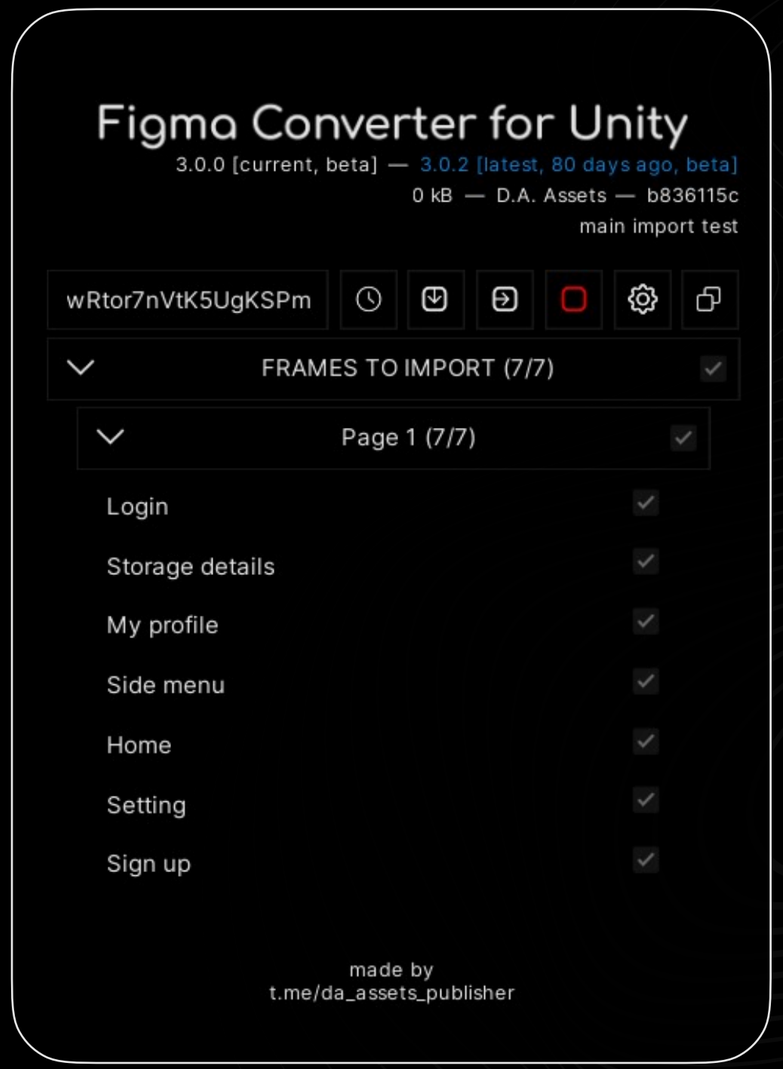


Figma Converter for Unity

manual for developers

3.1.5

Introduction

I strongly recommend reading
this guide before using the asset.

1. This asset can work in conjunction with other graphics assets using their capabilities.
Currently supported assets:
Shapes2D, TextMeshPro, **True Shadow, Modern Procedural UI Kit, Procedural UI Image** and **I2Localization.**
In order to work with these assets, you need to **buy** them from the **Asset Store** and **import** them **into** your **project**.
After you have done this, carefully **read** both **manuals** - for developers and for designers.
In the contents of these manuals, you will find **page numbers** for information on the **use** of these **assets** and their corresponding **tags**, if any.

2. 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 errors in the upcoming updates.

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

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

Telegram Support: https://t.me/da_assets

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

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

Changelog: https://t.me/s/fcu_changelog

3. Usually, **to reproduce your issue, I need access to your project in Figma.**
If you are working in a company, you might need to coordinate granting Figma project access with your management.
I follow a confidentiality policy, your project will not be used for any purposes other than assisting with your issue.

4. You can earn a percentage from sales of this asset through the Unity Affiliate Program.
If this interests you, please contact me via PM or email.

Contents

4 Scene setup

5 Installing Json.NET

8 Auth

11 Google Fonts

12 Import frames

14 Creating prefabs

16 Import issues

19 Layout updating

25 Scene backups and project cache

26 Asset UI

27 Main settings tab

30 Unity components tab

32 Fonts tab

34 Dependencies tab

37 Shapes2D, MPUIKit, Procedural UI Image

38 TrueShadow

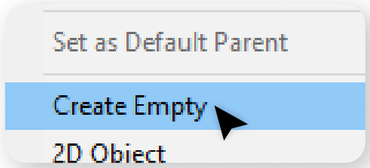
39 Localization

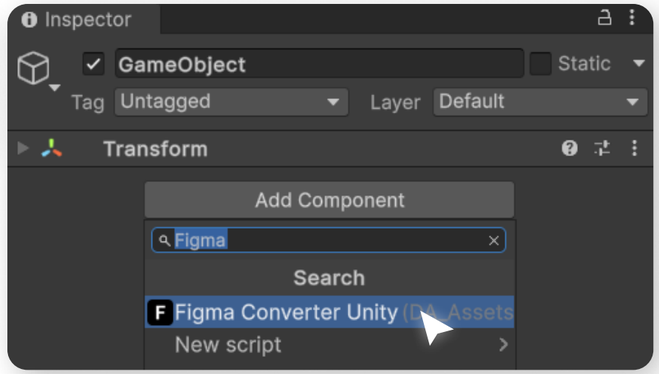
40 Buttons

41 Context menu

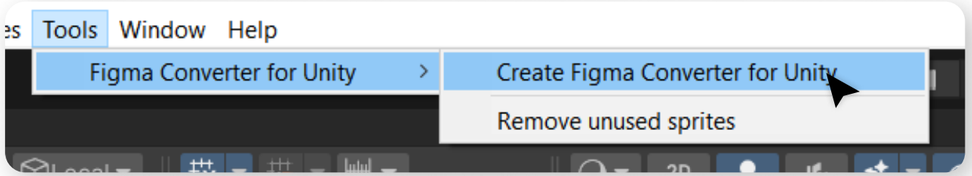
Scene Setup

1. In order to start using the **FCU** asset, create an empty GameObject on the scene, and then add the **FigmaConverterUnity** script on it.



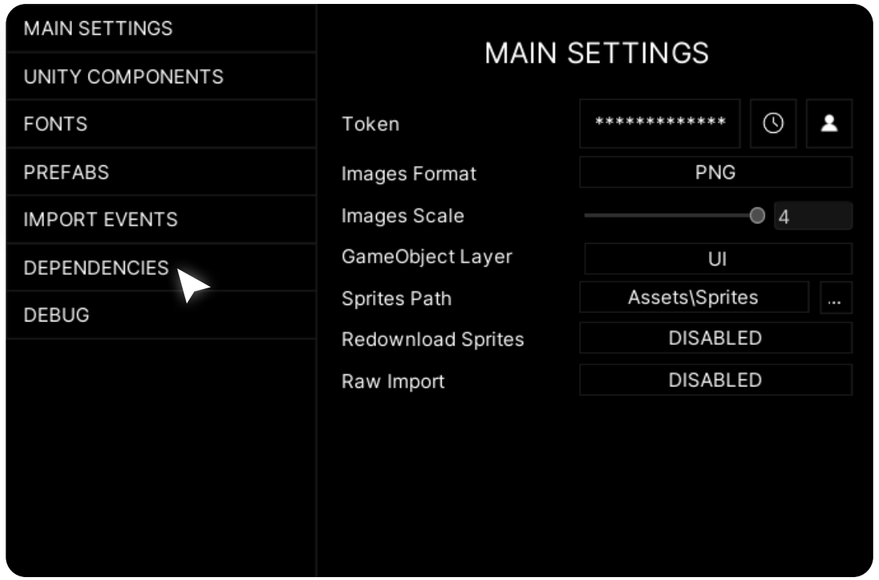


2. Also, you can create an asset in the scene using the menu.



Json.NET

1. The asset **requires the Json.NET** library for its operation.
To check if you have Json.NET installed, open the asset settings and go to the “**DEPENDENCIES**” tab.

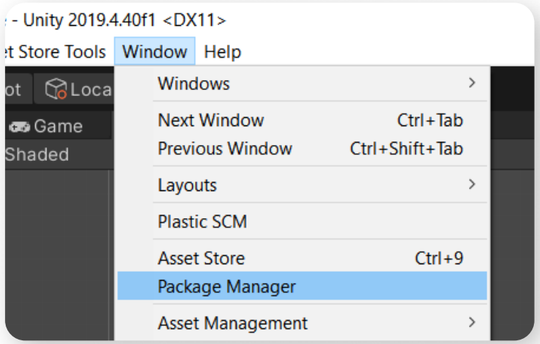


2. If Json.NET is marked as **ENABLED**, you **don't need to install** Json.NET and can proceed to the next section of the manual.

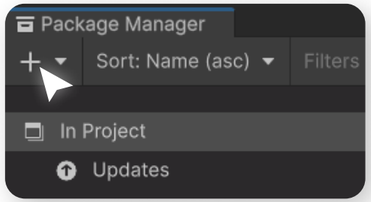
If Json.NET is marked as **DISABLED**, follow the instructions below.

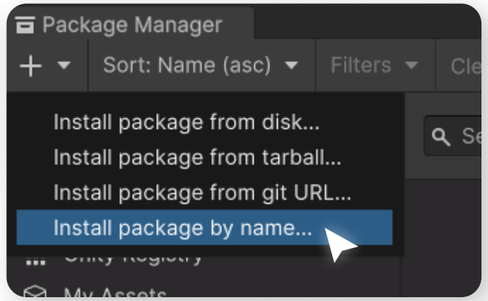
Json.NET

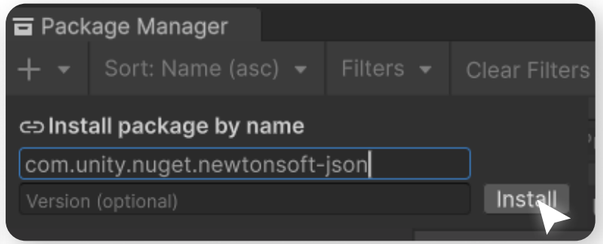
3 To install **Json.NET**, open the Unity Package Manager.



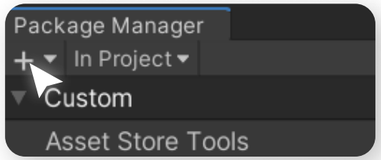
4 Click on the "**+**" button, and then, in the menu that appears,
click on "**Install package by name**" menu item.
Enter the package name "**com.unity.nuget.newtonsoft-json**" and click on the "Install" button.

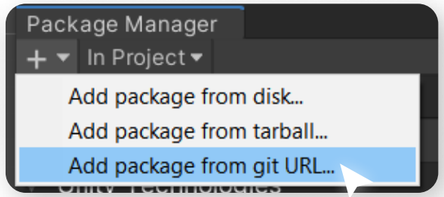


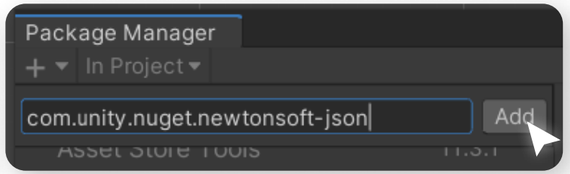




5 As an alternative, you can use the "**Install package from git URL**" function.



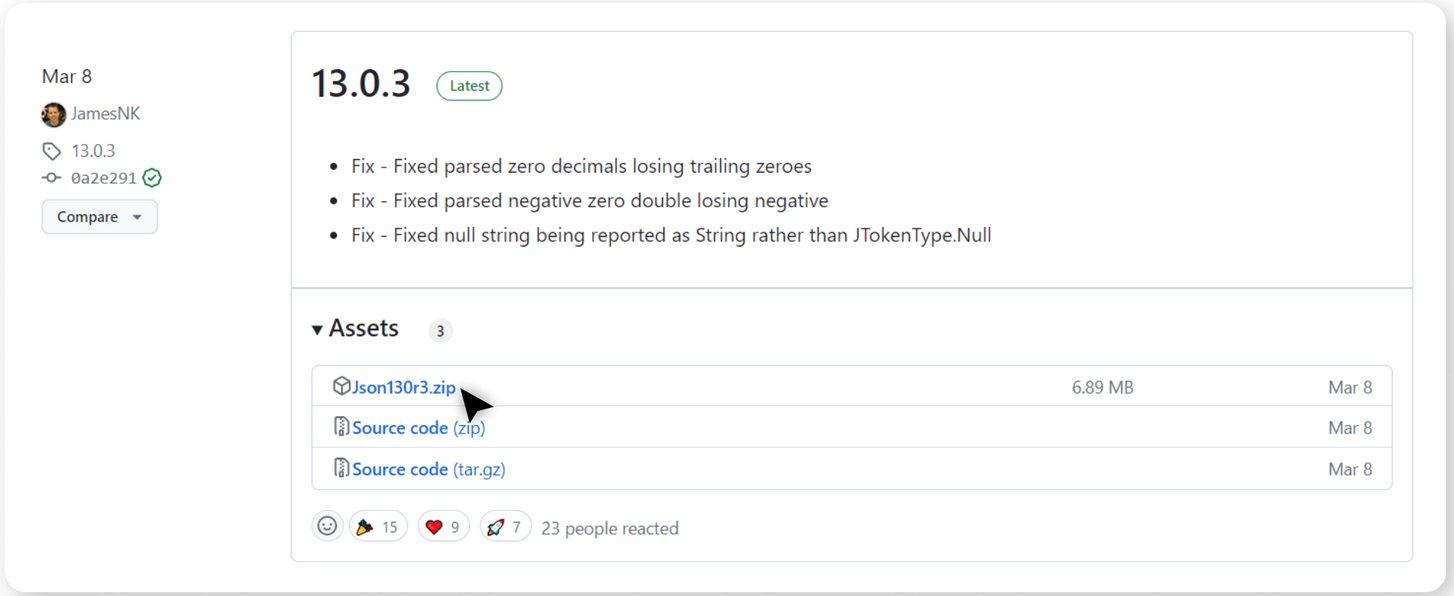




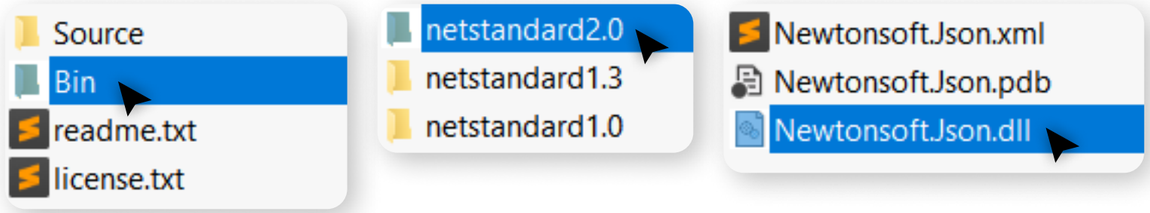
After installing Json.NET, you can continue using the asset.

Json.NET

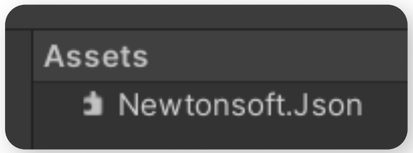
6 If the installation of Json.NET through the Package Manager was unsuccessful, download the latest release of Json.NET from the official repository: https://github.com/JamesNK/Newtonsoft.Json/releases



7 **Unzip the archive**, open the **Bin** folder, then **netstandard2.0**, and drag the **Newtonsoft.Json.dll** into the **Assets** folder in your project.

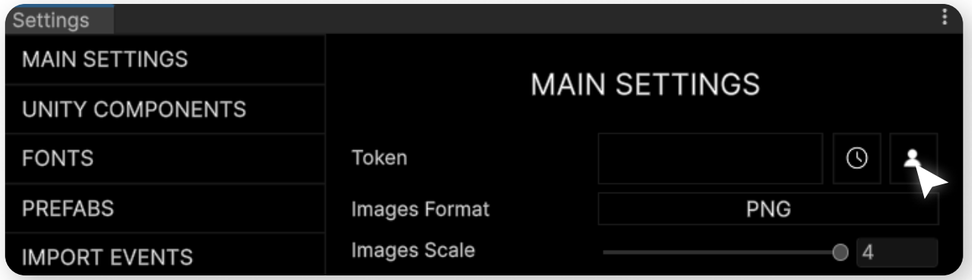


8 After installing Json.NET, you can continue using the asset.

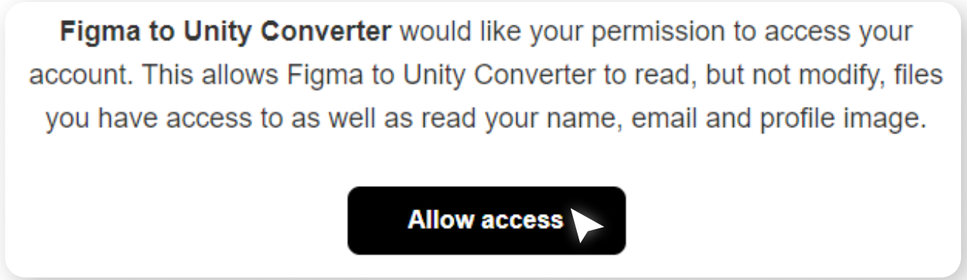


Auth

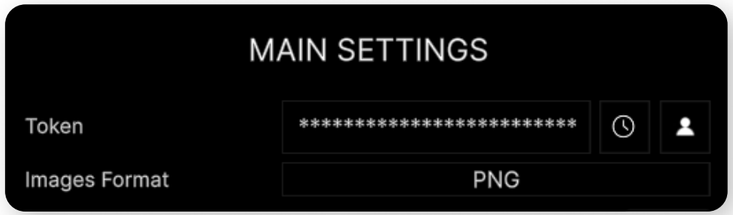
1. Now, you need to log in to your Figma account inside the asset.
To do this, **open** the asset's **settings**, then open “**MAIN SETTINGS**” tab and press "**Auth**" button.



2. In the browser that opens, click on the "**Allow access**" button.

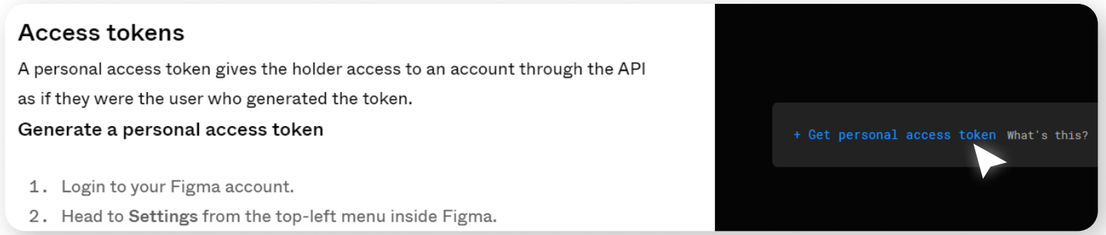


3. After that, under the logo you will see the name of your authorized account - this means that the authorization was successful, and now you can proceed with the import.

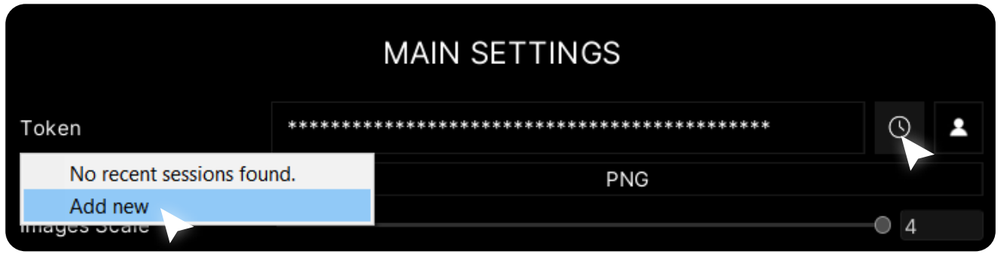


Auth

4 If for some reason you are unable to obtain the token using the asset, you can get it on the Figma website.
To do this, follow this link: https://www.figma.com/developers/api#access-tokens
Please check if you are logged in to this website with the Figma account that has access to the project you want to import.
To obtain the token, click on the "**Get personal access token**" button.



5 Then, copy the obtained value and paste it into the "**Token**" field, click on the "**Recent Sessions**" button, and then click on the "**Add New**" button.



6 After this, authentication will occur based on the entered token, and you will see a message in the console.

Auth

7 If you do not want to receive a token manually and when you try to receive a token using an asset, you see the error "**SocketException: An attempt was made to access a socket in a way forbidden by its access permissions**", you can use the solution suggested by one of the users of the asset.

The author of the asset has not tested this solution and **not responsible for the consequences of its use**.

Steps (for Windows):

1. Open CMD.exe as administrator and type “net stop winnat”, then press Enter;
2. Type “netsh int ipv4 add excludedportrange protocol=tcp startport=1923 numberofports=1“, then press Enter;
3. Type “net start winnat”, then press Enter;
4. Try auth in the asset again.

Google Fonts

Some fonts might be missing from the Google Fonts repository.
In that case, they won't be downloaded automatically, and you will see an error in the console.
You'll need to manually import those fonts into your project.

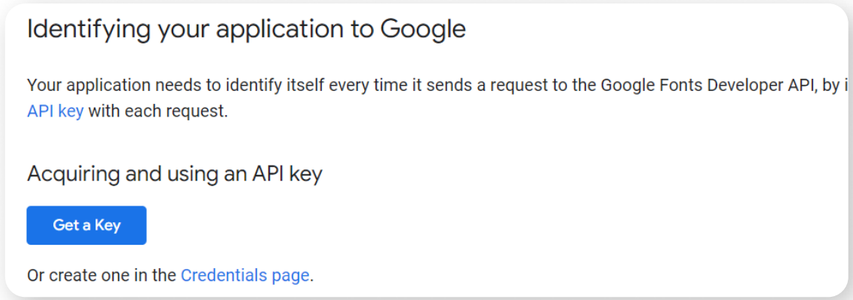
Fonts naming for latin subset fonts:
FontFamily Weight
Segoe UI Medium

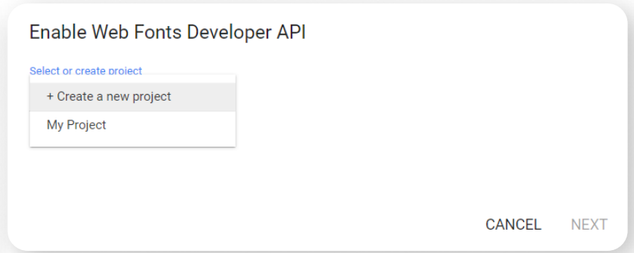
Fonts naming for other subset fonts:
FontFamily Weight Subset
Segoe UI Medium Greek

D.A. Assets

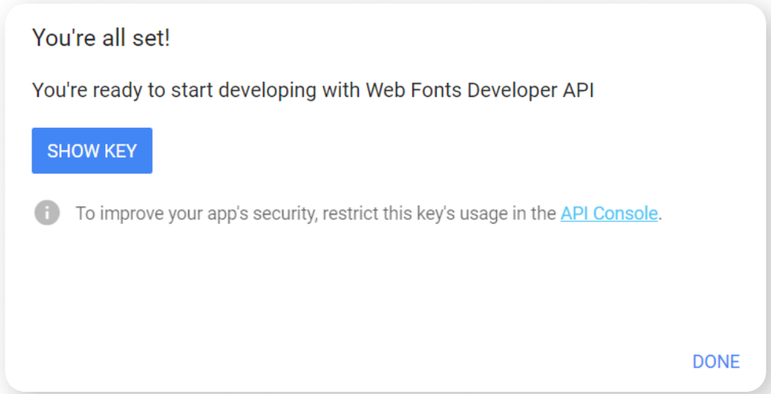
1. In order for the asset to automatically download missing fonts, you need to obtain a Google Fonts API key.
Go to: https://developers.google.com/fonts/docs/developer_api#identifying_your_application_to_google

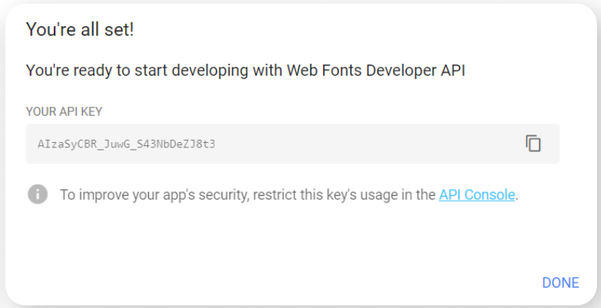
2. Click on “**Get a Key**” button, then create new project or select existing.





3. Click on “**SHOW KEY**” button, then copy your api key.



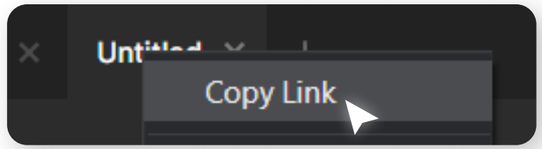


4. Open the "**FONTS**" tab and paste the obtained key into the "**Google Fonts Api Key**".
Now your fonts will be automatically downloaded from the Google Fonts repository.

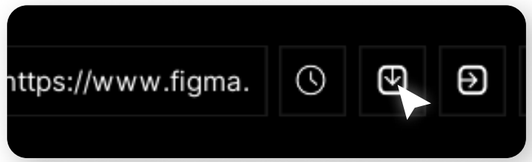
Import Frames

Before importing a layout, check **project permissions** for editing (see the **Teamwork** section in the **Manual for designers**).

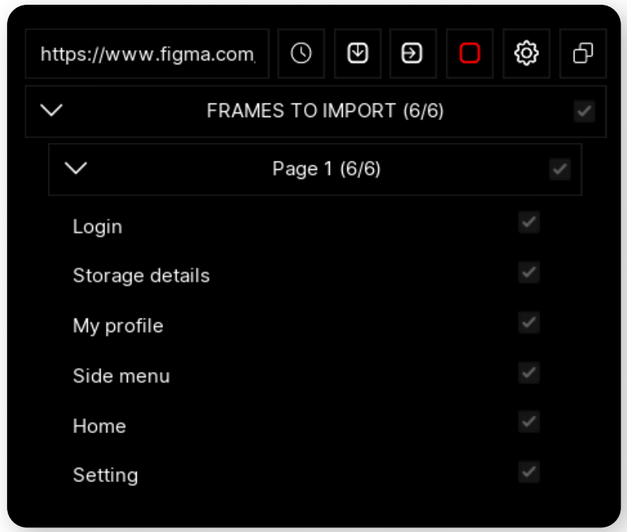
1. Open the figma project you are about to import and get a link to it. It can be obtained by **right-clicking on the tab** with an open project.
Example link: https://www.figma.com/file/XXXXXXXXXXXXXXXXX...



2. Press "**Download**" button to download your project and get a list of its pages and frames.



3. After the project has downloaded, you can select the pages and frames you want to import.

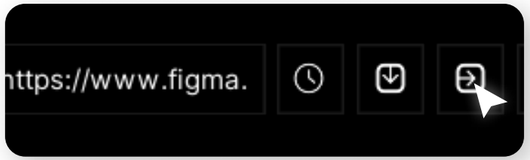


To have the components you want to import appear in the "FRAMES TO IMPORT" list, you must place them in a **Frame**.

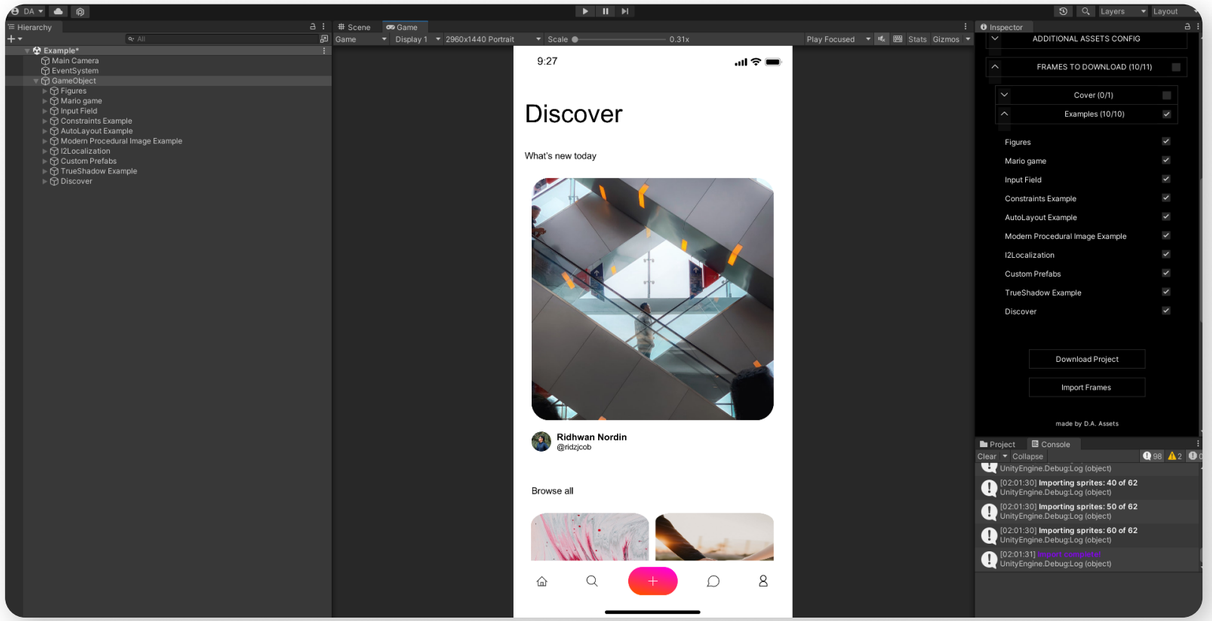
At the moment, the asset does not support importing **Sections** due to API limitations. To import the contents of **Sections**, place them in a **Frame**.

Import Frames

5 Press on the "**Import**" button, to start the import.

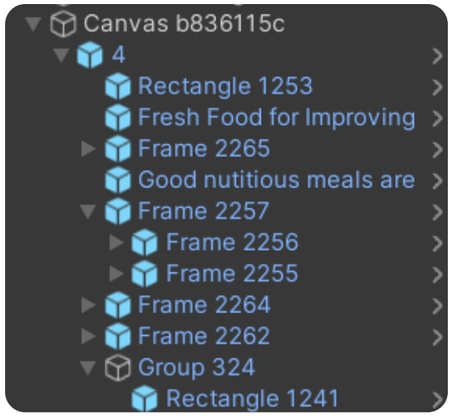


6 At the end of the import, you will see a message in the console - "**Import complete!**".



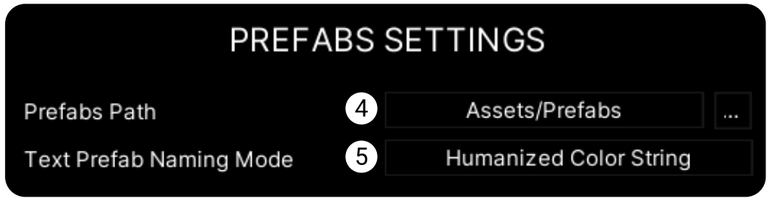
Creating prefabs

1. To create prefabs from the components, just right-click on your canvas where the FigmaConverterUnity.cs script is added, then click “**Tools/Figma Converter for Unity/Create Prefabs**” item.



2. After you've created prefabs, when re-importing the project **without removing** the old components from the scene, you'll **only** be able to synchronize the **transform** of existing objects. **New objects inside prefabs won't be created.** Information about updating existing objects can be found in "**Layout updating**" section.

3. In the "**PREFABS**" tab of the asset settings, you can find additional settings for prefab creation.



4. Folder where prefabs will be saved when creating prefabs using the asset.
You can set your own folder by clicking the button with three dots.

Creating prefabs

5 Naming type for text prefabs. Modes:

- **Humanized Color String** - The name of the prefab includes the name of the most suitable color, which is determined automatically.
  Example of a name: "TextMeshPro white 12px".
- **Humanized Color HEX** - The color is indicated in HEX format in the prefab name.
  Example of a name: "TextMeshPro #0C8CE9 12px".
- **Figma** - The text is named the same as its component in Figma.

Import Issues

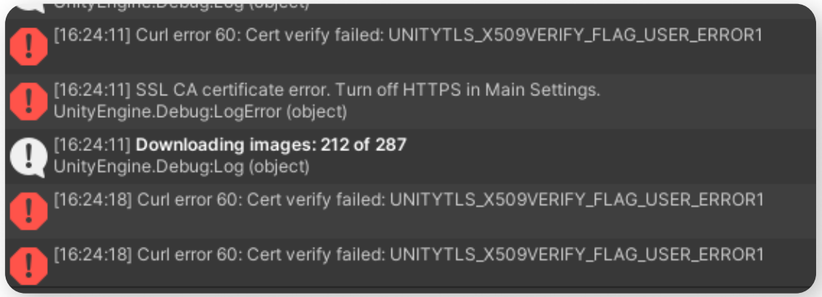
This section will be updated.

1. My frame doesn't look the same after import as it does in Figma's layout. Why?
The answer to this question can be found in the “**Layout Rules**” section of the **Manual for designers**.

2. My components merged into a **single image**, and I want to separate them.
My components consist of **several images**, and I want to combine them into one.
You will find the solution to this problem in the “**Naming and tags**” section of the **Manual for designers**.

3. "**Either this file doesn't exist or you don't have permission to view it. Ask the file owner to verify the link and/or update permissions**".
If you see this error, you need to read section "**Teamwork**" in the "**Manual for designers.pdf**".
"**Teamwork**" section of the designer guide will **help you if all the images** in the imported frame **are missing**.

4. If you see these errors, you may have reached your API request limit.

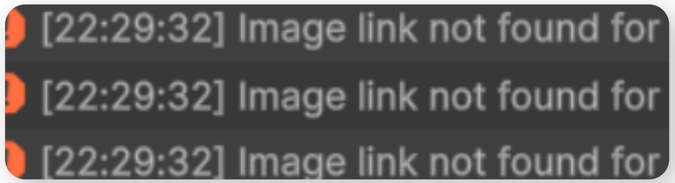


You can reach the limit on Figma API requests, which will prevent you from importing your frames for a while.
To avoid this, follow these guidelines:
• Don't import **more than 100** frames at a time;
• **Don’t store** design system components in your project if there are a lot of them;

If you've **reached** the **limit,** you'll **need to wait** a while to be able to import frames again, or create a new project.
These are **not requirements**, **but recommendations** that **based on personal experience.**

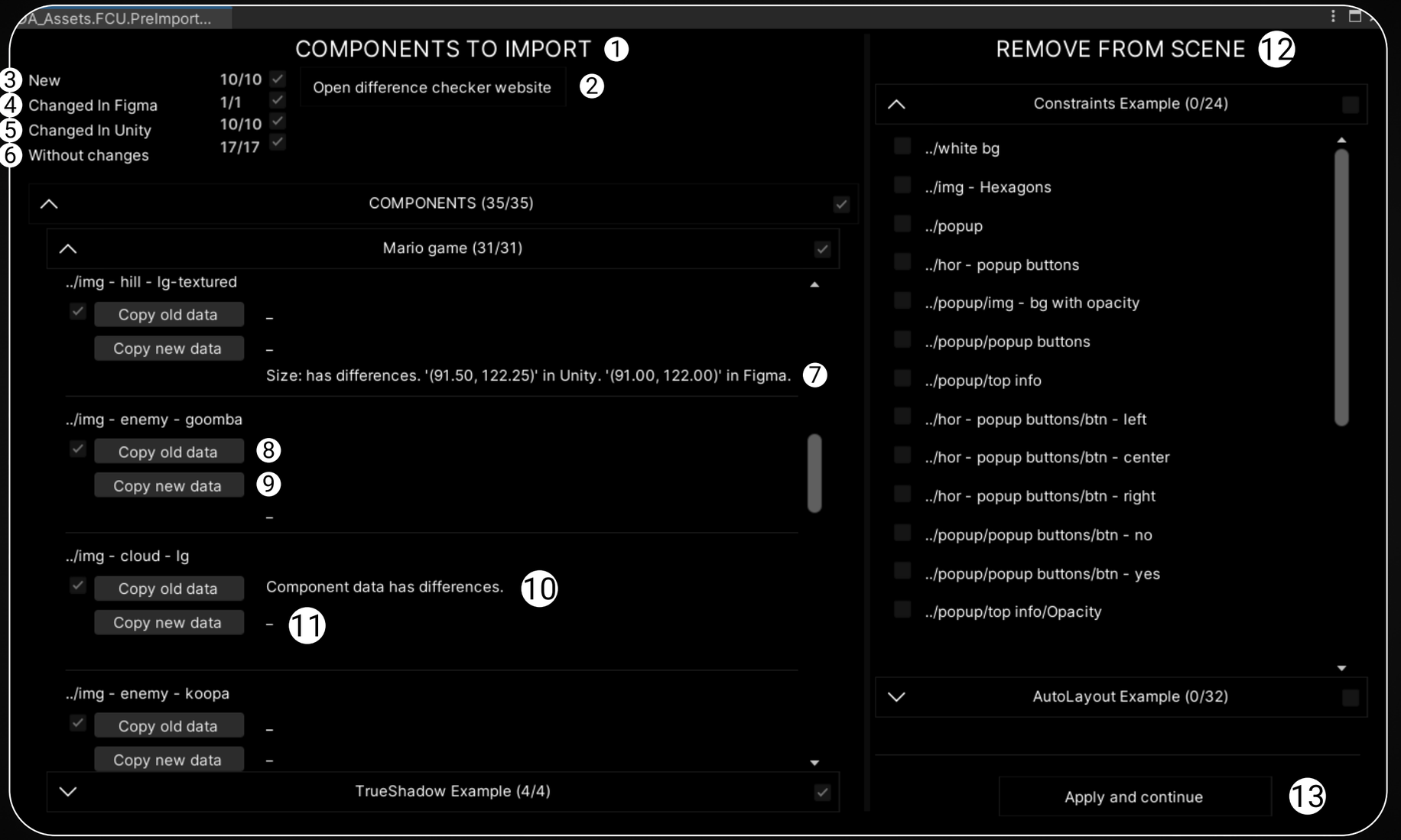
Import Issues

5 If you see this error, it's possible that your component is too large in pixels, and you need to reduce the Image Scale in the Main Settings of the asset.
If this does not help, please contact the asset developer.



You have read the basic manual

Below is extended information about the asset's capabilities, function descriptions, usage nuances, and solutions to potential issues.


A_Assets.FCU.PreImport...
COMPONENTS TO IMPORT 1
3 New 10/10
4 Changed In Figma 1/1
5 Changed In Unity 10/10
6 Without changes 17/17
Open difference checker website 2
COMPONENTS (35/35)
Mario game (31/31)
../img - hill - lg-textured
Copy old data
Copy new data
Size: has differences. '(91.50, 122.25)' in Unity. '(91.00, 122.00)' in Figma. 7
../img - enemy - goomba
Copy old data 8
Copy new data 9
../img - cloud - lg
Copy old data
Component data has differences. 10
Copy new data 11
../img - enemy - koopa
Copy old data
Copy new data
TrueShadow Example (4/4)
REMOVE FROM SCENE 12
Constraints Example (0/24)
../white bg
../img - Hexagons
../popup
../hor - popup buttons
../popup/img - bg with opacity
../popup/popup buttons
../popup/top info
../hor - popup buttons/btn - left
../hor - popup buttons/btn - center
../hor - popup buttons/btn - right
../popup/popup buttons/btn - no
../popup/popup buttons/btn - yes
../popup/top info/Opacity
AutoLayout Example (0/32)
Apply and continue 13

Layout updating

If **previously imported components** are present on your scene, attempting a new import will open the **PreImportWindow**. With its help, you can more precisely adjust your new import, specifically - **update** the **existing** components on the scene (**synchronize** them **with Figma**), see how the imported components differ from those in Figma and **compare** their **properties**, as well as **remove** unwanted components **from the scene**.

Below you will find a description of the PreImportWindow interface elements with an explanation of its functionality.

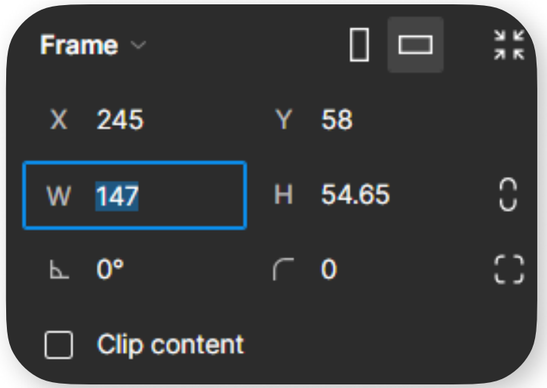
1. The section where you can analyze and configure the import of components to the scene.
2. Button to open the "**Diffchecker**" website.
3. Components that exist in the Figma project but not on the Unity scene (new components).
4. Components that have been changed in Figma since the last import.
5. Components that have been changed in Unity since the last import.
6. Components that have not changed.
7. The RectTransform size in Unity differs from the component size in Figma.
8. Lists of all properties of a Figma component.
   Old data - data of the component currently on the scene.
   New data - data of the component that is only in Figma and has not yet been imported onto the scene.

   The **data about the properties** of the Figma component is **captured at the time of import** and remains unaffected by any changes to the GameObject on the scene.
   In some cases, you may need this information, for example, made manipulations with the component in Figma that

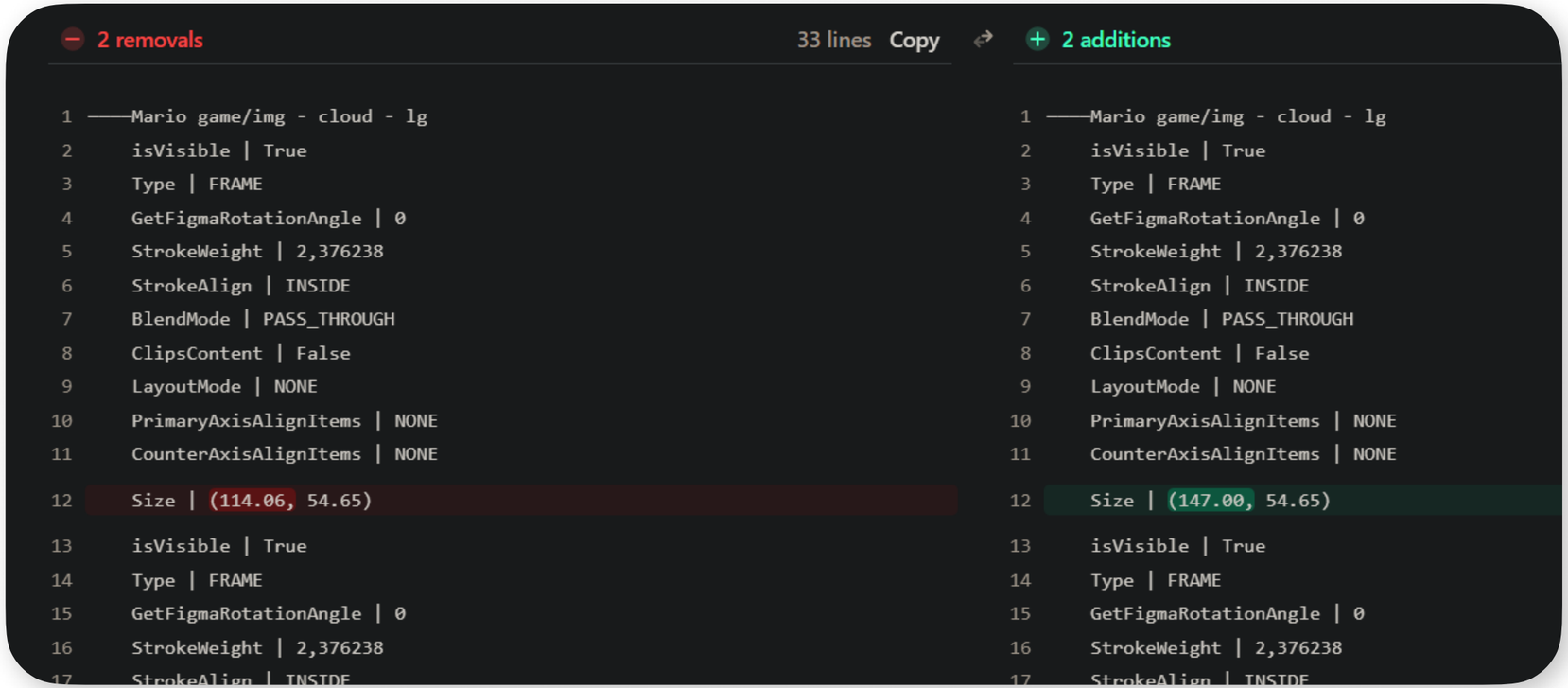
Layout updating

was imported into Unity as a sprite, and you need to decide whether to update the sprite during re-import, or not.

If this data differs, you will see the text "**Component data has differences**" in the component item.
To see how the new component differs from the existing one, you can open the website "**Diffchecker**" using the "**Open difference checker website**" button, and sequentially copy the **Old** and **New** data into the "**Original text**" and "**Changed text**" fields, then press the "**Find difference**" button.



In the example below, I changed the size of the component in Figma but did not change the RectTransform in Unity.



Layout updating

10 A message about data differences, related to points 7 and 8.

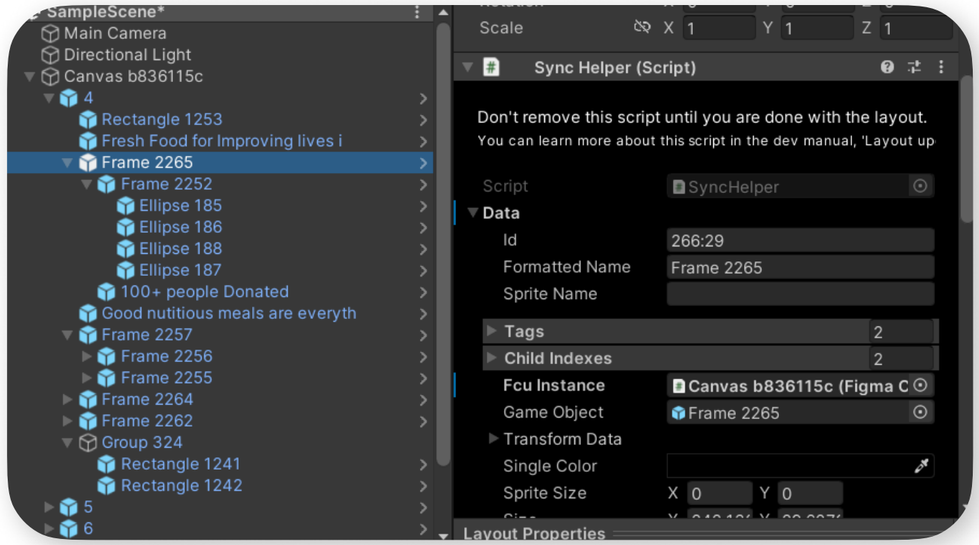
11 The **Color** property of **Graphic** component in Unity differs from the component color in Figma.

12 In this section, you can select frames or individual components that will be removed from the scene during the new import.

13 Click this button to continue the import with the parameters you have selected.
By default, if you have not made any changes in PreImportWindow - all components that are both on the scene and in Figma are synchronized, new components are imported, and components from the old import are not deleted.

Layout updating

1. After importing the project, a script "**SyncHelper.cs**" will be added to all imported objects.
This script is needed to synchronize objects between Figma and Unity during the import.



2. You can perform a re-import of your project to update the transforms and properties of objects (color, text), as well as add new objects and remove missing ones.

3. If you've created prefabs from your imported objects, during subsequent imports, you'll be able to update only their transforms and properties. New objects within the prefabs won't be added.

4. Avoid creating duplicates of objects with the same "**SyncHelper.cs**" script in your Unity scene.
To duplicate an object, create its duplicate in Figma, and then repeat the import process.

Layout updating

5. Do not remove the "**SyncHelper.cs**" scripts until you have finished importing your project from Figma to Unity.

6. After you've completed your work on the project and are sure you won't be updating it further, you can remove the "**SyncHelper.cs**" scripts from your objects using the corresponding function in the asset's context menu (more details in the "**Context Menu**" section).

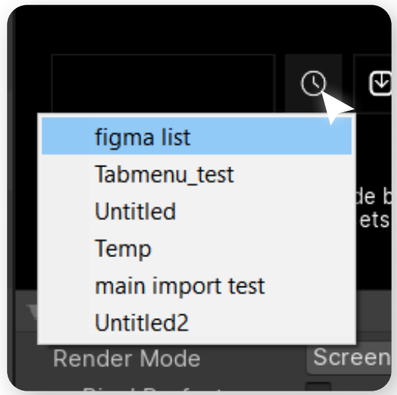
Scene backups and project cache

1. The folder with backups of your active scene is located here: **Library\Backup\Scene**

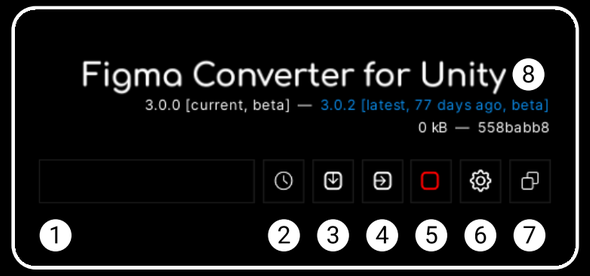
2. Backups are automatically creates before each import and before creating prefabs.

3. A backup is created for a **previously saved local scene file**. If you see a asterisk (*) next to the project name in the Unity interface, it indicates that changes you made to the scene without saving it will not be included in the backup.

4. With each project download, the transform and properties of objects from your Figma project are cached.
To avoid downloading it again, you can choose the cached version from the dropdown menu.



Asset UI



1. Link to your project in figma.

2. Open the list of cached projects that you have previously imported.
   From the list, **you can select** the **project** you want to import.

3. Download the project from the link.
   All downloaded projects are automatically cached.

4. Import selected frames from the downloaded project.

5. Stop import.

6. Open asset settings.

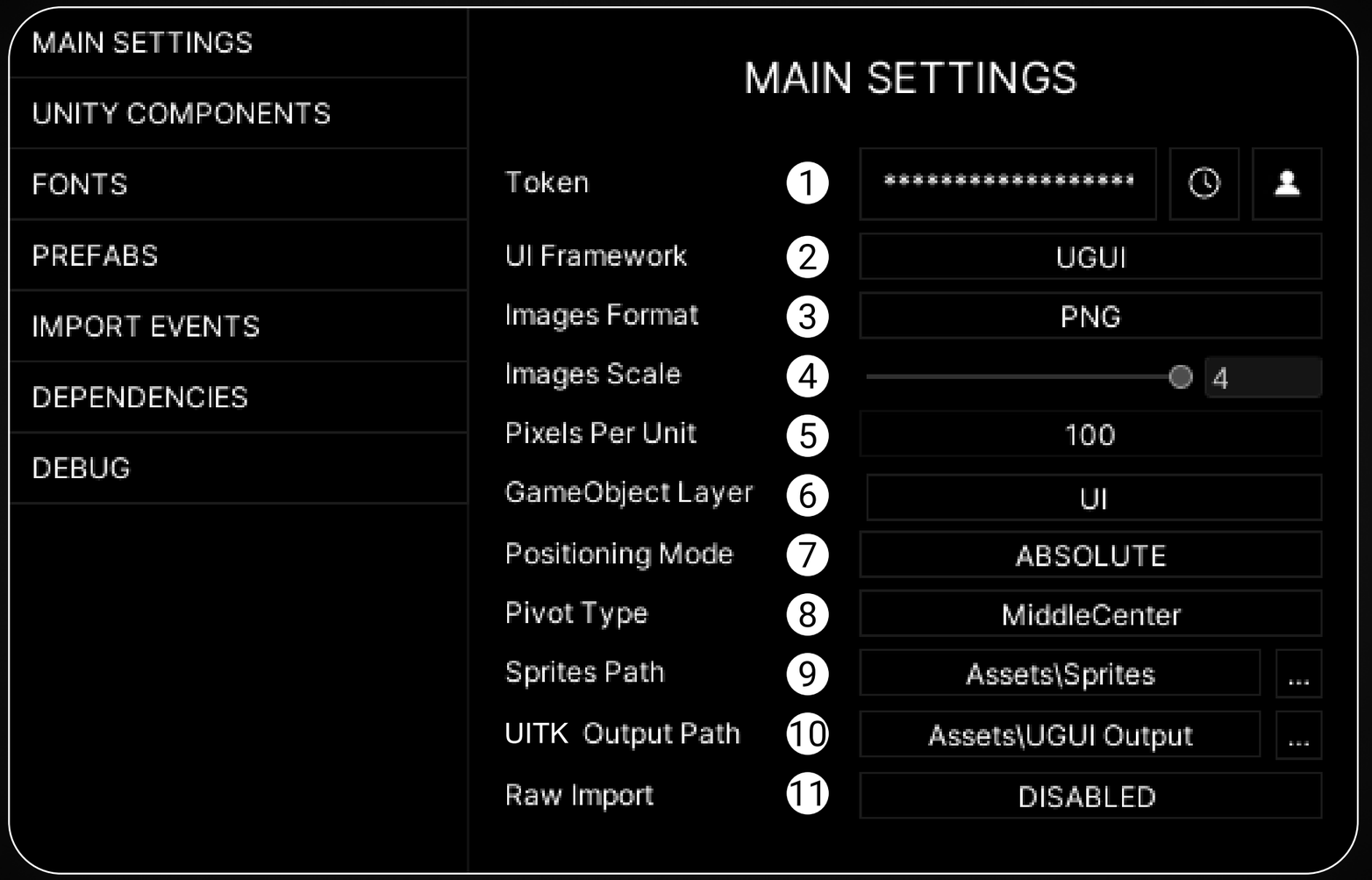
7. Switch the asset display mode. Available modes:
   - In the inspector
   - Windowed

8. Label displaying the current and latest version of the asset.

   If the version is colored blue, it means that too much time has passed since the release of the latest version, and you are recommended to update the asset.

   If the version is colored red, it indicates that it contains errors, and it is strongly recommended to update the asset.

   If you hover your cursor over the version, you will see a tooltip with detailed information about that version.


MAIN SETTINGS
UNITY COMPONENTS
FONTS
PREFABS
IMPORT EVENTS
DEPENDENCIES
DEBUG
MAIN SETTINGS
Token
1
Ul Framework
2
UGUI
Images Format
3
PNG
Images Scale
4
4
Pixels Per Unit
5
100
GameObject Layer
6
UI
Positioning Mode
7
ABSOLUTE
Pivot Type
8
MiddleCenter
Sprites Path
9
Assets\Sprites
...
UITK Output Path
10
Assets\UGUI Output
...
Raw Import
11
DISABLED

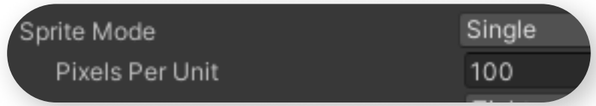
Main Settings Tab

1. The **token** of your account, which is **necessary for downloading** content from Figma.
You can select a previously saved session (button with a clock), or re-authenticate (button with a person).

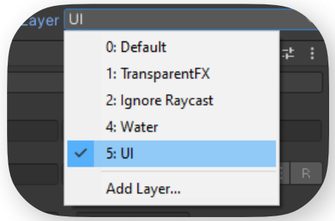
2. **UGUI** - layout import into **Canvas**.
UITK - layout import into **UI Builder**.

3. The format of the downloaded images. Can be **PNG** or **JPG**.

4. The scale of the downloaded images.
This option is identical to the same option when exporting image from Figma.

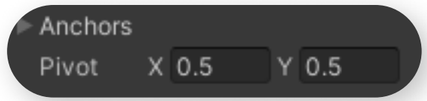
5. 

The value that will be assigned to all imported sprites.
When using the “**SpriteRenderer”** component, this value is ignored, and instead, the “**Image Scale**“ value is used for sprites.

6. 

Sets the Layer value for all imported GameObjects.

7. **ABSOLUTE** - positioning of frames on the canvas as in Figma.
GAMEVIEW - anchoring frames to the edges of GameView (does not work in UITK mode).

8. 

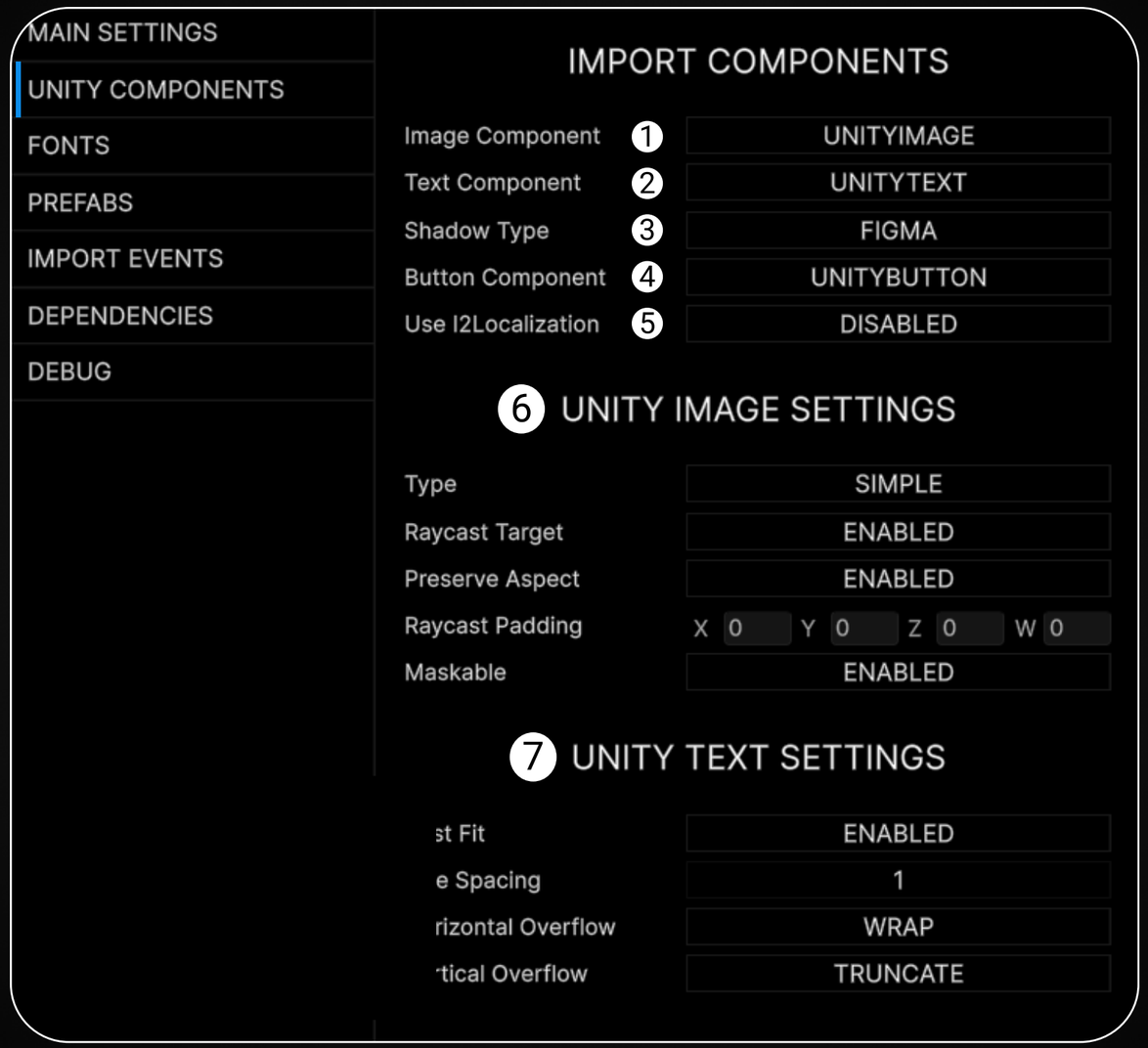
The value that will be assigned to all imported GameObjects.

9. The **path** to the folder where the **sprites** are downloaded.
You can **set your own** path by clicking the button with **three dots**.

10. The folder where the result of the import into **UITK** will be saved.

Main Settings Tab

11 If enabled, your project is imported "as is", i.e., without "smart" merging of individual vectors into single sprites. The function is in beta stage, and errors may occur during its using.



1. The value that will be assigned to all imported sprites. When using the **SpriteRenderer** component, this value is ignored, and ImageScale is used for sprites instead.

2. The component that will be drawn on the scene when importing **texts** from Figma.
   Available components:
   - **UNITY TEXT** (built-in)
   - **TEXTMESHPRO** (need **TextMeshPro** asset)

3. The asset supports two methods of importing shadow components:
   - **FIGMA**: The shadow is part of the downloaded sprite.
   - **TRUESHADOW**: The shadow is **procedurally rendered** using the asset.

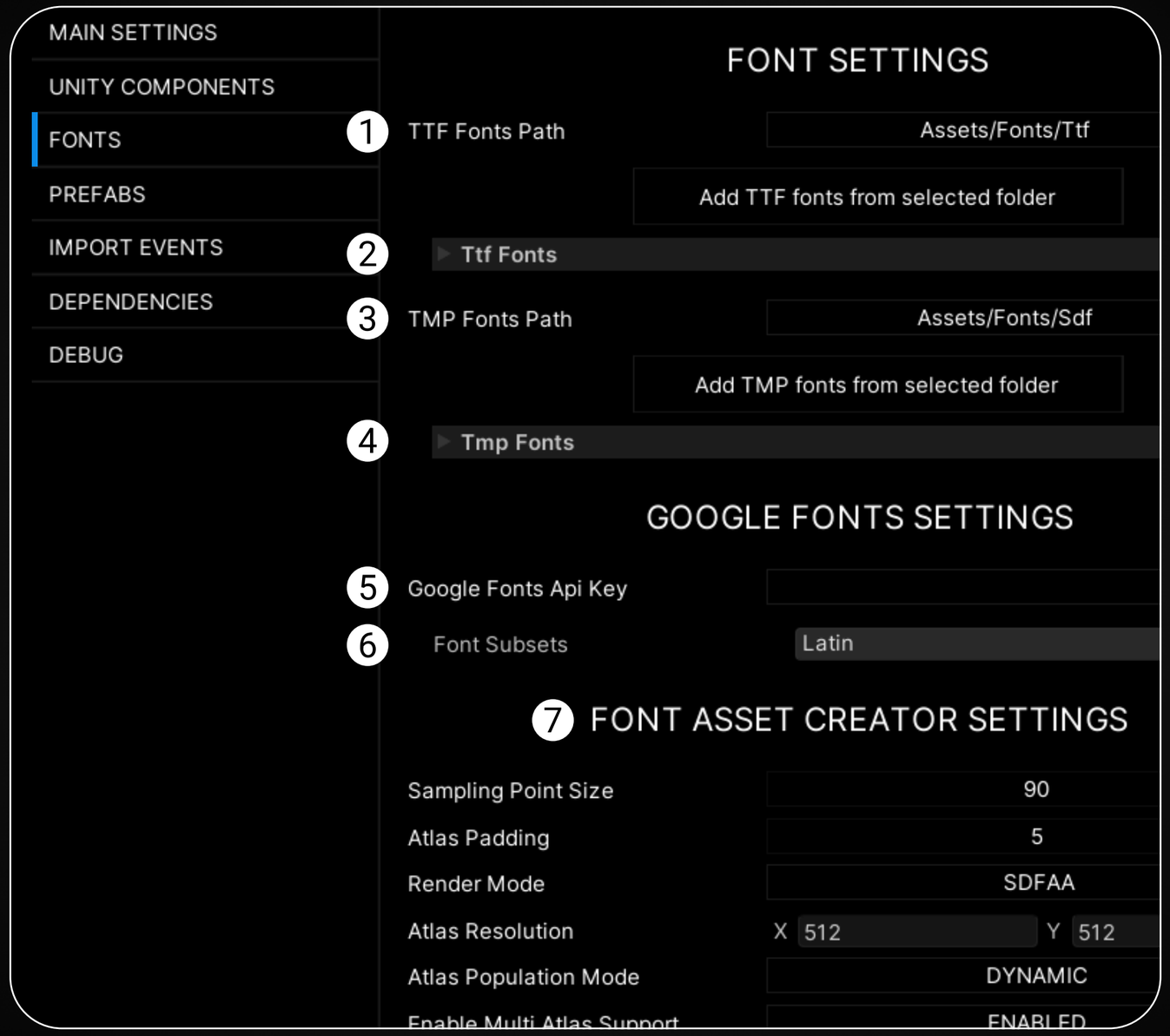
4. The component that will be drawn on the scene when importing components with the "**btn**" tag from Figma.
   Available components:
   - **UNITY BUTTON** (built-in)
   - **DABUTTON** (replacement of the legacy **FCU_Button** script, will be available in future versions)

Unity Components Tab

5. This function automatically **adds** a **localization script** from the "**I2Localization**" asset and **generates** a **key** for it **based on** the **name** of the text component in Figma. After that, it **saves** the current **text** and its key as English localization in an automatically generated localization file.
This function only **works if you have** the "**I2Localization**" asset installed.
You can find **more information** about this on the **"Localization" slide.**

6. The settings duplicate the settings of the **UnityEngine.UI.Image** component (or another text rendering asset you've chosen) and **apply to all** the **image components** created during project import.
You can find a description of these settings in the official Unity documentation:
https://docs.unity3d.com/Packages/com.unity.ugui@2.0/manual/script-Image.html

7. Same as in point 6, but for text components.
Unity documentation:
https://docs.unity3d.com/Packages/com.unity.ugui@2.0/manual/script-Text.html


MAIN SETTINGS
UNITY COMPONENTS
FONTS
PREFABS
IMPORT EVENTS
DEPENDENCIES
DEBUG
FONT SETTINGS
1
TTF Fonts Path
Assets/Fonts/Ttf
Add TTF fonts from selected folder
2
Ttf Fonts
3
TMP Fonts Path
Assets/Fonts/Sdf
Add TMP fonts from selected folder
4
Tmp Fonts
GOOGLE FONTS SETTINGS
5
Google Fonts Api Key
6
Font Subsets
Latin
7
FONT ASSET CREATOR SETTINGS
Sampling Point Size
90
Atlas Padding
5
Render Mode
SDFAA
Atlas Resolution
X
512
Y
512
Atlas Population Mode
DYNAMIC
Enable Multi Atlas Support
ENABLED

Fonts Tab

1. Folder containing your **TTF** fonts. Additionally, any missing **TTF** fonts for the project will be downloaded into this folder (provided that you've set up the **Google Fonts API Key**, more information below).
You can set your own folder by clicking the button with three dots.

2. Serialized array with your **TTF** fonts. Add your fonts to it so that they are used when importing the Figma project.
You can manually add your fonts or use the "**Add TTF fonts**" button which, when pressed, will add fonts from the folder specified in the "**TTF Fonts Path**" field to the array.

3. Same as in **point 1**, but for **TextMeshPro** fonts.
This setting is **visible** only **if** you have **imported** the **TextMeshPro** asset.

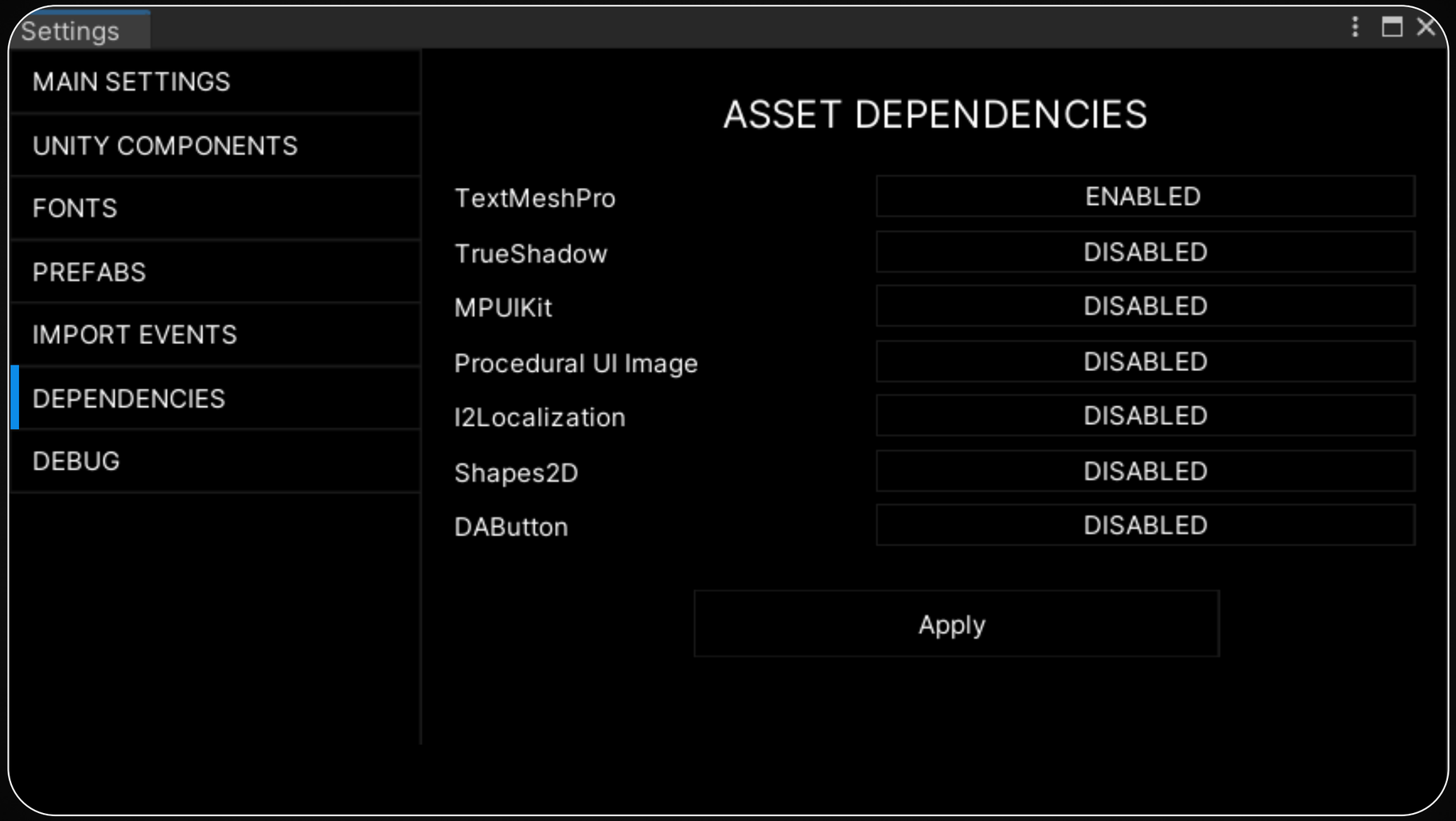
4. Same as in **point 2**, but for **TextMeshPro** fonts.
This setting is **visible** only **if** you have **imported** the **TextMeshPro** asset.

5. API key **required for downloading missing fonts** from the Google repository. More details about this can be found in the "**Google Fonts**" section.

6. If necessary, you can select **multiple language groups** if the "Latin" group doesn't contain the glyphs you need.
When selecting multiple groups, multiple fonts of the same family containing different glyph tables will be downloaded.
You can utilize these tables within a **single font** by using the **TextMeshPro** asset (more details below).

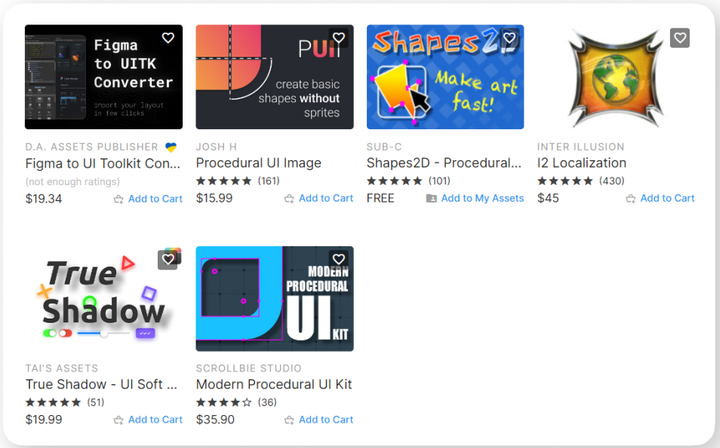
7. Settings same to the settings of the **Font Asset Creator** utility.
These settings are **used when** automatically **generating missing TextMeshPro fonts.**
You can find the description of these settings for your Unity version in the **official documentation:**
https://docs.unity3d.com/Packages/com.unity.textmeshpro@3.2/manual/FontAssetsCreator.html

8. Button for **downloading** and generating **missing fonts without importing** frames.
To use the button, you **need** to **select** a text **component** in the “**UNITY COMPONENTS**” tab, and then download the project using the download button (the import button does not need to be pressed).


Settings
MAIN SETTINGS
UNITY COMPONENTS
FONTS
PREFABS
IMPORT EVENTS
DEPENDENCIES
DEBUG
ASSET DEPENDENCIES
TextMeshPro
ENABLED
TrueShadow
DISABLED
MPUIKit
DISABLED
Procedural UI Image
DISABLED
I2Localization
DISABLED
Shapes2D
DISABLED
DAButton
DISABLED
Apply

Dependencies

1. By using additional assets, you can:
   - **avoid blurring** fonts with "**TextMeshPro**";
   - **reduce** application **weight** with one of the **procedural image** assets;
   - ease the **localization** process with "**I2Localization**".



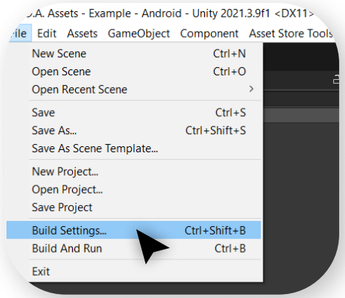
2. **Before** these assets can be **activated**, it's **must** be **imported** into the project.

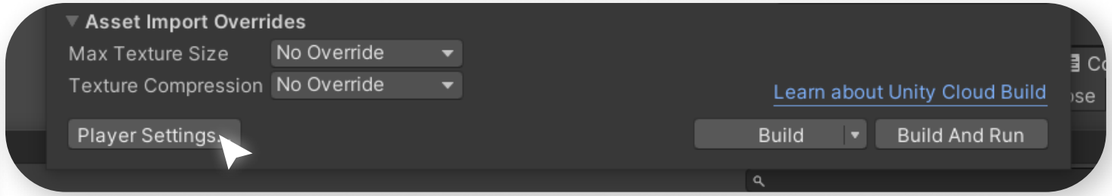
3. Assets are automatically activated after being imported into the project, but if this didn't happen, use the manual switching according to the following scheme:
   switch to "**ENABLED**" if asset is imported.
   switch to "**DISABLED**" if the asset is not imported.

4. **After removing** an additional asset from the project, **disable it manually.**

5. If, for some reason, you **cannot activate or deactivate** an additional asset and/or see errors in your **Editor**, you can activate or deactivate assets using "**Project Settings**":

6. Go to **File > Build Settings > Player Settings**





Dependencies

7. Find "**Script Compilation**" section in the "**Project Settings**".

8. Add or remove row from the list that correspond to additional asset:

Asset	Row
TextMeshPro	TextMeshPro
TrueShadow	TRUESHADOW_EXISTS
I2Localization	I2LOC_EXISTS
Shapes2D	SHAPES_EXISTS
Modern Procedural UI Kit	MPUIKIT_EXISTS
Procedural UI Image	PUI_EXISTS
Figma to UITK Converter	UITKPLUGIN_EXISTS
D.A. Button	DABUTTON_EXISTS

9. Press "**Apply**" button.

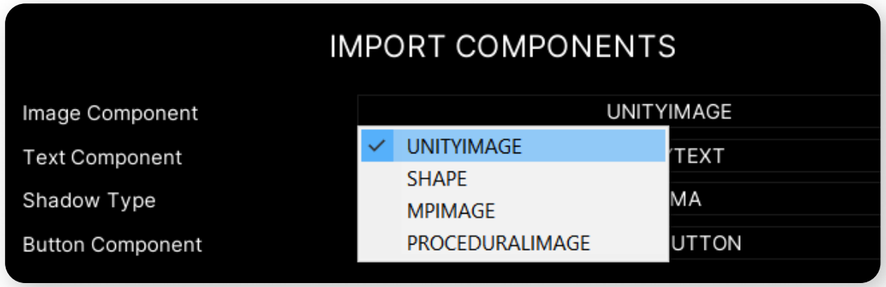
10. After the scripts in your project are recompiled, you can continue to use the asset.

Shapes2D, MPUIKit, Procedural UI Image

1. Instead of the standard Unity image component, you can use one of the additional assets, the links to which you can find in the "**Package dependencies**" section on the asset's page in the Asset Store: https://assetstore.unity.com/packages/tools/utilities/198134

2. When importing assets, follow the instructions in the "**Dependencies**" section of this manual.

3. After successfully importing the asset, you will be able to select it as the image component in the "**UNITY COMPONENTS**" tab in asset.



4. For design-specific details about working with these assets, you can read the "**Manual for designers**": *Assets\D.A. Assets\Figma Converter for Unity\Manual for designers.pdf*

5. If you are using the function to create prefabs using the asset, please note that updating prefabs upon re-import will not work if the image components inside the prefabs are different from the component selected in the "**UNITY COMPONENTS**" tab at the time of import.

True Shadow

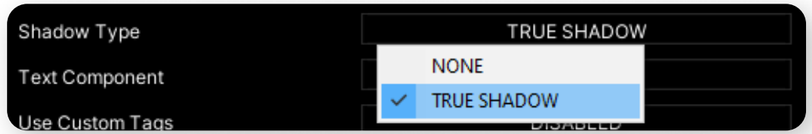
Video manual:

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

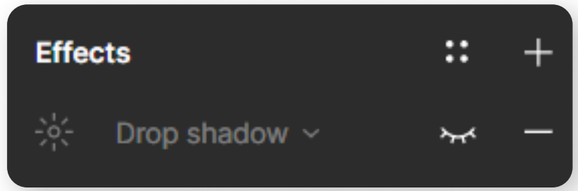
1. If you want the component's shadow not to be part of your sprite but rendered procedurally, you can use the TrueShadow asset.

2. When importing asset, follow the instructions in the "**DEPENDENCIES**" section of this manual.

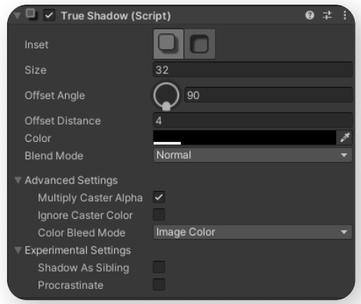
3. In the "**UNITY COMPONENTS**" tab switch parameter "**Shadow Type**" to **“True Shadow**".



4. Before importing your layout using the **“True Shadow”** mode, you need to make all the shadows in your Figma project invisible.



5. After import, **all** your **components** that have a **shadow** in the Figma layout **will have** a **shadow** script from "**TrueShadow**" asset.
To use this functionality properly, **read** the section of this asset in the "**Manual for designers**".



D.A. Assets

I2Localization

You can purchase "**I2Localization**" asset and use it in conjunction with "**Figma Converter for Unity**".

Video manual:
https://www.youtube.com/watch?v=Rn_Fv-oory8

D.A. Assets

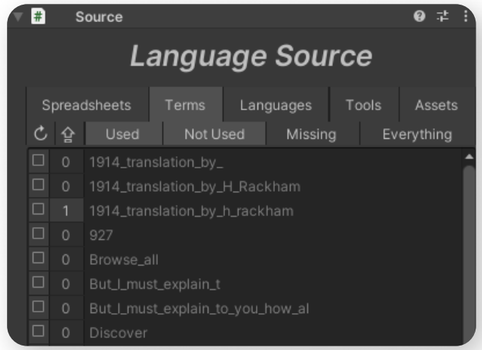
1. Add the purchased asset to your project.

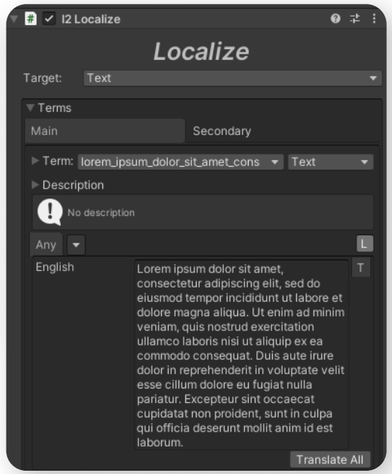
2. When importing assets, follow the instructions in the "**Dependencies**" section of this manual.

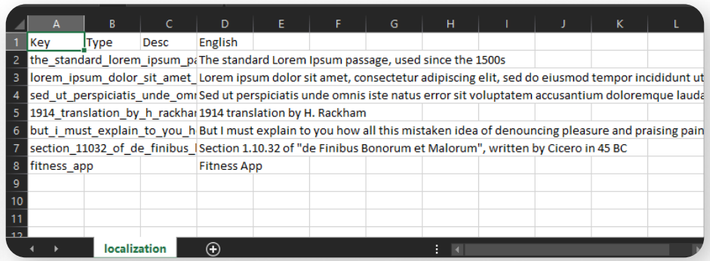
3. After a successful asset import, in the "**MAIN SETTINGS**" tab, switch the "**Use I2Localization**" parameter to "**ENABLED**".

4. Import your layout as you normally would.

5. **After import**, script "**I2Localize**" will be **added to all text components**, their text will be written to the localization file "**localization.csv**" (**separator** - **semicolon**) as an **english** localization, the localization corresponding to the text component will be **selected** in the script.
You can open the localization file in **Excel**.
All further **instructions** are detailed in the **manual** for "**I2Localization**" asset.







Buttons

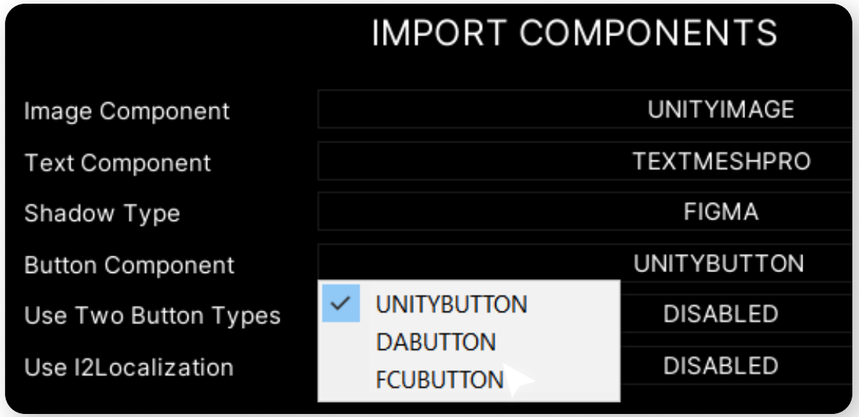
1. For importing buttons from Figma, you can use one of several components of your choice.
All components implement the ability to transfer color from Figma for different button states using tags:

   - **Unity Button** - built-in component.

   - **D.A. Button** - supports **multiple TargetGraphics**, **color**/**size**/**position** animations for various states via AnimationCurve. Supports **sprite swapping** and **looped** animations.
   Information on using **D.A. Button** can be found in the manual attached to **D.A. Button** asset.
   Sold separately.

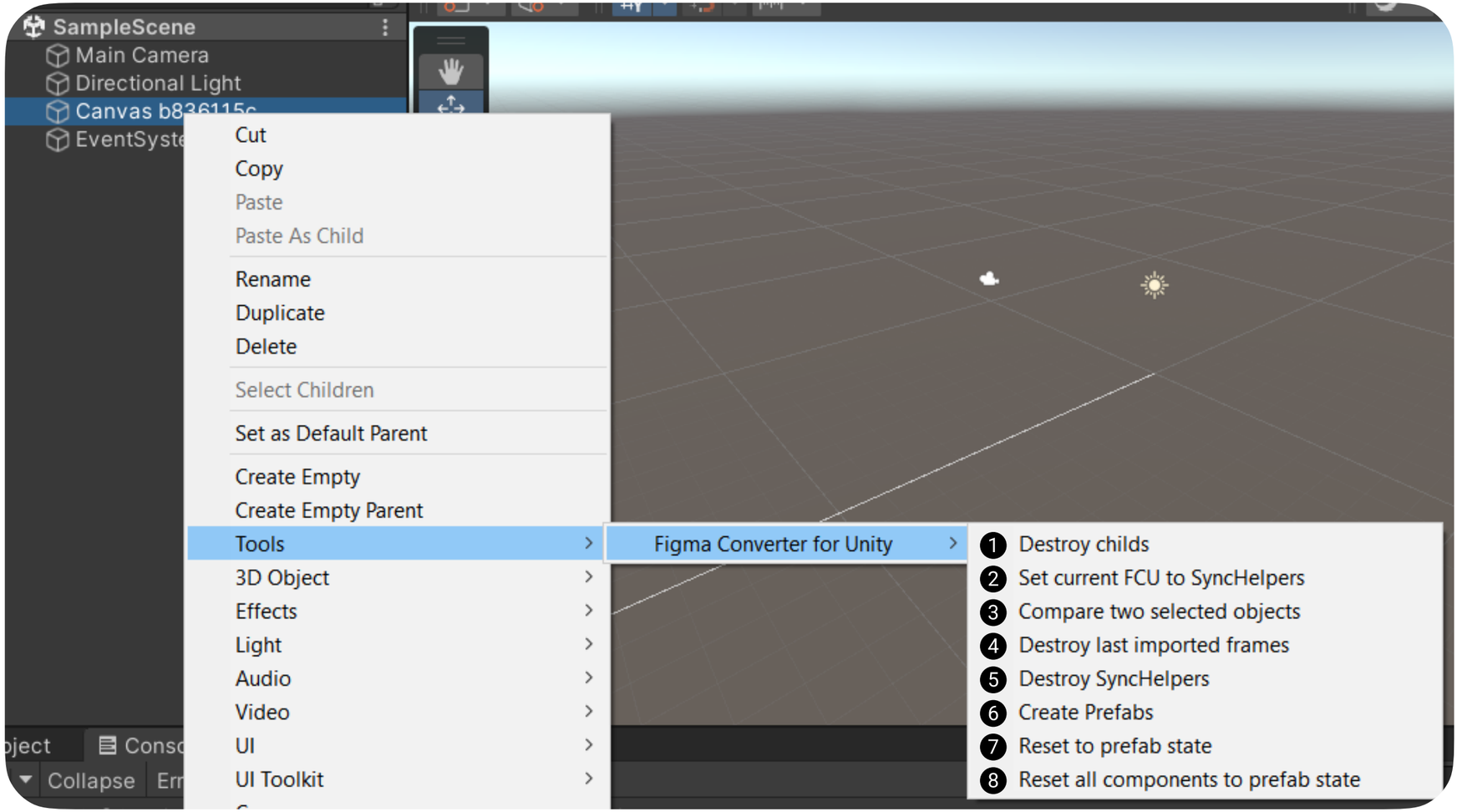
   - **FcuButton** - a copy of the Unity Button with the ability to change the color not only of the main TargetGraphic but also of the text. Already included in the asset.

2. To select the button component you need, switch the “**Button Component**” in the “**IMPORT COMPONENTS**” tab in the asset settings.



3. Information on setting up button component in Figma can be found in the **Manual for Designers** in the relevant section.

Context menu



1. Deletes the child objects of the current canvas.

2. Assigns the main script of the asset to the serialized field of child objects of the current canvas. This is necessary for creating prefabs and updating the project during re-import.
   Works only for objects that have the SyncHelper.cs script attached to them.

Context menu

3. Compares two objects that have the SyncHelper script attached to them. Using this function, you can determine the differences between two objects to avoid duplication in your Figma and Unity projects.

4. Removes the last imported frames. Please note that this function is temporarily not operational.

5. Removes the SyncHelper.cs script from all child objects of the current canvas. Please note that after removing these scripts from objects, you won't be able to synchronize your Unity project with the Figma project. Only delete SyncHelper.cs if you are certain that you won't need to synchronize your layout anymore.

6. Creates prefabs from the objects of the current canvas. Creating prefabs is only possible if all objects on the canvas have the SyncHelper.cs script attached.

7. Resets the selected GameObject to the state of the prefab. Child objects are not reset.
Resets the selected object and all its child objects to the state of the prefab. The SyncHelper.cs script is not needed for these functions to work.

8. Resets the selected GameObject and all its child GameObjects to the state of the prefab. The SyncHelper.cs script is not needed for this function to work.

Context menu

1. Creates a GameObject with the FigmaConverterUnity script on the scene.

2. Opens a window where you can specify a folder containing your sprites, from which you want to remove all sprites that are not used in the Image components of all objects in the current open scene.