v0.2 6/24/2017

Introduction

Thanks for the purchase and support! We are a community of VR & game devs, working together to create games, experiences, development tools, and tutorials to empower emerging indie developers worldwide. Join us here: https://www.youtube.com/nurfacegames/

What is Mobile VR Inventory?

Making an Inventory System for games, along with the UI to show the inventory items and everything else that goes along with it, is a big challenge! This pack aims to help reduce that workload and make it faster and easier to get an Inventory System for Mobile VR games specifically.

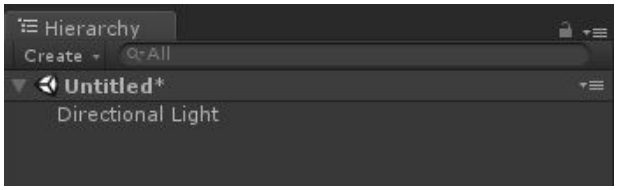
Video Tutorials

Intro v0.2: https://www.youtube.com/watch?v=yMZDXc_7KY8
Tutorial v0.2: https://www.youtube.com/watch?v=fhIBiQ5QeKg

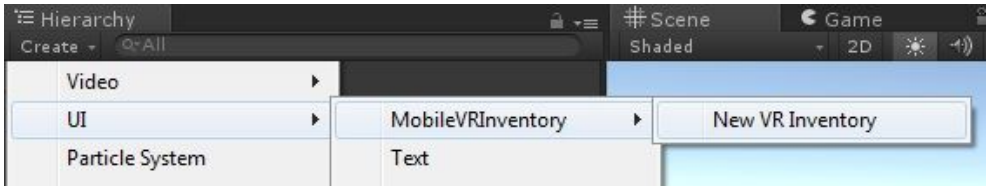
How To Use

Initial Setup

1. Open a new Unity Scene and delete 'Main Camera':



2. From the Hierarchy Windows, select: *Create > UI > MobileVRInventory > New VR Inventory:*



3. Create an "Inventory Item Database" somewhere in the /Assets folder by right clicking in the Project Explorer and selecting: *Create > MobileVRInventory > NewInventoryItemDatabase*

4. Now we can start adding some items to the database.

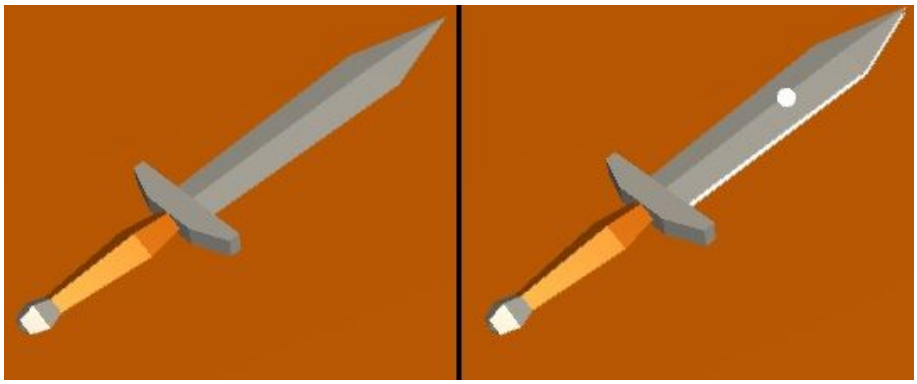
Inventory Item Database

All items that the system will use must be saved in an "Inventory Item Database".

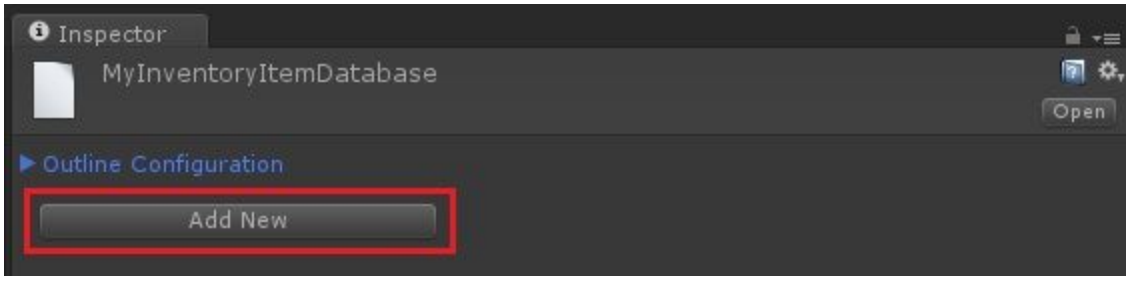
1. A new database is created by going into the Project Explorer window and selecting *Create > MobileVRInventory > NewInventoryItemDatabase:*

2. Select the Database file in the Project File to view the Inspector:

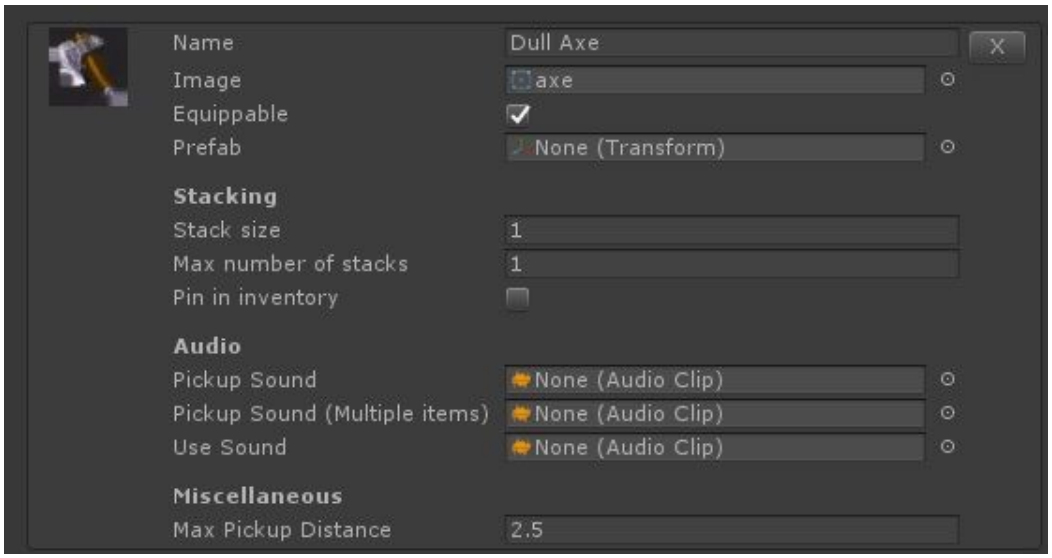
Outline Configuration:
Inventory Items, the actual 3D model in the scene, may show an outline when gazed at with this option.
Use Outline: Turn outlines on/off for the entire database of items.
Outline Color: The color of the outline, transparency also works.
Outline Width: The width of the outline.
Outline Adjust Speed: How fast the outline will grow to full width, and shrink back down.
An example from the demo scene, the sword shows a white outline when gazed at:



Adding Items to the Database
The “Add New” button will add a new item to the database.



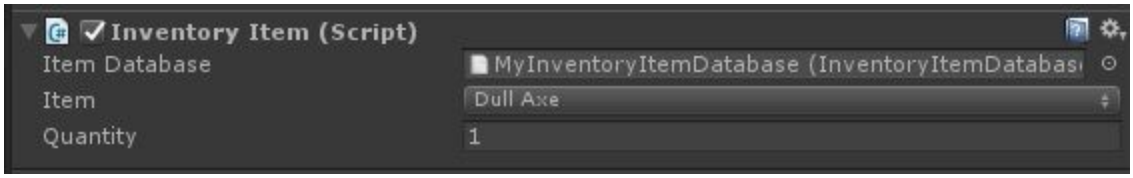
Item Configuration



Name: Give an individual name for each item.
Image: A sprite for this item which will be shown on the UI.
(Easy Icon Maker from the Asset Store makes this really easy!)
Equippable: Can the item be equipped into the player's hand? A 3D model of this object will be spawned at the player's hand position when equipped.
Prefab: The prefab that will be spawned into hand position when this item is equipped.
Stack Size: How big of a stack does this item allow? -1 for Infinity, such as gold.
Max Number of Stacks: How many stacks of this item can the player carry?
Pin in Inventory: This item will appear before any other non-pinned items in the inventory.
Pickup Sound: A sound that will be played when this item is picked up.
Pickup Sound (Multiple): The sound that will be played if more than 1 of this item is picked up (single gold coin versus a stack of gold coins).
Use Sound: The Sound that will be played when this item is used or selected in the Inventory UI.
Max Pickup Distance: How far the player must be from this item to pick it up from the world.

Inventory Item in the Scene

When you add a 3D model to the scene and want to specify it as in Inventory Item, add the *InventoryItem.cs* script to it.



Item Database: The Item Database that this item is located in.
Item: The item this 3D model corresponds to from the database.
Quantity: When this item is picked up, what quantity will it add to the database? (Stack of gold)

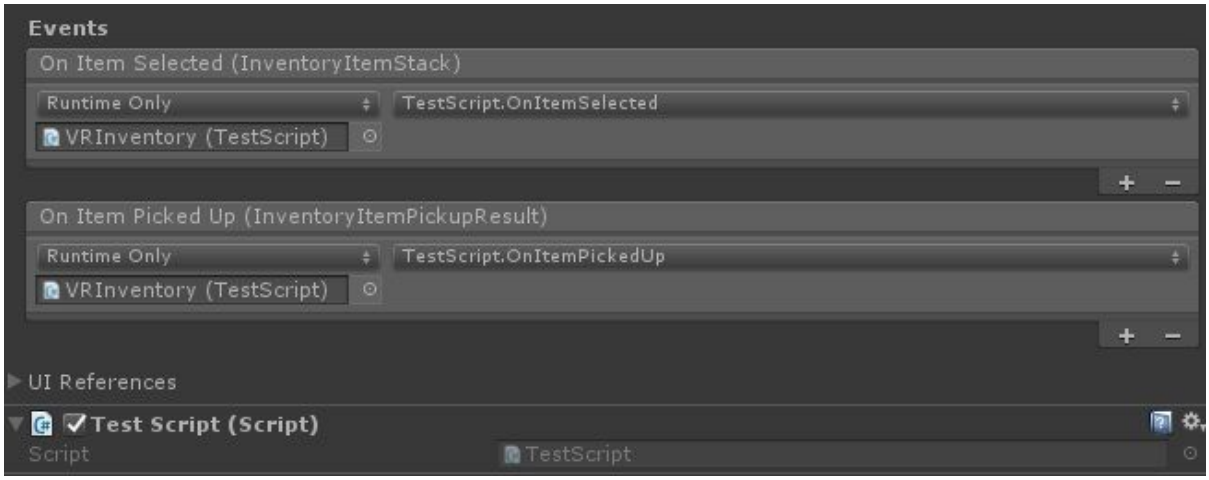
Responding to Events

The *VRInventory.cs* script currently responds to 2 events:

- When the item is picked up from the world.
- When the item is clicked on in the Inventory UI.

Assigning Functions in the Inspector

Custom functions can be assigned in the VRInventory.cs Inspector:



In the image above, a custom script *TestScript.cs* has 2 custom functions, *OnItemSelected* and *OnItemPickedUp*, which are assigned in the inspector. Here's what the script looks like:

```
using UnityEngine;
using MobileVRInventory;

public class TestScript : MonoBehaviour {

    public void OnItemSelected(InventoryItemStack stack) {
        Debug.Log(stack.item.name);
    }

    public void OnItemPickedUp(VRInventory.InventoryItemPickupResult result) {
        Debug.Log(result.item.name);
    }
}
```

Notes:

- Script must be *using MobileVRInventory;*
- The **OnItemSelected** event will pass an **InventoryItemStack** class, which is used to store information about the item or stack that was selected in the inventory.

- The **OnItemPickedUp** event will pass an **InventoryItemPickupResult** class, which contains data about the inventory item, quantity, and success of picking the item up.
- In the example above, we are logging the item's name in both cases.

Adding Listeners for the Events

The included demo shows how to add listeners in the script *VRInventoryExampleSceneController.cs*

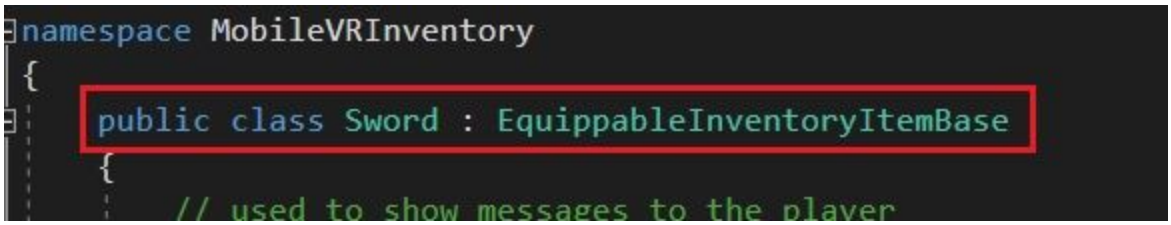
```
void Start() {
    // Listen to VR Inventory events
    VRInventory.onItemSelected.AddListener(ItemSelected);
    VRInventory.onItemPickedUp.AddListener(ItemPickedUp);

void ItemSelected(InventoryItemStack stack) {
    switch (stack.item.name) {
        case "Health Potion" : HandleHealthPotionUse(stack); break;
        case "Mana Potion": HandleManaPotionUse(stack); break;
    }
}

void ItemPickedUp(VRInventory.InventoryItemPickupResult result) {
    if (result.result == MobileVRInventory.VRInventory.ePickUpResult.Success) {
        switch (result.item.name) {
            case "Coin": UpdateCoinTextAnimated(); break;
        }
    } else {
        ShowMessage("You cannot carry anymore of those.");
    }
}
```

Using Equippable Items

This section will explain how to use equippable items such as the sword and the key.

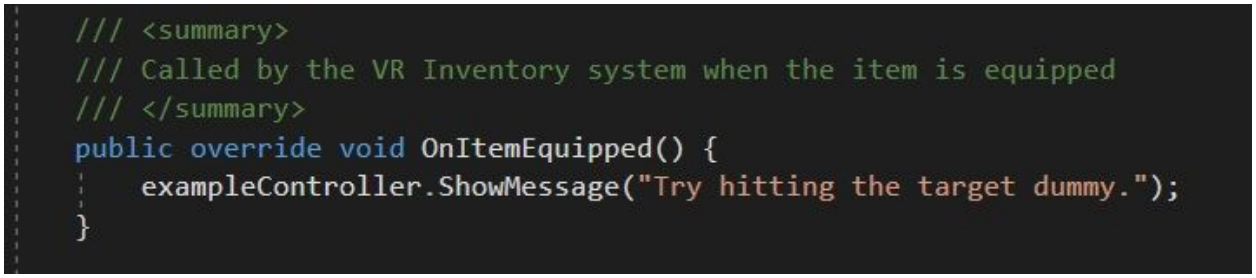
To explain how this works, we will reference the *Sword.cs* script located at */Nurface/VRInventory/Demo/EquippableItems.*

First, the script must extend the *EquippableInventoryItemBase* class:

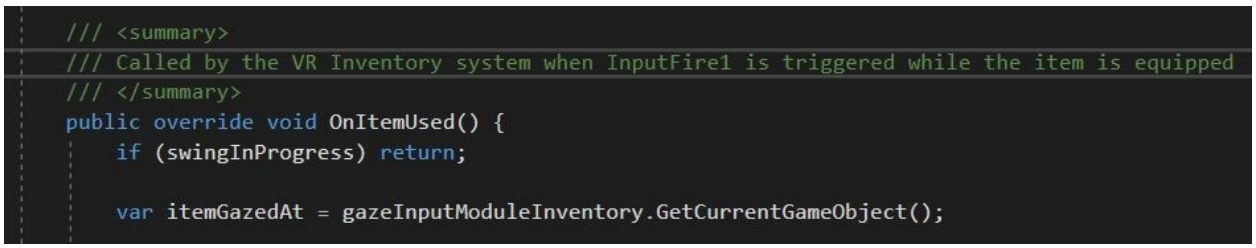


There are three useful events for equippable items:
OnItemEquipped: Called when the item is equipped.
OnItemUnequipped: Called when the item is unequipped.
OnItemUsed: Called when the item is used (when Input.Fire1 is triggered while holding the item)

In the demo, equipping the Sword gives a UI notification:



And using the sword triggers the sword's swing animation, checks distance from the enemy, and call the enemy's 'TakeDamage' method:



Please see *Sword.cs* for more details. The logic for the key is in script *DemoDoorInventory*.

VRInventory Useful Functions

The *VRInventory.cs* class contains functions related to inventory management. Please see the script itself for more details.

public ePickUpResult AddItem(string name, int quantity = 1)
Add the specified item to the inventory. An item with this name must exist in the inventory item database.

public void RemoveItem (string name, int quantity = 1, InventoryItemStack stackToRemoveFrom = null)
Remove the specified item from the inventory. Specify a quantity of -1 in order to remove all items.

public bool HasItem (string name)
Is the specified item currently in the inventory?

public int GetItemQuantity (string name)
Check how many items we are carrying of a particular type.

public void Save (string saveSlotName = null)
Save the inventory via PlayerPrefs.

public void Load (string saveSlotName = null)
Load the inventory via PlayerPrefs.

Help And Support

For a video tutorial related to this asset, please click here:
Intro v0.2: https://www.youtube.com/watch?v=yMZDXc_7KY8
Tutorial v0.2: https://www.youtube.com/watch?v=fhIBiQ5QeKg

Join our VR community here for VR tutorials and videos:
https://www.youtube.com/nurfacegames/

For any questions or support, please email:
nurfacegames@gmail.com

Credits

Sounds:
All credit to artisticdude and MedicineStorm from OpenGameArt.org!
https://opengameart.org/content/rpg-sound-pack
https://opengameart.org/content/superpowers-assets-sound-effects
(http://superpowers-html5.com/)

3D Models:
Credit to Fi Silva and his "Low Poly RPG Item Pack"! This pack includes the Sword, Potions, and Key. Get all the items from the official pack:
https://www.assetstore.unity3d.com/en/#!/content/76088
And please visit his asset store here:
https://www.assetstore.unity3d.com/en/#!/search/page=1/sortby=popularity/query=publisher:25003