

Getting Started





Visit the [Nova website](#) for comprehensive documentation.

Welcome to Nova! This guide aims to provide some basic information about using Nova in your own project as well as point you in the right direction to learn more about specific features or topics.

Creating Content

UIBlocks

All content in Nova is made up of [UIBlocks](#), which are Nova's building blocks (hence the name, **UIBlock**). There are a few types of [UIBlocks](#), each of which can be used for specific types of content:

ICON	TYPE	DESCRIPTION
	UIBlock	Non-visual base type for all other UIBlocks useful for hierarchical purposes or sizing, positioning, and laying out other content.
	UIBlock2D	Renders 2D content such as images , gradients , borders , drop shadows , and more.
	TextBlock	Renders text.
	UIBlock3D	Renders 3D, volumetric content such as rounded-corner cubes, spheres, cylinders, capsules, and more.

Adding [UIBlocks](#) is as simple as right-clicking in the [hierarchy](#) window and selecting `Nova > UIBlock 2D` (or alternatively with the `Add Component` menu to add to an existing `GameObject`).

Sizing and Positioning

[UIBlocks](#) are rendered in world-space using the [Transforms](#) on their respective [GameObjects](#), similar to [MeshRenderers](#) (although Nova does not actually use any [MeshRenderers](#)). This means when you want to position, rotate, or scale an entire menu/panel/etc. built with Nova, simply position, rotate, or scale the [Transform](#) on its root [UIBlock](#) (or one of its ancestors).

Because [UIBlocks](#) are rendered in world space, the values assigned to the various [UIBlock](#) properties, such as [Size](#), [Position](#), [Corner Radius](#), [Border Width](#), and so on, correspond to standard world-space units – meaning an unscaled [UIBlock](#) with a width of `1` will be `1m` wide in the scene. What this often translates to in practice (depending on your desired final world-space size) is requiring very small values in certain situations, such as a [Border Width](#) of `0.01` to achieve a thin [Border](#) on a `1m` wide [UIBlock2D](#).

For this reason, it can be convenient to construct Nova content at much larger sizes, allowing you to operate in larger units across all your Nova components and custom scripts. and then apply a small, **uniform** scale to the [Transform](#) on your content's root [UIBlock](#). For example, most of the Nova [Samples](#) are built with a `1920 x 1080` reference size in mind and then have a scale of `0.001` applied to their UI root [Transforms](#). Constructing content with larger units and scaling down the UI root, as opposed to leaving the root unscaled and using smaller world-space values, is not required and subject to user preference/scenario. The table below highlights some benefits and tradeoffs of both options:

ROOT SCALE	PROS	CONS
Scaled down	More user-friendly adjustments (e.g. +/- 1 rather than +/- 0.001). Easier portability between Nova and other design tools because it's closer to working in "pixel" units.	The size of your Nova content relative to other scene objects needs to be calculated.
Unscaled	Size of Nova content relative to other objects in scene is easily determined.	Will likely require small decimals in many places and makes minor adjustments less user-friendly.

NOTE

Once you have selected a size and scale to work in, you can easily [modify the default values of UIBlocks](#) to better match your scenario.

WARNING

[UIBlocks](#) do not support non-uniform scaling. See [Unity's Transform documentation](#) for more about issues with non-uniform scaling.

API Usage

With few exceptions, the API for modifying [UIBlocks](#) is made up of [get-properties](#) which return the type of the property by [ref](#). This allows direction modification and manipulation of the returned type without requiring a reassignment.

For example, even though the type of the [Border](#) property on the [UIBlock2D](#) is a `struct`, you can see in the following example usage that the properties of the returned [Border](#) can be modified directly.

Property Declaration:

```
public ref Border Border { get; }
```



Example Usage:



```
// Constructs a new Border and assigns it
uiBlock.Border = new Border(Color.white, Length.FixedValue(1));

// Sets the border width to be 10%
uiBlock.Border.Width.Percent = 0.1f;

// Makes the border expand outward
uiBlock.Border.Direction = BorderDirection.Out;
```

Explore More

ICON	TYPE	DESCRIPTION
	Input and Events	Add mouse , touch , and XR interactions with Nova's Input and Event system.
	Data Binding	Create rich, dynamic visualizations for collections with the ListView and GridView .

ICON	TYPE	DESCRIPTION
	Animations	Bring your content to life with Nova's easy-to-use Animation system.
	Editor Workflow	See how Nova's tooling makes building UI faster and easier than ever before.