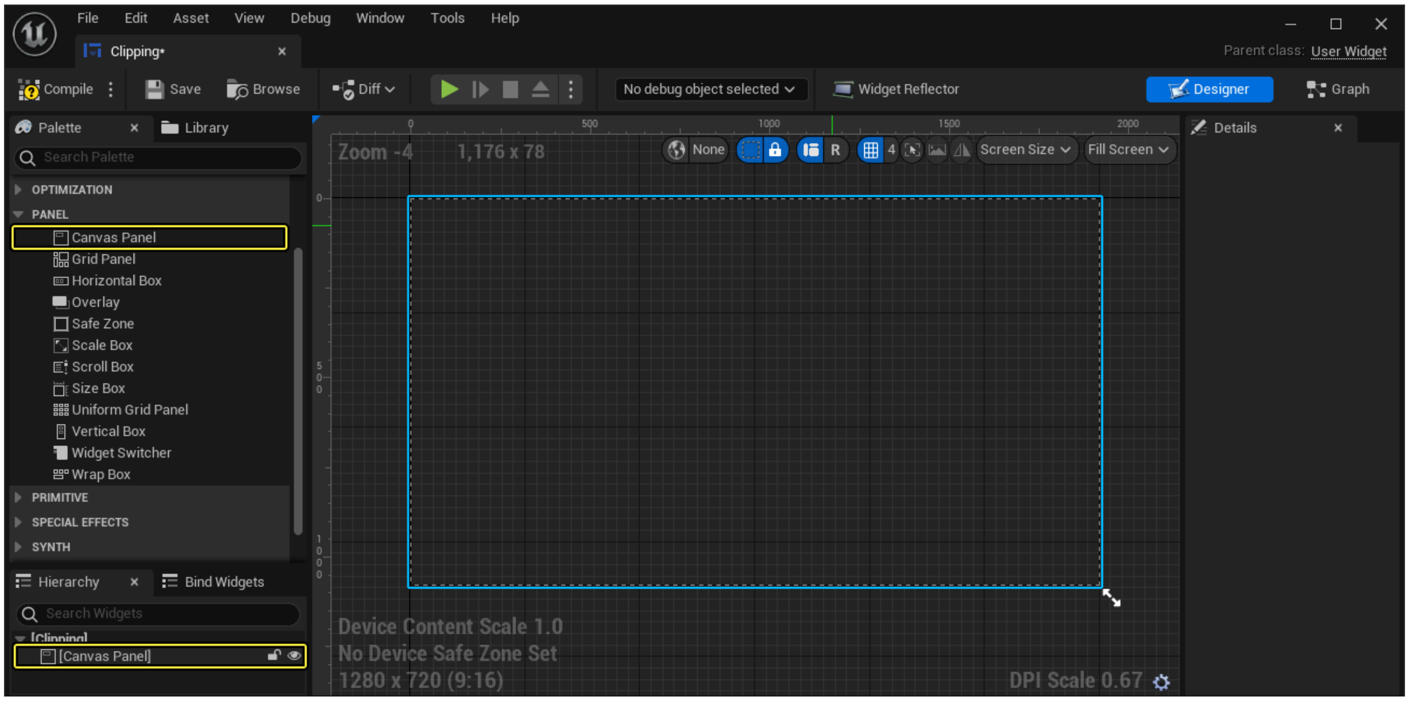
The clipping system in UMG uses Slate's Clipping System as a framework to control how text, images, or content is shown for Widgets (as well as the rest of the Editor). **Clipping** works by restricting rendered objects (graphics and text) to a region using a bounding box so that anything outside of it is not shown. The clipping system is now axis-aligned meaning that it can clip any rotation, which was not possible before because of the way transforms were handled.



In this example, each of the buttons is a parent to the displayed text. These examples demonstrate whether the button or the text is responsible for clipping.

- Left - No clipping is enabled on the button or the text.
- Center - Clipping is enabled on the text
- Right - Clipping is enabled on the button.

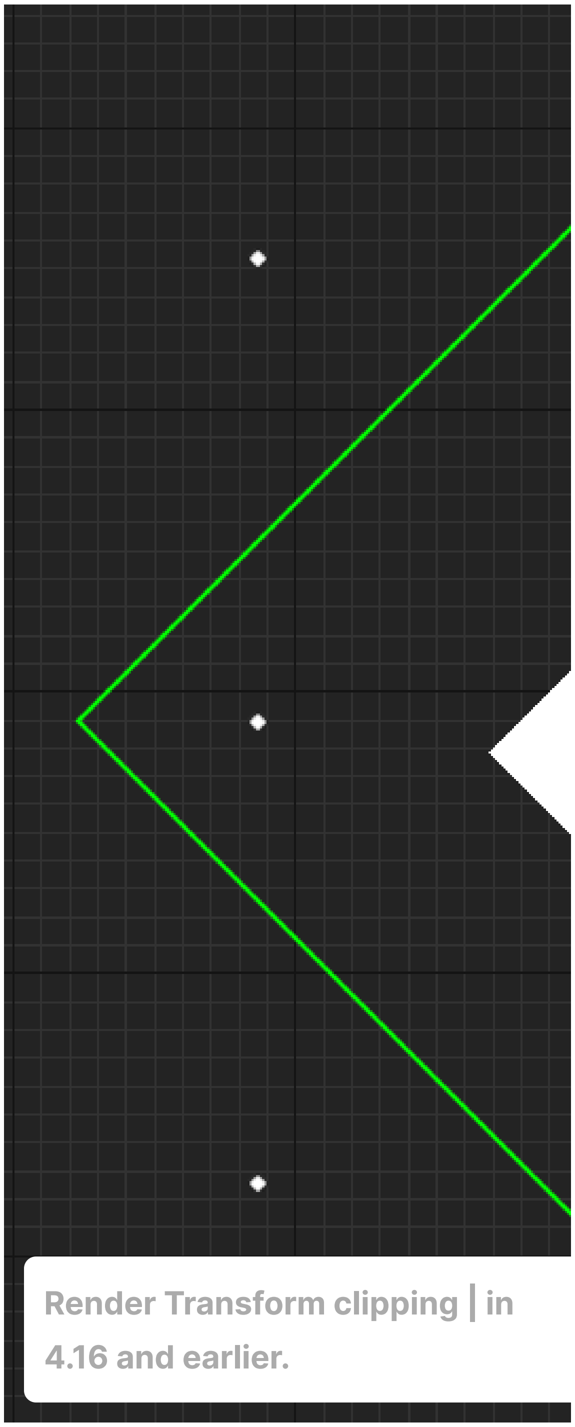
The **Canvas** panel (or clipping zone), outlined in blue, represents your game screen and will clip (or not draw) anything outside of this area for your game.

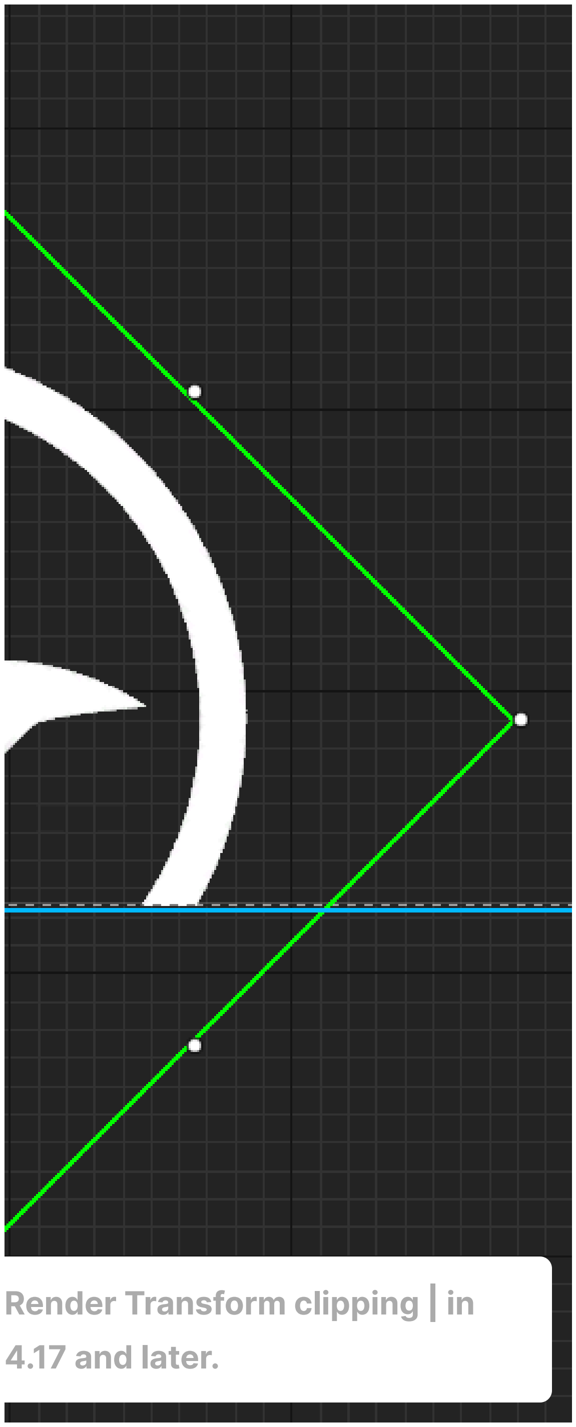


In the UMG Designer Graph, the Canvas panel (blue) represents the clipping zone for your game screen. Click image for full view.

In Unreal Engine 4.16 and earlier, clipping for Widgets was handled using layout-space which prevented anything from rendering outside of the Canvas panel. So if part of a Widget's bounding box was outside of the Canvas panel it would no longer rendered, even if the Widget were rotated, its bounding box would not be leaving parts of the graphic or text cut off even if it's inside the Canvas panel.

Take for instance these examples showing a comparison before and after the changes:

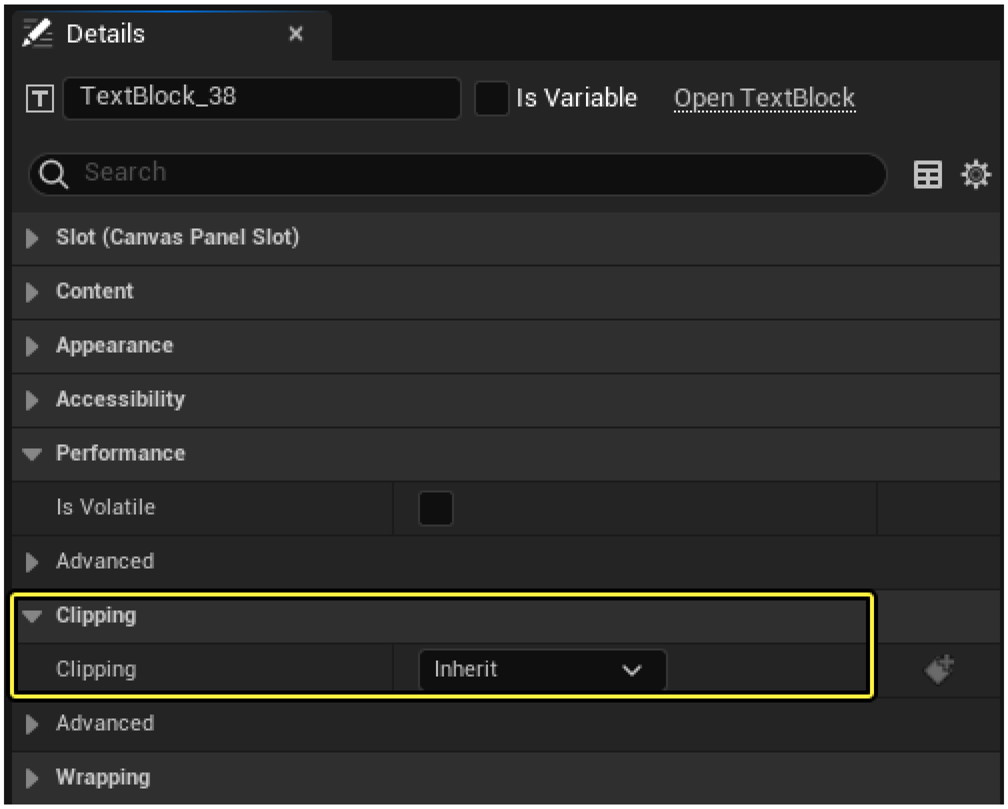

Render Transform clipping | in 4.16 and earlier.


Render Transform clipping | in 4.17 and later.

The Image Widget (left) has been placed at the edge of the Canvas panel so that part of its bounding box is outside. Now that the clipping system is axis-aligned, instead of using layout-space, clipping which resolves artifacts and issues like the comparison (right).

Clipping Properties

You can change how clipping is handled based on the Widget selected in the UMG **Details** panel under **Clipping**.



Property	Description
Inherit	This widget does not clip children and obeys whatever clipping / culling it was passed in from a parent widget.
Clip to Bounds	This widget clips content to the bounds of the widget. It intersects those bounds with any previous clipping area.
Clip to Bounds - Without Intersecting	This widget clips to its bounds. It does not intersect with any existing clipping geometry, it pushes a new clipping state. Effectively allowing it to render outside the bounds of hierarchy that does clip. This will not allow you to ignore the clipping zone that is set to **Clip to Bounds - Always**
Clip to Bounds - Always	This widget clips to its bounds. It intersects those bounds with any previous clipping area.

Property	Description
	This clipping area cannot be ignored, it will always clip children. This is useful for hard barriers in the UI where you never want animations or other effects to break this region.
On Demand	This widget clips to its bound when it's Desired Size is larger than the allocated geometry the widget is given. If that occurs, it will work like **Clip to Bounds**. This mode was primarily added for **Text**, which is often placed into containers that eventually are resized to not be able to support the length of the text. So rather than needing to tag every container that could contain text with [YES], which would result in almost no batching, this mode was added to dynamically adjust the clipping if needed. The reason not every is not set to **On Demand** is because not every panel returns a Desired Size that matches what it plans to render at.

Additonal Considerations

- In most cases, you should not need to adjust the clipping method except in instances where you can't control the length of the text and need to clip it. An example of this would be Scroll Box and Editable Text Widgets, which are set to **Clip to Bounds** instead of Inherit.
- Elements in different clipping spaces cannot be batched together, so there is a performance cost that comes with clipping. For this reason, do not enable clipping unless a panel actually needs to prevent content from showing up outside of its bounds.