Overview

Project Search and Replace (PSR) is a plugin for Unity's Editor which provides the ability to search and replace text, numerical data, images, materials, and other content on a project-wide basis.

Videos

Here is a quick introduction to the tool:

Here is a video playlist for all Project Search & Replace video tutorials:

https://www.youtube.com/watch?list=PLBUqXH-FZtWkyDU0tKksA_oo_9RtrQGNG&v=0qdMWRnqKOY

Product Comparison

There are two versions of the tool: Project Search and Project Search & Replace. Project Search is the free version of the product with some restrictions. Upgrading to Project Search & Replace gets you the following additional features:

- Search and *replace* values: scripts, prefabs, materials, text, images, and more.
- Search a specific location: A folder, prefab(s), scene(s), object(s), or the current selection.
- Chain searches together with subsearches to filter your results.
- Make 'Conditional' searches where the value matches if it is equal, does not equal, or is any valid value.
- Search dependencies of items in the current search scope.
- Search and replace while the application is playing to update the existing scene at runtime.
- Copy search results to the clipboard.

This documentation covers both versions.

Caution: Use version control or have a backup.

With great search and replace comes great responsibility. The intent of this program is to allow you freedom in searching and replacing content on a *project-wide* basis. This means you have the freedom to completely break your project in unique and unexpected ways.

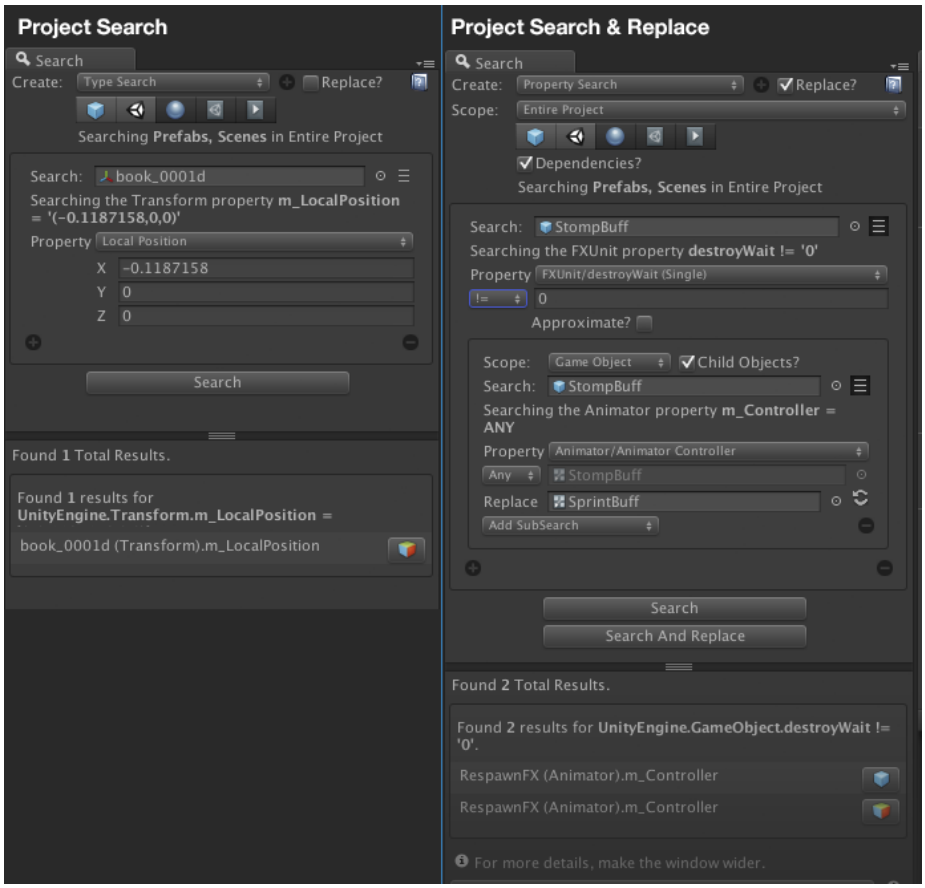

Figure 1: Illustrating some of the different options available in the demo vs. full versions.

Requirements

PSR currently requires Unity 5.1.2 or later. PSR contains two sub-packages depending on the version of Unity you are using. For more info: http://enemyhideout.com/2016/08/project-search-replace-1-2-1-update/

Installation

- Open Unity.
- Go to `Assets -> Import Package -> Custom Package...`
- Navigate to the location of the `unitypackage` file.
- Click 'Import' to add the files in the package.
- There will be two sub-packages added. Double-click the version that matches your Unity version to install the package'.
- Click 'Import' to add the fiels in the package.

The Search & Replace window should now be available under `Window -> Search + Replace` or by pressing Cmd/Ctrl-Shift-F.

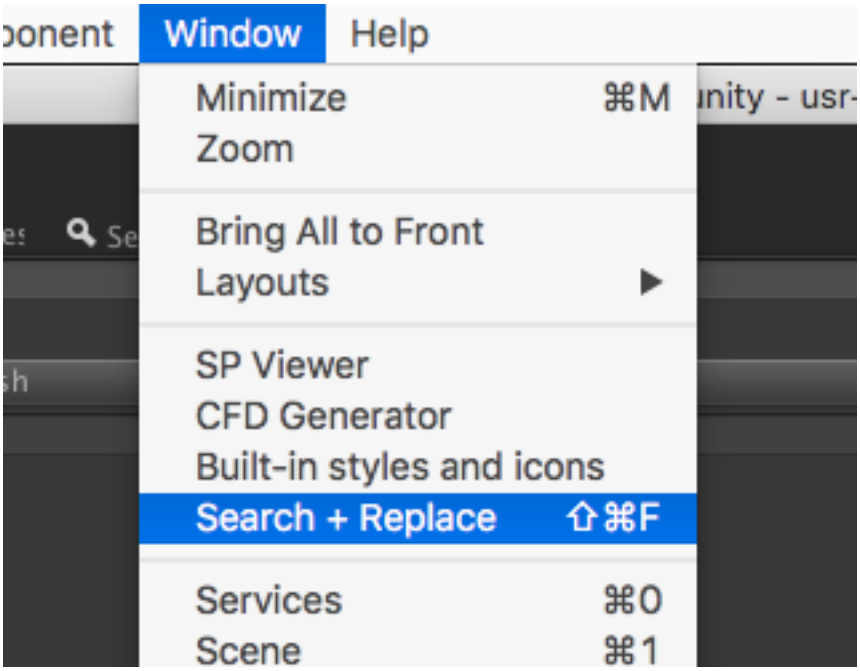

Figure 2: Search & Replace Menu Item

Getting Help

You can e-mail me at chris@enemyhideout.com

The help files you are currently reading are available:

- Online: http://support.enemyhideout.com/searchandreplace
- As a PDF: ProjectSearchReplace.pdf
- As a downloadable HTML zip: ProjectSearchReplaceDocs.zip

The Basics

The Search Window

The Search Window provides you with the ability to search and optionally replace content. The window attempts to provide both a Compact and Horizontal layout, with more information shown in the Horizontal format. Search information is saved to disk so that your search will stick around between compiling and restart.

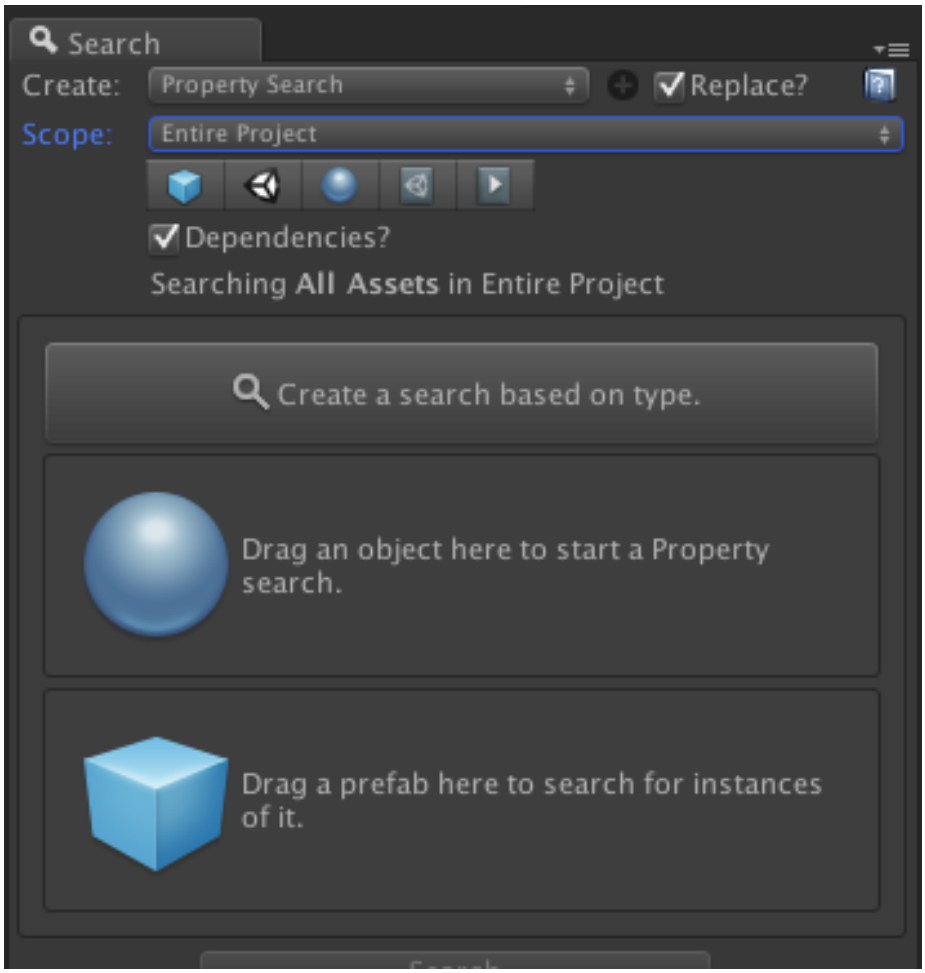

Figure 3: Search & Replace Start Window

When the window is first opened, the window will prompt you to create a 'search item'. You can create a new search item through the `Create` dropdown on the top, or one of the controls shown.

Search Items

There are three types of search items: Property, Type, and Prefab Instance. Property Searches allow you to granularly search inside one property of an Object. For example, you can search for the number 42 on the position of a Transform. Type searches search over everything. For example, you can search for the number 42 in every Unity Object. Prefab Instance searches search and replace instances of prefabs.

Type-based Searches

When searching you first create one or more Search Items by picking the type of search you want from the `Create` dropdown.

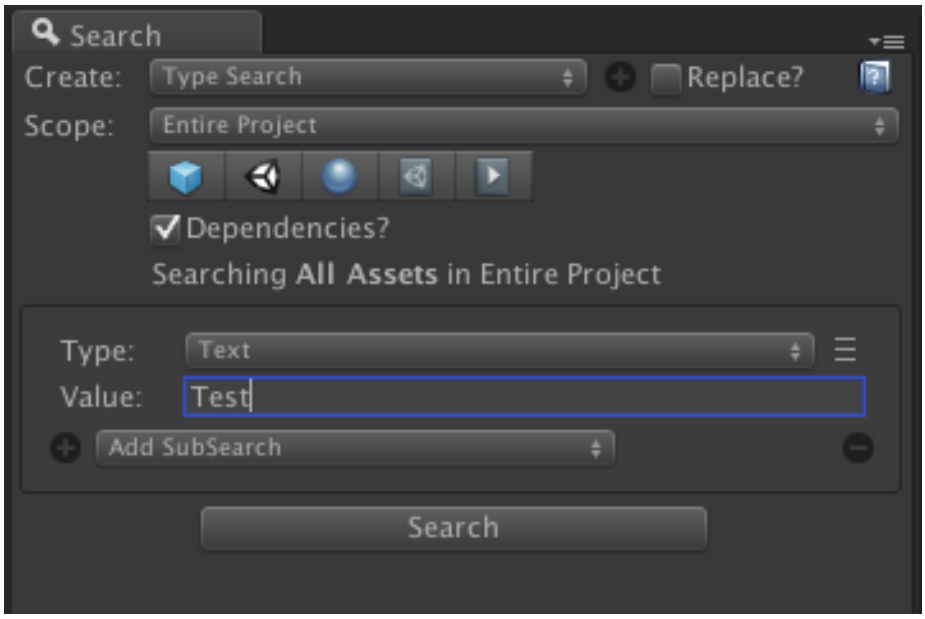

Figure 4: A Global Text Search

The above image will search for the string 'Test' in all properties of all game objects. Pressing the Search button will iterate over all game objects and provide a list of results.

Results for each search show you the full path to the object found in the format:

`Folder1/Folder2/Asset.ext/Object1:SubObject2.MonoBehaviourName.property.subproperty1`

A button with a GameObject or Prefab icon is shown to the right of the result. Clicking this button will highlight the object in the Inspector and select it in

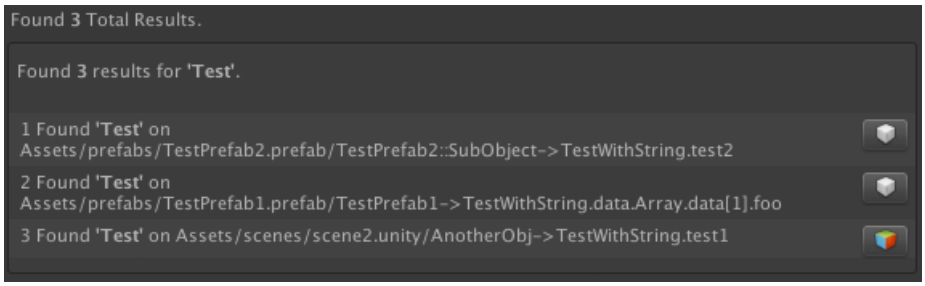

Figure 5: Global Search Results

the Project tab. If the object is a GameObject in a scene it will open the scene and highlight the object.

If the search result's object is a modified instance of a prefab it is more difficult to select the GameObject. In this case the tool attempts to guess the object as best it can.

Property Searches

It is possible to search inside a specific property on a MonoBehaviour, Component, Material, AnimationClip and other Unity Objects.

To search within a specific property on an object:

- In the `Create:` dropdown select 'Property Search' and click the `+` next to it.
- A Property Search item will be added to your search items:
- The 'Object' field is used to find the property to search for.
  - This is not the object that you are searching for.
  - It is not searching only inside this object.

In order to search a Property, drag an object from the Project or Inspector windows into the Object field.

Upon choosing the object you can then choose the property you are interested in searching inside. Depending on the content you are searching, different controls will show up. The search will initialize to the values of the object you dragged in. For example if you drag a Transform in, it will default to the localPosition value, and input the values for the position into the box automatically.

Searching Built-in Components

Unity's built-in Components save and display data differently depending on the object. This makes it difficult to search certain things. For example, the Albedo

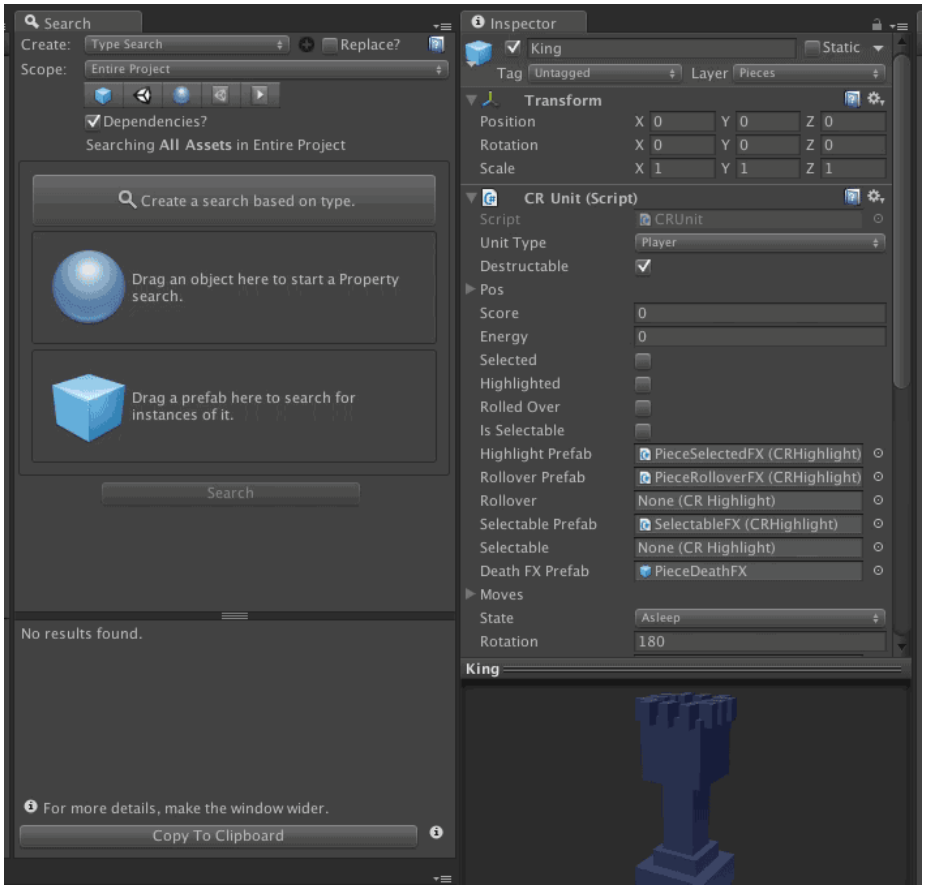

Figure 6: Dragging a GameObject will allow you to choose from any property on any Component within the game object.

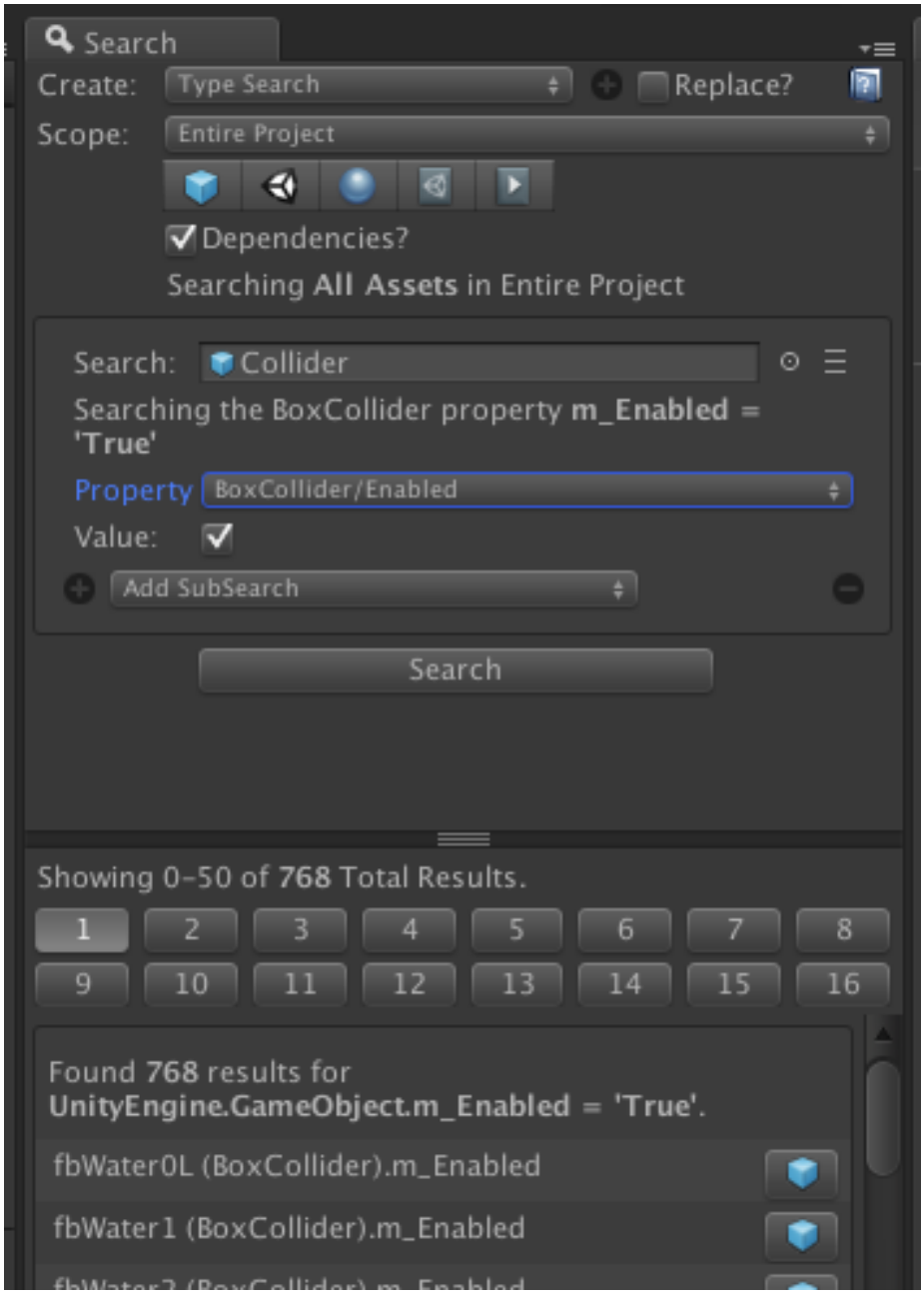

Figure 7: Example: Searching for all enabled Box Colliders.

texture inside a material is not called `Albedo` but instead saved as a property called `m_SavedProperties.m_TexEnvs.Array.data[0].second.m_Texture`. A lot of effort has been made to provide a better experience searching built-in Components but each Component is different and not all properties are easily searchable.

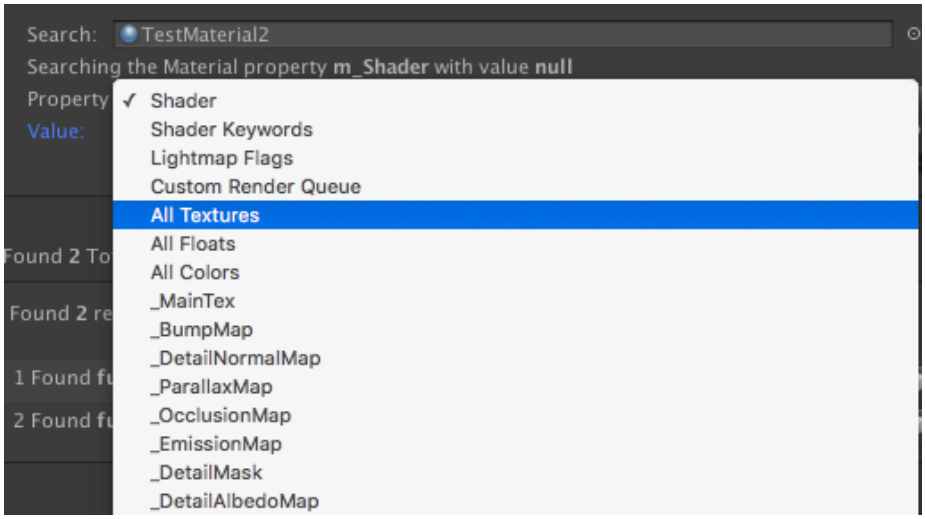

Figure 8: Common built-in Components have specially built searches to facilitate searching these items. For example you can search a specific texture slot or 'All Textures'

Cancelling a Search

Searches are cancellable by clicking the 'X' in the dialog as the search progresses.

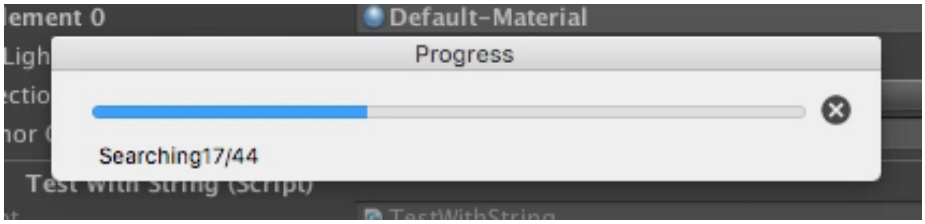

Figure 9:

Extended Search

Searching Custom Data Types in Properties

It is quite common to create special classes in Unity to serialize data. For example:

[System.Serializable] public class CRSquare { public CRUnit unitPrefab; public CRUnitType unitType; public IntVector2 pos; }

And used like:

```
public CRSquare[] squares;
```

Classes like this are not primitive data types and are searched differently. When selecting the property a 'subtype' field will display. This sub-type is one of the known primitive types that can be searched. This will search all data within the object's type recursively. For example if you search for text, and your data type has an array of an array of an array, it will search as deep as the rabbit hole goes.

Searching Multiple Properties

It is possible to search multiple items simultaneously. The + button under each search item allows you to quickly duplicate an existing search item and modify it.

About Searches

What Is Searched?

Searched:

- Prefabs (.prefab)
- Scenes (.unity)
- Materials (.mat)
- ScriptableObjects (.asset)
- AnimationClips (.anim)

Not Searched:

- Code (.cs files or .dlls) are not searched.
- Any text assets.
- Shaders (compiled or not) are not searched (Shaders *can* be searched *for*).
- Textures (Textures *can* be searched *for*).

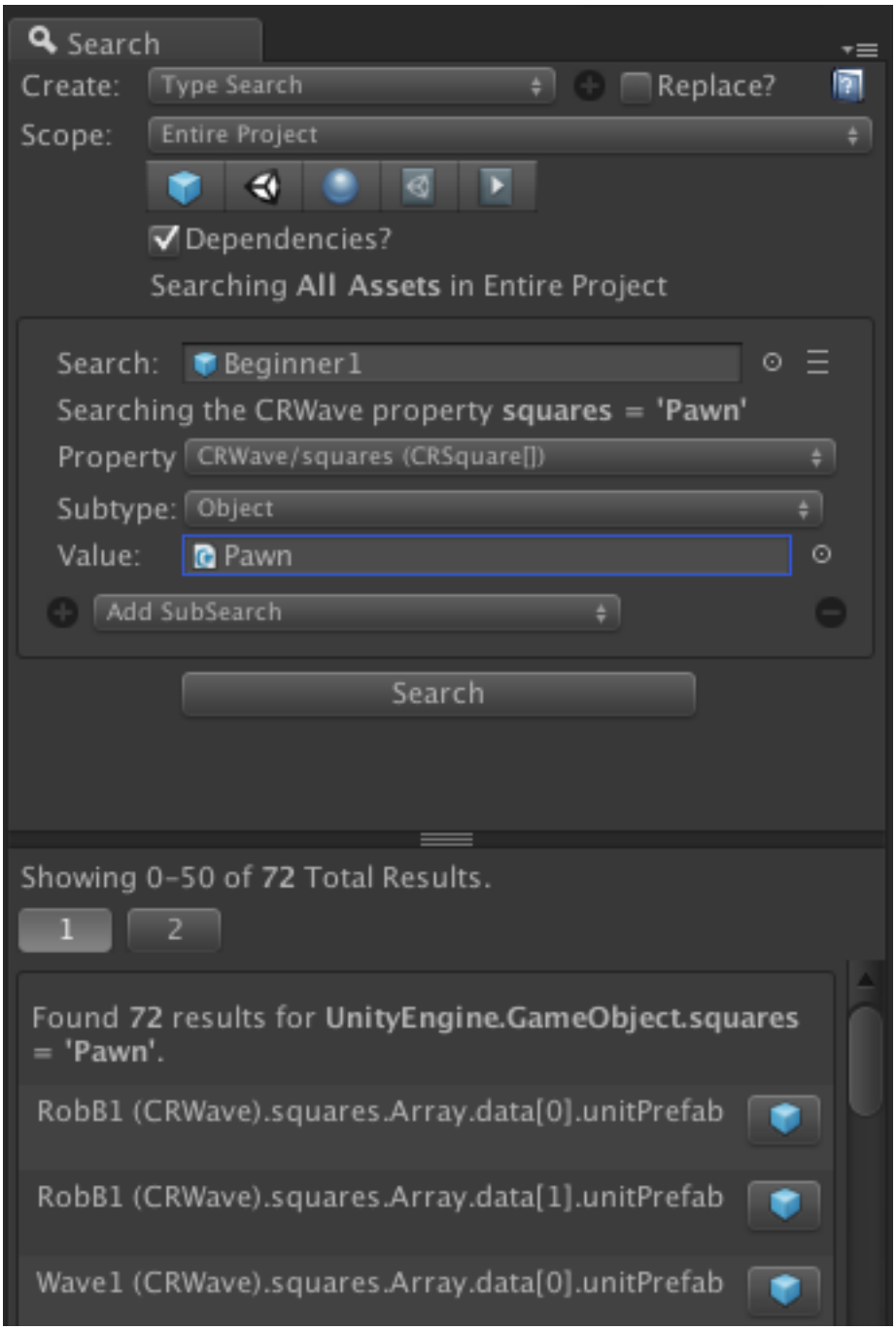

Figure 10: Example: Searching inside the squares property of the CRWave class. The tool has found 72 usages of the CRUnit instance 'Pawn'

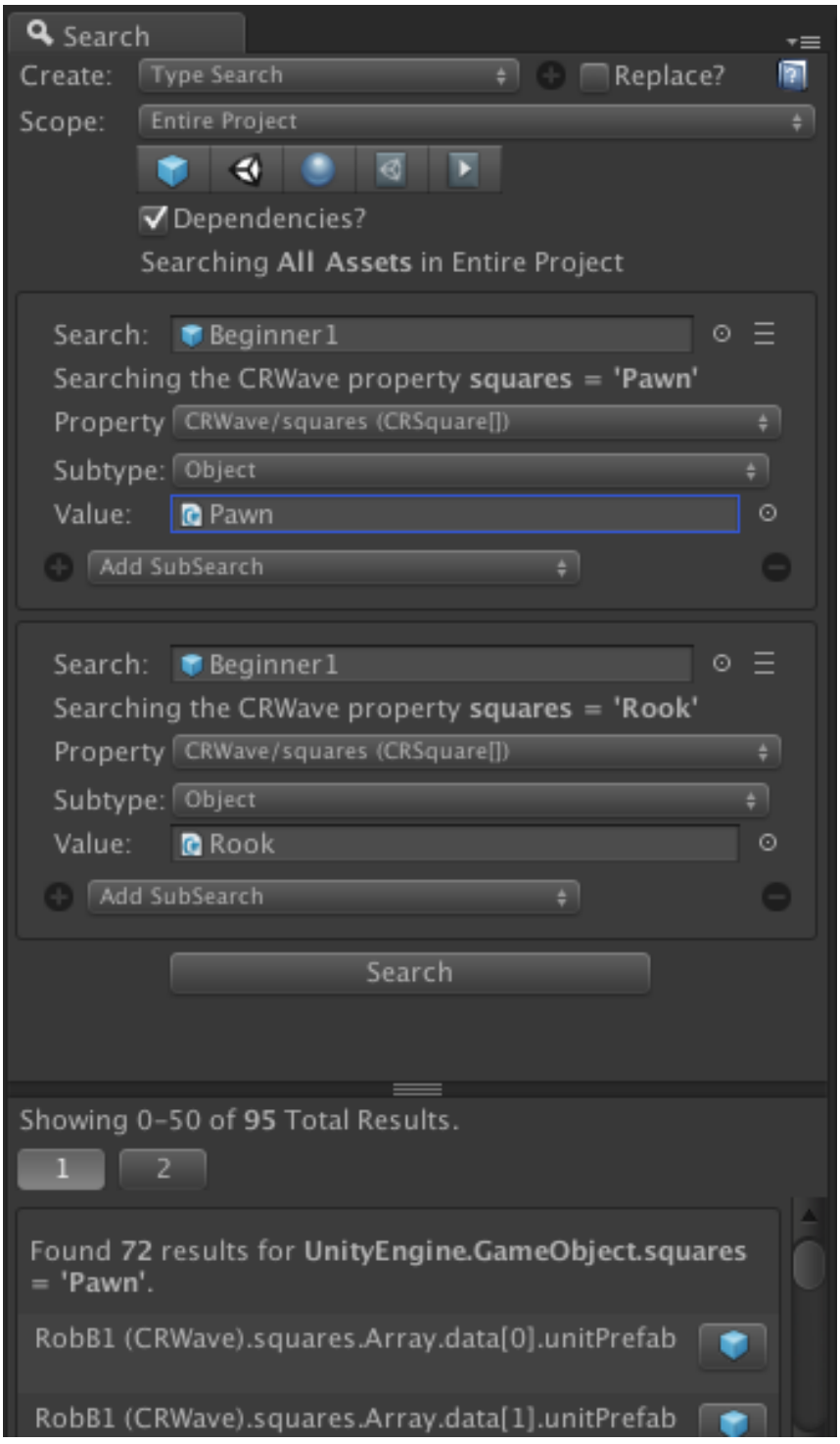

Figure 11: Example: Duplicating the previous search and searching for the value 'Rook' in addition to 'Pawn'.

What is a 'Unity Object'?

Most items in Unity inherit from UnityEngine.Object: GameObjects, Textures, Materials, Animations, Shaders, and many more. Properties on these items can be searched, in addition to the objects themselves being searched for.

Search Limits and Search Performance

Care has been taken to ensure that references to content are not saved as part of the Search Result. Search Results contain only strings and integers. Unity's internal resource loading/unloading mechanisms are used to load and unload assets, and no direct pointers to objects are saved as part of searching. The tool uses numeric IDs and guids to identify resources for load.

Pagination is used when the results reach a certain threshold. This ensures that Unity's UI is not slowed down displaying a large number of results and you can easily reach the end of the results.

The search system will warn you after finding 10,000 items. You may then cancel out or continue with the search. The tool has been tested to function with over 200,000 results, but the UI will make it difficult to page through.

Searching can load textures and other data into memory while searching. Testing has been done with large textures to make sure that the application is stable. If a large number of textures is being loaded/unloaded it can slow down your search. As a workaround you can change the scope of your search to speed up your search, and/or search specific asset types.

Scenes

When searching inside scenes the current scene will be closed before opening the new scene. The tool will display a prompt if there are changes to the current scene.

Search Options

More Options

In the upper right of each Search Item `Options` toggle can appear. When it is active additional options are shown. For example the Vector3 Type Search displays the following extra controls:

The `Approximate?` checkbox defines whether you want floating points to be exact matches or use Unity's built-in Mathf.Approximately() functionality.

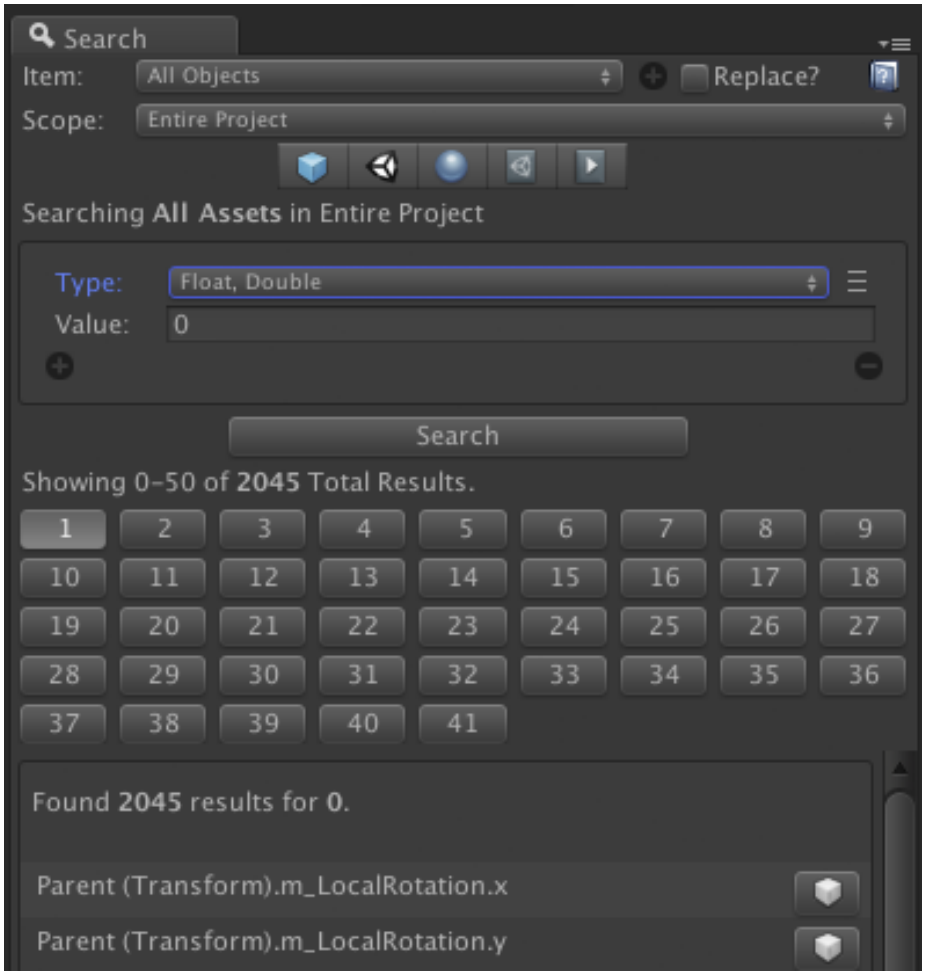
Search
Item:
All Objects
Replace?
Scope:
Entire Project
Searching All Assets in Entire Project
Type:
Float, Double
Value:
0
Search
Showing 0-50 of 2045 Total Results.
1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18
19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36
37 38 39 40 41
Found 2045 results for 0.
Parent (Transform).m_LocalRotation.x
Parent (Transform).m_LocalRotation.y

Figure 12:

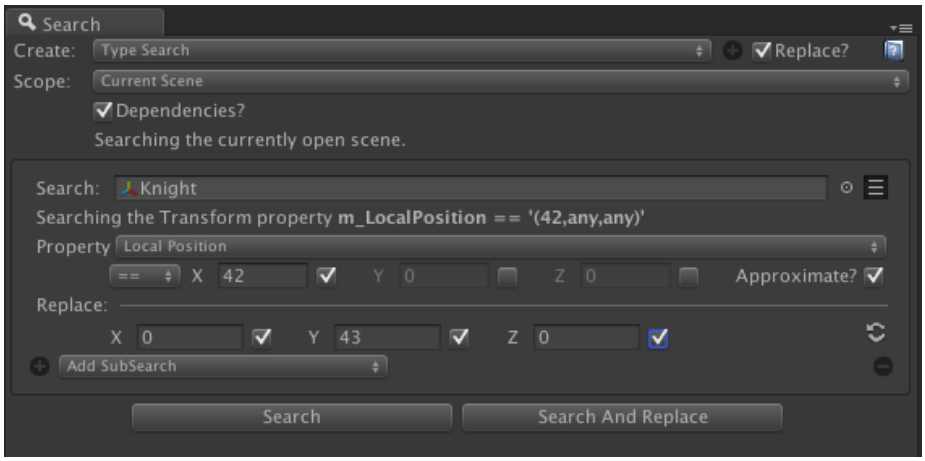

Figure 13:

The checkboxes next to each field defines the fields we are interested in searching for. In the above example we only care if the `x` of the Vector3 is approximately 42, and we then set all values to (0,43,0).

Text Search Options

When searching with text there are additional options available.

- Ignore Case - Searching does not pay attention to case. For example 'foo' will match 'foo', 'FOO', and 'Foo'.
- Contains - Search inside of the text, instead of matching the text exactly. For example 'foo' will match 'foo', 'foot', and 'bigfoot'. Multiple matches are supported.
- Regex - This is for advanced users. Regex uses the built-in regex of Mono. Please see Microsoft's documentation here.. Grouping is supported. For example you can search for `^(/d{3}) is before (/d{3})` and use `$2 is not before $1` to replace '123 is before 456' with '456 is not before 123'.

Conditional Searches

Sometimes you may want to search for all values that do *not* match a specific value, or search for *any* value to view all values used. Conditional searches help accomplish this. To create a conditional search:

- Create either a Property or Type search item.
- Click the `More Options` button.
- The conditional search control should display.

- Choose one of the options:
  - == matches if the values are equal (the default functionality).
  - **!=** matches if the values are not equal.
  - **Any** matches all values.

Special Considerations For Conditional Searches

For the most part conditional searches work as you might expect. But there are a couple advanced scenarios that do not.

- It is not a good idea to execute a Type Search and replace with `!=` or `Any`. It will most likely break your project. You may want to reduce your search scope or use a property search.
- Special care should be used when doing a text search with either 'Contains' or 'Regex'. Unique rules apply when using `!=` and `Any`. These searches rely on a 'match' being returned from the search, but in these instances the 'match' returned will be the entire string value. For example: searching for the string 'hip' Contain'ed in 'hip ship' and replacing with 'hop' would *usually* yield 'hop shop'. But when `!=` or `Any` is used no match for 'hip' would be found, instead the entire string value will simply be replaced with 'hop'.

Subsearches

Subsearches provide the ability to chain searches together and filter results further. For example it is possible to search for all 18 pixel tall text, then only text that is yellow. To add a subsearch:

- Create a Property or Type search.
- At the bottom of the search item click `Add Subsearch`.
- Choose the subsearch item type you are interested in.

When searching the top-most search item is executed and if a match is found, then the sub-search executes, and so on. If all searches return a match then a search result is found.

Using Subsearches with Conditionals

Subsearches can be used with conditionals to create more complex searches. For example you may search for all text that is *not* red, and chain those results into a subsearch.

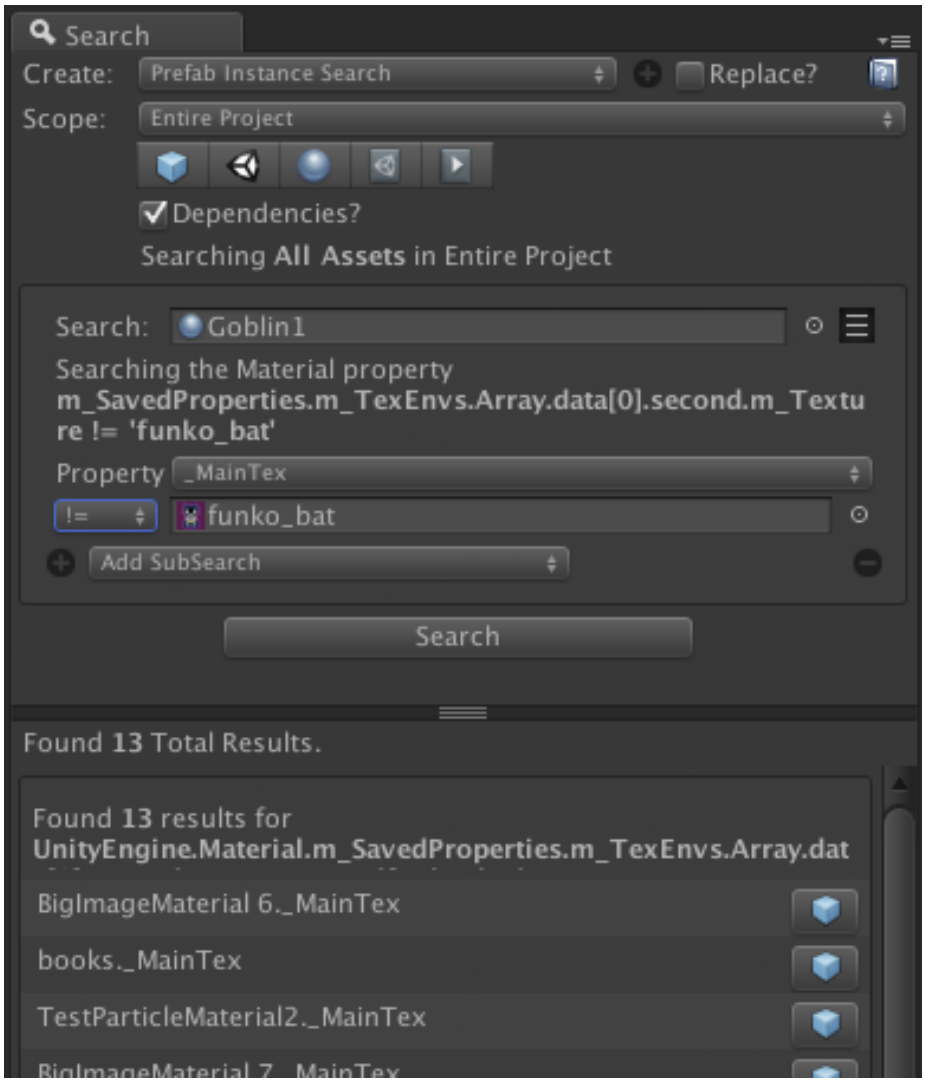

Figure 14: Example: Finding all materials that do not use 'funko_bat' as the _MainTex value.

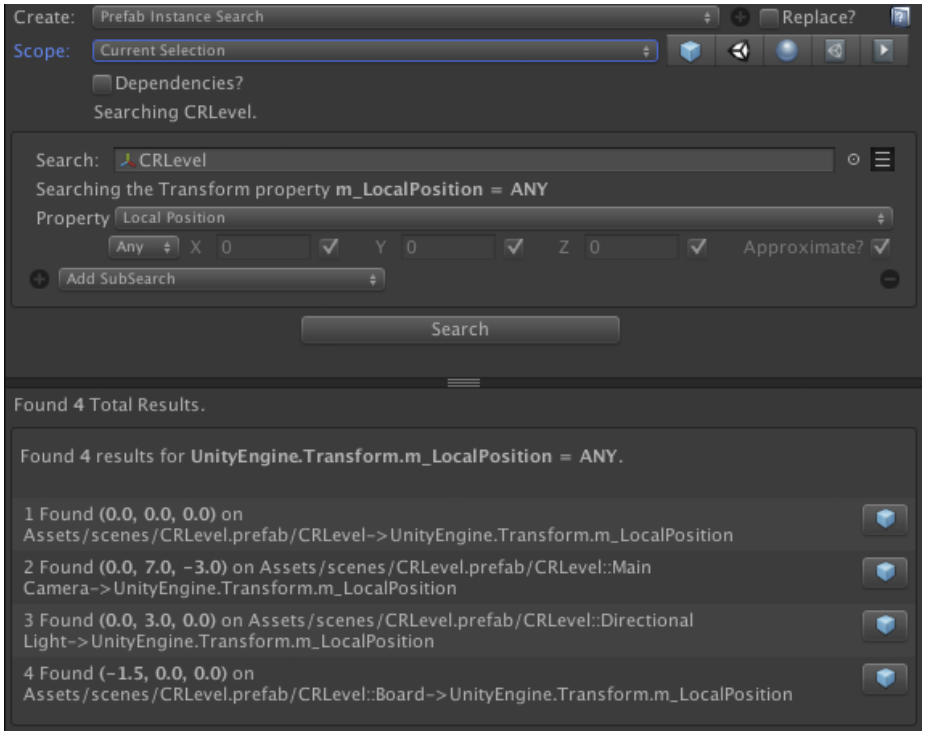

Figure 15: Example: Viewing all transform positions within the current selection using the 'Any' conditional.

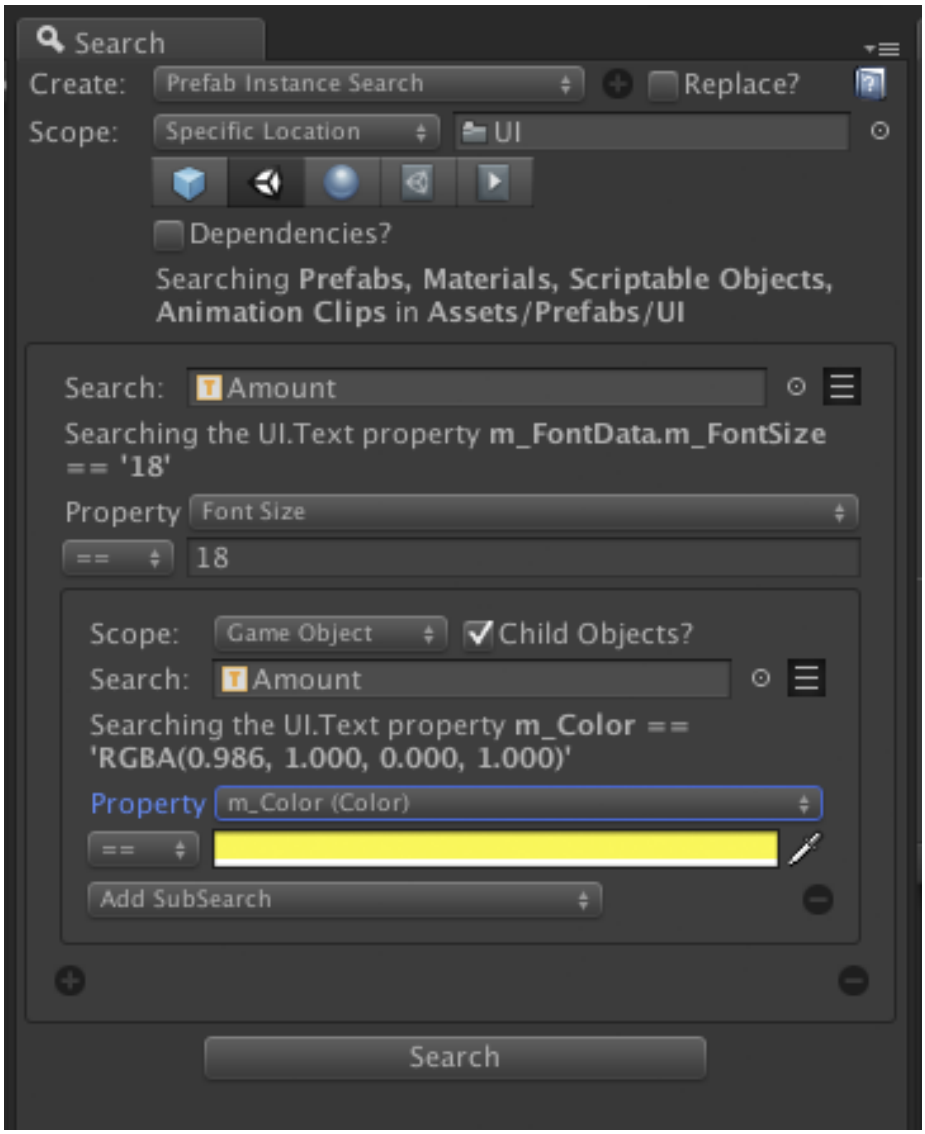

Figure 16: Example: Searching for text with a font size of 18 that is also yellow.

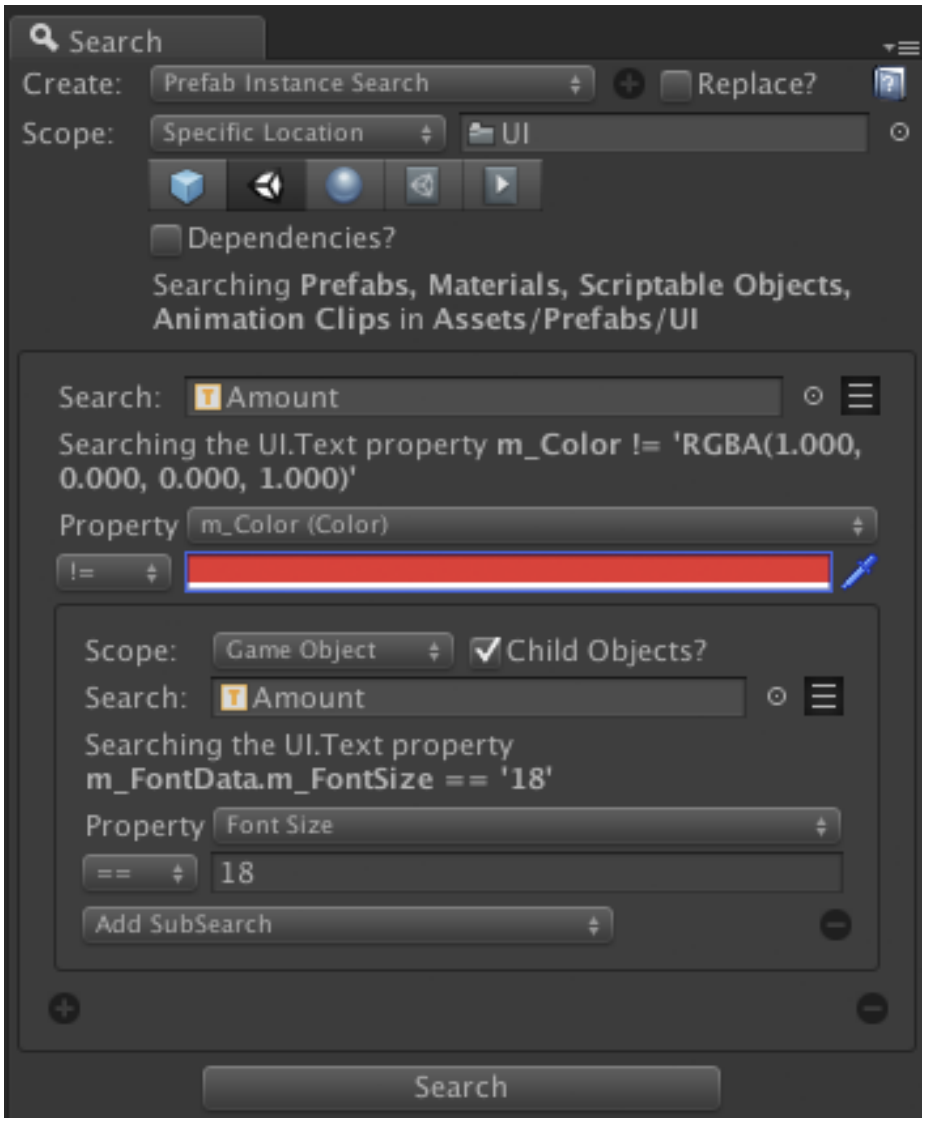

Figure 17: Example: Searching for text that is not red with a font size of 18.

Subsearch Scope

Sometimes you may want to search for a value on one object, and then subsearch for a different value on a related object. The subsearch item's **scope** provides this functionality.

Figure 18:

When a match is found, the subsearch will look at the scope value and search the objects defined within the scope. The different scope values are:

- `Return Value` - Use the value in the parent search field returned as the scope. When `Child Objects` is checked then the GameObject and Components that are part of the
- `Same Object` - Only look at this specific object and no other for a match.
- `Game Object` - Look at the Game Object and Components attached to this object (Default Value).
- `Parent` - Look at the transform's parent and use that as the scope. If there is no parent then no matches are found.
- `Children` - Do not look at the Game Object or Components of the match, but instead its children. If there are no children then no matches are found.
- `Prefab` - Look and see if we are in a prefab and search the Game Object and Components of the prefab. If the object is not part of a prefab then no matches are found.

When `Child Objects` is checked then the child Game Objects and their Components are searched in addition to the values above. This is checked by default.

Searching Specific Locations

By default the entire project is searched. It can be useful to only search and replace certain areas of your project instead. The scope options help accomplish this.

These are the Scope Options:

- *The Location Field* - The location where you want to search.
- *The Asset Field* - What kinds of assets you want to search inside.
- *The Scope Description* - A human readable explanation of the current scope.

In order to search inside a specific location:

- Check `Specific Location`. An input field will appear.
- Drag an item into the Location Field. Possible values are:

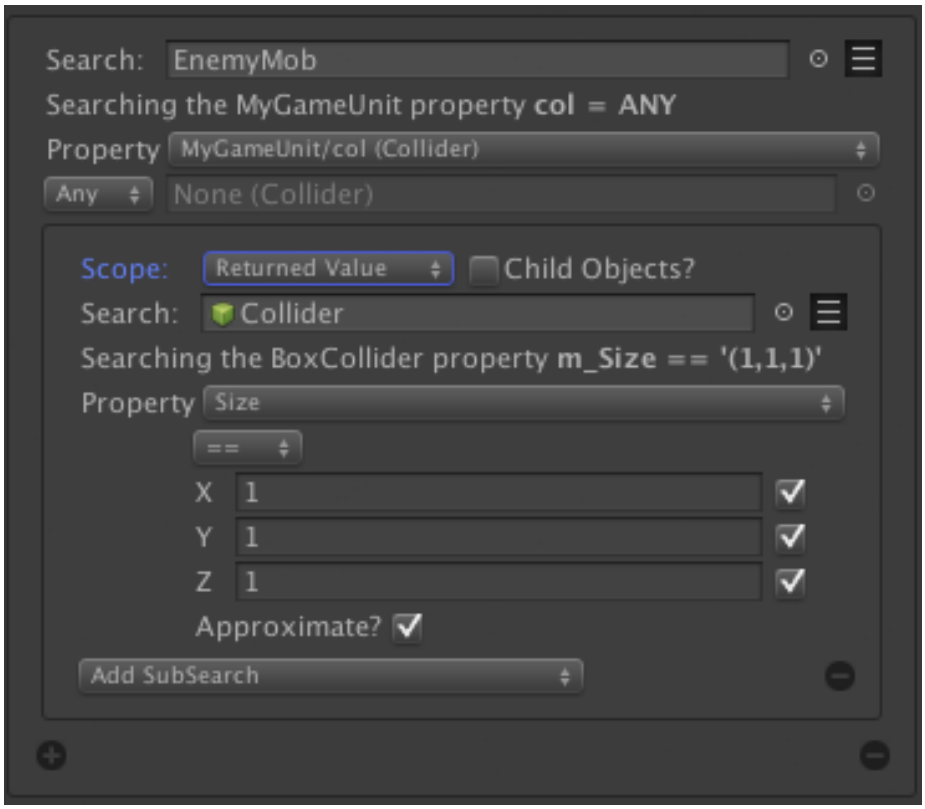

Figure 19: Example: Search the Collider returned from the parent search inside the 'col' value, and only match BoxColliders with a size of Vector3.one

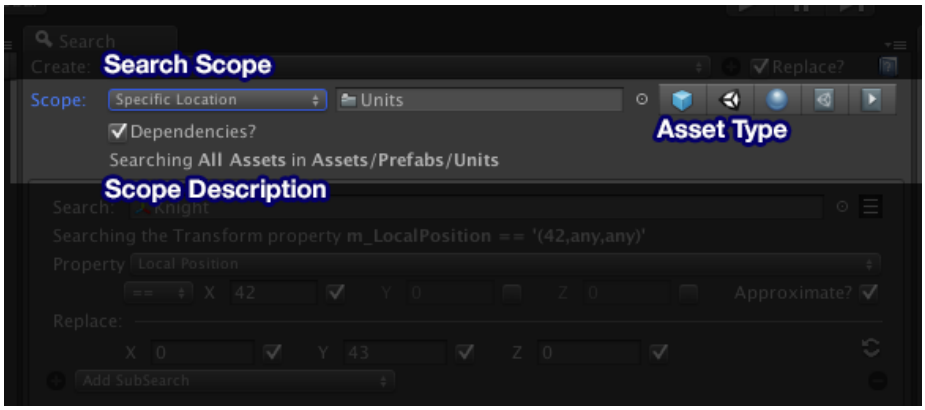

Figure 20:

- Folders
- Scenes
- GameObjects in the currently open scene.
- Prefabs
- Materials
- ScriptableObjects
- Animation Clips
- MonoBehaviours and Components.

Searching Specific Types of Assets

The *Asset Field* provides the ability to search specific types of assets. This field is only available if you are searching the entire project or a specific directory.

Figure 21: The Scope Description above shows that we are searching only for Prefabs & Scenes inside the Assets/Books/Prefabs folder.

Figure 22: The Scope Description above searches all materials in the entire project.

The Asset Field allows you to select any combination of:

- Prefabs
- Scenes
- Materials
- Scriptable Objects
- Animation Clips

Searching Folders

Figure 23:

When searching in a folder you are searching within the folder, and all sub folders.

Searching Inside A Specific Scene

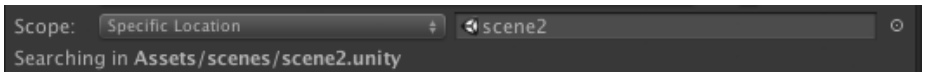

Figure 24:

It is possible to search inside a specific scene by dragging a scene into the field as well. In order to search the current scene select `Current Scene` from the `Search Scope` dropdown.

Searching Items Within The Current Scene

It is possible to search GameObjects within the current scene by selectiong `Specific Location` and dragging them into the field. Unfortunately when doing so your search cannot be saved. Upon recompile or scene change the search location will be lost.

Searching Items while in Play Mode

It is possible to search within the current scene, or specific locations within the scene while in Play mode. You can either:

- Select `Current Scene` and the search will apply to all items within the scene.
- Select `Specific Location` and drag an object from the scene into the Location field.

Searching Inside MonoBehaviours and Components

It is also possible to search a specific MonoBehaviour or Component on a prefab or scene object. This can be useful if your MonoBehaviour has large data structures attached to it and you're looking for that needle in a haystack.

Searching Dependencies

A 'dependency' is an object that requires another object to function correctly. For example a material may require a texture in order to work correctly. Dependencies in PSR can be searched within your current scope by checking the `Dependencies?` checkbox.

Dependencies are resolved using Unity's internal EditorUtility.CollectDependencies function. Dependencies are then added to the list of items being searched, and always only searched once.

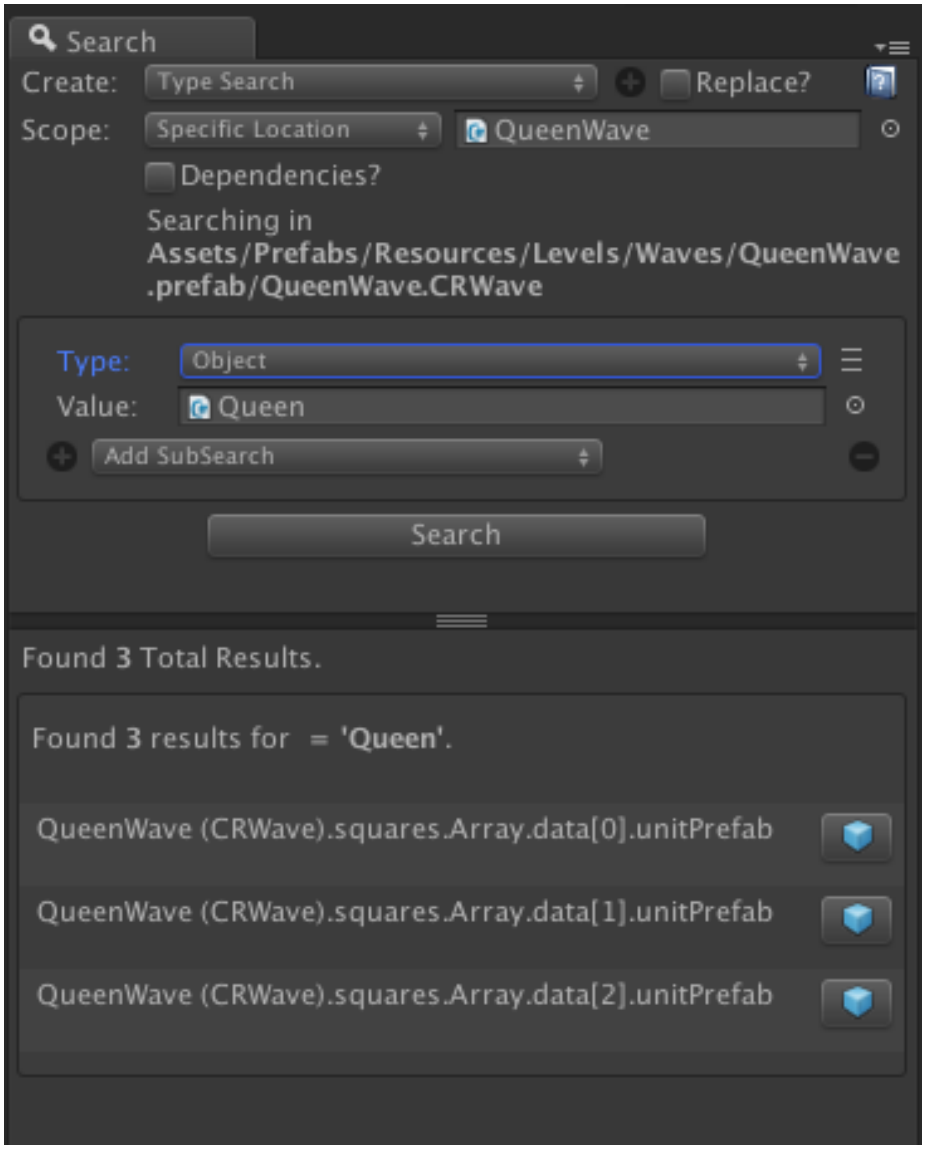

Figure 25: The above search is a specific instance of the CRWave MonoBehaviour in a scene. The search looked at every property of the MonoBehaviour and found a deep nested link to Queen's CRUnit Monobehaviour inside an array inside an object inside the CRWave.

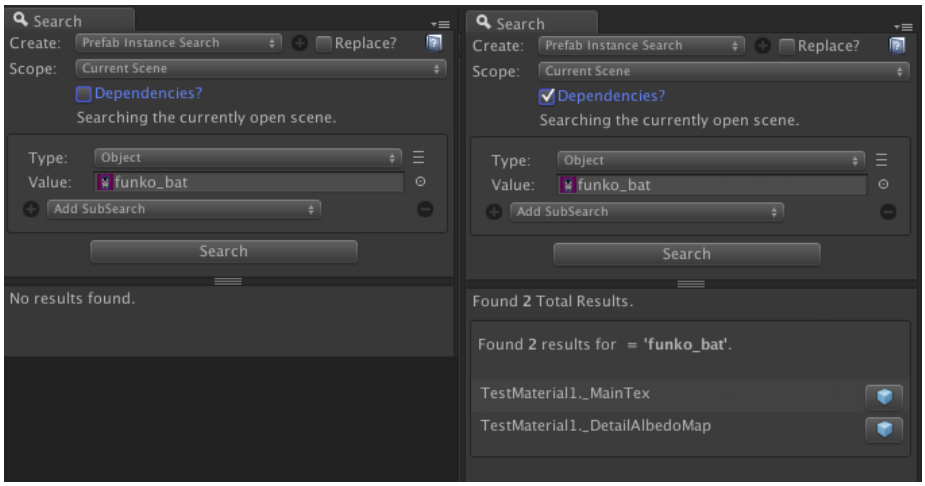

Figure 26: Searching for a texture within a scene with/without dependency search.

Searching and Replacing

When searching and replacing it is imperative to have a backup or use version control.

You can switch between Search and Search/Replace modes at any time by changing the `Replace` checkbox. When you do, then additional fields will display, providing the ability to replace the results given.

More Replace Options

When a Search Item's `Options` are active, it is possible to more granularly replace values. In the above example when a Transform with an approximate position.x of 42 is found, it will set the position.y to 43. All unchecked values are ignored.

Searching For Scripts

It is possible to search for usages of specific scripts. For example, you might be interested to know whether a specific MonoBehaviour is used at all, so that you can safely delete it.

Scripts are:

- MonoBehaviours

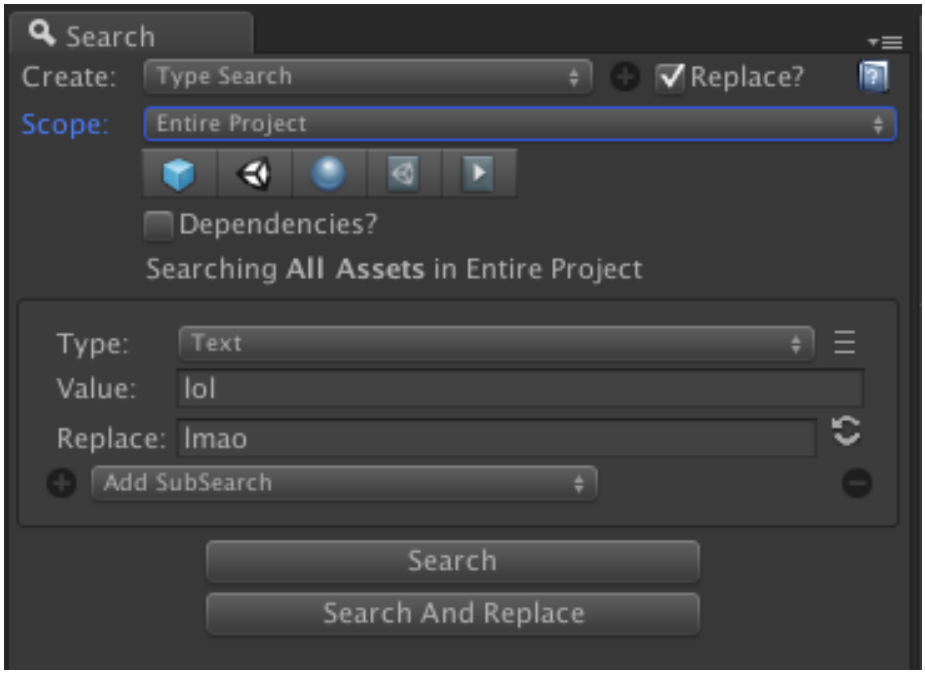

Figure 27:

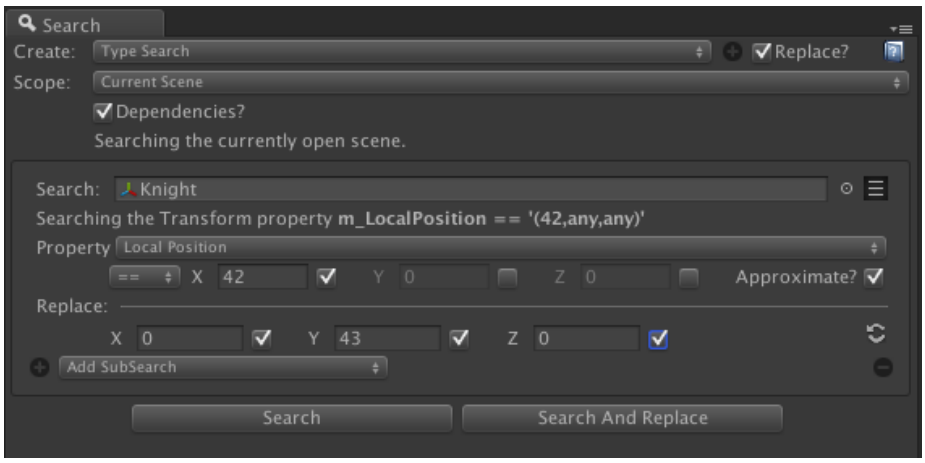

Figure 28:

- ScriptableObjects
- Components

To search for a script:

- In the `Create` dropdown choose 'Type Search' and press the `+` button.
- In the `Type` dropdown select `Scripts`.
- Select the script you are interested in by:
  - Dragging the Monobehaviour class into the field from the Project Tab.
  - For built-in MonoBehaviours such as the UI classes you can drag from a prefab or scene object.
  - Dragging a Component into the field.
- Click `Search`

This will show all usages of the script within the given scope, including prefab instances.

When searching for `Objects` in a Type Search you are searching for any usages of an object. For example, if I drag a Camera Component over into an `Objects` field then the search is looking only for usage of that specific Camera.

When searching for `Scripts` in a Global Search you are searching for any usages of the script. For example, if I drag a Camera Component over into a `Scripts` field it will return all Cameras in the given scope.

Searching and Replacing Scripts

It is also possible to swap out the script that a MonoBehaviour uses. For example you might want to search for MyBehaviour and replace the usage with MyBehaviourExtended.

Replacing scripts comes with a few caveats. MonoBehaviours contain references to objects called MonoScripts. In order to search and replace a MonoBehaviour to use a different class, you can switch the underlying MonoScript and it is now using the new script. All the saved property data for the behaviour will attempt to be re-mapped to use the new script as best as possible.

Example Interactions

It is not important that the scripts extend from each other, but that can help. It is more important that the property names match.

Given the following classes:

```
public class MyBehaviour : MonoBehaviour
{
  public string myString;
}
```

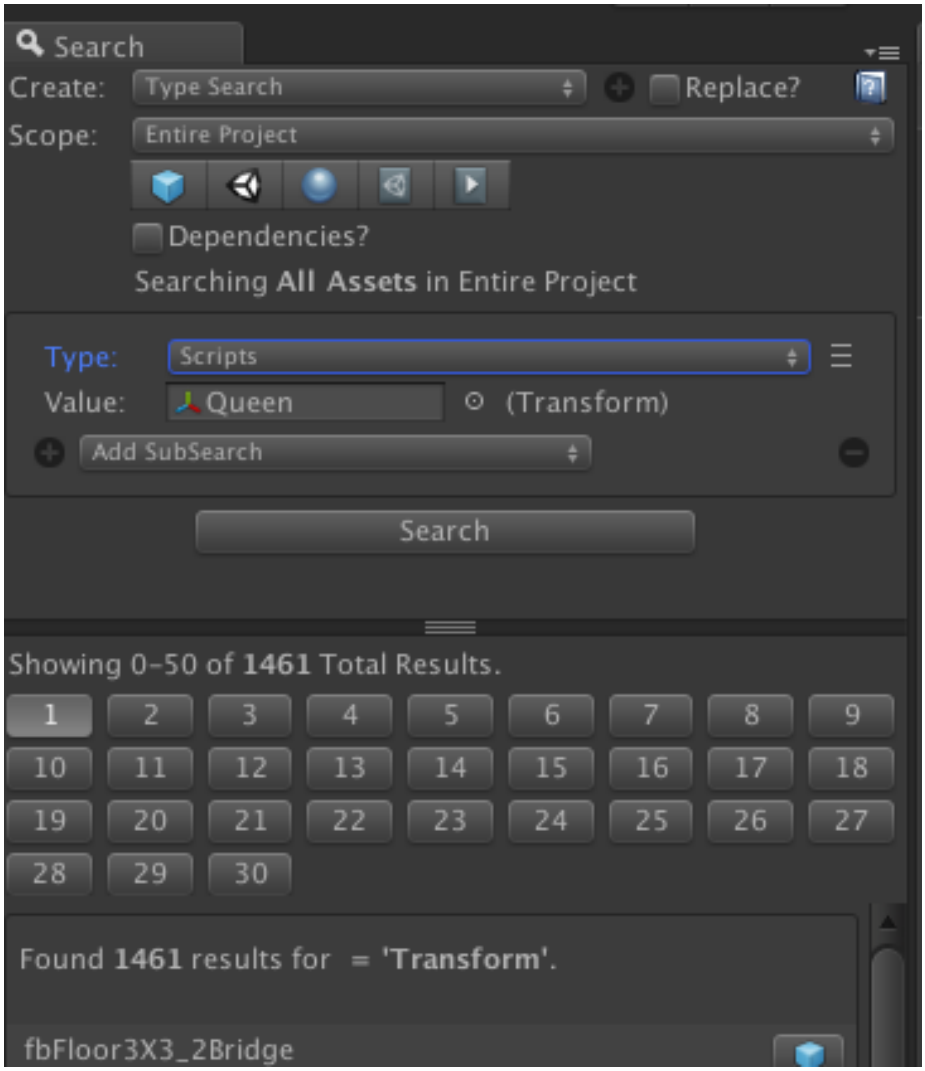

Figure 29: Example: Searching for all Transforms in the game.

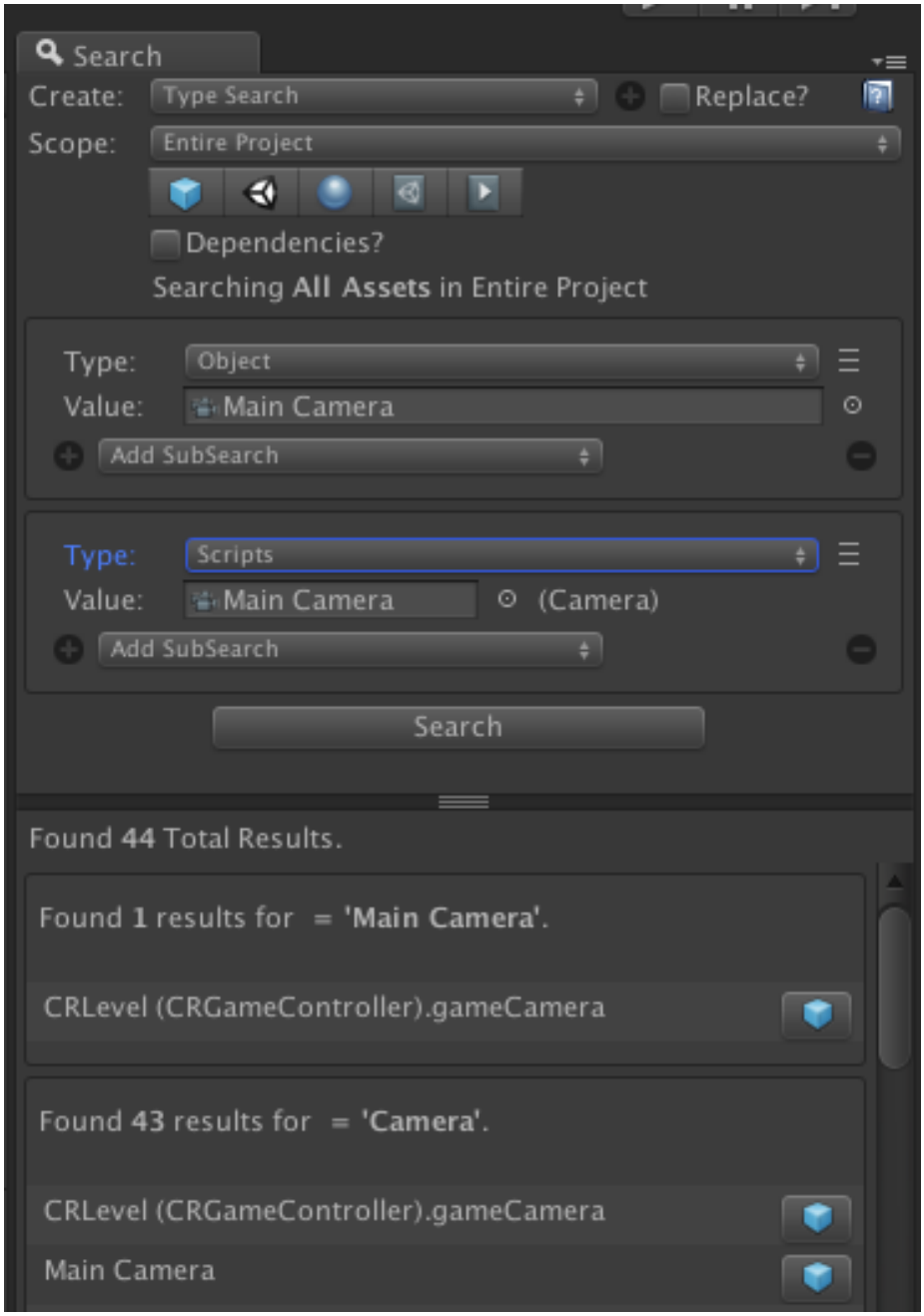

Figure 30: Example: Comparison of searching for a specific Camera vs. all Cameras

```
public class MyBehaviourExtended : MyBehaviour
{
  public string anotherString;
}

public class AnotherBehaviour : MonoBehaviour
{
  public string myString;
  public string anotherString;
}

public class YetAnotherBehaviour : MonoBehaviour
{
  public string unrelatedString;
}
```

- Replacing MyBehaviour with MyBehaviourExtended has no property data loss because it extends it.
- Replacing MyBehaviourExtended with MyBehaviour would lose the data for `anotherString` because the property only exists in the subclass.
- Replacing MyBehaviourExtended with AnotherBehaviour would cause no data loss because the property names match exactly.
- Replacing any behaviour with YetAnotherBehaviour would lose all property data.
- Replacing AnotherBehaviour with MyBehaviour would save `myString`.

Searching for Subclasses

When a Search Item's `Options` are active, it is possible to search for classes that inherit from the given class. For example, searching for Transform will also yield all RectTransforms. With the above example you can search for MyBehaviour and results would also return for MyBehaviourExtended.

Script References When Replacing

When it comes to searching and replacing scripts it is useful to understand the difference between a reference and an instance. A reference is a 'usage' of an object, while an instance is the object itself.

In the above image the first three items (GameUnit1, GameUnit2, and Player1) are the object *instances*. The GameController.gameUnits array contain *references* to the objects.

```
public MyGameUnit[] gameUnits;
```

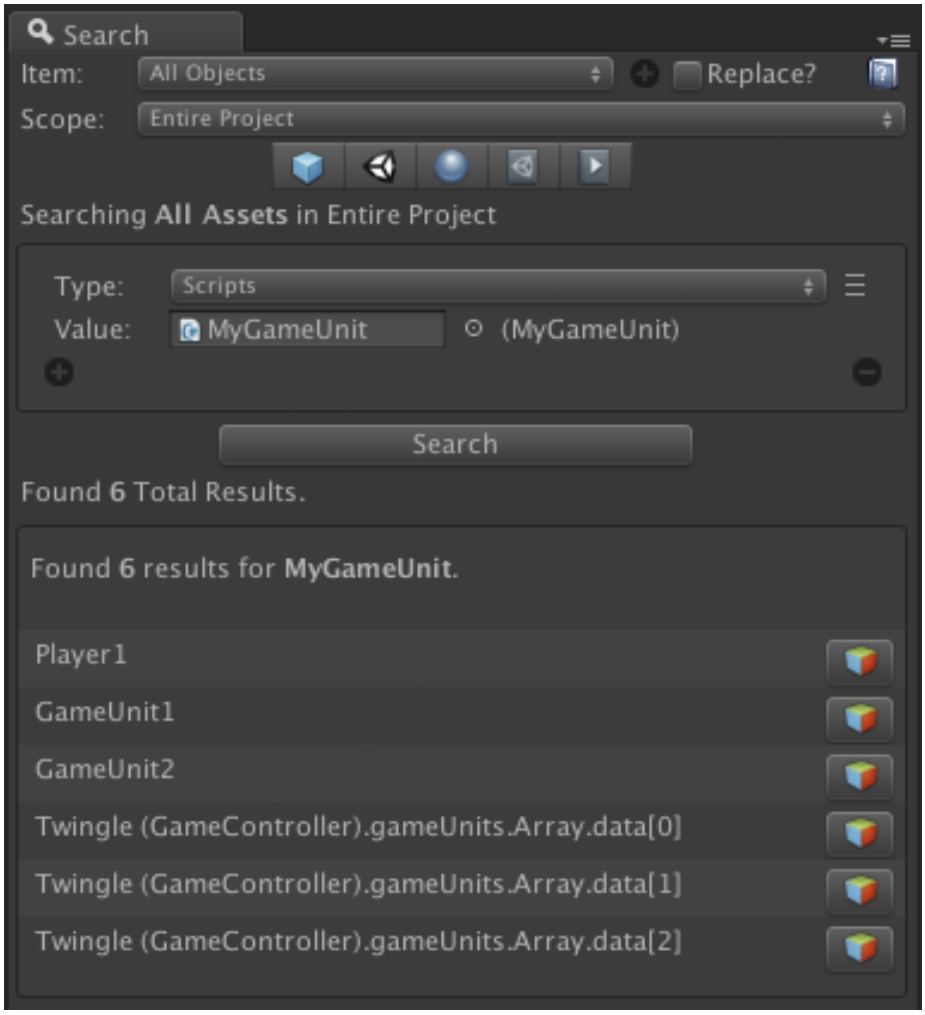
Search
Item:
All Objects
Replace?
Scope:
Entire Project
Searching All Assets in Entire Project
Type:
Scripts
Value:
MyGameUnit
(MyGameUnit)
Search
Found 6 Total Results.
Found 6 results for MyGameUnit.
Player1
GameUnit1
GameUnit2
Twingle (GameController).gameUnits.Array.data[0]
Twingle (GameController).gameUnits.Array.data[1]
Twingle (GameController).gameUnits.Array.data[2]

Figure 31:

When searching and replacing these references will still reference the same object, only the MonoScript on the object changes. But if the type changes to a type that is incompatible with the reference, the reference will break. For example, if I were to replace all `PlayerUnit` scripts with `MyBehaviour`, then the reference inside the `GameController.gameUnits` array would be destroyed because the declared array type is `MyGameUnit` not `MyBehaviour`.

Before doing a search and replace, it is recommended to look for references and whether they would break if a replace would occur .

Searching Components and Prefab Instances

When searching for scripts, components are built-in Unity constructs and cannot be replaced.

When searching inside Prefab instances, it is not possible for a prefab instance's MonoBehaviour to differ from its prefab and it is not replaced.

Searching and Replacing inside Animation Clips

Currently most of the contents of an AnimationClip is not easily modifiable. The only currently searchable data is the animation 'path'. The 'path' of an animation is a string that describes the hierarchy of objects from the animation 'root' to the object being animated. Some examples:

- Parent/Child
- Parent/Child/Grandparent
- MyObject

When a GameObject is moved in Unity, there is no built-in facility to re-map the path to the new location.

It is possible to fix broken paths using Search & Replace.

- Create a new search item of type 'Property Search'
- Drag an AnimationClip into the `Object` field.
  - This is not the clip we are searching! It is only used to tell PSR that we want to search for AnimationClip properties.
- Enter the value of the broken path into the `Search` field.
- Enter the correct value into the `Replace` field, making sure to add slashes between items.
- Verify the search scope correctly includes the AnimationClips we want to fix.
- Press `Search` to test your search and verify the values are correct.
- Press `Search and Replace` to modify all AnimationClips.

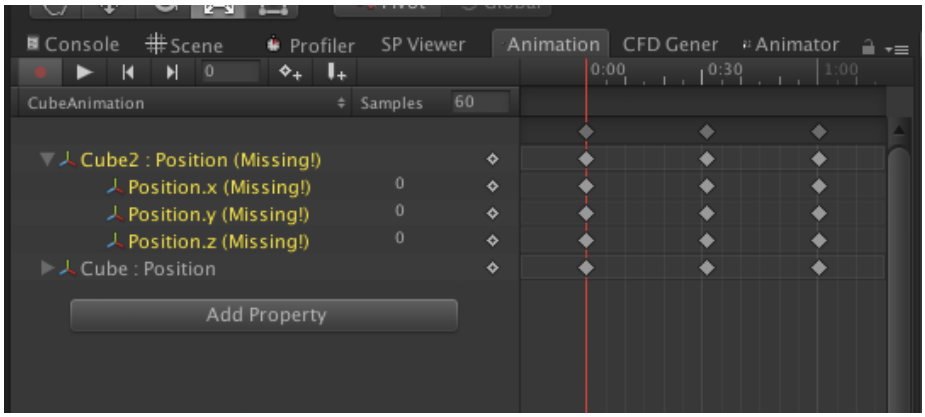

Figure 32: The object Cube2 has been moved to a new location, and the animation is now broken.

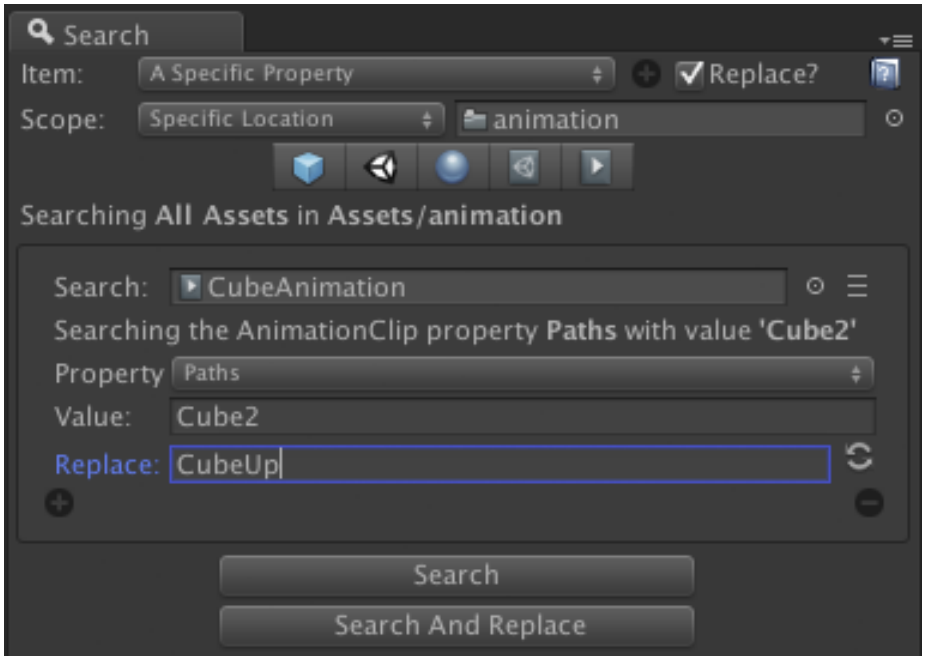

Figure 33: Example: Using search and replace to fix the Cube2 animation path to the correct name.

Search and Replacing Prefab Instances

It is possible to search for instances of prefabs. Searching and replacing works similarly to type and property-based searches.

- Create a new search item of type 'Prefab Instances'.
- Drag the prefab you want to search for from the Project tab into the `Search:` field.
- Click `Search` to search for the instance of the prefab.
- If the search type is set to `Search And Replace` you may add a prefab to replace with.
- Click `Search and Replace` to replace all instances in the current search scope.

Missing Information when Replacing Prefab Instances

Searching and replacing prefab instances can lead to a loss of information. This is an *expected side effect* due to prefabs being completely different objects. For example I may search for PrefabA and replace with PrefabB.

PrefabA:

- PrefabA
  - Child
    - GrandChild

PrefabB:

- PrefabB
  - Child
  - Child2

When replacing PrefabA with PrefabB the game object 'GrandChild' will be removed, and 'Child2' will be created. Any modifications to the instance of 'GrandChild' will be destroyed because no such object exists in PrefabB. It is not possible to keep old game objects when replacing, since these objects are technically part of the original prefab, not the instance.

PSR attempts to map information between the search prefab and replace prefab but cannot fix situations where the replacement prefab doesn't have the object.

To continue with the above example:

- The transform.x of `PrefabA` has been set to 100.
- `PrefabA/Child` has a script called 'MyBehaviour' on it and the value of 'myString' has been changed to 'foo'.
- The transform.y of `PrefabA/Child/Grandparent` has been set to 75.
- `PrefabB/Child` also has a script called MyBehaviour on it.

When a prefab instance in a scene overrides a value in the original prefab these are called 'property modifications'. The above property modifications on PrefabA will map onto PrefabB like so:

- PrefabA and PrefabB both have transforms, and so the modification of tranform.x will migrate successfully.
- PrefabA/Child and PrefabB/Child both have a 'MyBehaviour' MonoBehaviour and so 'foo' will migrate successfully.
- PrefabB does not have a 'GrandChild' object and so PSR can not map the modification onto the instance.

When property modifications are mapped the following edge cases will not replace as expected:

- When sub-objects have been renamed.
- When the order of the game objects has changed.
- Prefabs that contain multiple of the same component type.
- Instances that have removed components.

Advanced Feature: Instance Mappings

Instance Mappings are an advanced feature which allows you to attempt to fix potential data loss when searching and replacing prefabs. If you have prefabs with different hierarchies, order, or names you can use Instance Mappings to define explicit mappings between your search prefab and your replace prefab.

In the above example the data for GrandChild would simply be lost. But we can instead apply the Transform modification to `PrefabB/Child2`. This is possible through Instance Mappings.

- Click the More Options icon to show options.
- Click the '+' symbol to add an Instance Mapping.
- Drag the GameObject of `GrandChild` from the Project tab into the 'From' field.
- Drag the GameObject of `Child2` from the Project tab into the 'To' field.

Now any property modifications made to GrandChild will now instead by applied to Child2. If you drag the GameObject then it will map the GameObject and all Commponents. If you only wanted changes from the Transform to be applied, then drag only the Transforms into the 'From' and 'To' fields.

Most of the time complicated mappings should not be necessary. This is an advanced feature and should only be used if you feel comfortable using it. It is possible to map modifications arbitrarily between object, from a MeshRenderer to a Transform. Maybe that is what you want... or possibly not. :)

Accessing deep objects in Prefabs

By default Unity will not allow easy access to objects nested below 1 level deep within the Project Inspector. In order to access these objects you may also drag from the scene into the Instance Mappings fields. PSR will automatically find the parent prefab object and use that instead.

If the object is not in a scene, or you do not want to drag it into a scene, you may also use my free Prefab Hierarchy Inspector. This allows you to view the full hierarchy and drag items from it into PSR.

Potentially Invalid Replaces

If searching specific types globally, it is possible to search for an object of type A and replace it with B, but B is not a valid type.

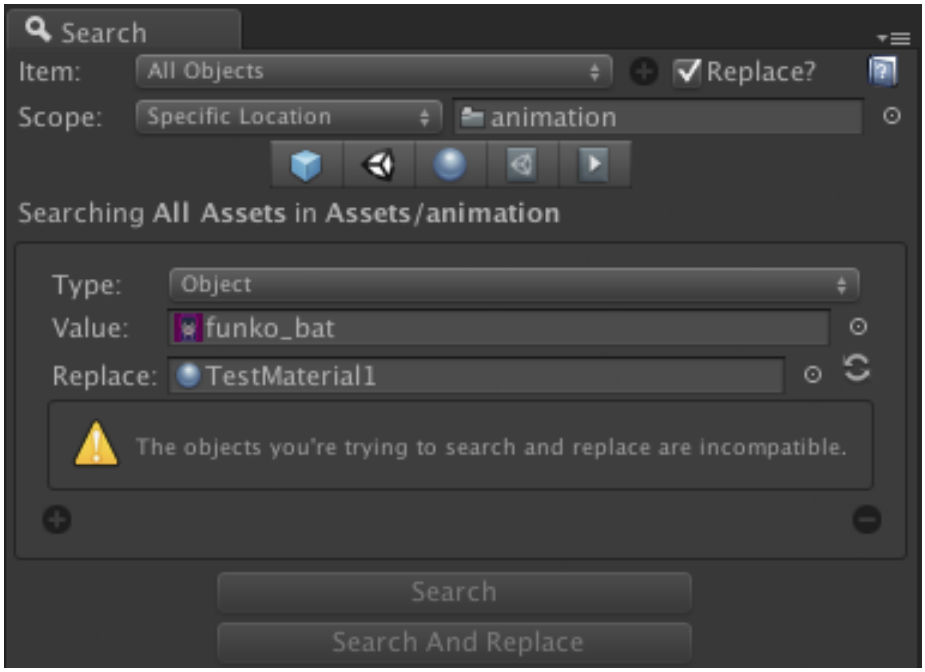

Figure 34:

In the above example if we were to replace, we would attempt to replace a texture with a material. This is not a valid replacement and 'undefined behavior' may occur. In this example nothing will happen because Unity is smart enough to route around the potential damage.

Currently Unsupported Searches

Certain search and replace operations are more difficult than others. Many aspects of Animation Clips and Particle Systems are not easily searchable due to the implementation details of these objects.

For example in Particle Systems things like the 'Start Delay' value is a mixture of 4 values (2 floats and 2 curves) that are computed at runtime.

Animations have similar issues where the values are stored as curves which are currently unsupported.

Keyboard Shortcuts

On Mac please use the 'Command' key instead of 'Ctrl'. Keyboard shortcuts only work while the tool window is currently focused.

- `Ctrl-Enter` - Execute a search.
- `Ctrl-Shift-Enter` - Execute a search and replace.

FAQ

How does it work?

It uses reflection, SerializedObject, and SerializedProperties to search and replace the serialized representation of the project. Certain internal properties are ignored. Additional magic is used to handle things like built-in components and modifications to prefab instances.

Invalid AssetDatabase Path Warning

I'm seeing a warning that starts with "Invalid AssetDatabase path:". What does this mean?

When we search and replace items in dlls such as UnityEngine.UI this warning shows up. It is a benign error but unfortunately unavoidable.

Scene Objects Warning

I'm seeing a warning that starts with "Scene objects can only be searched for inside the current scene." What does this mean?

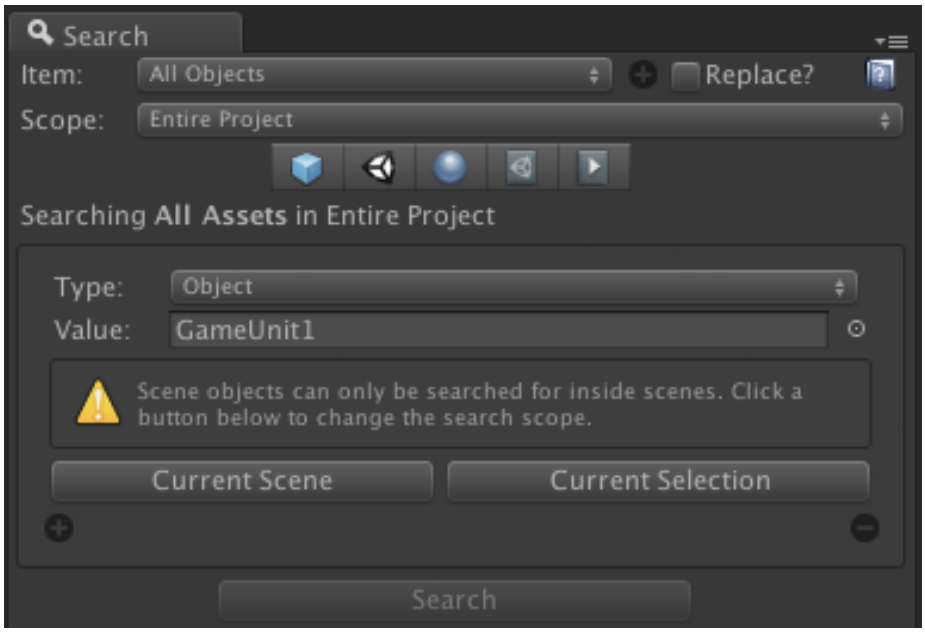

Figure 35:

Chances are you have dragged an object into a field that only exists within a scene, but are trying to search across the project. Scene objects only exist while the scene is open, and cannot exist in other parts of the project. As soon as you change scenes, the object disappears.

You can click the button to switch the search's scope to be within the scene, or search with an object that does not exist solely within a scene.

Looks Like This Object Doesn't Exist Anymore

This will display when you click on the 'Go To' button and the tool cannot find the item you are looking for. Prefab instances and unsaved changes do not have identifiers that allow the tool to find the object, but it will try to guess. This usually occurs when the hierarchy has changed (objects have been added or deleted) and the object is a prefab instance or an unsaved change.

Searching in a scene but this is not a scene object

Scene objects are items such as MonoBehaviours, GameObjects, and Components. Items such as Materials, Textures, and Animations are not a part of scenes and exist within the project. They are referenced by scene objects and are scene 'dependencies'.

When searching scenes and dependencies is unchecked, PSR searches only scene objects, and not inside of dependencies. This means that you won't find properties of Materials in scenes. For example: searching for all Materials with a red color inside a scene will not find any results. A possible workaround is to use a 2-step process. First, find any red colored Materials in the project. Then search for those Materials within the scene.

When the search scope is set to search scene objects, but the current object type is not an object that can exist in a scene (such as a GameObject, MonoBehaviour, or Component) then this warning will display.

I'd like feature X. Can you make it happen?

I'm happy to try. Please email me at chris@enemyhideout.com and let's chat.