XEntity Kit 1.0 Documentation

Documentation for Editor scripts or Unity specific methods like Awake, Start, Update, etc. are not included. Documentation for those are available on the Unity manual.

Table of contents

FAQ

❖ **How do I use the Inventory / Item Container system?**

➢ The inventory system requires setting up the item container UI game objects and they must be set up in a very specific way with very specific namings. A premade inventory prefab is already provided with the kit in the folder:
XEntity GameKit >> Prefabs >> Inventory

- ➢ Add the inventory prefab in the scene or create your own based on the example provided (Must be a children of a canvas because it contains the item container UI)
  - ■ If creating your own, remember to disable the layout group components and content size filter. Otherwise when the container slots are being dragged their positions will be messed up.

- ➢ Now you grab a reference to the ItemContainer and call 'AddItem' and pass in the item you want to add to the container.

❖ **How do I pick up an item GameObject and add it to the inventory?**

- ➢ Attach the Harvestable script to the Item GameObject and assign the Item they should drop and the amount of it or attach the InstantHarvest script if you want only one Item to be added instantly when it is clicked.
  - ■ The scene item GameObject must have a collider.

- ➢ You can write your own interactable detection script or use the interactor script provided which is to be attached to a player or someone who can interact with interactable objects. Adding this script allows the player to pick up Items when left clicked over a Harvestable or InstantHarvest object.

- ➢ You can write your own scripts that derive from the base class Interactable and override the OnInteract method to make custom interactions with interactable objects.

❖ **How do I make custom Items and custom item uses?**

- ➢ If not done already add the provided ItemManager prefab to the scene or create a new GameObject and attach an ItemManager to it.

- ➢ Go into the Item script and add custom data to your needs.

  - ■ Some of the data are essential that must not be removed from the Item script because other systems in the kit depend on them. Essential Data Includes:

    - ◆ public ItemType type
    - ◆ public string itemName
    - ◆ public Sprite Icon
    - ◆ public int ItemPerSlot

- public GameObject prefab

  - You can add more unique item types in the ItemType enum if you want more unique functionalities. For example: you might add a new type called Armor if you wanted armor equipment functionalities.
  - Create Item scriptable objects to define a new item. Create as many unique items as you want.
  - Once you're done creating your items, select all of them, you can add them to the item list in the ItemManager object manually or automatically > select all the item scriptable objects, right click and select "Add to ItemList" from the menu. Don't worry about redundant elements when using the automated function, duplicates won't be added.
  - To add custom use events
    - Go to the ItemManager script and create a new method for your custom item use that takes in an ItemSlot as parameter which will be the inventory slot the item is being contained in. Add your custom use events for the item in that method. Make sure to remove the item from the slot by calling slot.Remove if your item is supposed to be permanently gone after the use.
    - In the UseItem method add your ItemType case in the switch statement and call your custom use method for that case and pass in the slot as the argument which is a parameter of the UseItem method.

- **How do I save and load the item container data between game sessions/scenes?**
  - Call ItemContainer.SaveData(*"some unique id"*) to save
  - Call ItemContainer.LoadData*("some unique id"*) to load
  - If two item containers are saved with the same id, data will be overwritten. Make the ID something unique and persistent to the container because that same ID will need to be used to load that data for the container.
  - Best way to ensure unique ID is to have an array/List of all the ItemContainers in the scene, loop through them and Save/Load with the array index as their ID.

---

namespace XEntity

★ All XEntity scripts and tools are contained within this namespace.

---

class ItemContainer : Monobehaviour

★ This script must be attached to the UI object of the container. The children of the UI object should be the container UI.
★ The gameobject with an ItemContainer attached must be active when the game starts, it will deactivate itself after initializing all the variables.

Attributes and Properties

- **public Transform carrier**
  - This should be the transform of the object the container is linked to or the transform of the owner of the container. For example: the carrier for a player inventory would be the player.

- **public KeyCode UIToggleKey**
  - When this key is pressed the container UI is toggled on/off.

- **public bool dropRemovedItemPrefabs**
  - If true, when an item is removed from a slot, an item prefab will be instantiated in front of the carrier which can be picked up again.
  - if false, when an item is removed from a slot, it will be permanently gone with no reference left.

- **private ItemSlot[] slots**
  - This is an array of all the slots of the container.

- **private Transform containerUI**
  - The corresponding UI for this container which contains all the slots & container menus.
  - The 'container UI' must be set up in the exact way and with the exact names as in the example 'container UI' prefab that comes with the kit.

- **private GameObject slotOptionsUI**
    - This is the UI GameObject of the slot options menu.

- **private Button itemUseButton**
    - This button is a child of the slotOptionsUI.

- **private Button itemRemoveButton**
    - This button is a child of the slotOptionsUI.

---

protected virtual void InitContainer(Transform container)

- Takes in the transform of the container UI.
- This method initializes the UI variables and the container slots.

---

public bool AddItem(Item item)

- returns true if the container adds/stacks the item in one of the slots.

---

public ContainsItem(Item item)

- returns true if the container contains the provided item in any of the slots.

---

public bool ContainsItemQuantity(Item item, int amount)

- returns true if the container contains the specified amount or more of the provided item in the slots.

---

private void ToggleUI()

- Activates/Deactivates the container UI.

---

private void OnSlotClicked(ItemSlot slot)

- This method is called when the slot UI button is clicked.
- By default it activates/deactivates the slot options menu and adds new listeners to the menu buttons for the slot passed in.

---

private void OnRemoveItemClicked(ItemSlot slot)

- This method is called when the remove item button is clicked for the slot.

---

public void SaveData(string id)

- Saves all the container slot item data using Untiy's JsonUtility.
- The data directory is acquired by calling GetIDPath(id).
- If data exists with the same id, it will be overwritten.

---

public void LoadData(string id)

- Loads all the container slot item data of the provided id using Unity's JsonUtility.
- The data directory is acquired by calling GetIDPath(id).

---

protected virtual string GetIDPath(string id)

- returns the universal path for saving/loading data of the provided id.

---

class ItemManager : Monobehaviour

★ This class has a singleton pattern implemented.

★ This must exist in a scene if using any of the item and item container systems.

Attributes and Properties

- **public static ItemManager Instance { get; private set; }**
    - This is the static instance of the class assigned in the singleton pattern.

- **public List<Item> itemList**
    - All the items in the game must be added to this list either manually or using the MenuItem function 'Add to ItemList'.
    - If adding items manually, do not add duplicates as it will add inefficiency in the code. Adding items with the MenuItem function won't add any duplicates.

---

public void UseItem(ItemSlot slot)

- This method is called when the 'use item' button is clicked on the slot's options menu.
- This method is not called if the slot is empty.
- The appropriate use method is called based on the type of item in the slot.

---

public void ConsumeItem(ItemSlot slot)

- This method is called when the 'UseItem' method is called, if the slot item type is Consumable.
- The item is removed from the slot once its used.

---

public void EquipItem(ItemSlot slot)

- This method is called when the 'UseItem' method is called, if the item type is ToolOrWeapon.
- The item will be equipped and moved to the equipment slot.
- If the equipment slot already contains an item, the items will be swapped.

---

public void PlaceItem(ItemSlot slot)

- This method is called when the 'UseItem' method is called, if the slot item type is Placeable.

- The player enters builder mode allowing the items to be removed if placed.

---

public void DefaultItemUse(ItemSlot slot)

- This method is called when the 'UseItem' method is called, if the slot item type is Default.
- By default it just prints out "Using [item name]." to the console

---

public Item GetItemByIndex(int index)

- Returns the Item from the itemList with the index.

---

public Item GetItemByName(string name)

- Returns the Item from the itemList with the name.

---

public int GetItemIndex(Item item)

- Returns the corresponding index of the item from the itemList.

---

class ItemSlot: Monobehaviour

★ This class must be attached to a slot UI object with a very specific setup with specific names. A premade slot UI object prefab comes with the kit.

Attributes and Properties

- Item SlotItem { get; private set; }

    - This is the item that's contained in this slot, the value is null if the slot is empty.

- **public int ItemCount { get; private set; }**
    - The quantity of the item contained in this slot.
    - The slot is empty if this value is zero.

- **public bool IsEmpty { get; }**
    - This returns true if the ItemCount is zero.

- **private UnityEngine.UI.Image iconImage**
    - This is the image of the item slot icon
    - This object is a child of the item slot UI.

- **private UnityEngine.UI.Text countText**
    - This is the text that shows the item count of the slot.
    - This object is a child of the item slot UI.

---

public bool Add(Item item)

- Tries to add the provided item and returns true if is able to add/stack the item to the slot.

---

public void RemoveAndDrop(int amount, Vector3 dropPosition)

- Removes the passed in amount of item and instantiates an item prefab at the dropPosition for each of the items.

---

public void Remove(int amount)

- Removes the passed in amount of items.

---

public void Clear()

- Removes all the items from the slot.

---

public void ClearAndDrop(Vector3 dropPosition)

- Removes all the items from the slot and instantiates an item prefab at the dropPosition for each of the items.

---

private bool IsAddable(Item item)

- returns true if the item can be added or stacked

---

private void OnSlotModified()

- This method is called every time a value in the slot is changed.

---

class ItemSlotUIEvents: Monobehaviour

★ This class should be attached to the slot UI object.

Attributes and Properties

- **private ItemSlot mySlot**
  - This is the ItemSlot this script is attached to.
- **private UnityEngine.UI.Image slotUI**
  - Image of the slot object this script is attached to.
- **private Vector3 dragOffset**
  - The offset of the slot UI from the mouse cursor position when being dragged.

- **private Vector3 origin**
  - The original position of the slot UI.
- **private Color regularColor**
  - The regular color of the slot UI image.
- **private Color dragColor**
  - The color of the slot UI when being dragged.

---

public void OnDrop()

- This method is called if a dragging slot is dropped on top of another slot UI.
- Tries to transfer/swap the items of the slots.

---

class Item: ScriptableObject

★ A base class for all items.
★ Asset Menu Path : "XEntity/Item"

---

abstract class Interactable: Monobehaviour

★ A base class for all interactable objects.

---

public virtual void OnInteract(Interactor interactor)

- This method is called when an Interactor tries to interact with this object.

---

class Harvestable: Interactable

★ Inherits from interactable.
★ This is attached to any object that can be harvested for items.

Attributes and Properties

- public HarvestDrop[] harvestDrops
    - This is an array of all the harvest drops that can be dropped when this object is harvested.

- public float HP
    - The hitpoints of the harvestable object.
    - The object is harvested when this value is zero.

---

public override void OnInteract(Interactor interactor)

- This is called when the player interacts with the harvestable by clicking on it.

---

private void Harvest(Interactor interactor)

- This method is called when the HP hits zero.
- The object is destroyed after the drops are harvested.

---

class InstantHarvest: Interactable

★ Inherits from interactable.
★ This is attached to any object that can be harvested for items.

Attributes and Properties

- public Item harvestItem
    - This item is added when an Interactor interacts with this object.

---

public override void OnInteract(Interactor interactor)

- This method is called when this object is interacted with by the interactor.

---

class Interactor: Monobehaviour

★ This script is attached to any entity in the game that can interact with an interactable object.

Attributes and Properties

- [SerializeField] private float interactionRange
  - The minimum distance the interactor must be to an object to interact.
- [SerializeField] private Camera mainCamera
  - The camera the game is mainly viewed from.
- [SerializeField] private ItemContainer inventory
  - This is the inventory of the interactor entity.
  - When the interactor tries to add an item, it will be added to this container.
- private Intractable interactionTarget
  - The interaction target. Value is null if there is no target.

---

private void HandleInteractions()

- Handles the interactions and target detection.
- This method is called every frame.

---

private bool InRange(Vector3 targetPosition)

- returns true if the distance between the targetPosition and the interactors position is less than or equal to the interactionRange.

---

private void InitInteraction()

- This method is called when an Interactor tries to interact with the interactionTarget.

---

public Vector3 ItemDropPosition { get }

- Returns the ideal drop position for an item which is one unit vector away from the interactors current position in the interactors forward direction.

---

public void AddToInventory(Item item, GameObject instance)

- Tries to add the item to the inventory and if successful the GameObject instance of the item is destroyed.

---

class ItemCollector: Monobehaviour

★ This script is attached to an item collector GameObject.
★ Collectors are intermediate item container objects that are instantiated when an item is dropped and the collector holds a reference to the dropped item.
★ When an interactors collider comes in contact with a collector, the item in the collector attempts to add itself to the interactors inventory.
★ If the item is successfully added, the collector is destroyed.

---

enum ItemType

★ An enum for all the possible types of items.

---

readonly struct Utils

★ This struct contains various helper functions.

---

public static void TransferItem(ItemSlot trigger, ItemSlot target)

- Tries to transfer items from trigger to target, if they are unable to be transferred the slots are swapped and stacked

---

public static IEnumerator TweenScaleIn (Transform obj, float durationInFrames, Vector3 maxScale)

- Activates obj.gameobject.
- After activating it lerps the obj's localScale from Vector3.zero to maxScale in the period of durationInFrames.

---

public static void HighlightObject(GameObject obj, Color highlightColor)

- Changes the material color of the passed in object to the highlight color.
- The object must have a valid mesh renderer and material.

---

public static void Unhighlight(GameObject obj, Color original)

- Changes the material color of the passed in object to the original color.
- If no original color is passed in, by default it's set to Color.white.

---

public static IEnumerator TweenScaleOut(Transform obj, float durationInFrames)

- Lerps the obj's localScale from its current scale to Vector3.zero in the period of durationInFrames.
- After the lerping is done the obj.gameobject is deactivated.

---

struct HarvestDrop

★ This structure holds the data of a harvestable item.

If there's any question regarding any of the code contact me in discord: shariar#1029