OldSchool Grid Inventory v1.2

Contents

Introduction

OldSchool Grid Inventory is a grid inventory system inspired from good-old RPG and Hack&Slash like games.

System is coming with ground and UI item scripts that you can use with your inventory. All scripts are extendable and reusable for your system.

Package hierarchy is below:

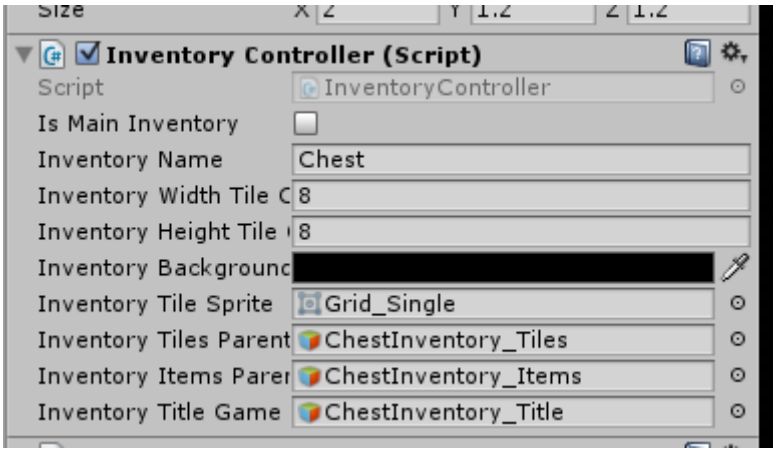
- Demo
    - Prefabs
        - InventoryItemPrefabs
            - HoveringItemCanvas
            - InventoryCanvas
            - LogBoxCanvas
            - PropertyViewerCanvas
        - DemoSceneWorld
    - Demo.scene
    - DemoTerrain
    - ExampleItems.cs
    - BasicMovement.cs
    - Chest.cs
- Resources
    - Prefabs
        - InventoryPrefabs
            - Tile
        - ItemPrefabs
            - Grounded_Item
            - Inventory_Item
    - Sprites
        - Items
            - onexone.png
            - onextwo.png
            - onexthree.png
            - onexfour.png
            - twoxtwo.png
            - twoxfour.png
            - threexone.png
            - threextwo.png
            - threexthree.png
            - threexfour.png
            - fourxone.png
            - fourxtwo.png
            - fourxthree.png
            - fourxfour.png
            - GenericItems.png
            - Single_Tile.png

            - Chest_closed.png
            - ItemPropertyViewImage.png
- Scrips
    - Inventory
        - Ground
            - GroundItem.cs
        - Logic
            - Inventory.cs
            - Item.cs
            - ItemSlot.cs
        - UI
            - FaceToCamera.cs
            - InventoryUI.cs
            - ItemSlotUI.cs
            - ItemUI.cs
    - Managers
        - GameManager.cs
        - InventoryController.cs
        - LogBoxController.cs
        - PropertyWindowController.cs
    - Utility
        - Constants.cs
        - Enums.cs
        - Interfaces.cs
        - RaycastHelper.cs

Thank you for using OldSchool Grid Inventory.

Demo Scene Elements

With the project, there comes a demo scene. In this scene, there are all the elements that one need to construct an Inventory. Using the InventoryController.cs script, user can define an inventory with all the width and height grids he/she wishes.



Demo scene elements are:

World Elements

- **Player:** Contains the Controller script and the camera, also the manager scripts: GameManager, LogBoxController and PropertyWindowController..
- **Terrain:** Sample empty terrain to contain controller and the items and provide gravitation.
- Chest: A 3D box with a sample sprite on it. It has the InventoryController.cs, Chest.cs and a FaceToCamera.cs scripts on it. Chest.cs and FaceToCamera.cs scripts are for demo scene and not the main purpose of this content. It's there for testing multi-inventory features.

UI Elements

- **PlayerInventoryCanvas:** Main inventory canvas for demo scene. Contains Inventory Rect with InventoryController.cs script on it. Inventory Rect has three more children, Inventory_Tiles, Inventory_Items and Inventory_Title. These three gameobjects are used in InventoryController.cs script.
- **ChestInventoryCanvas:** Chest inventory canvas for demo scene. It has the same children as playerinventory canvas, but related InventoryController.cs script is in Chest gameobject located below.
- **LogBoxCanvas:** Simple screen box that shows item's names on screen. Has a Text gameobject to show names on screen.
- **PropertyViewerCanvas:** Contains and image, a header text and a body test. Shows item's name and properties on right click on item.
- **InformationCanvas:** Contains a text object explaining about the key mappings.
- **EventSystem:** Default Unity EventSystem object.

Creating The Inventory

In order to create a new Inventory, user must follow this instructions;

- Create a gameobject to contain InventoryController.cs script. This gameobject must never be disabled in hierarchy.
- Create a Canvas into the scene and put three empty gameobjects as children of it, then reference them approprietaly in the InventoryController.cs.
- Enter the rest of needed information on InventoryController.cs: MainInventory checkbox, name of the inventory, width and height tile counts, background color and Sprite.

User can use the demo scene and prefabs as reference and example for creating the inventory.

Creating Items

There are two create item operations, creating ground and UI items.

Creating Ground Item

Ground items appear in world space and they are interactable in a different way. User should use *Grounded_Item* prefab in order to generate a grounded item. After that, user should call the *Initialize* method of the prefab's GroundItem.cs script. The *Grounded_Item* prefab located in 'Resources/Prefabs/ItemPrefabs/Grounded_Item' path.

```
GameObject groundItem = Instantiate(Resources.Load(InventoryConstants.GroundItemPrefabPath)) as GameObject;
groundItem.transform.position = GameManager.Player.transform.position + (GameManager.Player.transform.forward * 5);
groundItem.GetComponent<GroundItem>().SendMessage("Initialize", item);
```

Creating a Ground Item

Creating UI Item

UI items appear in a 2D UI space and they are interactable in a different way. They can be drag&drap in the inventory. They can be moved into an another inventory too. After generating UI inventory, user should call UIItem's *Initialize* method too. The *Inventory_Item* prefab located in 'Resources/Prefabs/ItemPrefabs/Inventory_Item' path.

```
GameObject itemUI = Instantiate(Resources.Load(InventoryConstants.InventoryItemPrefabPath)) as GameObject;
itemUI.transform.SetParent(GameObject.Find(GameObjectConstants.InventoryItems).transform);
itemUI.transform.localScale = new Vector3(1, 1, 1);
itemUI.GetComponent<ItemUI>().SendMessage("Initialize", item);
```

Creating an UI Item

Scripts

Inventory/Ground/GroundItem.cs

This class is used in Grounded_Item prefab and responsible for initialization and behavior of the world-space ground item.

Inventory/Logic/Inventory.cs

This is one of the main classes in package. Inventory.cs contains most of the logical features of inventory. It is responsible for;

- Checking if there is a proper spot for the new item in inventory,
- Adding items to the queue until the inventory is opened
- Placing items working with *InventoryUI.cs* script.
- Dropping items from inventory

Inventory/Logic/Item.cs

This is the main Item class used by GroundItem prefab. It holds the physical information of related item, which are;

- Id of the item,
- Type,
- Name,
- Width and Height information,
- Sprites

Also the factory methods of the item class resides in here.

Inventory/Logic/ItemSlot.cs

ItemSlot.cs represents one logical slot of the inventory. It holds the index and position information of the slot. *Inventory* class uses this position information to place items.

Inventory/UI/FaceToCamera.cs

This script is used by grounded items, it makes all world-space ground items look to player.

Inventory/UI/InventoryUI.cs

InventoryUI is the MonoBehaviour part of the Inventory.cs logic class. *InventoryUI.cs* helps Inventory.cs class to placing and dropping the items.

Inventory/UI/ItemSlotUI.cs

ItemSlotUI.cs does the same job InventoryUI does for Inventory class. Also it contains two coloring methods using for debugging purposes.

Inventory/UI/ItemUI.cs

ItemUI.cs holds the drag&drop actions of the items. *ItemUI.cs* hosts the Unity's EventTrigger System's;

- Begin Drag
- Drag
- End Drag
- Pointer Enter
- Pointer Exit

methods.

Managers/GameManager.cs

GameManager main controller method of the system. It holds the getter properties of *LogBoxController* and *PropertyWindowController* scripts and acts as a wrapper to their methods. GameManager is also triggers the redraw function if there is been a resize action.

Input handlers are also in the GameManager script.

Managers/InventoryController.cs

InventoryController.cs controls the initiated Inventory object(s) and handles their open/close methods. It is also responsible for item generation operations.

Managers/LogBoxController.cs

LogBoxController.cs controls the LogBox object itself. Show and Hide LogBox methods are in it and it also has the ability to Override the LogBox. With overriding, one can give a message to player without any interaction related operations.

Managers/PropertyWindowController.cs

PropertyWindowController controls the PropertyViewer canvas. Property window canvas appears when player right clicks on any item and shows its name and properties that player define.

Utility/Constants.cs

Constants are holding inventory related constant values, prefab and sprite paths. They are seperated by this classes;

- ScreenConstants
- PlayerConstants
- GameObjectConstants
- InputConstants
- InventoryConstants

Utility/Enums.cs

- ItemsEnum: Items should be added to this enum value after adding to system.
- GroundSpriteCommandEnum: This enum is used in sprite positioning.

Utility/Interfaces.cs

- IItemObject: Shows that the object is an item.
- IInteractable: Shows that the object is an interactable object. Taking items from ground uses this interface to work.

Utility/RaycastHelper.cs

RaycastFromCamera method is used to raycast and locate the item for player to take.

DrawRayFromPlayer method is for debugging purposes.

Demo/ExampleItems.cs

ExampleItems.cs class holds some items the make an example for users.

Demo/Chest.cs

Simple chest behaviour script for demo screen.