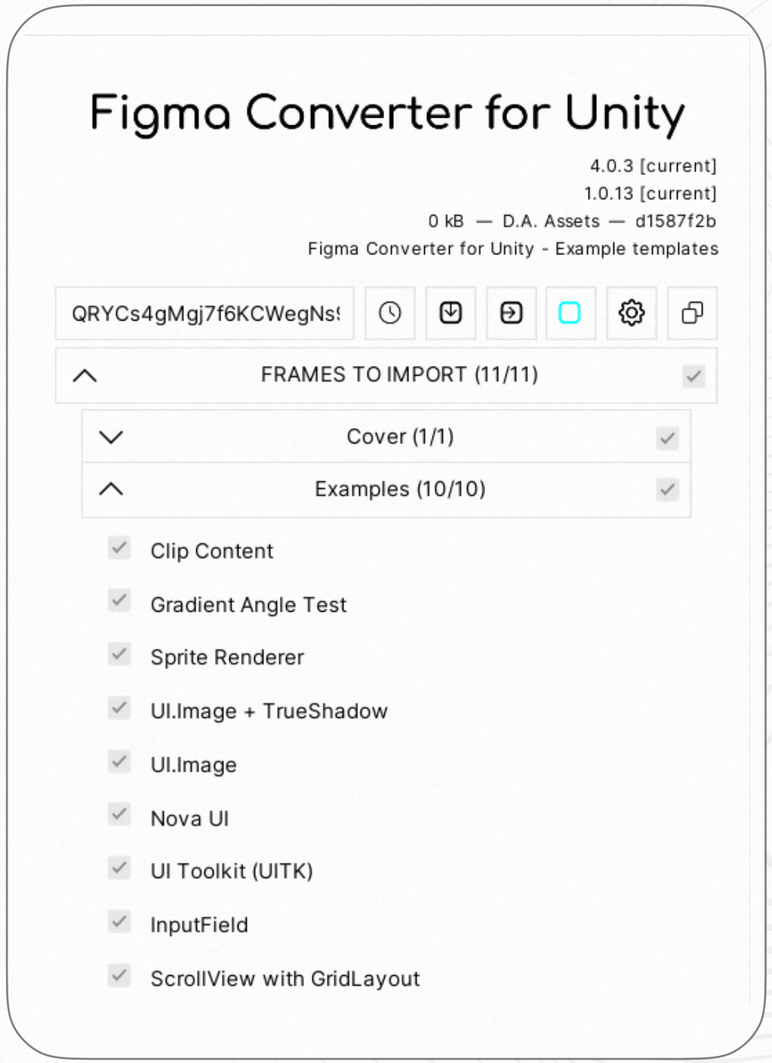


Figma Converter for Unity

manual for designers
5.4.0

Introduction

I recommend reading this manual in its entirety.

1. If you encounter any errors while working with the asset, please write me about it at provided contacts.
I typically respond quickly to messages, offer assistance on an individual basis, and address any identified bugs in the upcoming updates.
You can leave comments about the features that you want to see in the asset - it's will also be considered.

   Discord Server: https://discord.com/invite/ZsnDffV5eE

   Telegram Group: https://t.me/da_assets_publisher

   Email Support: da.assets.publisher@gmail.com

   Website: https://da-assets.github.io/site/

2. Information about changes in the manual can be found in the changelog available on the developer's website.

3. If you see any mistakes in the manual, or oddities or bugs in the operation of the asset, please report it to developer using known contacts.

Contents

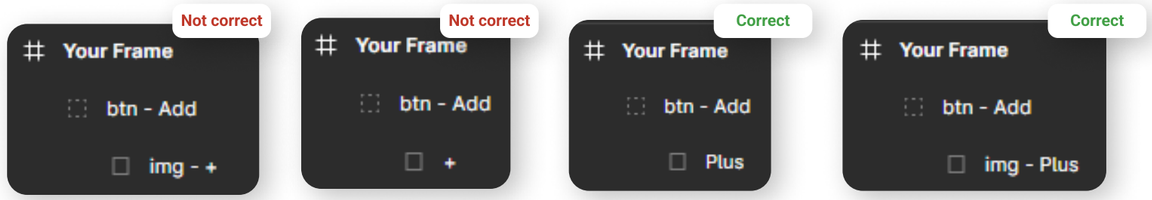
Layout rules

The first 3 points are mandatory.

In most cases, you don't need to make any other changes to your layout before importing to achieve a correct result,

but if you encounter any issues, please review **points 6 and 7** of this slide.

1. Disable "**Clip Content**" on frames before importing.
2. To enable the asset to import **Sections**, apply "**Frame selection**" to them (place the section inside a frame).
3. The text after the tag **should not** consist only of **special characters.**



4. If you wanted a certain component to be imported as a single image, and this did not happen by default, see the **Image** slide.
5. If you want the components within your component to not be merged into a single image, see the **Container** slide.
6. Unity offers various tools to replicate Figma's graphical capabilities, such as procedural images with support for gradients and outlines. Please review "Compatibility Table for Figma Components with Unity Components" to better understand how your imported design will look.
7. Please review the "**Example templates**" to understand what to focus on when designing layouts in Figma for import.

Naming and tags

Actual tag list for **Canvas**:

- img
- cont
- btn
- ignore
- scroll
- fld
- pwd
- ph
- tg
- tgr

Actual tag list for **UITK**:

- img
- cont
- ignore

Actual tag list for **Nova**:

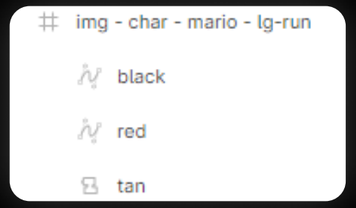
- img
- cont
- ignore

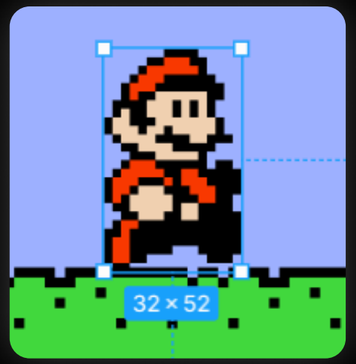
Tags are needed to **clarify the import** of your layout, **or** to **assign scripts** to some objects in Unity.
Component name format with tag: "**tag - name - info**".
Most components don't need a tag.

1. "**tag**" - tag of the component. It is needed so that the importer can understand exactly how to import this object. The **tag is part of** the component **name**.
2. "-" or "/" - required separators. Separates the tag from everything after it.
3. "**name**" - component name. Specify the logical purpose of the component here, for example "**btn - menu open**".
4. "**info**" - any information you want to include in the name of this component.
5. Examples with "**-**" separator:
   btn - menu open
   bg - circle pattern
   img - avatar
6. If tags have been renamed in the new version of the asset and you are experiencing compatibility issues, you can ask the developers to change the values in the "**Figma Tag**" field in the "**Tags**" array in the "**FcuConfig**" file.

Image/Container

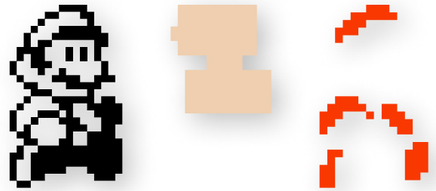
cont - char - mario - lg-run
img - char - mario - lg-run





a1 If your object is imported as a single image, but you want its **internal structure** to be imported, give it a "**cont**" tag.

With cont tag:

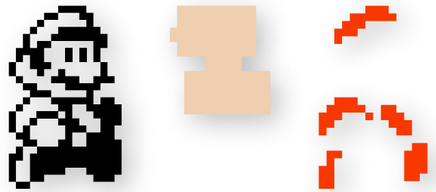


Without cont tag:

a2 If one of the child components of an object contains the word "icon", it is the same as if the parent component had the "cont" tag.

b1 Apply this tag to a parent object if you want that component, along with its child components, to become a **single image** when imported.

Without img tag:

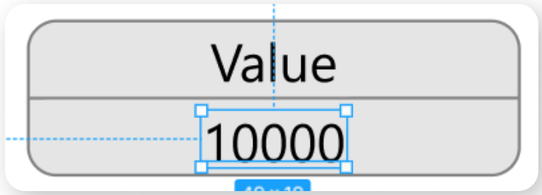


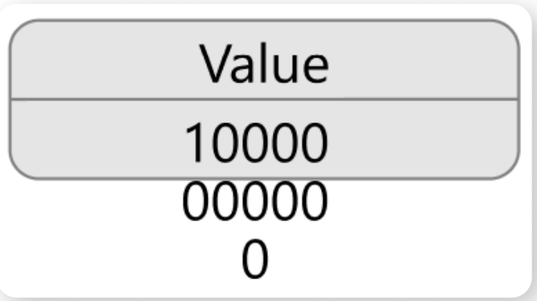
With img tag:

b2 If all children of the parent component are vectors, the parent component automatically will import as a single image.

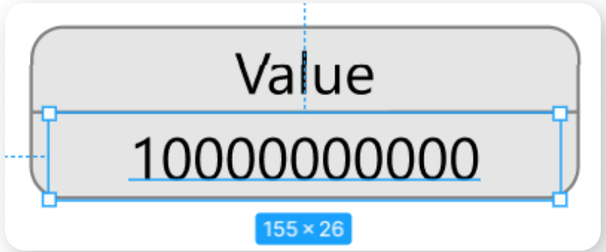
Text

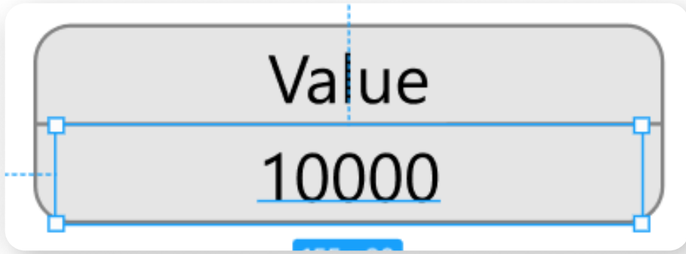
1. Example of how not to do it. If in the future, inside Unity, using an algorithm or manually, we want to set this text component to the value "1000000000", it will not fit and will get out of bounds.





2. Example of the correct size for a text component.





3. The rendering principles for fonts in Figma and Unity differ, so **you may need** to **make adjustments** to your **font file** in Unity (for example, **margins** from the edges) if you're using **TextMeshPro** to achieve an identical look.

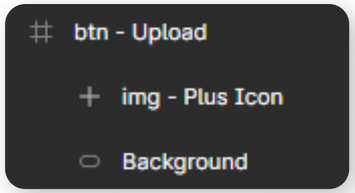
   If you're using **UI.Text**, adjustments cannot be made, so it is recommended to use **TextMeshPro**.

4. **If** you **don't want** to **make** any **changes** to the Figma layout or the font file, you can enable the "**Auto-Size**" option in the "**TEXT & FONTS**" tab in the asset settings. This option is enabled by default.

   In this case, **if** the **text doesn't fit**, its **size** will **be** automatically **reduced**.
   The maximum size will be the size you set in Figma.

UI.Button

1. In order for the button script to be added to it's GameObject during import, you need to add the "**btn**" tag to the beginning of its name in Figma.

2. 

   If you use the "**Unity Button**" without states (def, pres etc.), and you want the "**Color Tint**" animation to be applied to the button (changing color when pressed), the button must be assigned a "**Target Graphic**" - usually an image that is the background of the button.
   In order for the "**Target Graphic**" to be assigned automatically during import, the **image** that is the **background** of the button **must be at the end of the hierarchy** of its child objects.

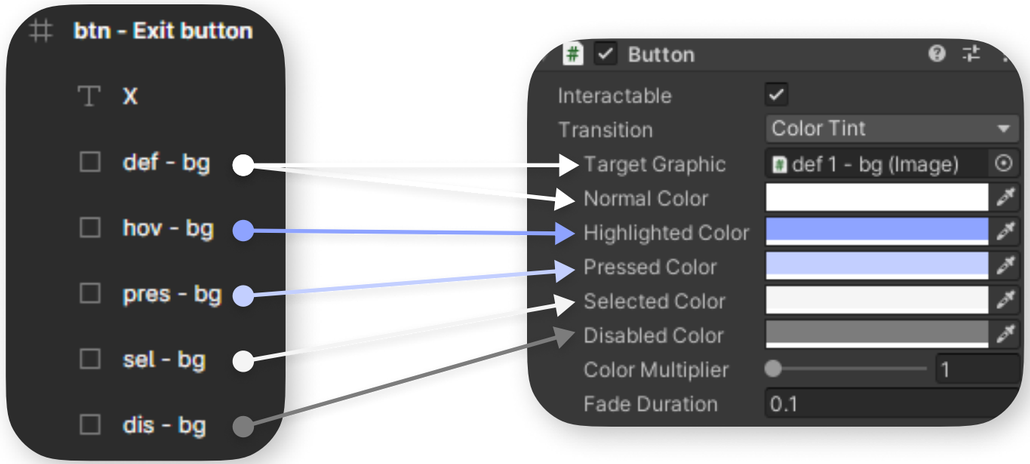
2. **For each** of supported button assets, you can specify special behavior for states "**pressed**", "**hovered**", "**disabled**" (and "**selected**" for ”Unity Button”).
   For this, you need to assign special tags to the button's child components.

   Special tags:
   def - default button state, used when nothing happens to the button;
   pres - used when a button is clicked;
   hov - used when hovering over a button;
   dis - used when a button is disabled (when it cannot be clicked).

   Click here to see an example button in Figma.

3. Please take a look at the scheme showing how objects with tags are assigned to the button component.

UI.Button

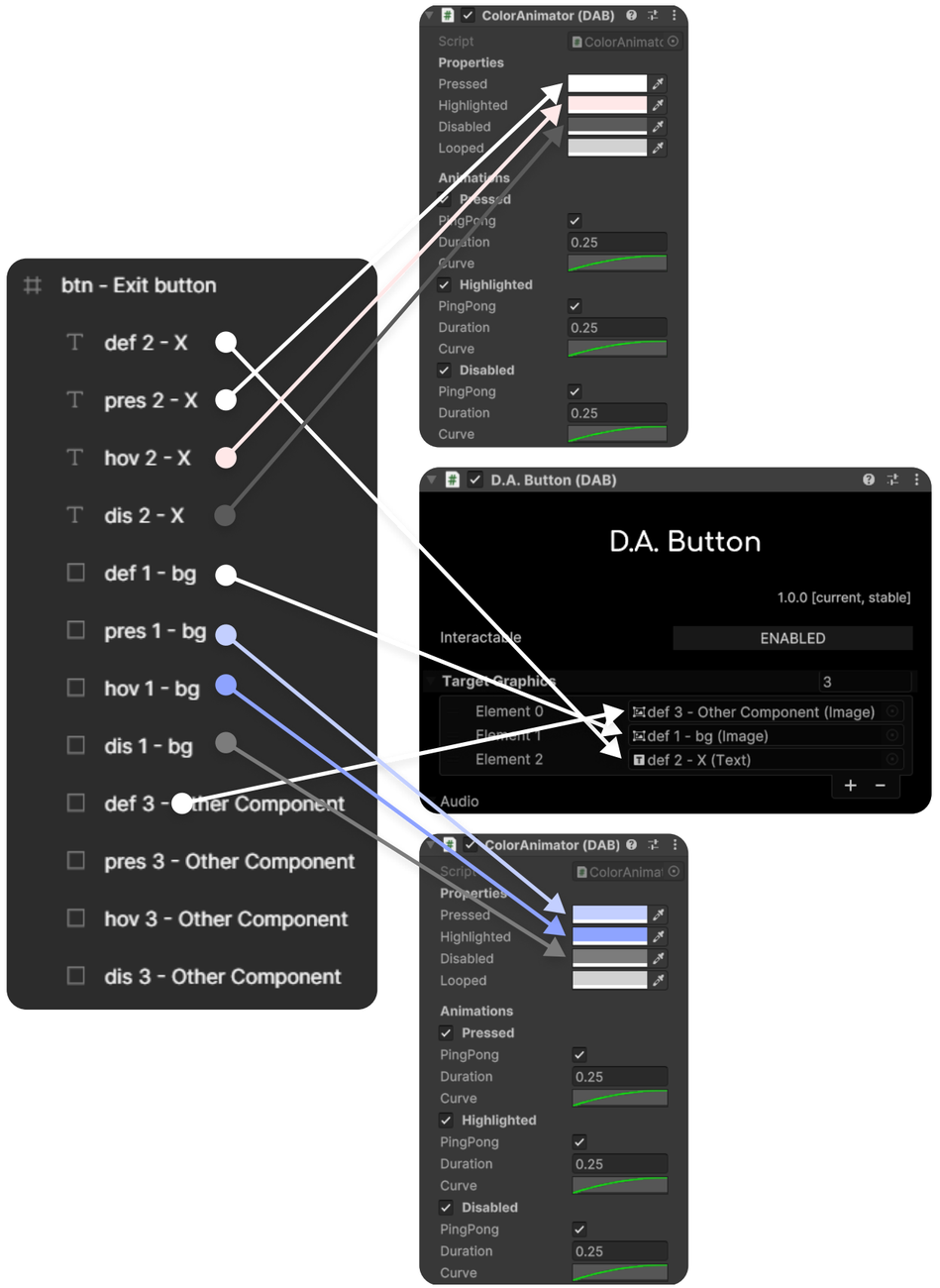


4. As you can see, the component with the "**def**" tag is recognized as TargetGraphic, and from the other components with state tags, only the color is taken, which will then be assigned to the component with the "**def**" tag in Unity.

5. When the button is in **ColorTint** mode, **child objects** of the button with tags "hov", "pres", "sel", "dis" are removed from the scene.

6. If your **TargetGraphic** component cannot be recreated on the scene using Unity's means or with procedural assets, it will be downloaded, and the button transition type will be switched to **SpriteSwap** mode.
In **SpriteSwap** mode, child **image objects** of the button are not removed, regardless of the tag.

D.A. Button



1. D.A. Button supports multiple TargetGraphics, meaning all of them can be animated on click, hover, or disable.

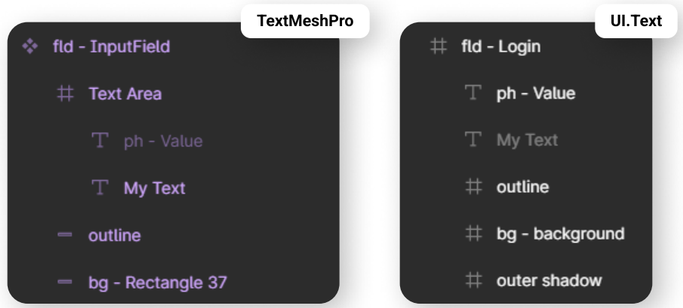
The principle of setting states is the same as in UI.Button (refer to the UI.Button section), but now you can configure similar behavior for an unlimited number of TargetGraphics.

Input Field

fld - Username
pwd - Password

Example: link.

1. To have the **InputField** script automatically added to your objects in Unity, you need to include the tag "**fld**" in the name of your component in Figma.
   You can also use the "**pwd**" tag if you want to create a password input field.
2. For the placeholder, you should use the tag "**ph**".
3. If your InputField contains multiple images, such as a background and a outline, use the tag "**bg**" for the background.
4. Other components within the InputField do not require tags.
5. In Unity, there are two scripts for **InputField**: one supporting **TextMeshPro** and the other supporting **UI.Text**.
   Below you can see examples of the hierarchy for these two scripts.
   Link to the frame with InputField examples: link.

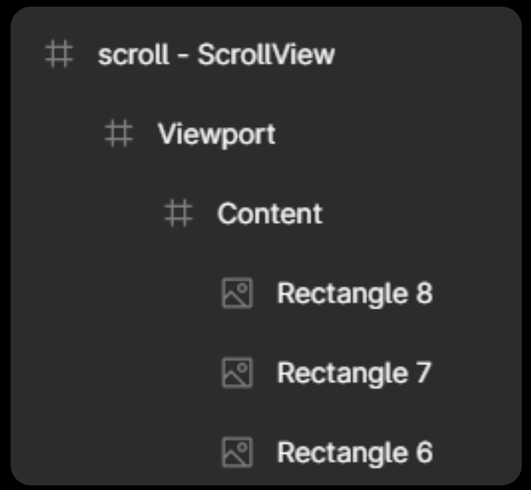


6. You can toggle the **visibility of text components** inside the fld or TextArea, but in Unity, all text components within the InputField **will be visible** regardless.

Scroll View

scroll - My Scroll

Example: link.



1. To have the asset automatically add a **ScrollView** to your Unity scene, the hierarchy of your **ScrollView** must match the one shown in the screenshot.

2. The component with the "**scroll**" tag is the parent component, onto which the "**ScrollRect**" script will be added during the Unity import.

   Disable the "**Clip Content**" option for it.

3. On the Viewport, you need to configure the scroll prototype, where you can set the **Overflow** property.
   Supported "**Overflow**" options:
   • Horizontal
   • Vertical
   • Both directions
   The "**Position**" property configuration is currently not supported.

   The Viewport should be the same size as the parent component with the "scroll" tag.
   Disable the "**Clip Content**" option for it. The RectMask2D component will be automatically added to the Viewport during import (analogous to Figma's Clip Content).

4. Your content, which will be scrolled, should be placed inside the Content object.
   Disable the "**Clip Content**" option for it.
   The Content must always be larger than the Viewport.

5. Regardless of the Constraints you set for these components, the Figma Converter will automatically set the correct Constraints during import.

Toggle/Checkbox

tg - My Toggle
tgr - My Toggle Group

Example: link.

1. To have the "**Toggle**" script automatically added to your objects in Unity, you need to use the "**tg**" tag and the keyword "**Checkmark**".
2. The object that serves as the checkmark in your "**Toggle**" should be named "**Checkmark**."
3. If your "**Toggle**" contains multiple images, such as a background and a outline, use the tag "**bg**" for the background.
4. To group your Toggles, you need to add the "**tgr**" tag to their common parent.
   The parent with the "**tgr**" tag can be at any level of the hierarchy and can group any number of Toggles.

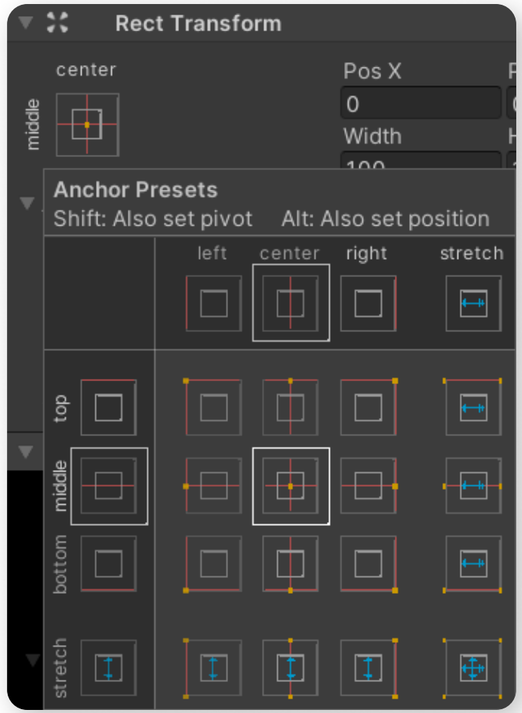
Ignore

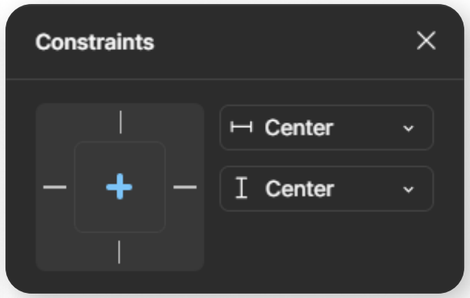
ignore - my object

1. Ignores the current object and all its child objects (the object will not be imported).
The tag is equivalent to the function of disabling the visibility of an object (the closed eye icon in the Figma hierarchy).

UI Responsivity

1. If you need your interface to look correct at different resolutions in Unity, you'll need to configure "**Constraints**." In Unity, "**Constraints**" are called "**Anchors**," and they work identically to Figma.





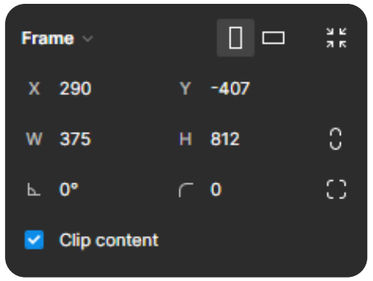
2. The only exception is that in Unity, you can't use a "**Scale**" anchor; instead, you need to use "**Left and Right**" or "**Top and Bottom**" constraints in Figma.

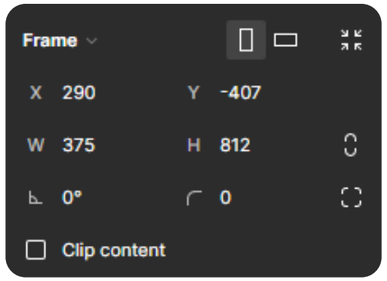
3. You can learn how to set up constraints in Figma from this tutorial: https://www.youtube.com/watch?v=U8N1gLcnATE

Shadows and True Shadow asset

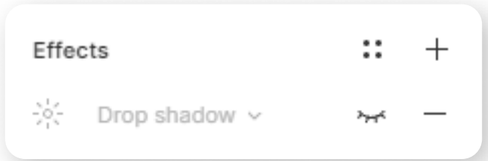
Video manual:
https://www.youtube.com/watch?v=ckyS96RmsY8

1. Starting with version 2.0.0, the **asset supports importing Figma shadows**, so the **shadow** component **don't need to be disabled** before the layout is imported.
If the Figma's shadow extends beyond the container containing the shadow component, the component will be distorted.
To prevent this, **disable** "**Clip content**" feature **for** your **frames**.





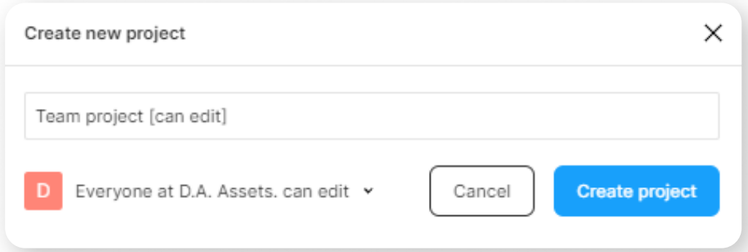
2. If you want to use "**TrueShadow**" asset, you **need to disable** the component's shadow before importing the layout, otherwise "**TrueShadow**" asset's shadow will duplicate the Figma's shadow, and overlay it.



D.A. Assets

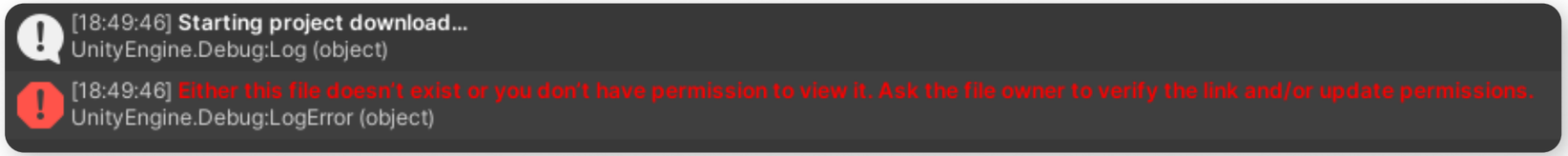
Teamwork

1. You can create a team project with the following **permissions**:
   - can edit
   - can view
   - invite-only



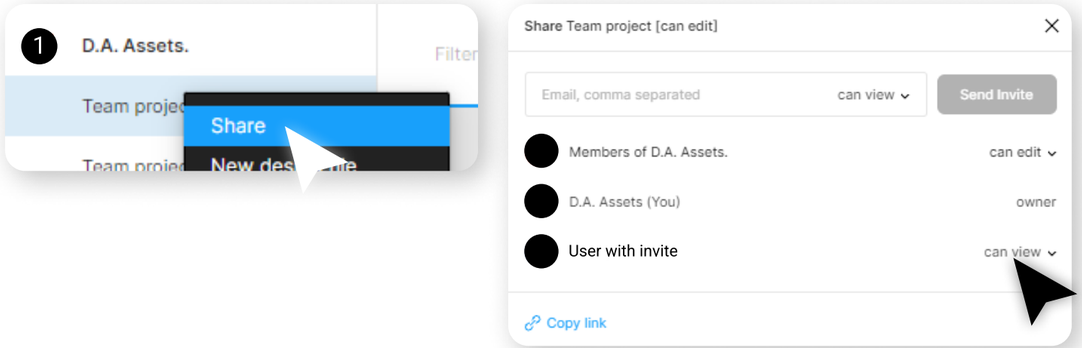
Regardless of which permission you choose, if you select the **account** from which this **project** was **created** when **authorizing** inside an **asset** in Unity, you will be able to **import frames** from it.
But in order for the frames to be imported through **other accounts** that you want to **give access to the project**, you need to follow certain steps. See below.

2. Consider a team-project with the "**can edit**" permission.
If user **A** is the **owner** of the project, and user **B** tries to download the project from the link, and at the same time, user **B** does **not have acces**s to the project, an error will occur during import:

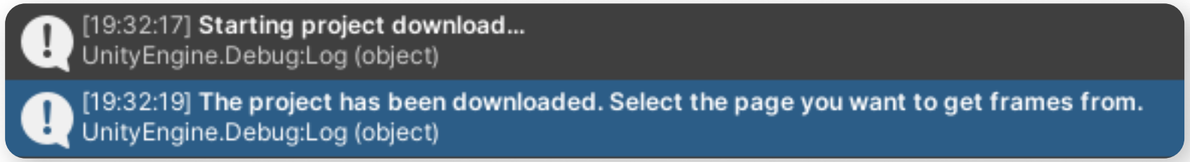


Teamwork

3 Send an invite to the **email** of the **user** you want to **give access** to the project.
In order for this user to be able to import a project using an asset, the "**can view**" permission is enough.



4 **After** the user **accepts** an **invite** with the correct permissions, he will be able to **import** the project.



5 If, for some reason, you have an **error** with permissions, or layout pictures are not downloading, **try re-inviting** the user who imports the layout.
Also, some users were helped by **re-creating** the project with the components of the **design system** and the project in which **instances** of these **components** are **used**.
I have kept the original wording of the "**Either this file doesn't exist**" error, so information on it **can be found** on the **Figma forum** or on Google. You may be able to find more precise instructions on how to resolve the permission problem for your situation.
If none of the above helped and the project does not import - create your own project using the account you are logged into the asset, copy the necessary frames to this project, and try to import this project.

Uniqueness of the components

To prevent components with the same names from **overwriting** each other **after downloading**, a random set of numbers is added to their names before downloading, which is calculated based on the properties of these components and all the components that are inside it.
If these properties change, the set of numbers also changes, and then a new file will be downloaded.

To avoid duplicating prefabs and images in your project, use the component creation function in Figma.

Since version 2.1.0 you **can give** components the **same name**, and if the properties of these components and their child components are different, it’s will **not overwrite each other.**