xDoc

Manual

Installation, Setup, User and Script Manual

Version 1.09-Letter

Thank you for purchasing xDoc!

Your support allows us to update and develop great tools for Unity.

VERSION HISTORY

Version	Comment	Author	Date
1.09	Initial version	Dr. S. Issever	14. May 2016

TABLE OF CONTENTS

1 HELPFUL LINKS

1.1 @ XOX INTERACTIVE

- xox interactive
  http://www.xoxinteractive.com
- xDoc product page
  http://xoxinteractive.com/utilities/xdoc/
- xDoc tutorials page
  http://xoxinteractive.com/utilities/xdoc/xdoc-tutorials/
- xDoc support forum
  Place our support requests, feature requests, bug reports, etc. here
  http://xoxinteractive.com/questions/category/xdoc/

1.2 @ YOUTUBE

- xDoc tutorials playlist
  https://www.youtube.com/watch?v=GPPuwUxf-bI&list=PLuqlqrwUXc5P5fXISC-LoJDFRgC67LDD5

1.3 EMAILS

- Support: support@xoxinteractive.com
- Contact: contact@xoxinteractive.com

2 ABOUT THIS DOCUMENT

We recommend that you start using xDoc right away. And in addition:

PLEASE, TAKE THE TIME AND WATCH AT LEAST A FEW OF THE INTRODUCTORY TUTORIAL VIDEOS.

You will get an xDoc pro in no time.

xDoc is quite intuitive to use – in contrast to the length of this document!

This document tries to explain every detail, even if it is obvious to common understanding. Therefore. instead of "studying" this document, it makes more sense to use it as reference and to look up specific topics.

2.1 USEFUL KNOW-HOW

Knowing about and understanding the following topics will help you to understand this document more easily.

- Unity Editor
- Inspector
- Game Objects
- Game Object Components
- Hierarchy View
- Scene View
- Scenes

Further helpful topics are:

- Unity Rich-text
- GUI Style
- Builds
- Conditional Compilation

2.2 FEEDBACK ABOUT THIS DOCUMENT

Please feedback via email or in our support forum, if anything in this document is

- wrong,
- not understandable,

- badly explained,
- not explained or
- needs optimization.

This will give us the chance to rectify the shortcomings and release a new and better version.

3 INSTALLATION, SETUP AND EXPORT

3.1 INSTALLATION

Installation of xDoc is very easy. Just follow this simple step:

1) Import the xDoc package from the Unity Asset Store.

Please, make sure that all items are imported: toggles have to be turned on, on every item.

Please do not change the import paths. Everything has to be and stay exactly at their designed position in the folders. Also, do not rename any files or folders. Unity and xDoc rely on folder names and locations. E.g., Unity does not import assets in the 'Assets/Editor Default Resources' folder into builds. By using this folder, we can ensure to keep your builds as clean as possible.

Also, do not edit anything in these folders, unless explicitly told by this manual or by xox interactive.

xDoc will create two folders in the 'Editor Default Resources' folder:

a) '`xDoc`' and
b) '`xDoc-FreeReader`'

The folder structure has to be as shown in the figure below.

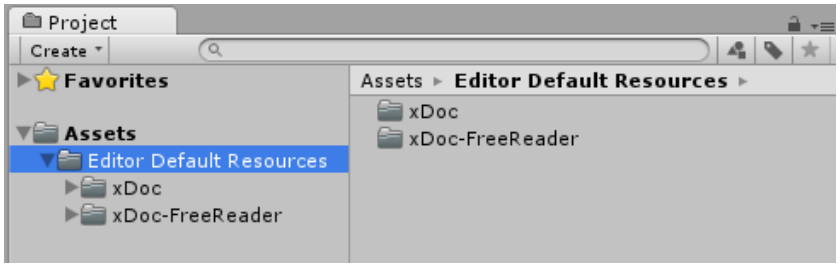


Figure 1: xDoc folder structure

3.2 SETUP

xDoc is functional immediately after installing the xDoc package from the Asset Store. You do not have to set it up or configure it specifically.

Nonetheless, you can customize xDoc to your liking. Please refer to chapter 7 'The xDoc Window'.

3.3 EFFECTS ON THE UNITY EDITOR

After the installation, you get access to two new features in the Unity editor:

➢ *Annotation Component*

The annotation component is a usual game object component, which you can add to game objects or prefabs. Actually, it is a C# script.

The annotation component is the basic building block of your documentation. You build your documentation by adding more and more annotation components to your game objects.

➢ *xDoc Window*

The xDoc window is the central hub for all operation and maintenance of your xDoc documentations.

3.4 EXPORT

You may want to export or share assets, which contain an xDoc documentation and want to export the documentation as well. You can do this easily:

1. Just keep the xDoc Annotation Components in the respective objects and
2. make sure to include the folder '`Assets\Editor Default Resources\xDoc-FreeReader`' into your export file. This folder contains the free xDoc reader.

THE FOLDER 'Assets\Editor Default Resources\xDoc' MAY NOT BE DISTRIBUTED!

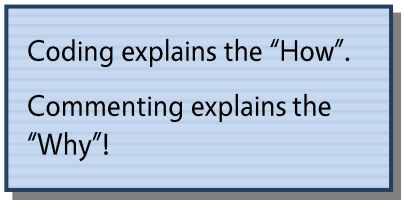
YOU NEED TO HAVE A FULL XDOC LICENSE TO USE THIS FOLDER AND ITS CONTENT. YOU CAN BUY A LICENSE IN THE UNITY ASSET STORE.

With the xDoc reader, the recipient will be able to read your documentation, but will not be able to edit it. Please note that the recipient will also be able to delete your documentation, if he wants to.

For the recipients convenience the xDoc reader includes the xDoc search engine; please see section 7.2 ‘The xDoc Search Engine’.

4 DOCUMENTATION IN SOFTWARE ENGINEERING

Documentation is a very important part of software engineering. However, software engineers have quite different opinions on how documentation has to be done the right way – if at all. Self-documenting code, literate programming and excess documentation are a few key words in these vivid discussions.



We respect everybody's attitude to this topic and do not think that there is only one right approach. Nevertheless, at least the tools should be available, so that everybody is able to implement his own style.

One best practice and a very effective way of technical documentation is embedded documentation. It has proven itself since decades and is available for almost any development environment. The developer can write the documentation while referring to the code he is just working on, and can use the same tools to make the documentation he used to create the code. This of course makes it much easier to keep the documentation up-to-date!

If embedded documentation is so great, how can you use it in Unity? In Unity, development takes place in two distinct places: the script editor (e.g. Mono) on the one hand side and the Unity Editor on the other hand side.

While it is possible to create embedded documentation in all script editors, this is not possible in the Unity editor – at least not out of the box. This missing piece is now covered by xDoc: **XDOC BRINGS EMBEDDED DOCUMENTATION INTO THE UNITY EDITOR!**

4.1 FROM EMBEDDED COMMENTS TO DOCUMENTATION

An embedded comment alone does not make documentation. All comments, reminders, notes, explanations, etc. together form the embedded documentation of your project.

However, it is not enough that your pieces of documentation – your content– are just available somewhere in your project. The real value of documentation arises when you can access it

easily, whenever you need the information – and not just because you are a genius and can keep it all in your mind or accidentally stumble over it. You have to be able

- to *browse* your content (*structured access*) and
- to *search* your content (*unstructured access*).

In order to facilitate this

- *content data* has to be available of course and has to be enriched by
- *meta data* for classification and versatile usage.

Classified data can be structured for browsing or be searched effectively.

Additionally, a documentation system needs

- *bulk operations* to facilitate effective operation and maintenance in a real life project.

4.2 XDOC'S BASIC CONCEPT

The main development work in the Unity editor happens while creating scenes: game objects are created and their behaviors are designed by adding and customizing so called 'Components' to them.

Therefore, xDoc is also designed to add 'Components' to game objects in order to annotate them; and by doing so, to – step by step – build the documentation.

This way the developer or designer can use the same tools to create the documentation in the Unity editor as he is using for building the scenes.

➢ *xDoc Annotations*

xDoc Annotations are components, which hold a piece of text and some meta data.

Every xDoc Annotation has an assigned annotation type, which is the most important piece of meta data.

➢ *Annotation Types*

Annotation Types classify your annotations. They group annotations into distinct sets (types) of annotations – each with a specific documentation function.

The grouping of annotations is reflected in three areas. Annotations of the same type

- share the same look and feel,
- are used in the same placed in the Unity editor (inspector, hierarchy view, scene view) and
- have the same custom meta data structure.

With Annotations and Annotation Types, you can create and maintain your documentation content and their meta data.

➢ xDoc Window

The xDoc window is the central hub for all annotation related operations. It gives access to the xDoc search engine, bulk operations, the customization of annotation types and user settings.

You can browse Annotations in the bulk operations screen; either to access your content or to perform operations on them.

In addition, you can use the hierarchy view to browse annotations, if the respective annotation type is set up accordingly.

➢ Annotations, Types and the Operations Hub

xDoc Annotations, Annotation Types and the xDoc Window as central hub for operations are the three main conceptual building blocks of xDoc. Together they form an efficient and practical documentation system for Unity projects. You can now build your documentation embedded right in the Unity editor and have access to it.

We will elaborate each of these building blocks in deep detail in the next chapters.

5 XDOC ANNOTATION

xDoc Annotations are Components to Game Objects and are edited in the Inspector or in the Hierarchy View. They are shown in the Inspector, the Hierarchy View or Scene View – depending on their Annotation Type.

The following figures show annotations in various settings.

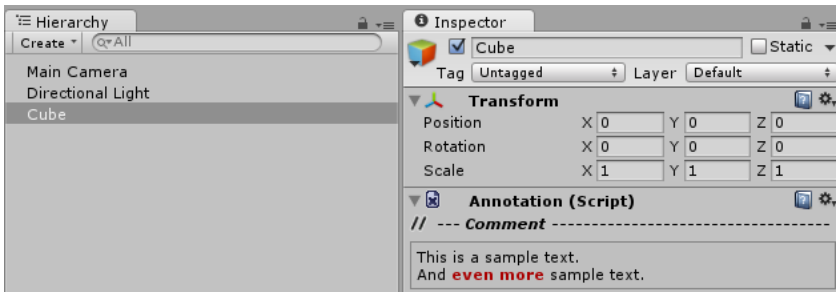


Figure 2: A simple xDoc Annotation

Annotations support text in rich text format. In the figure above the annotation is only shown in the inspector.

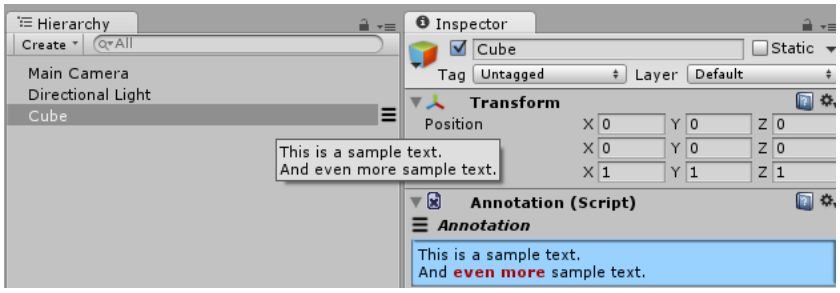


Figure 3: An xDoc Annotation with an icon in the Hierarchy View

If configured accordingly, the annotation can also be shown as an icon in the hierarchy view. Hovering over the icon shows a popup with the annotation text.

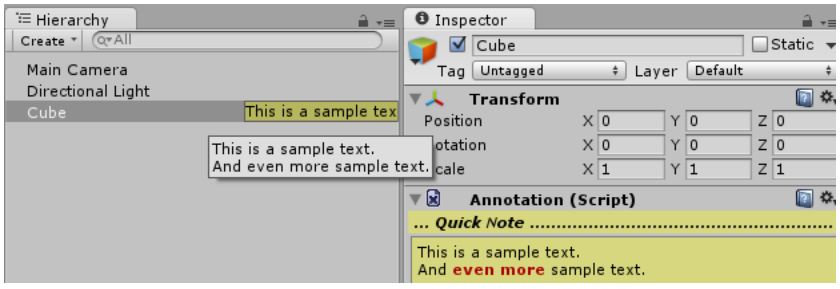


Figure 4: An xDoc Annotation with a text entry in the Hierarchy View

Instead of the icon, the annotation can also use a text field in the hierarchy view. You can edit the annotation directly in the text field; and again hovering over the text field will show a popup with the annotation text.

Up to here, we have seen examples for annotations in the inspector and hierarchy view. The next figure shows an example for an annotation in the scene view.

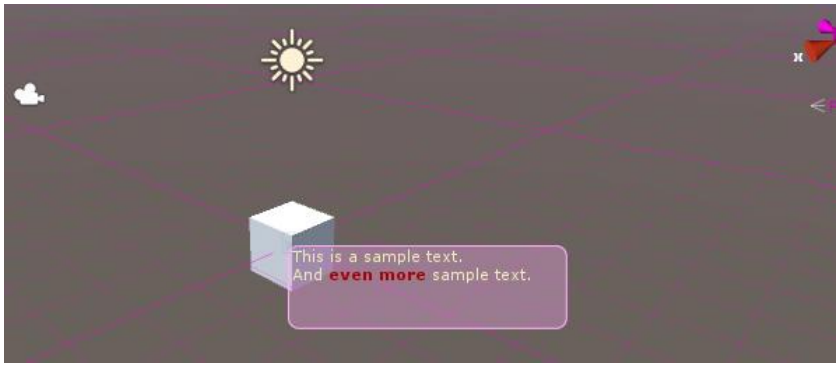


Figure 5: An xDoc Annotation in Scene View

As you can see in the examples above, the presentation style changes with the annotation type: background color, frames, icons, title styles, etc. In the examples above the following annotation types have been used: Comment, Annotation, Quick Note and Scene Note. Annotation Types are not static or fixed. Change, create or delete annotation types as you like.

You can adapt the presentation style of annotations to your taste and needs,

- either by changing the settings for all annotations of a type by customizing the annotation type or
- by adjusting the settings of specific annotations.

Possible customizations include the max-allowed height for the text or data section, foreground and background colors, fonts, background textures, etc.

This way you can define if an annotation does or does not take a prominent role in the game object or project – whatever suits your documentation needs best.

5.1.1 ANNOTATION CUSTOM DATA / META DATA

Annotations can have additional custom data, if the corresponding Annotation Type defined and enabled them. Custom data are additional data fields under the text field as

shown in the next figure. Each Annotation Type can define its own set of data fields.

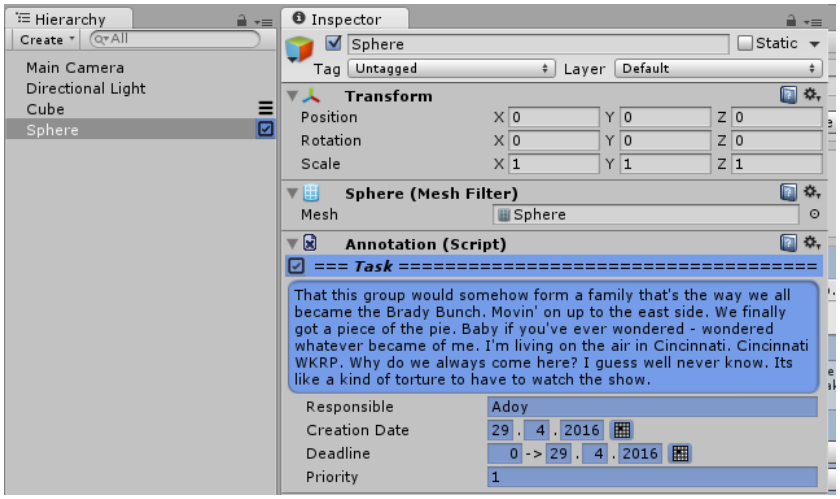


Figure 6: Annotation with Custom Data / Meta Data

The following data field types exist:

- String
- Date
- Date Span
- Time
- Bool
- Int
- Float
- Link
- Game Object
- Object

The next figure shows an annotation with all possible data field types.

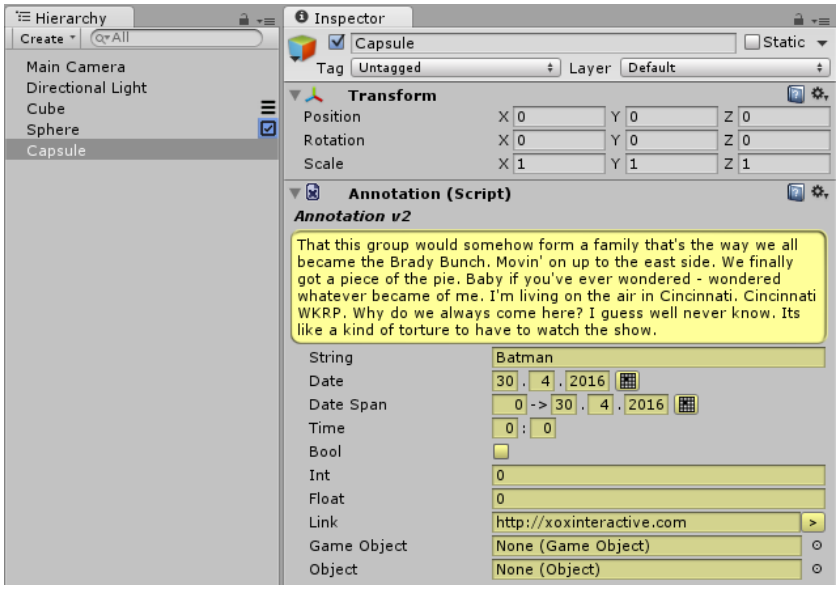


Figure 7: Annotation with all possible custom data field types.

➢ *String*

This data type holds any string; e.g. a name.

➢ *Date*

This data type holds a normal calendar date in the format 'day.month.year'; e.g. the creation date of the annotation.

The button at the right hand side is only shown, if space is available and will set the date to the current system date.

➢ *Date Span*

This data type holds a date span; i.e. the number of days between the current system date and the date in the date entry. The date span field has the format
'number of days -> date entry'. The date entry field has the same format as above.

The number of days can be positive, zero or negative. A possible usage for a date span field would be a deadline e.g.

The button at the right hand side is only shown, if space is available and will set the date to the current system date. You can either enter the number of days or the date to set the date span.

➢ *Time*

This data type holds a time in the 24-hour format.

➢ *Bool*

This data type holds usual bool data: yes/no, true/false, on/off, etc.

➢ *Int*

This data type holds an integer number.

➢ *Float*

This data type holds a float number.

➢ *Link*

This data type holds a link to a webpage, web-document or documents in your file system. Just enter the link address to the document as you would enter it in a web browser.

The button on the right side of the link entry will open the linked document with the appropriate application.

➢ *Game Object*

This data type will hold a game object reference.

➢ *Object*

This data type will hold an object reference.

5.1.1.1 OBJECT REFERENCE LIST

Annotations can optionally use an Object Reference List, if it is enabled in the annotation type. The figure below shows an annotation with an Object Reference List.

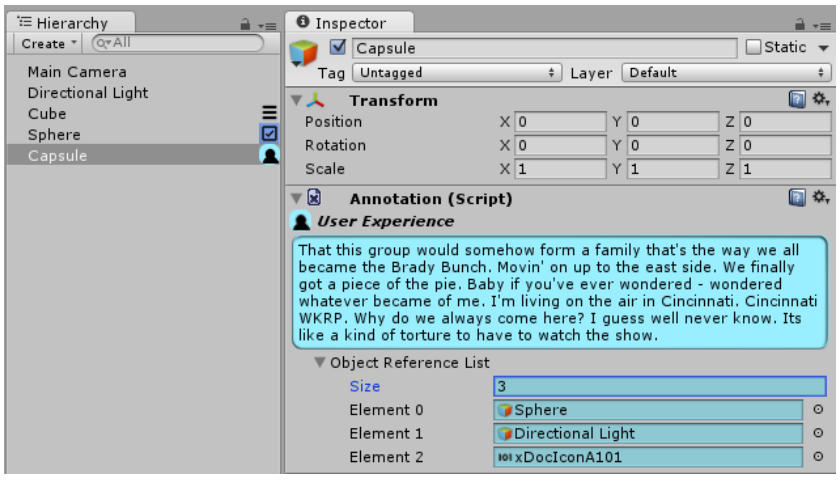


Figure 8: Annotation with an Object Reference List

You can use the Object Reference List to maintain a list of references to objects in the scene or in the project folder. First, you have to set the size of the list and can then set the object references per list element. A possible usage would be a bug, which is affecting more than one game object; or you might want to keep references to several alternative textures for a 3D-model.

Game objects in the Object Reference List can be raycasted in the scene view, if the corresponding Annotation Type has been set up accordingly. Please refer to the section about Annotation Type customization.

5.2 CREATE AN ANNOTATION

As discussed already, xDoc Annotations are just components. Therefore, to create an annotation you just have to add this type of component to the respective game object. You can use any of the standard ways to do this. Here are three possibilities:

1) First, select the game object you want to annotate. Then use the 'Add Component' button in the inspector and select the 'xDoc Annotation' entry. The 'xDoc Annotation' entry should be at the top usually.
2) First, select the game object you want to annotate. Then use the 'Component' menu in the Unity Editor and select the 'xDoc Annotation' menu entry.
3) First, select the game object you want to annotate. Then use the shortcut button in the xDoc window. The shortcut button is located in the upper right corner of the xDoc window. It is labeled with a '+'-sign and has a green background. Just click it.

As result the annotation component will be added to the selected game object.

The following figure shows these three ways to create an xDoc annotation.

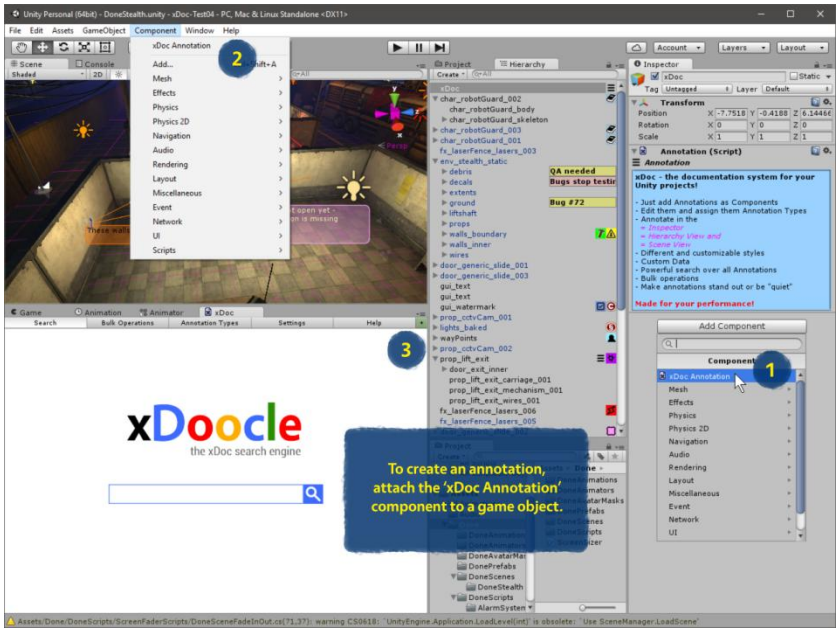


Figure 9: Create an xDoc Annotation

5.3 EDIT AN ANNOTATION

The main element of the annotation in the inspector is the text field, which holds your content data. Depending on the annotation type, custom data fields may exist as well.

The annotation component has two modes in the inspector:

- view mode and
- edit mode.

All figures we have seen up to here have shown the annotation in view mode, which is the default mode. As soon as the annotation component gets the focus, the mode is switched to edit mode. As soon as it loses the focus the mode goes back to view mode again.

In view mode, the text field can display text in rich-text format. Please refer to the Unity documentation for rich-text and the tags available for rich-text formatted text. However, you don't have to memorize them. The menu bar of the Annotation component gives access to all available rich-text tags. Please read on.

The next figure shows the annotation in edit mode.

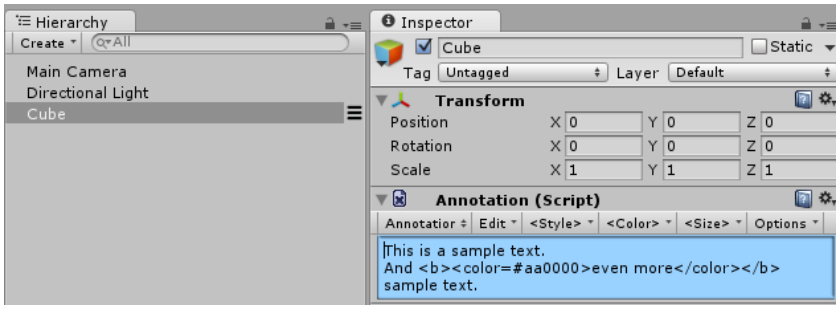


Figure 10: xDoc Annotation in the Inspector in Edit Mode

Two things happen, when the mode switches to edit mode:

1) the annotation title – which shows the annotation type in display mode – is replaced by a menu bar and
2) the edited text is now shown in plain text. This way you are able to edit the rich-text tags manually, if you want to.

Please note, whenever the annotation component switches to edit mode the focus will be in the text field of the annotation component – even if you clicked into a custom data field.

In edit mode, the text field behaves just like an usual, simple text editor: writing text, selecting text[1], moving the cursor with the mouse or arrow keys, paging up / down, deleting, copying, pasting text, etc. works just as expected.

The menu bar gives access to editing commands, which will make text editing easier. Please read on for a detailed description of every menu bar item.

5.3.1 ANNOTATION TYPE MENU

The Annotation Type Menu (the left most menu) will show a drop down menu with all your annotation types. They are displayed in the order you have assigned to them (please refer to section 7.1.1 Annotation Types List).

The figure below shows the annotation types menu for the default setup with the 20 pre-defined annotation types.

[1] Click and drag with the mouse or 'Shift'-key press and move the cursor with the arrow keys to select text.

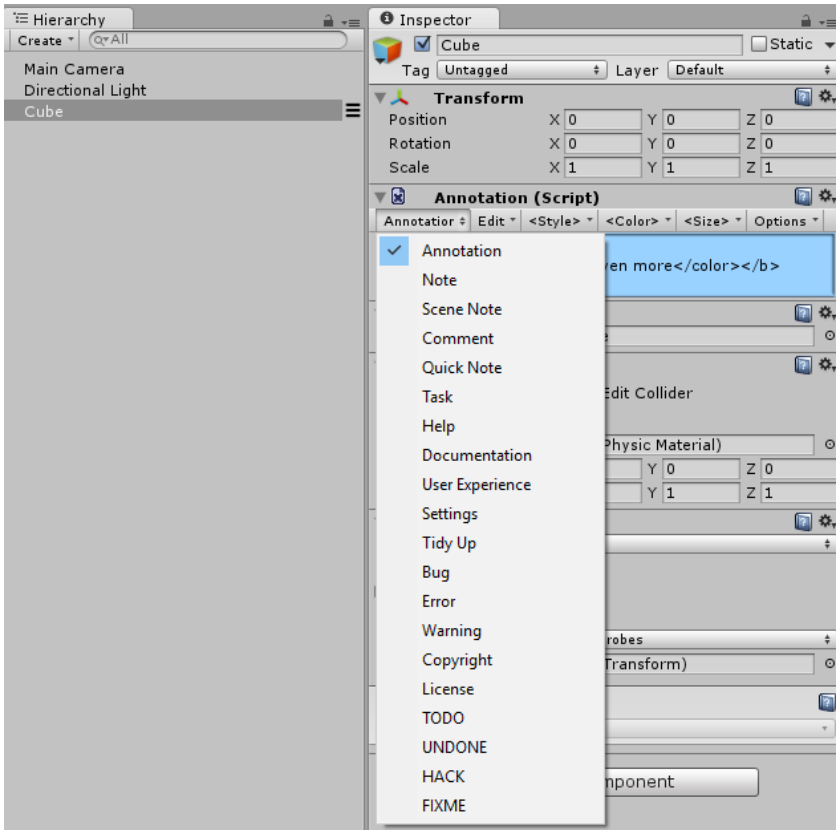


Figure 11: Annotation Edit Mode, Annotation Type Menu

Selecting an entry will change the annotation type: The style will change and the custom data will be adapted to the new custom data structure. xDoc tries to match and keep as much of the custom data as possible. But you may lose some of the custom data, if matching is not possible or custom data fields are not available in the new annotation type. For more information about custom data, please refer to section 7.1.3 Custom Data / Meta Data)

5.3.2 EDIT MENU

The edit menu has the usual set of 'Select All', 'Cut', 'Copy' and 'Paste' commands. They behave the expected way – as in any other text editor.

The next figure shows the edit menu.

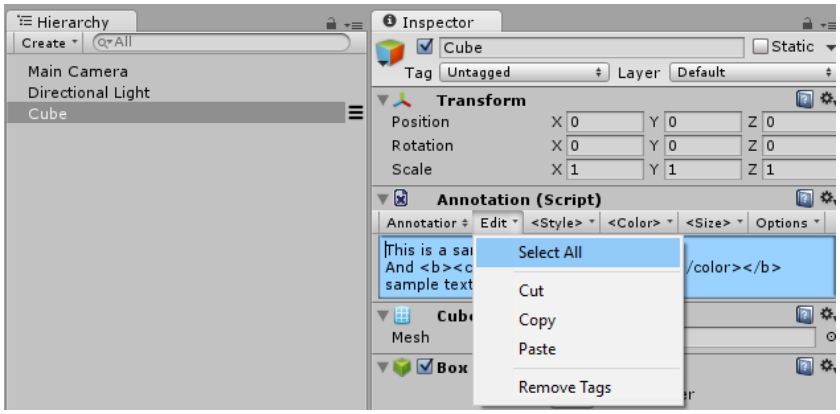


Figure 12: Annotation Edit Mode, Edit Menu

Additionally, the 'Remove Tags' command is also accessible in the edit menu. The 'Remove Tags' command will remove all[2] bracket tag commands from the selected text. This way you can easily remove rich-text tags and clear the formatting.

5.3.3 STYLE MENU

The next figure shows the '<Style>' menu.

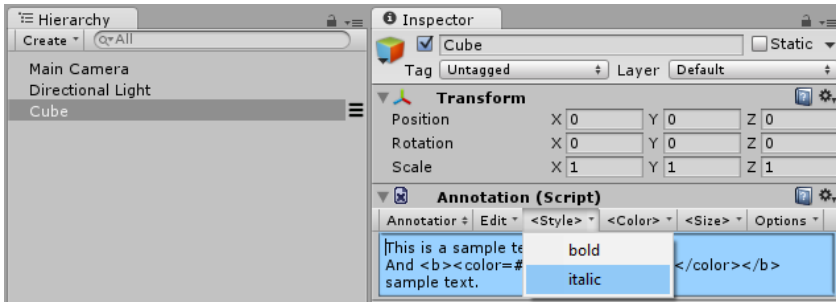


Figure 13: Annotation Edit Mode, Style Menu

The style menu has two entries:

- bold and
- italic.

Selecting an entry adds bold or italic rich-text tags around the currently selected text.

[2] 'All' means *all* in this case and not only rich-text tags. Anything within brackets ("<…>") will be removed together with the brackets.

5.3.4 COLOR MENU

The next figure shows the '<Color>' menu.

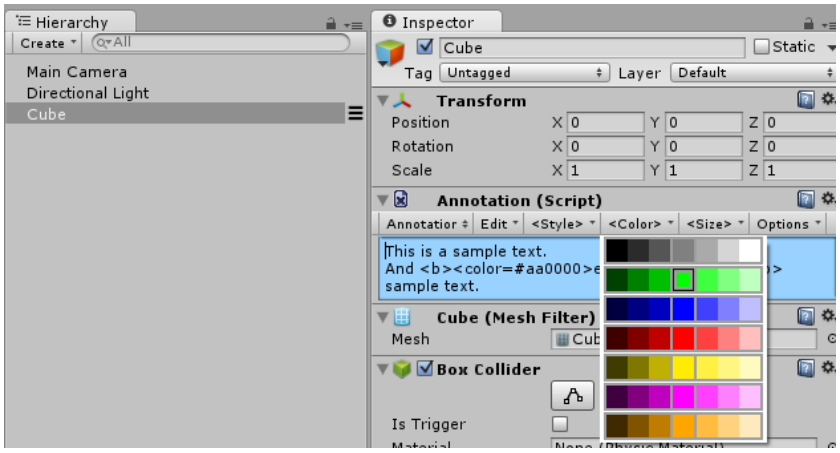


Figure 14: Annotation Edit Mode, Color Menu

The color menu will display the color selection window. Selecting an entry adds color rich-text tags around the currently selected text.

Optionally, you can fine-tune the color by editing the rich-text tag directly in the text field.

5.3.5 SIZE MENU

The next figure shows the '<Size>' menu.

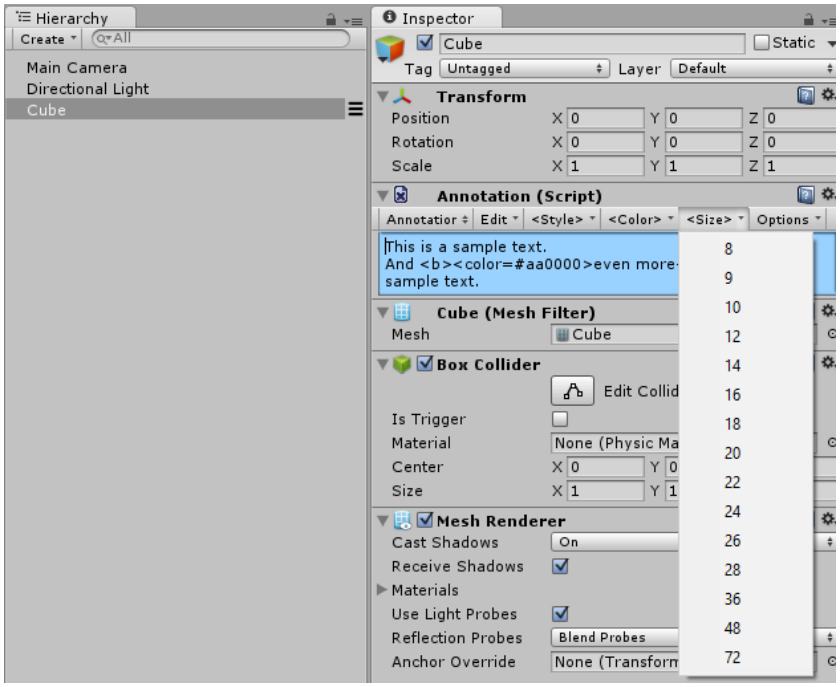


Figure 15: Annotation Edit Mode, Size Menu

The size menu has a couple of preconfigured text sizes.

Selecting an entry adds size rich-text tags around the currently selected text. You can fine-tune the size by editing the rich-text tag directly in the text field.

5.3.6 OPTIONS MENU

The options menu gives access to general style settings for the specific annotation in the inspector and in the scene view. This way you can adapt the display settings of the annotation exactly to your needs and taste. The following settings are available:

- Inspector Options
  - Word-wrap
  - Rich-text
  - Max. Height Text
  - Max. Height Data
- Scene View Options
  - Word-wrap
  - Rich-text
  - Max. Width
  - Max. Height
- Game Object
  - Place After

The next figure shows the options menu.

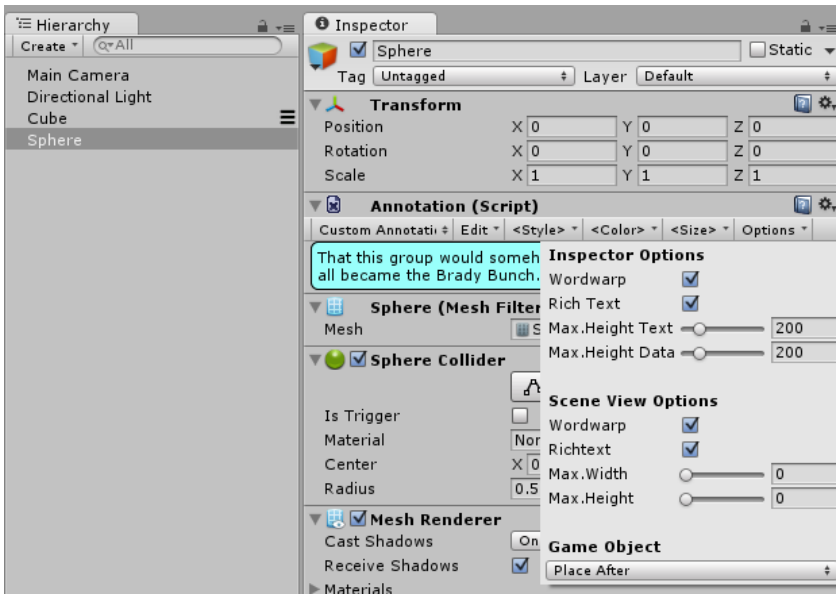


Figure 16: Annotation Edit Mode, Options Menu

5.3.6.1 INSPECTOR OPTIONS – WORDWARP

With word-wrap enabled, line breaks are inserted, if a line is longer than the width of the inspector. This way all of the text can be kept on screen. Without word-wrap, the original formatting is kept as is. The effect of word-wrap on and off is shown in the following two figures.

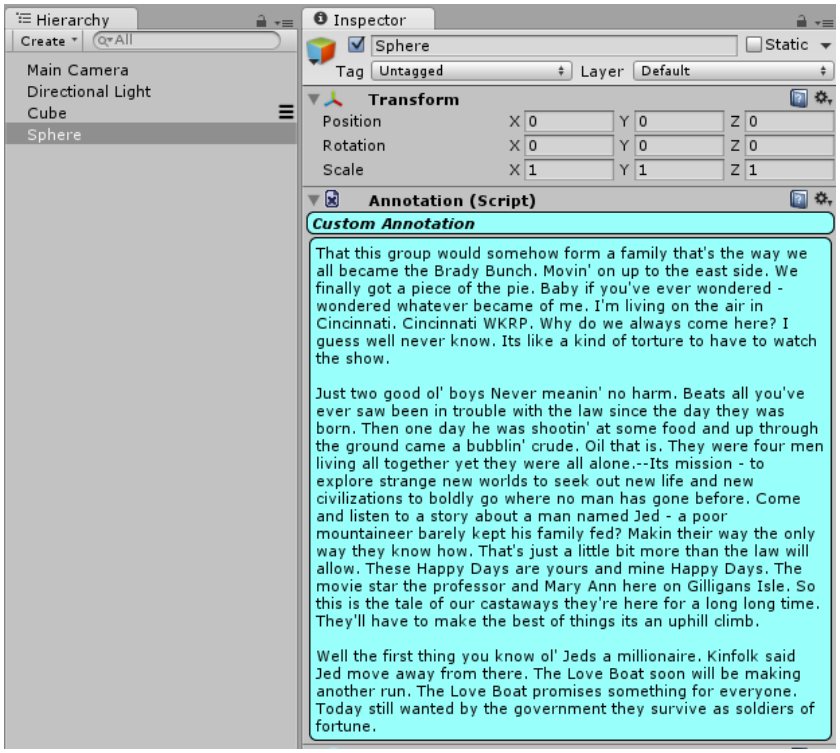


Figure 17: Long annotation with three paragraphs and word-wrap turned on

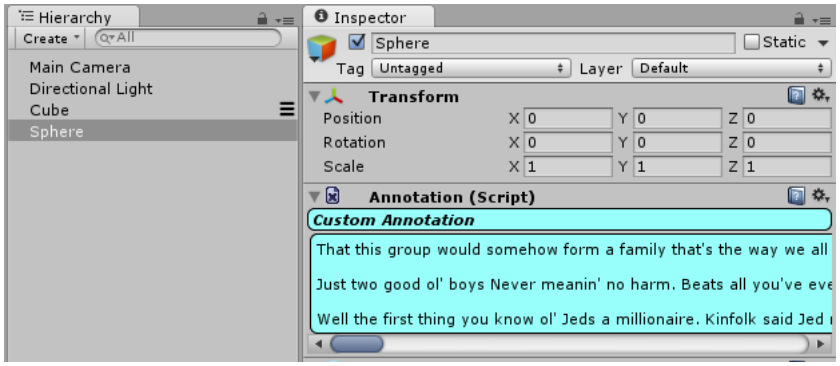


Figure 18: Same long annotation with three paragraphs and word-wrap turned off

5.3.6.2 INSPECTOR OPTIONS – RICH-TEXT

With rich-text turned on, text in brackets "<…>" will be interpreted as rich-text tags. This way it is possible to display bold, italic and normal text in different colors and sizes.

With rich-text turned off, the text will be displayed as is.

5.3.6.3 INSPECTOR OPTIONS – MAX. HEIGHT TEXT

The text field adapts its height automatically to the size of its content. For very long texts, the annotation may take up nearly the whole display space and hinder your workflow. Display space is very precious in the Unity editor and therefore you have the option to limit the maximum height of the text area.

Please compare Figure 19 (next figure) with Figure 17 to see the effect of parameter "Max. Height Text". In Figure 19 the parameter is set to a smaller value than in Figure 17. When the text is longer than the maximum allowed height, the text area is displayed as scrollable text area.

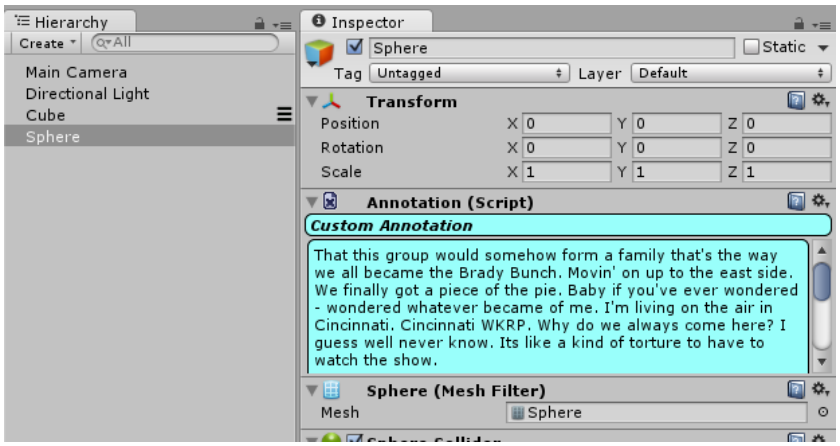


Figure 19: "Max. Height Text" limits the space the text area can take

5.3.6.4 INSPECTOR OPTIONS – MAX. HEIGHT DATA

The "Max. Height Data" parameter works as the "Max. Height Text" parameter – just for the data section instead of the text section. Figure 20 and Figure 21 show the effect of this option: large value vs. small value.

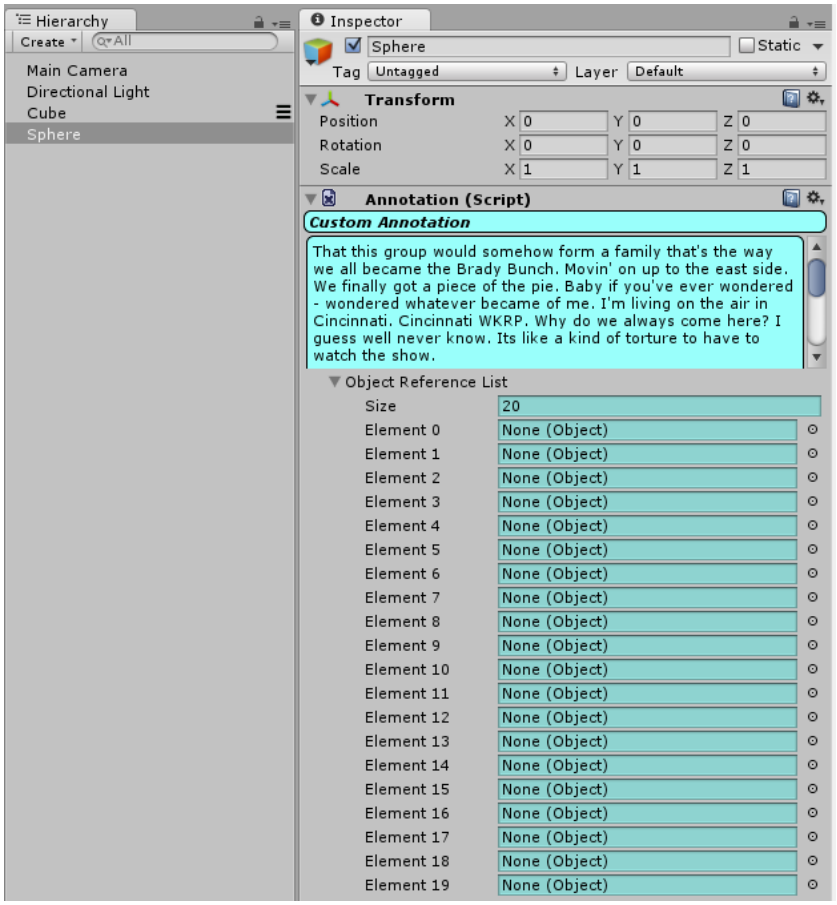


Figure 20: An annotations with a long data section

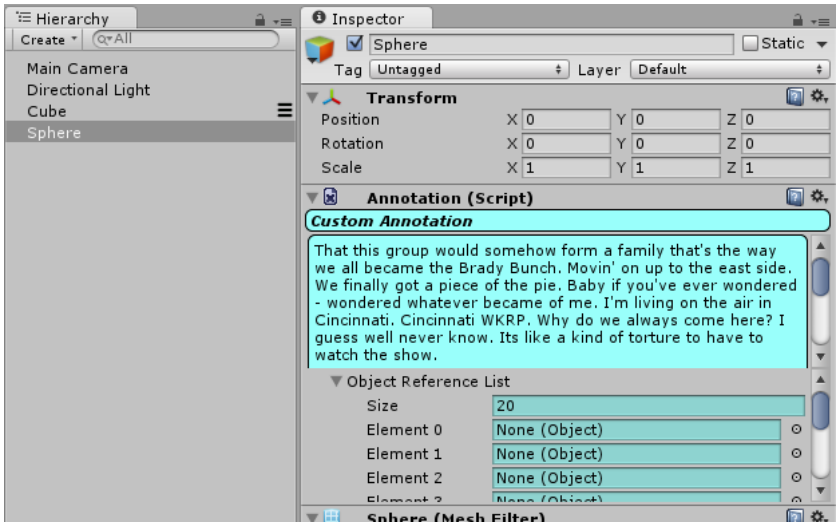


Figure 21: An annotation with a long data section, delimited by the "Max. Height Data" parameter

5.3.6.5 SCENE VIEW OPTIONS – WORDWARP

This option works the same way as the word-wrap option for the inspector, but just for the text in the scene view: turns word-wrap on or off for the scene view text.

5.3.6.6 SCENE VIEW OPTIONS – RICH-TEXT

This options works the same way as the rich-text option for the inspector, but just for the text in the scene view: turns rich-text on or off.

5.3.6.7 SCENE VIEW OPTIONS – MAX. WIDTH

This option sets the width for the text in scene view. If it is set to zero, the default setting form the annotation type is taken.

5.3.6.8 SCENE VIEW OPTIONS – MAX. HEIGHT

This option sets the height for the text in scene view. If it is set to zero, the default setting form the annotation type is taken.

5.3.6.9 GAME OBJECT – PLACE AFTER

The place after option opens a menu, which lists all current components in the game object. The next figure shows an example of the place after menu.

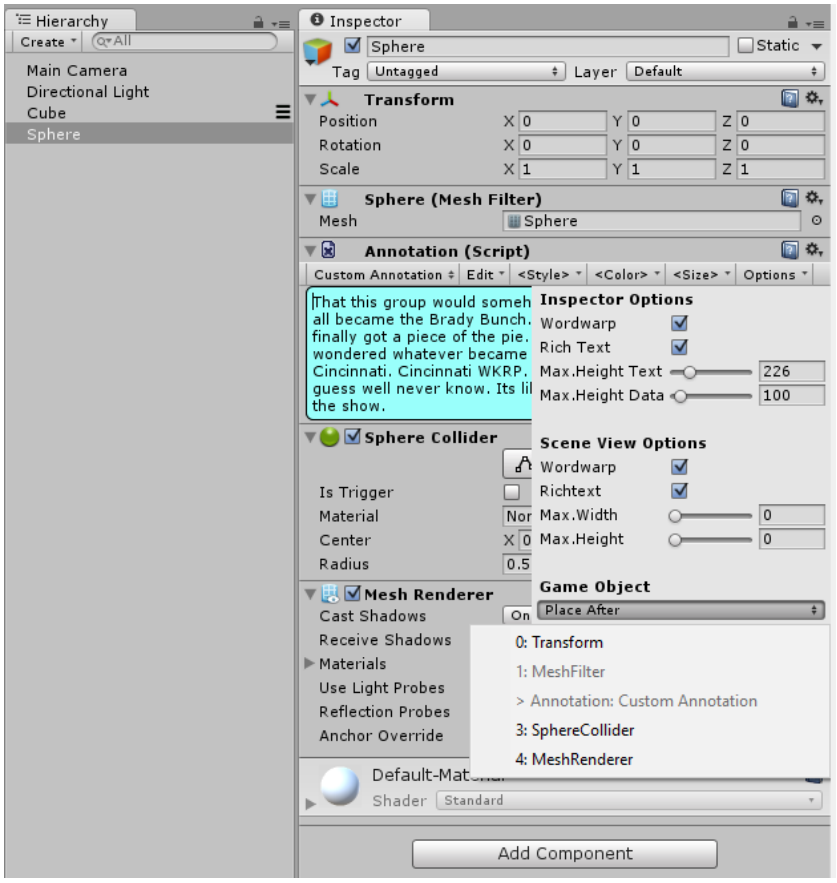


Figure 22: Annotation Edit Mode, "Place After" Menu

The place after option is used to easily position the annotation component within the game object. When a menu entry is selected, the annotation is placed right after the selected component.

6 ANNOTATION TYPE

Annotation Types group annotations into sets: E.g. Bug-type annotations, feature request type annotations, to-do or task type annotations, etc.

Each Annotation Type has its own display style in the Inspector, the Hierarchy View and the Scene View. Furthermore, each annotation type can define its own custom meta data structure, which is used by annotations of this type.

All these topics will be presented in detail in the next chapter in the context of the xDoc Window.

xDoc comes with 20 predefined annotation types. However, you are strongly recommended to change, add and delete annotation styles. Please customize xDoc to match your style of working and to cover your documentation needs best.

Every individuals, every teams, every projects needs in terms of documentation will differ. Depending on the current set up and your style of working

- lot of icons in the hierarchy view may help you to quickly get an overview; or make just too much clutter and make you less efficient.
- you may concentrate better, if colors are used sparsely; or your attention is guided better, if colors are used more prominently.
- the usage of meta data may help you to use the documentation more effectively; or may hinder your workflow, because it may be too bureaucratic, cumbersome and individuals may be not accurate when entering data.
- Etc.

Please, set xDoc up in such a way that it supports your and your teams style of working best.

6.1 THE INVALID ANNOTATION TYPE

When an annotation type is deleted, while annotations still use it (still reference it), the type of these annotations is set to the invalid annotation type.

This annotation type is just an usual annotation type; the only difference is that it is not available for the user for customization.

7 THE XDOC WINDOW

The xDoc window is the main hub for all operation and maintenance topics around annotations.

To open the xDoc window open the 'Windows'-menu in the Unity editor and select the 'xDoc' entry as shown in the following figure.

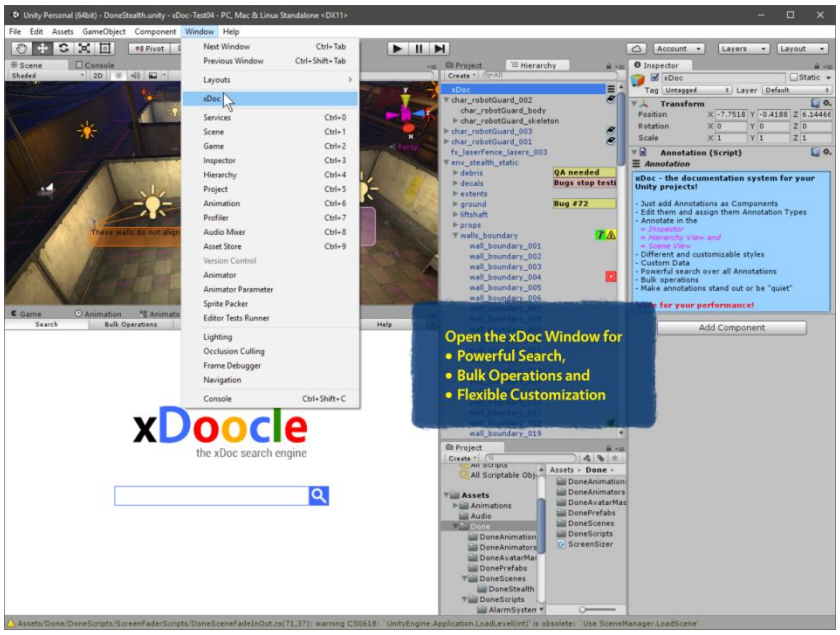


Figure 23: Open the xDoc Window

The xDoc window gives access to

- the xDoc search engine,
- bulk operations,
- customization of the Annotation Types,
- user settings and
- help.

Please refer to the following sections for detailed descriptions.

7.1 ANNOTATION TYPES CUSTOMIZATION

Annotations and annotation types go hand in hand. Every annotation must have and does have an annotation type – even if it is the invalid annotation type, which will be discussed later.

Annotation types have two important functions:

1) They define the look and feel of the respective annotations and
2) they define a custom meta data structure per annotation type.

For each annotation type, you can define the style (look and feel) of the following five elements of an annotation:

- the icon in the hierarchy view and in the title line,
- the title line in the inspector,
- the text in the inspector,
- the text in the scene view and
- the text in the hierarchy view.

An Unity IMGUI style element, a background color and a couple of specific options define each of the five style settings mentioned above.

The Unity IMGUI System is described in the official Unity documentation: http://docs.unity3d.com/Manual/class-GUIStyle.html. Please refer to the link above, if you have detailed questions about how to tweak Unity GUI styles to your liking.

You can customize each of the five styles independently from each other. However, usually it makes sense to have a common look and feel for the same annotation type, regardless of where it is used; e.g. a same background color may be a good idea. To make this synchronization easy, a simple style synchronization tool exists. You can use it or adjust the settings manually, if you want to.

Do not be overwhelmed by all the options. You CAN use all of them, but you do not have to.

As next, we will discuss all the details about how to create, delete and customize annotation types.

7.1.1 ANNOTATION TYPES LIST

The next figure shows the Annotation Types Tab in the xDoc Window. The left pane shows the list of all Annotation Types and the right pane shows the details of the selected Annotation Type. In the current case, no annotation type is selected and a message is reminding this fact.

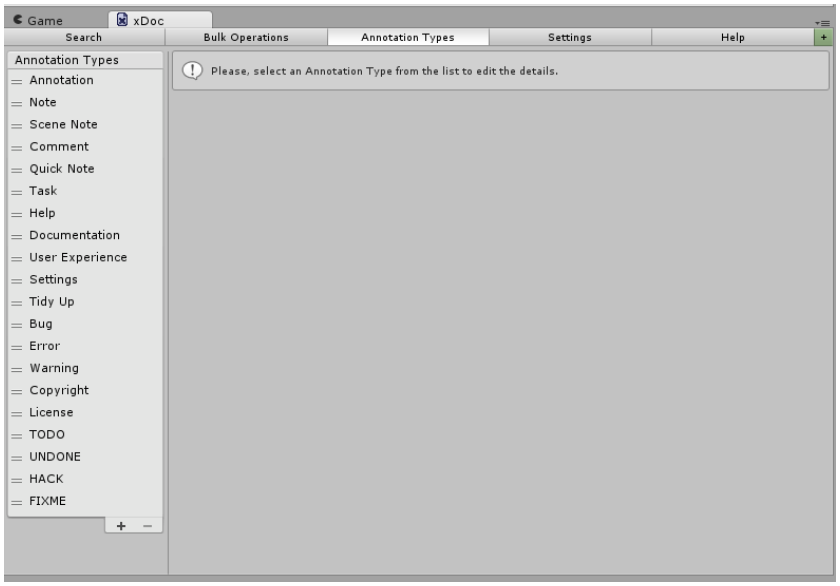


Figure 24: xDoc Window, Annotation Types Tab

7.1.1.1 ADD A NEW ANNOTATION TYPE

You can add a new Annotation Type just by clicking the '+'-button in the footer of the list. An annotation type is created with the default settings, which you can customize now.

PLEASE NOTE THAT ANNOTATION TYPE NAMES HAVE TO START WITH A LOWER CASE OR UPPER CASE LETTER!

7.1.1.2 DELETE AN ANNOTATION TYPE

To delete an annotation type just select the respective annotation type and click the '-'-button in the footer of the list. Then click 'Yes' in the confirmation dialog and the currently selected annotation type is deleted.

Please be aware that annotations still referring to the deleted annotation type will now have the 'invalid' annotation type. You can either set a new annotation type per annotation or use bulk operations to set them all. Alternatively, you can delete these Annotations, or leave them as they are.

7.1.1.3 REORDER THE ANNOTATION TYPES LIST

To reorder the Annotation Types List, just grab the '='-handle of an annotation type in the list and drag and drop it into the position where you would like it to be.

Please note that the Annotation Type menu in the inspector will show the annotation types in the order of this list. Also note, that the Annotation Type at the first position will be the default annotation type: newly created annotations will be created with this type.

7.1.2 EDIT ANNOTATION TYPES

To edit an annotation type you have to open the 'Annotation Types' tab in the xDoc window and select an Annotation Type from the list. The next figure shows an example of the resulting window.

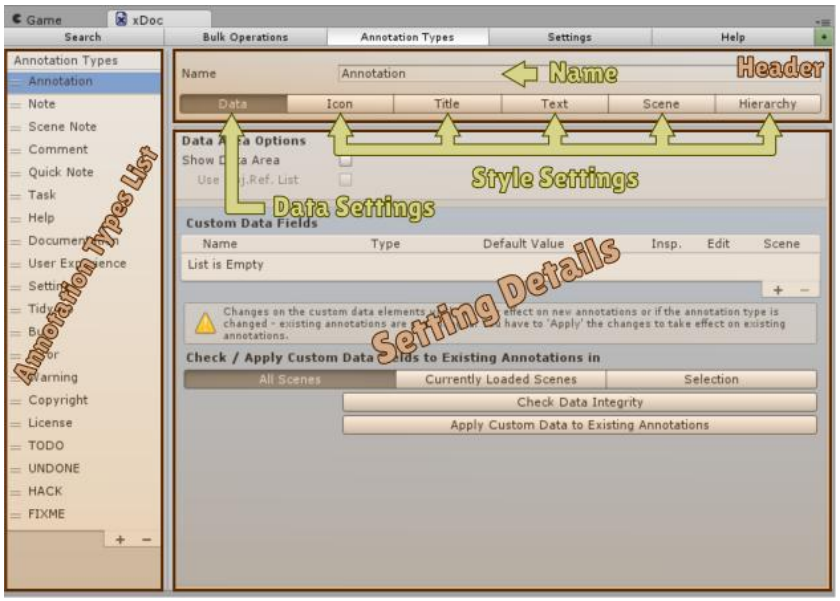


Figure 25: Annotation Type Settings

The Annotation Types tab has three main sections:

- the Annotation Types List,
- the Header section
  of the selected Annotation Type and
- the Setting Details
  of the selected set of Settings
  of the selected Annotation Type.

7.1.2.1 ANNOTATION TYPE NAME

The Header section contains the name field, where you can edit the name of the annotation type. Annotation Types must have unique names. xDoc will add postfixes to names, if they are not unique.

PLEASE NOTE THAT ANNOTATION TYPE NAMES HAVE TO START WITH A LOWER CASE OR UPPER CASE LETTER!

7.1.2.2 SETTING TABS

The Header section also contains the 6 setting tabs, which switch between the settings of the selected annotation type; 1 data setting and 5 style settings exist, which can be edited in the 'Setting Details' section.

7.1.3 CUSTOM DATA / META DATA

This section discusses the custom data (meta data) settings for annotation types. First:

1. Open the xDoc Window.
2. Select the 'Annotation Types' tab.
3. Select the Annotation Type you want to edit.
4. Select the 'Data' Setting

The next figure shows the resulting xDoc window. You can now edit the Custom Data settings in the Setting Details section.

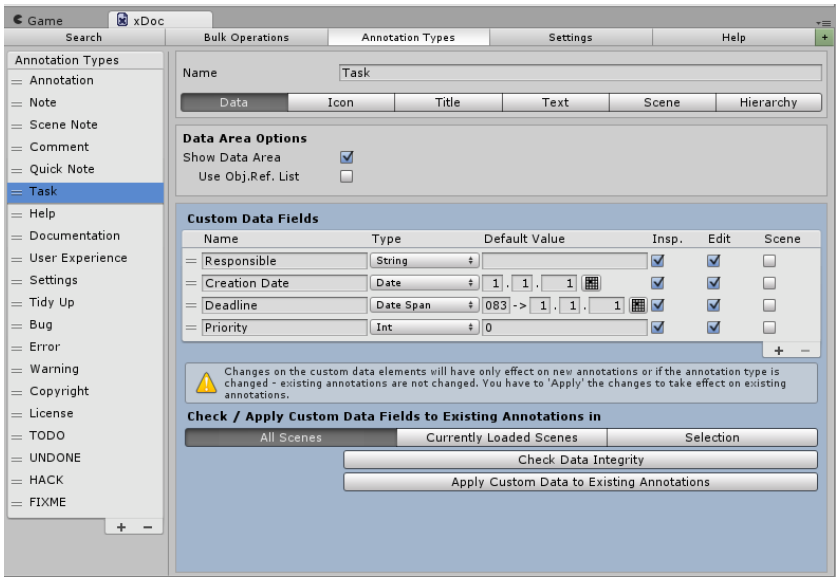


Figure 26: Annotation Type, Custom Data Customization

➢ *Option: Show Data Area*

Show Data Area has to be enabled, if you want the custom data or the object reference list to be shown in the inspector.

➢ *Option: Use Obj.Ref. List*

This option enables the Object Reference List.

7.1.3.1 CUSTOM DATA FIELDS

Custom Data Fields are managed in a reorderable list, similar to the reorderable list of annotation types:

- Click the '+'-button to add a new custom data field.
- Click the '-'-button to delete the selected data field.
- Grab the '='-handle and drag and drop the data field to reorder the list.

Each data field has

- a name,
- a type,
- a default value and
- three view settings.

➢ *Parameter: Name*

Select any but unique name for the data field.

➢ *Parameter: Data type*

Select a data type from the dropdown menu.

➢ *Parameter: Default Value*

You can set a default value, if you need one. The default value is used when an annotation is newly created; the respective field gets this value assigned as first value.

You could e.g. have an author field and set the default value to your name; this way every created annotation automatically gets the right author assigned.

Date and Date Span types have a special default value: 1.1.1. This value will be replaced with the current system date when an annotation is newly created.

➢ *Parameter: View Settings*

For each data field you can set, if it is going to be shown in

- the inspector in view mode,
- the inspector in edit mode and / or
- in the scene view.

E.g., you might want to hide an author field or creation date field in edit mode, but show it in view mode. Or you might want to hide data fields in scene view to avoid clutter. Just adjust these settings to your needs and taste.

7.1.4 CUSTOM DATA MIGRATION

Sometimes your custom data has to be migrated (changed). This happens because you directly or indirectly requested this change. In some cases, this change is initiated automatically and in some cases, you have to initiate it manually.

This section will help you to understand what is going on when; and therefore it will help you to initiate custom data migrations only knowingly.

Data migration might sound intimidating. However, there is nothing to be hesitant about. It is straightforward, easy to understand and face it, part of every IT professional's life.

The parameter values of the style options (of annotation types) are stored in the annotation type. In contrast to this, custom data does not reside inside the annotation type. Instead, custom

data is *instantiated within every single Annotation*. This means, custom data is not referenced data like e.g. a background color, icon texture, etc. This makes sense of course, as every annotation can have different values for its own custom data fields.

The Annotation Type only defines the common data structure of the custom data. If you create an Annotation, the Annotation is created with an own set of custom data fields. They will have the same structure as in the Annotation Type, but will be distinct data containers.

At some point, you might want to change this structure. You do so by changing it in the Annotation Type. However, if you change this structure, it will result in a difference between the newly defined structure in the *Annotation Type* and the already existing structures in the *Annotations* – the data integrity is broken, which is not a good thing generally.

One could propose to adapt the existing data in the annotations to the new structure – we call this data migration – immediately and automatically, but this not a good idea as well for a couple of reasons.

First of all, data migration should not be a behind-the-scenes, hidden process. A person should be responsible and aware about what is happening. The migration process should be initiated intentionally – especially because in this case it is a bulk operation and can have a wide effect. And e.g., you don't want constant data migrations, while you are still in the middle of setting up the new structure.

Secondly, immediate data migration does not solve the problem of possible broken data integrity in a real life scenario. If you are using a versioning system and are getting back an older version of game object (scene), you might get back the object with an old data structure.

The solution is not to avoid broken data integrity in our case, but to deal with it. And xDoc can deal with it and work in such a situation just as normal. The broken data integrity does no harm to xDoc or xDoc functioning. It is bad from a more practical point of view. Let us say you have a 'Task' Annotation Type, which has a priority field. Usually you want to be able to rely on that all annotations have this data field. If data integrity is broken, you can't.

xDoc offers to check the data integrity and (if needed) to migrate the data as bulk operation. You can select the scope of such a *data bulk operation*.

7.1.4.1 SCOPE OF CUSTOM DATA OPERATIONS

Before initiating a data bulk operation, you have to select the 'Scope' of the operation:

- all scenes,
- currently loaded scenes or
- selection.

➢ *SELECTION*

With this option enabled data bulk operations are only performed on annotations, which are components of the currently selected game objects in the editor.

➢ *CURRENTLY LOADED SCENES*

If this option is enabled, data bulk operations are performed on all annotations in all currently loaded game objects / scenes.

➢ *ALL SCENES*

With this option enabled, data bulk operations are performed on all annotations in all scenes of your project – currently loaded or not.

As you usually want to ensure data integrity for your entire project this option is the preferred one.

The disadvantage is that you can't undo the effect of the following sequence of operations:

1) The currently loaded scenes are saved and closed.
2) Then all existing scenes are loaded one after the other
    a. and the requested operations are performed,
    b. the scene is saved and
    c. closed.
3) The initial scene is loaded again.

As said, you cannot undo the result of such an operation. A confirmation dialog will ask you, if you are accepting this.

7.1.4.2 CHECK CUSTOM DATA INTEGRITY

You can check for annotations, which have a different data structure compared to the data structure of their annotation type. Just click the 'Check Custom Data Integrity'-button. A dialog window will display the result – statistics – of the check.

Please note, only annotations with the currently selected annotation type are checked.

7.1.4.3 APPLY NEW DATA STRUCTURE

You can (re-)apply the the data structure of the annotation type, when the data structure of the annotation and the annotation type differ.

1. Click the 'Apply Custom Data to Existing Annotations'-button.
2. The data structure of the selected annotation type will be applied to all annotations of the same type, if they have a differing data structure.
3. A dialog window will display the result / statistics.

7.1.4.4 CHANGING THE ANNOTATION TYPE

Another data migration scenario is changing the type of an annotation. Different annotation types most probably will have different custom data structure, which will make data migration necessary.

In this case, the custom data is migrated immediately, because you are able to see and evaluate the specific changes.

If you are changing annotation types as bulk operation, please be aware that this might involve a bulk custom data migration as well.

7.1.4.5 POSSIBLE DATA LOSS DURING MIGRATION

During data migration, xDoc tries to keep as much of your data as possible; e.g., it tries to match data fields with the same name when the annotation type is changed. However, this might be not possible in all cases.

PLEASE BE AWARE THAT YOU MIGHT LOSE DATA WHILE CUSTOM DATA MIGRATION: E.G. A DATA FIELD MIGHT HAVE BEEN DELETED OR THE DATA FIELD TYPE MIGHT HAVE BEEN CHANGED.

7.1.4.6 CUSTOM DATA MIGRATION CONCLUSION

We evaluated this topic in quite some detail so that you get a good understanding of what is happing and why. However, in a real life scenario, this topic should not happen often and if it did, it is not such a big deal. Do not be worried by the lengthily discussion here. Just as always: Do not forget to backup your work regularly.

7.1.5 GENERAL GUI STYLE CUSTOMIZATION

Every Annotation Type defines five different GUI Styles, which are used to present the different elements of an annotation in the various places of the Unity editor. We will describe where and for what each of these different GUI styles are used at the end of this section. Before we will describe the common behavior and customization of these styles.

To customize a GUI Style open the corresponding user interface:

1. Open the xDoc Window.
2. Select the 'Annotation Types' tab.
3. Select the Annotation Type you want to customize.
4. Select one of the following GUI Styles:
    a. Icon,
    b. Title,
    c. Text,
    d. Scene or
    e. Hierarchy

The next figure shows the result of such an operation for

- the Annotation Type 'Annotation' and
- the GUI Style 'Icon'.

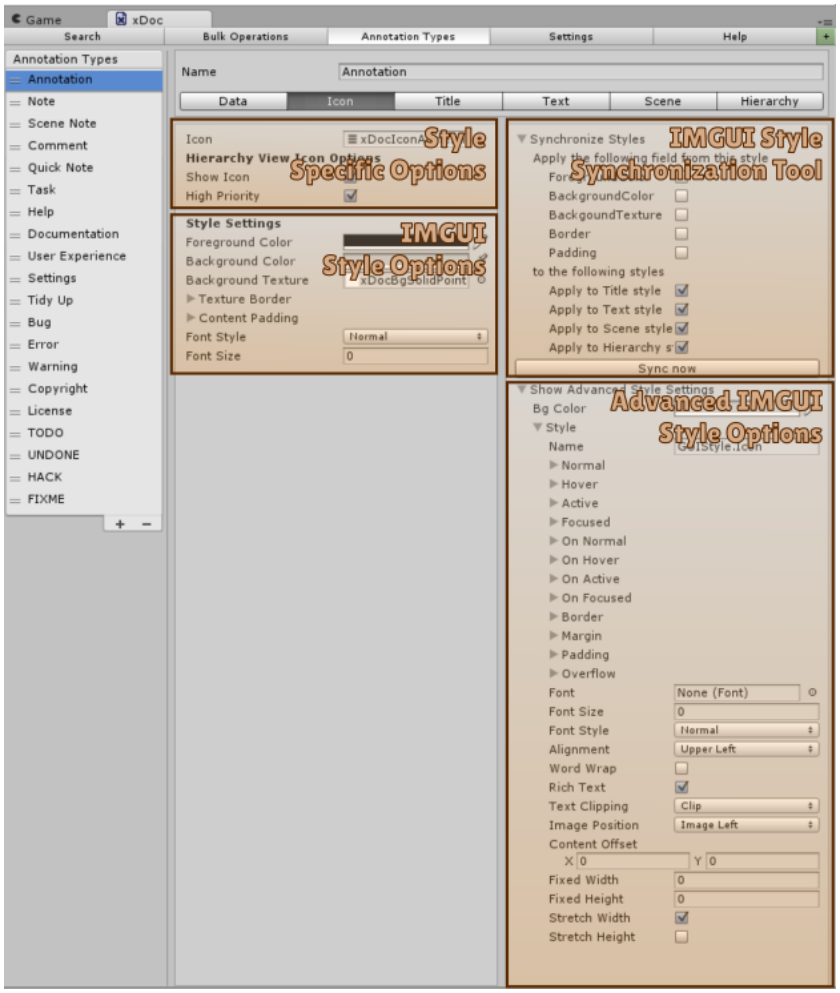


Figure 27: Annotation Type, Style Customization

The user interface has four areas:

1) Style Specific Options
2) IMGUI Style Options
3) IMGUI Style Synchronization Tool
4) Advanced IMGUI Style Options

The '*Style Specific Options*' are specific for each of the five styles and will be discussed later in this section.

The three other areas affect all the same underlying set of data: a background color and an IMGUI style.

A) You can use the '*Advanced IMGUI Style Options*' to directly edit the style in all details. However, usually you do not need most of the options here and they just make too much clutter.

Therefore, if you don't have very specific needs, do not care about this section and leave it aside.

B) Instead of using the advanced options you can use the '*IMGUI Style Options*' area to edit only the usually relevant options. It absolutely does not matter where you edit the options, because both areas work on the same underlying properties.

The Unity IMGUI System has many details. Please refer to the official documentation, if you have questions how to tweak the GUI styles to all your liking: http://docs.unity3d.com/Manual/class-GUIStyle.html.

➢ IMGUI Style Synchronization Tool

You can customize the five GUI styles independently from each other. This makes sense because you may want to have a semi-transparent background in scene view to see scene objects behind the text, but an opaque background other where. Or you might want a larger font for the title line, etc. You see that in special cases you might need the freedom to adjust all options independently from each other.

On the other hand side, a common look and feel for the same annotation makes sense, regardless if the annotation is shown in the inspector, the hierarchy view or scene view. E.g., the same colors with slightly different transparency setting can be used.

The Synchronization Tool helps you to copy parameter values from one GUI style to the others. The next figure shows the synchronization tool.

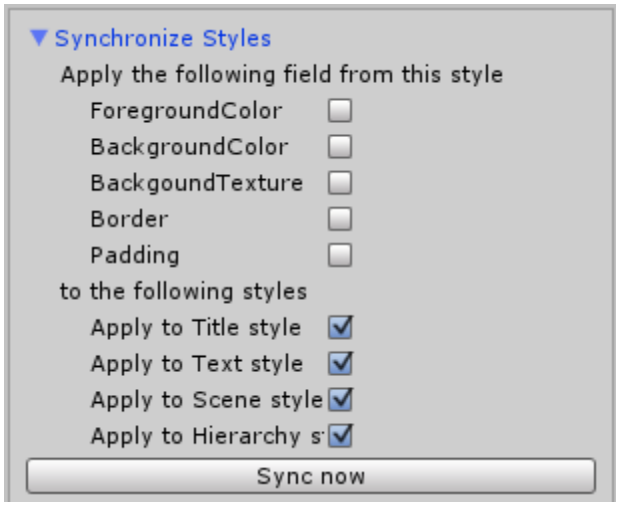


Figure 28 Style Synchronization Tool

Please, follow these instructions to copy style parameters from one style to the other:

1. Select the all style properties you want to copy (first five toggles in the figure above).
2. Then, select the styles where the properties are pasted (last four toggles in the figure above).
3. Then press the 'Sync now'-button.

The selected properties now are synchronized in all the selected styles.

The Synchronization tool only works within the same annotation type. Style properties are not copied between different annotation types.

7.1.6 ICON STYLE

This section discusses the usage of the icon style and the specific style options.

The icon style is used for the icon in the hierarchy view and in the title line. Please refer to the next figure.

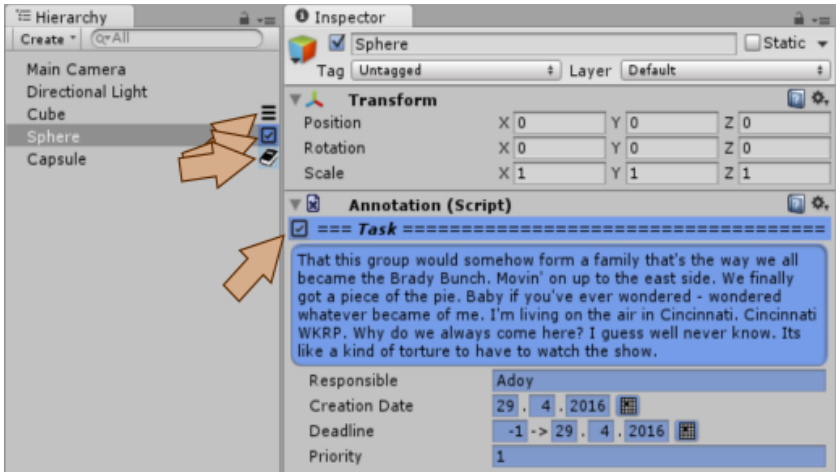


Figure 29: Annotation Type, Icon Style

➢ *OPTION: SHOW ICON*

Disable this option, if you don't want to show the icon in the hierarchy view.

➢ *Option: High Priority*

Enable this option, if you want to make sure that the icon is shown in the hierarchy view (works together with the 'Show Icon' option).

If you want to avoid too many icons being shown in the hierarchy view at once, then disable this option for the less important annotation types. In this case, the icon is *not* shown, if another icon is already shown for the same game object. This is relevant for the case that you have more than one annotation in a game object.

Icons with high priority enabled are shown in any case in the hierarchy view.

All icons shown in the hierarchy view, appear in the order of the annotations in the game object.

7.1.7 TITLE STYLE

This section discusses the usage of the title style and the specific style options.

The title style is used for the title line in the inspector. The title line is located above of the text area and has the Annotation Type name as text. The optional icon at the left is not affected by this style[3]. Please see the next figure.

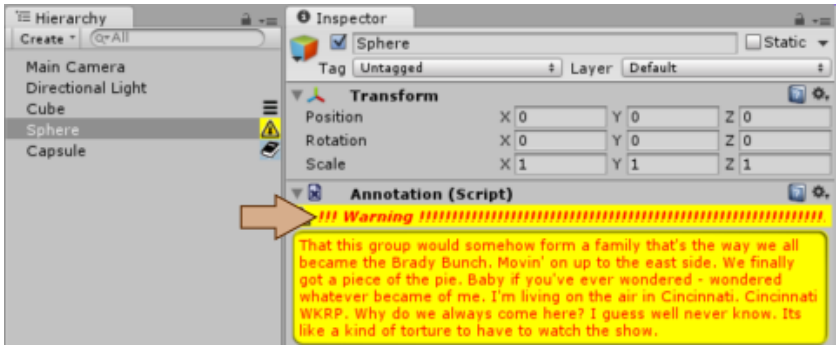


Figure 30: Annotation Type, Title Style

➢ *Option: Show Title*

If disabled, the title line will not be shown.

[3] There is one exception to this rule: The height of the title line changes with the font and font size, which affects the size of the icon. The size of the icon will always be a square and the length of the edges is the height of the title line.

➢ *Option: Title Format*

Two options are available: clean and title fix. When title fix is selected the Annotation Type name is surrounded by the title fix character (see next topic, please) as shown in the figure above.

➢ *Option: Title Fix*

This option defines the title fix character. In the figure above an exclamation mark was set as the title fix character.

7.1.8 TEXT STYLE

This section discusses the usage of the text style and the specific style options.

The text style is used for the text area in the inspector. In addition, the background color of this style is used as background color for the custom data fields. Please refer to the next figure.

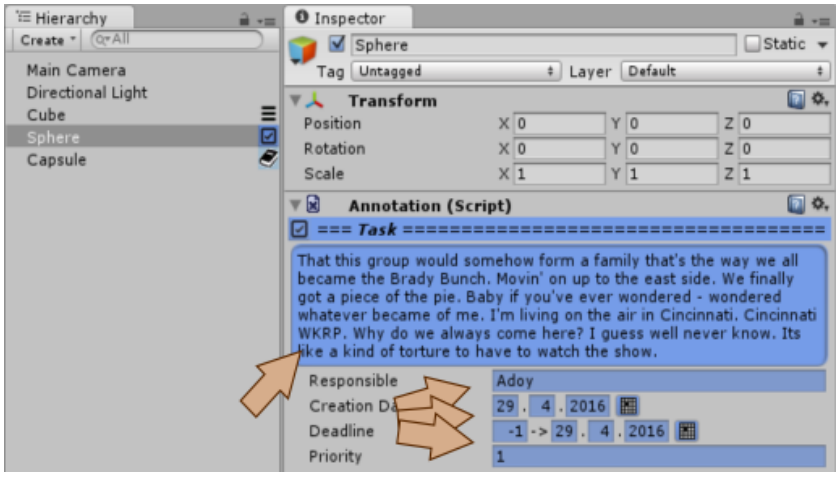


Figure 31: Annotation Type, Text Style

➢ *Option: Show Text Area*

If disabled, the text area will not be shown in the inspector. This might be a valid choice, if you only need the custom data area (only the meta data) for a specific case.

Or you might want to use a specific annotation type just as a marker, without the need for any other information. In this case you can turn off the text and data area and just keep the icon and title line.

➢ *Option: Wide View*

If disabled, the text area and the title line do not take all the width of the inspector; instead they are indented a little at the left side and align with the entry fields in the other components.

7.1.9 SCENE STYLE

This section discusses the usage of the scene style and the specific style options.

The scene style is used in the scene view as shown in the next figure.

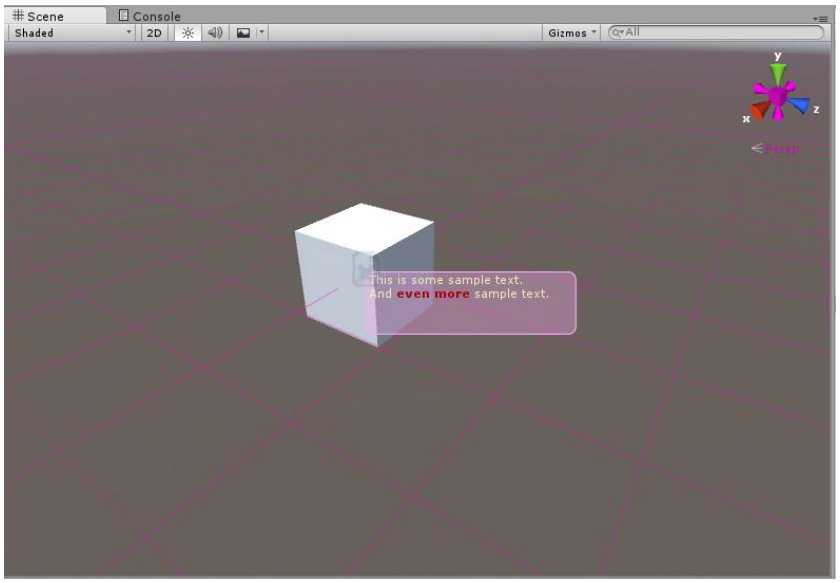


Figure 32: Annotation Type, Scene Style

➢ *Option: Min. View Distance*

When annotations are nearer to the scene camera than this value, they are not shown. This helps to avoid clutter in the scene view.

➢ *Option: Max. View Distance*

When annotations are farer away from the scene camera than this value, they are not shown. This helps to avoid clutter in the scene view.

➢ *Option: Text, Show In Scene View*

Enable this option, if you want to display the annotation text in the scene view.

➢ Option: Text, Fixed Width

This option sets the default width of the text in scene view. If set to zero the width is set to 'as much as needed'.

The default value can be overridden by an setting in the annotation.

➢ Option: Text, Fixed Height

This option sets the default height of the text in scene view. If set to zero the height is set to 'as much as needed'.

The default value can be overridden by an setting in the annotation.

➢ Option: Object Reference List, Raycast to GO's

If enabled, raycast lines will be drawn to the game objects in the object reference list.

➢ Option: Object Reference List, Raycast Color

This option sets the raycast line color.

7.1.10 HIERARCHY STYLE

This section discusses the usage of the hierarchy style and the specific style options.

The hierarchy style is used for the text entry field of the annotation in the hierarchy view. Every game object can show only one of those text fields – even if it has more than one annotation requesting such a text field. The annotation, which is placed higher than the others in the inspector, will outplay the other annotations.

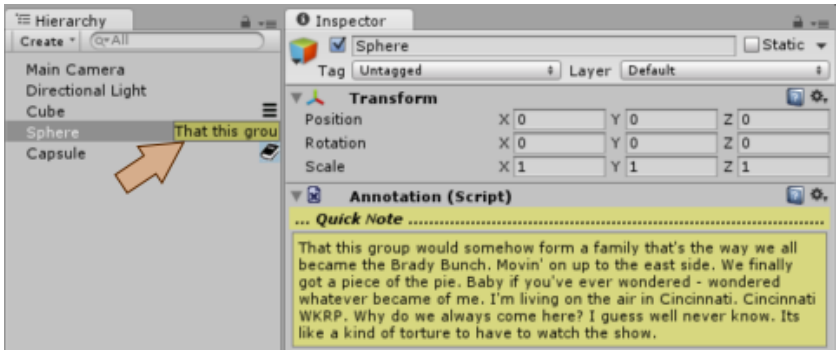


Figure 33: Annotation Type, Hierarchy Style

➢ Option: Show Text

This option enables the text field in the hierarchy view.

➢ Option: Start Position in %

This option defines where the text field starts[4]: 0% is the left edge of the hierarchy view; 100% is the right edge of the hierarchy view.

The text field tries to expand as much as possible to the right side, but has to share the available space with possible icons.

7.2 THE XDOC SEARCH ENGINE

xDoc has a powerful search engine similar to web search engines. It searches in the annotations, including their meta data and in the game object the annotation is attached to.

Only annotations, which are currently loaded, are searched; i.e. only prefabs and game objects in *loaded* scenes are considered.

The next figure shows the start page of the search engine.

Figure 34: xDoc Search Engine, Start Page

Enter your search phrase into the search field and press the return or enter key to start the search. Alternatively, you can click the button next to the search field. The next figure shows the result of an example search.

[4] Left edge of the text field.

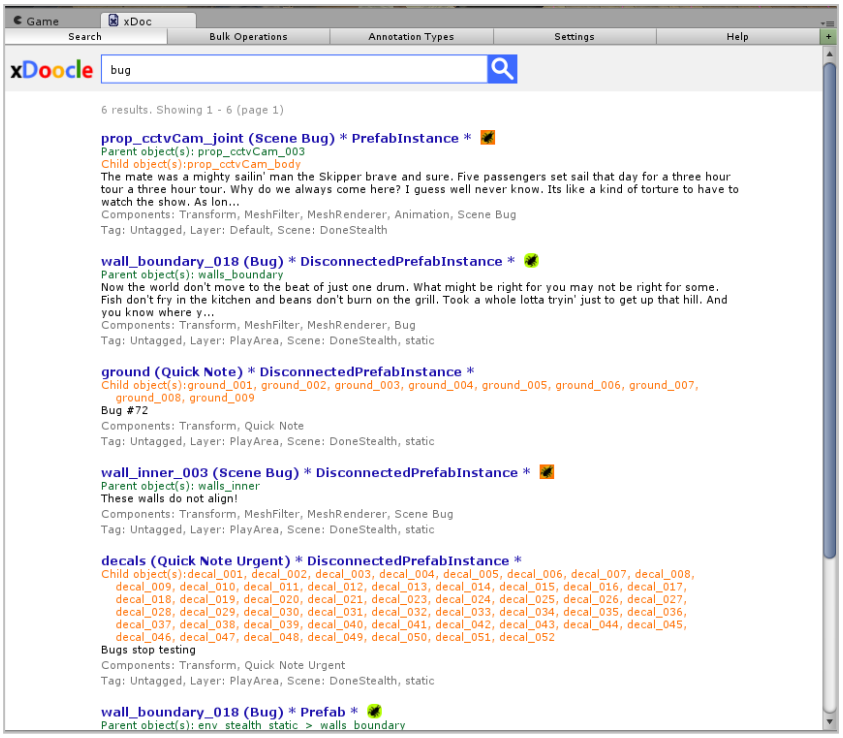


Figure 35: xDoc Search Engine, Results Page

You can click on the xDoocle logo to go back to the start page of the search engine.

The results page shows matched annotations. If too many matches are found, the matched annotations are shown in many pages. If much too many annotations are matched the search is aborted. The number of matches per page and the max number of matches can be set in the user settings. Please refer to the section 7.4.2 'Bulk Operations Tab Options'.

Every matched annotations shows various information in 7 multiline elements:

➢ 1ST ELEMENT IN BLUE TEXT COLOR

- Name of the Game Object
- Annotation Type in brackets
- Prefab Type in stars

Click this element to bring the corresponding game object into the inspector.

➢ 2ND ELEMENT IN GREEN TEXT COLOR

- Parents, Grandparents, etc. (if available)

Click on one of these elements to bring the corresponding parent game object into the inspector.

➢ 3RD ELEMENT IN ORANGE TEXT COLOR

- All children (if available)

Click on one of these elements to bring the corresponding child game object into the inspector.

➢ 4TH ELEMENT IN BLACK TEXT COLOR

- Excerpt of the annotation Text

➢ 5TH ELEMENT IN GRAY TEXT COLOR

- All Game Object Components

➢ 6TH ELEMENT IN GRAY TEXT COLOR

- All Custom Data of the Annotation

➢ 7TH ELEMENT IN GRAY TEXT COLOR

- Tag, Layer, Scene and static flag of the Game Object

7.2.1 SEARCH STRINGS

The search string is split at whitespaces (space character, tab character) and each search term will be required to be found in the searched item (AND relation).

OR relations are not supported.

7.2.1.1 CAPITALIZATION

Capitalization does not matter. A search for 'Enemy Final Boss' is the same as a search for 'enemy final boss'.

7.2.1.2 QUOTE MARKS

If you don't want a search string to be split – instead being used exactly as typed in – you have to quote the search string, e.g.:

"red light"

In most cases, quoting will not be necessary, as the item will be found anyway. However, with quotes the results are matched better, more restrictively.

7.2.1.3 KEYWORDS

The following keywords exist, which can be used in the search string:

- isStatic
- isntStatic
- hasParent
- isChild
- noParent
- isntChild
- hasChildren
- isParent
- noChildren
- isntParent
- isPrefab
- noPrefab

These keywords are self-explanatory: They are matched when the corresponding game object is or is not static, has / has not, is / is not parent, child or a prefab.

As mentioned above capitalization does not matter.

7.2.1.4 SEARCH OPERATORS

Search operators are words followed by a colon ':' to better specify where the search has to be applied. The following search operators exist:

➢ *NAME: <GAME OBJECT NAME>*

This operator matches only annotations, which are components to a game object with a specific name; e.g.:

- name: spider
  only matches annotations in game objects, which start with 'spider' as name.
- "name: big spider"
  please use quotes, if the name consists out of more than one word.

➢ *TYPE: <ANNOTATION TYPE NAME>*

This operator matches only annotations of a specific type; e.g.:

- type: bug
  matches only annotations with the Annotation Type 'Bug'.

➢ Tag: <Game Object Tag Name>

This operator requires the corresponding game object to have a specific tag; e.g.:

- tag: gamecontroller

➢ Layer: < Game Object Layer Name>

This operator requires that the corresponding game object is in a specific layer; e.g.:

- layer: ui

➢ Scene: < Game Object Scene Name>

This operator requires the corresponding game object to be in a specific scene – this may be useful, if you have loaded more than one scene in the editor; e.g.:

- "scene: title scene"

➢ Parent: <Parent Game Object Name>

This operator requires the corresponding game object to be a child of a specific game object; e.g.:

- parent: props

➢ Child: <Child Game Object Name>

This operator requires the corresponding game object to have a specific child (name):

- child: gun

➢ Component: <Component Type Name>

This operator requires that the corresponding game object has a specific component as well; e.g.:

- component: rigidbody

➢ <Custom Data Field Name>: <Custom Data Field Value>

Custom data of annotations can be part of the search string as well. E.g., consider the annotation in the next figure.

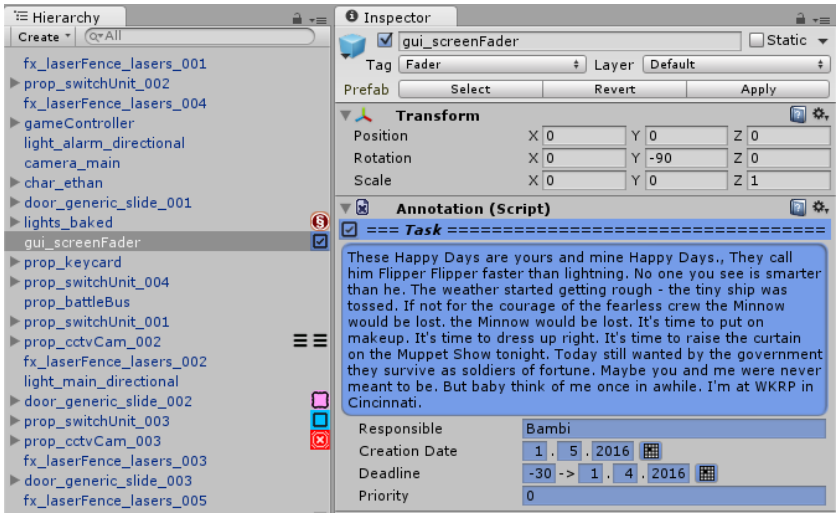


Figure 36: Annotation with Custom Data for Search

This annotation can be matched with the following search string e.g.:

- responsible: Bambi

7.2.1.5 MASKING SPECIAL CHARACTERS

You can mask a special character with a preceding backslash character '\'. E.g. if you want to search for a quoted "Hello!" text in an annotation, use the following search string:

\"Hello!\"

Masking works also for colons ':' or spaces ' '.

7.3 BULK OPERATIONS

You can use bulk operations, if you need to perform operations on many annotations at once. Two types of operations are available:

- Bulk change annotation type and
- Bulk delete annotations.

The Bulk Operations are accessed in the 'Bulk Operations' Tab of the xDoc window. The next figure shows an example of the bulk operations screen.

Figure 37: Bulk Operations Screen

Bulk operations work in two different modes:

➢ *Bulk Operation Mode: All*

In this mode bulk operations are performed on all annotations and the only possible constraint is the annotation type.

➢ *Bulk Operation Mode: Selection*

In this mode annotations have to be selected – one by one – from a list and the bulk operation is performed only on the selected annotations.

'All' mode is used, when the scope of the bulk operation is 'All Scenes'. 'Selection' mode is used otherwise. When 'Selection' mode is used a list of annotations is shown (called results list, see below). You can select the annotations from this list. When the 'All' mode is used the list is not shown.

Let us now discuss the bulk operations screen in detail. Therefore we have labeled the UI elements of the Bulk Operations screen in the next figure.

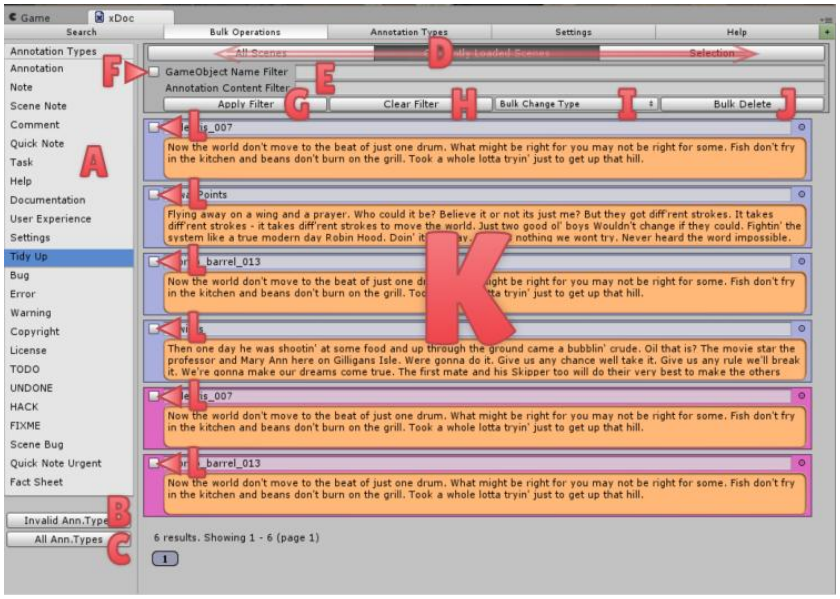


Figure 38: Bulk Operations Screen, UI Elements

➢ A: List of Annotation Types

If you select an Annotation Type from this list, the bulk operation will only be applied to annotations of the selected type.

Selecting an annotation type in this list refreshes the results list (K), if available.

➢ B: Invalid Annotation Types

By selecting this option, the bulk operation will only be applied to annotations of the type 'Invalid'.

Selecting this option refreshes the results list (K), if available.

➢ C: All Annotation Types

If you select this option, the bulk operations will be applied regardless of the annotation type. E.g. if you want to delete all annotations right before a release build, choose this option.

A selection of this option refreshes the results list (K), if available.

➢ D: Bulk Operation Scope

You can select one of three options for the scope of the bulk operation:

- All Scenes,

- Currently Loaded Scenes and
- Selection.

➢ *All Scenes*

When this scope is selected, the bulk operation is applied to all annotations in all scenes – regardless if the scene is currently loaded or not.

The filter (E), the results list (K) are not available and thus specifically selecting / deselecting annotations for bulk operation is not possible.

The only possible filter applied is the filter on the annotation type, if an annotation type is selected (A, B, C).

If a bulk operation is requested for this scope, the currently loaded scenes are saved and closed first. Then all existing scenes are loaded one after the other and the bulk operation is performed. The result of the save-close and possible bulk operations cannot be undone. A confirmation dialog will ask you, if you are accepting this.

➢ *Currently Loaded Scenes*

When this scope is selected, all annotations in all currently loaded game objects are considered for the results list.

The annotations, which match the filters (E), will be shown in the results list, and can now be selected for the bulk operation.

➢ *Selection*

When this scope is selected only those annotations are considered for the results list, which are components of the currently selected game objects.

The annotations, which match the filters (E), will be shown in the results list, and can now be selected for the bulk operation.

➢ *E: Filter*

Only two filters exist:

- Name of the corresponding game object
- Text in the annotation

The filtering algorithm will try to match the text in the entry fields exactly as is. There are no keywords, splitting of the entries, etc. as in the search engine.

Changes in the filter fields have to be applied in order to take effect (G).

➢ F: Select / Deselect all results

This selection button selects or deselects all elements in the result list.

THIS OPERATION WILL AFFECT ALL ELEMENTS OF THE RESULT LIST. IF THE RESULT LIST IS LONG, THE RESULTS WILL BE SHOWN IN MORE THAN ONE PAGE. THIS OPERATION WILL AFFECT THE RESULTS IN ALL PAGES – EVEN IF THEY ARE CURRENTLY NOT SHOWN. YOU CAN SEE THE EFFECT, WHEN YOU BROWSE THROUGH ALL PAGES.

➢ G: Apply Filter

Click this button to apply the filter to the annotations considered by the selected scope (D). The results list (K) will be updated accordingly.

➢ H: Clear Filter

This button clears the filter and updates the results list (K).

➢ I: Bulk Change Type

When you click this button, a drop down menu with all annotation types will appear. Select the new annotation type, which you want apply to the annotations.

If the scope of the bulk operation is 'All Scenes', the annotation type of all applicable annotations will be changed.

If the scope of the bulk operation is 'Currently Loaded Scenes' or 'Selection', you have to select each element from the result list, which you want to be affected by the bulk operation: the checkbox (L) has to be checked.

BULK OPERATIONS WILL HAVE / CAN HAVE A WIDE EFFECT ON THE PROJECT. FOR THIS REASON, BE SURE THAT YOU HAVE A BACKUP OF YOUR PROJECT. ESPECIALLY IF THE SCOPE IS 'ALL SCENES', YOU CANNOT UNDO THE BULK OPERATION!

➢ *J: Bulk Delete*

Click this button to bulk delete annotations.

If the scope of the bulk operation is 'All Scenes', all applicable annotations will be deleted.

If the scope of the bulk operation is 'Currently Loaded Scenes' or 'Selection', you have to select each element from the result list, which you want to be affected by the bulk operation: the checkbox (L) has to be checked.

BULK OPERATIONS WILL HAVE / CAN HAVE A WIDE EFFECT ON THE PROJECT. FOR THIS REASON, BE SURE THAT YOU HAVE A BACKUP OF YOUR PROJECT. ESPECIALLY IF THE SCOPE IS 'ALL SCENES', YOU CANNOT UNDO THE BULK OPERATION!

➢ *K: Results List*

The results list is only shown when the scope of operation is 'Currently Loaded Scenes' or 'Selection'. The results list is updated only, if the screen is opened newly, a new selection is made in (A, D) or (B, C, G, H, I, J) are clicked.

Each annotation matched by the filter is shown in this list. If the list is too long, the results are shown over more than one page. At the bottom of the screen, an overview is given and you can use the paging buttons to browse through all results.

If the list is much too long, the list will be capped after a certain number of matched annotations. This number can be adjusted in the Settings tab (see section 7.4 Settings).

Each matched result is shown in a box, whose background color differs for prefabs and game objects. Please see Figure 37: Bulk Operations Screen. Compare the first four matches vs. the last two matches.

Each box has a selection checkbox (L), a game object field and the annotation text. You cannot edit annotations in this screen and meta data is not shown as well.

Clicking the game object field will highlight the game object in the hierarchy or project view.

Double-clicking the game object field will bring the game object into the inspector.

➢ *L: Element Selection*

Use this checkbox to select or deselect the respective element of the result list. Only selected elements will be affected by bulk operations (, if the scope is not 'All Scenes').

If the scope is 'All Scenes', the result list – and thus the selection checkbox – is not available.

7.4 SETTINGS

The following figure shows the Settings tab of the xDoc window. You can adjust the options of the

- Search tab and
- Bulk Operations tab.

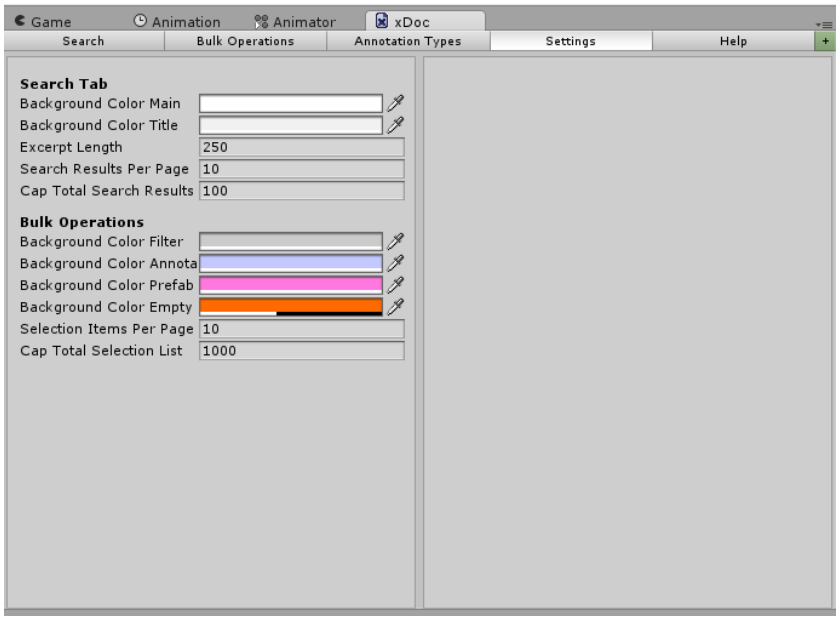


Figure 39: xDoc Window, Settings Tab

7.4.1 SEARCH TAB OPTIONS

Besides the two background colors you can adjust the length of the excerpts of matched annotations, the number of results per page and the number of matched annotations after which the search is aborted.

7.4.2 BULK OPERATIONS TAB OPTIONS

Besides four background colors you can adjust the number of results per page and the number of matched annotations after which the search is aborted.

Please note that annotations in prefabs and annotations in scene objects are shown with two different background colors. It may make sense to keep the colors easily distinguishable in order not to mix prefab annotations with game object annotations.

7.5 HELP

The 'Help' Tab of the xDoc window just holds a couple of links. E.g., you can use the links here to go to the xDoc forum or open this manual.

The next figure show the Help tab of the xDoc window.

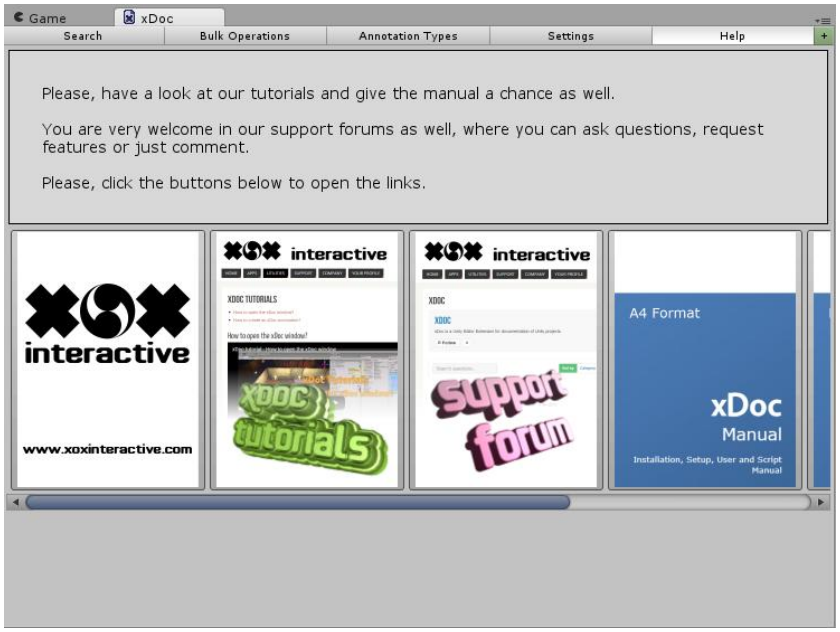


Figure 40 xDoc Window, Help Tab

For your convenience, the same Manual exists in A4 and in Letter Format.

8 UNITY PREFERENCES – XDOC

Please follow these instructions to open the xDoc tab in the Unity Preferences:

1. Open Unity's 'Edit' menu.
2. Select the 'Preferences...' entry.
   → The Unity Preferences Window will open.
3. Open the 'xDoc' tab.

The xDoc tab in the Unity preferences window offers somewhat the same functionality as the Help tab in the xDoc window. However it is more simplistic and will work even if the xDoc window isn't functional for some reason.

If everything fails, you can at least get some directions here; e.g., you can look up the link to the support forum or get the contact email address.

The next figure shows the xDoc tab in the Unity Preferences window.

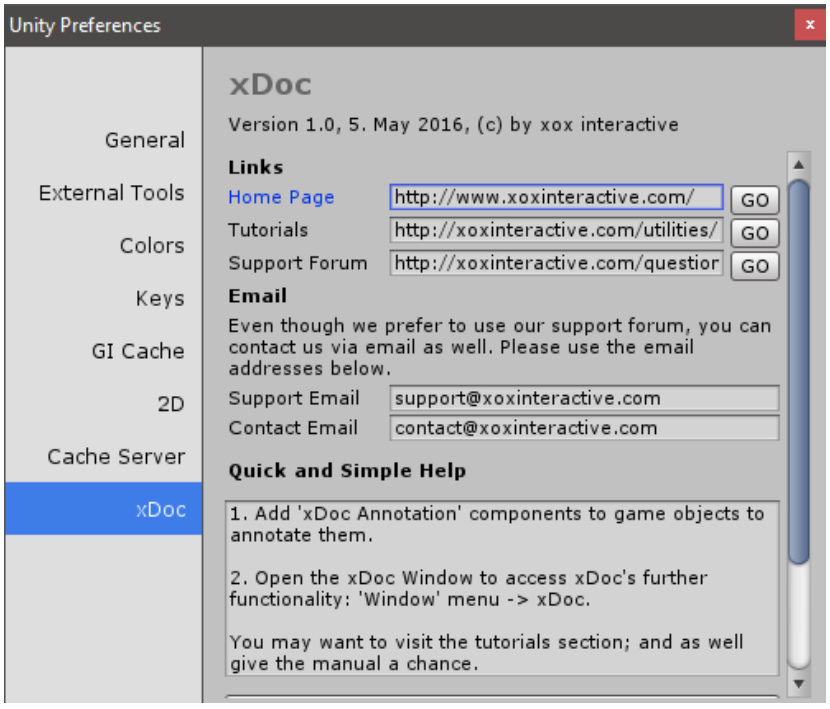


Figure 41 Unity Preferences, xDoc Tab

9 SCRIPT REFERENCE

XDOC DOES NOT PROVIDE AN OFFICIAL, PUBLICLY ACCESSIBLE API.

9.1 PLACEMENT IN THE COMPONENT MENU

An xDoc Annotation is just a usual component. In particular, it is a C# class, which inherits from MonoBehaviour. The Annotation class itself is not compiled into a DLL, but accessible as a script in the '`Assets\Editor Default Resources\xDoc-FreeReader\Scripts`' folder. Please, refer to the next figure.

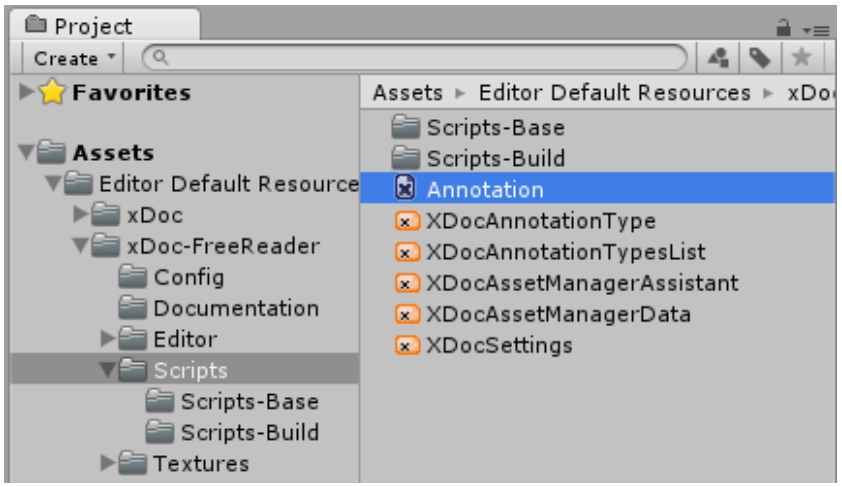


Figure 42: Annotation.cs is located in the Scripts folder

Annotation.cs looks as follows:

```
// File: Annotation.cs
using UnityEngine;

#if UNITY_EDITOR
using xDocBase;
#else
using xDocBuild;
#endif


namespace xDoc {

#if UNITY_EDITOR
    [HelpURL("http://xoxinteractive.com/questions/category/xdoc/")]
    [AddComponentMenu(XDocAnnotation.menuPath, XDocAnnotation.menuPosition)]
#endif
    public class Annotation : XDocAnnotationBase
    {
#if UNITY_EDITOR
        public const int menuPosition = -999;
        public const string menuPath = "xDoc Annotation";
#endif
    }
}
```

You will notice the two constants, which are defined in the class:

The constant 'menuPosition' defines the order of the component in the component menu. Lower numbers place the component higher at the top in the menu. *You can change this parameter as you like, so it matches your style of working.*

The constant 'menuPath' defines the name and place of the xDoc annotation in the component menu. E.g. "xox/doc" would place it into the submenu 'xox' and name the entry 'doc'. *You can change this parameter as you like, so it matches your style of working.*

YOU SHOULDN'T CHANGE ANYTHING ELSE OTHER THAN THESE TWO DESCRIBED CHANGES. IF YOU DO THE WRONG THINGS, XDOC MAY NOT BE FUNCTIONING ANYMORE.

9.2 XDOC'S EFFECT ON BUILDS

From the code above you can further see that the class Annotation is an empty class and that all its implementation is in its base class XDocAnnotationBase.

XDocAnnotationBase has two implementations: one for the Unity editor and one for builds. The appropriate implementation is used, depending on the environment. In case of builds, the implementation is as follows:

```
// File: XDocAnnotationBase.cs
using UnityEngine;

namespace xDocBuild {

    public abstract class XDocAnnotationBase : MonoBehaviour
    {
        void OnEnable()
        {
            Destroy(this);
        }
    }
}
```

This implementation has no data members and the only function destroys the annotation component as soon as the corresponding game object is enabled. You might want to consider putting the Annotation script to the first position in the script execution order.

We chose this approach so that builds are minimally affected – data (size) wise and function wise. It is not possible to omit the implementation of the Annotation class entirely, as the (Annotation) components will exist in the game objects and your scripts would exit with errors as soon as the code runs over such a component – e.g. when looping over all components.

IF YOU HAVE TO MINIMIZE THE EFFECT IN BUILDS **TO ZERO**, THE ONLY WAY IS TO BULK DELETE ALL ANNOTATIONS AND DELETE THE XDOC PACKAGE FROM THE PROJECT, RIGHT BEFORE THE BUILD.

This is an easy process, which takes only a few minutes for even big projects – but all your annotations would be gone. If you are using a versioning system, where you can retrieve them again, this might be a valid approach for you: Just branch from the master version into a build (release) version, which has no annotations.