Ultimate

crafting system

by **polyperfect**

Have a Suggestion?

info@polyperfect.com

Thanks!

First of all, thank you for purchasing our pack, we really appreciate that! We are putting a lot of effort into this.

We are also planning to expand the list of the characters and their animations in the future with free updates of the pack. Check out our **Facebook Page** for any news.

Handy Links ;)

Other Low Poly Packs

Low Poly Animated Animals

Low Poly Animated People

Low Poly Animated Dinosaurs

Low Poly Animated Prehistoric Animals

Low Poly Epic City

Low Poly Ultimate Pack

Low Poly War Pack

Toolkits

Ultimate Crafting System

2D Packs

Low Poly Icon Pack

Fancy Icon Pack

Follow us

Facebook

Instagram

Youtube

Polyperfect.com

Updates

VERSION 1.0
INITIAL RELEASE

Overview

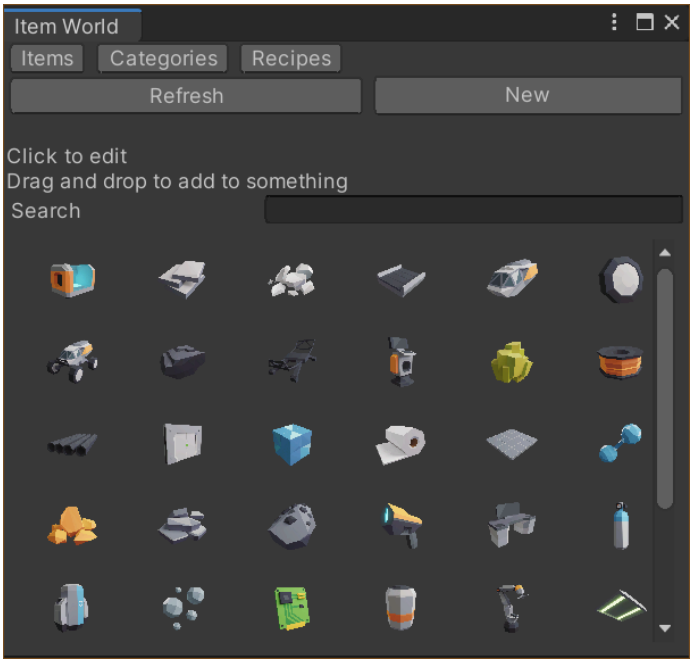
What does it do?

The Ultimate Crafting System is designed with ease of use in mind, and allows for the creation of “Item Worlds”, which are collections of Items, Recipes, and Categories that make up the core of versatile crafting games. Items, recipes, and their data can all be created directly in-editor. Of course, programmers are free to extend the system even more too, as the source code is included!

How do I get started?

The heart of the Ultimate Crafting System is the Item World Window, which you can open from the Window dropdown at the top of the screen. From here, you can edit your Item World by creating new objects or editing existing ones.

Item World Window



Navigating the Window

At the top of the window are three tabs: Items, Categories, and Recipes. Each of these will let you create and edit objects of the relevant type. Selecting one will change the item view to show only the objects of that type, and the New button will let you make objects of the same type as well.

Naturally, by typing in the Search field, you will filter the displayed objects by name. You can click and drag objects from the Item Window into slots or fields in other parts of the Unity UI to register them with scripts and other such things. If you simply click an object, you will open it for editing in a small window. Alternatively, if you click with the right mouse button, you'll have the options to Clone or Delete a

particular item. The refresh button just refreshes the window in case you added or deleted items outside the system, such as through Version Control.

Editing Objects

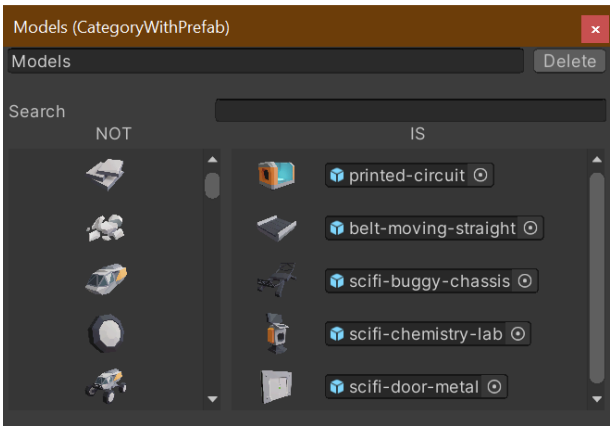
Items

If you create or open an object, this window will appear. You can edit the name of the object at the top, as well as delete it easily.

The first main section allows you to assign the Item to categories and give it associated values like icons or numbers. Drag a category from the left to the right to associate the Item with that Category, or vice-versa to remove that association. Double clicking a Category will open that Category up for direct editing, for convenience. The image in the "_Icon" Category will be what is used for displaying the item in the Item World Window, as well as other parts of the interface.

The bottom section allows you to see which recipes create the item, as well as which recipes the item gets used in. If you click a Recipe, it will open it up for easy editing.

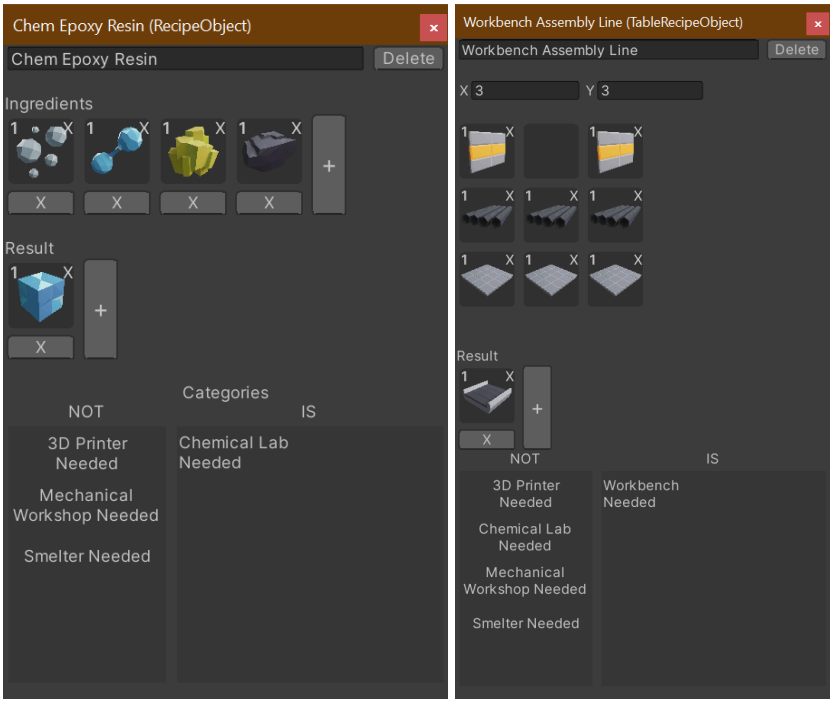
Categories



A Category can simply act as its namesake, but it can also allow you to associate data with objects. Editing one is straightforward--drag an object from the list on the left to the right, and that object will be associated with the current Category.

A simple use case would be to add descriptions to each item, or other flavor text to add depth to the world. For a more complicated use, you could have a “Model” Category that has a GameObject field, and scripts can grab that GameObject from the Category to bring the object into the 3D game world.

Recipes



Recipes naturally are the core of a crafting game. They let you specify how someone can turn some group of items into another. By default, there are two types of recipes you can create--Standard Recipes, which use some collection of ingredients in any order; and Table Recipes, which require items to be placed in a matching arrangement in order to be valid. You can drag items from the Item World Window into the slots in the Ingredients or Results sections to register them, and click the number to edit the amount required.
At the bottom you'll find the same Category-editing view as in the other editing windows. The most common use for Categories and Recipes is to restrict which recipes can be used in what contexts, such as only allowing large items to be crafted on a workbench, or hot items to be made with a furnace.

Making Gameplay

The scripts included in the Ultimate Crafting System are all rather generic, though due to its simplicity there are a number of scripts for UGUI, Unity's traditional UI System, included as well.

Item Stacks and Slots

Most of the scripts assume the use of ItemStacks, which are just an item with a quantity. For example, many of the scripts have an empty slot in them that you can drag an item into. After this, you can edit the number of items by clicking on the number itself. Some slots though are specifically for Categories, or others.

Inventories

An "Inventory" is simply a collection of Slots which can have items inserted or extracted. The **ChildSlotsInventory** component will automatically collect all Slots on child GameObjects. This can then be referenced by anything for insertion, saving or whatever else you want. One inventory can include others--for example, in the

demo scene, the player has a ChildSlotsInventory on it, as well as one on the hotbar and one on the hidden inventory page. Thus, the player inventory contains both the hotbar's slots as well as the hidden slots, while each of them contain only their own. This means with one inventory saving script you can save the contents of both.

Generic Scripts

These scripts don't really do too much by themselves, but they're used by many of the other scripts in the pack. Chief among these is the **ItemSlotComponent**. The ItemSlotComponent allows a GameObject to hold on to a singular item stack, as well as specify some constraints, like requiring a particular category. Some common uses for the ItemSlotComponent are for displaying items in… well, a slot in the UI. Or otherwise just keeping track of available space in a player's inventory.
Another generic script is the **Crafter**. The crafter just exposes various methods useful for crafting to other scripts, as well as to Unity Events. Because of this, you can have Unity GUI Buttons link to it and craft things.

Event System

Unity GUI leverages a very powerful event system, and this same event system can be used for interacting with objects. The Demo Scene in the project makes heavy use of this Event System. By attaching one of Unity's EventTrigger components to an object, you can receive various events like Click, Drag, and others. These are used in conjunction with the demo script called PlayerCommander, which lets you send commands like move, place, and interact, to the player.

Unity GUI (UGUI) Scripts

Various UGUI-specific scripts are included. These cover common tasks like hooking up inventories to slots in a UI, and so on. All UGUI-specific scripts are prefixed with "UGUI", so you can find them easily by searching in the Add Component menu.

Saving and Loading

The **InventorySaver** script allows you to quickly save and load inventories to a file on the device. You can specify the file name to be used, as well as what extension to give it. The JSON format is used so it can be interacted with easily. The file is saved to the Application.PersistentDataPath, which means it will work regardless of which device you're on. If you disable automatic saving and loading, you can call those methods from code easily as well, via SaveToFile and LoadFromFile.

Technical Details

While the Ultimate Crafting System is designed with ease of use in mind, more advanced users such as programmers looking to extend the various systems may find some of the following useful.

Runtime IDs

Every object in the Ultimate Crafting System has associated with it an ID, which is used for comparing or otherwise manipulating objects in the various systems. Runtime IDs are a blittable type, so they can be freely used in Unity's Job System, and do not accrue any garbage through use. An additional benefit is they play nicely with separate Asset Bundles using the Addressables system--standard ScriptableObjects are often duplicated as they are put into multiple bundles, making them inconsistent for references. However, because the runtime Item World only holds on to their IDs, a duplicated Item can be referenced from different Asset Bundles and still be treated as one.

Object Representation

The objects that you work with in the Unity Editor are ScriptableObjects, which makes them play nicely with the Unity UI, Version Control, and other features. However, as hinted at previously, at runtime what is used to identify an object is, and should only be, its Runtime ID. The data you set for a given object is instead stored in a lookup, with the Runtime ID as the key, much like in Unity's ECS implementation.

Category Data

As just described, the data for an Item, for example, is not actually stored on that Item. Instead, in edit mode it's stored within the Category. This means you can theoretically add an entirely new Category to your game through the use of Addressables and Asset Bundles, without updating the game client. When you drag a Category when editing an Item, behind the scenes it actually edits the Category's ScriptableObject.

At runtime, the Category is looked up by its Runtime ID, and then the data within it is looked up via the Item's ID. Because it's done via ID, the user must be aware of the type of data that it is expecting the Category to have. Categories without any data simply have a bool value of "true".

Interfaces

Instead of using ItemSlotComponents or the like directly, when coding it's recommended to use their implemented interfaces if possible. For example, an IInsert<ItemStack> will let you interact with anything that can have an ItemStack inserted in code. Unity expanded its serialization capabilities in 2020+, though there are still some limitations, so in some situations if you want to be able to drag and drop references you'll have to use the component type or an abstract class.

FAQ

They’re not stupid questions, just stupid answers from us.