Item System V0.1

Getting Started

To see the inventory system in action, drag the Inventory prefab in the Hierarchy, place some IDCard and Note prefabs around, and start collecting them with E.

The Inventory

The inventory is quite easy to set up. Just create an empty GameObject in the scene, name it whatever you feel like. Only two more things left to do. First go to 'Add Component' in the empty's Inspector and select scripts→Item Database. You now have all the item information available, for the whole game. Great!

But you want to use it, right? So we want a system to actually do something with the data. With the empty GameObject still selected, click 'Add Component' and scripts→Inventory Controller.

Look again in the Inspector. There is an options available. Stack Size, so the number of items of a kind – sharing the same ID – allowed to be stacked, i.e. allowed in the inventory.

The Item Database

It gets a little more technical, unfortunately. But not too much. The information, that are required for now are the following:

```
id:      A unique identifier. Only items sharing the same id will be
         stacked.
name:    That is the identifier, the user will end up seeing.
handle:  all data have a handle, textures, images, icons, sounds will use
         this name, in the          related file.
prefab:  the big exception. This is the name of an attached prefab. Objects
         may or may not share the same handle or id. They are independent.
descr:   The description, that may end up in an info box, displayed to the
         user. Get creative ;)
```

Why is that important to you? Well for a new item you are creating, you will need to provide at least some information. If you don’t know what the handle or the prefab will be, or miss a description, fear not, you will get an error message, but the system will work with the data provided. Only id is really necessary to introduce an item.

Unfortunately I don’t have a script, for adding stuff to the database right now. However it is stored in Assets/StreamingAssets/Items.json. The data format is easy to grasp, so for now you may want to add items manually. If there is a mistake (missing comma or quotes), the item system should warn you. If you are entering prefabs, that do not exist yet, or handles, the system will also warn you, but everything will work. Messages in the Unity console are mostly friendly reminder, to add a missing component.

The Items

That’s where the fun begins. Just create / import / draw / I don’t know, you guys are the artists :P your object and click ‘Add Component’ scripts→Item Controller. Fair enough. The downside, the

object must have a Box Collider attached. Or else, you won't be able to interact with the items altogether.

And a second look to the Inspector reveals the Item ID box for the script. 0 is an invalid id, so you must get one from the database.

Action Key is irrelevant to you and will be removed soon.

In fact, you can use the system right now. Run to your freshly created item, press the action key (E) and the item goes straight to the inventory. Great!

And stays there. You can test it, only if you use the test build and are using id 1, by pressing Q. That will dump the item to the ground. This 'feature' will be removed.

Also note, that the item can only ever be restored, if prefab to the ID is set to the right name of the objects prefab. That has performance reasons.

The Item Container

Only picking up items from the floor. They are dirty, and we are no animals, right? So you can store items on desks, dressers, in front, on top, under, left, right, inside, well everywhere to be honest. If the item itself is unreachable to the character, because of one of these reasons, make the item 'blocking' access a container, by adding the script ItemContainerController. All the items that are to be contained must have ItemController script attached (hope that is self explanatory) and must be children to the container object.

And you may notice, that it does not work. I would have wanted it like this, but Unity wouldn't let me. Attach a Physics→Rigidbody to it (you may want to check all Constraints→Freeze Position and Rotation – or else, the container will spin and move all over the place).

Pick up the items from a container: Great fun!

The Inventory Feed

Okay! You will have seen, … well nothing, cause there is nothing to be seen. What a shame.

Go ahead to the inventory object in your hierarchy. Right click→UI→Canvas. Here we will draw everything required. Add a Panel to this canvas (Right click→UI→Panel). This is the background graphic for the panel, be nice, attach a Layout (Add Component→Layout→*choose one*). It would work without I guess, but that is pure madness.

The great advantage comes with the next step, the actual display slots. A slot is just another panel and child to the feed label, we just created. And the slot must have one UI→Image and UI→Text as children.

If you let the Layout handle position and scale of this slot, you can now simply copy paste your blueprint slot and they will be nicely arranged.

As a last step, you can now revisit the canvas, we created in the first step and drag and drop it's child – the feed label – from the Hierarchy view right inside the Feed Label box in the Inspector under Feed Controller.

That's all there is to it. If you defined all prefabs and handles data to the database, you now have an inventory system.

Fine Tuning the Prefabs

In my great generosity I provided the prefab to the inventory. If you are lazy, and you should be, just replace the graphics, the size, color, opacity, and you save yourself some time.

What Is About to Come

The feed gets more fancy. For example, it will appear on inventory updates and fade away quickly. I don’t want the screen to be blocked.

A use case where you can actually do something with all the thousand things, we can collect.