HW Assignment 2: Inventory System

Purpose

This assignment is to learn how to create an inventory system which involves UI, usage of non-Monobehaviour class (i.e., Item.cs), and inventory management.

Note Before You Start

This is a not-that-easy assignment so start early. Seek for help in work & help class session or office hours.

Assignment Reference & Credit

This assignment is based on the following Brackeys' ***Making an RPG in Unity*** tutorial series. Before you work on the assignemnt, please watch the tutorials as preparation:

- MOVEMENT - Making an RPG in Unity (E01) https://youtu.be/S2mK6KFdv0I
- INTERACTION - Making an RPG in Unity (E02) https://youtu.be/9tePzyL6dgc
- INTEGRATION 01 - Making an RPG in Unity (E03) https://youtu.be/COckHIIO8vk
- ITEMS - Making an RPG in Unity (E04) https://youtu.be/HQNl3Ff2Lpo
- INVENTORY UI - Making an RPG in Unity (E05) https://youtu.be/w6_fetj9PIw
- INVENTORY CODE - Making an RPG in Unity (E06) https://youtu.be/YLhj7SfaxSE

Package Preparation

In Unity go to menu Window -> ***Package Manager***, select the ***Unity Registry*** category, then search for ***2D Sprite*** package and install it.

This package helps to edit the sprites as needed.

Assets Preparation

Download the Assets.zip from BB. Unzip it. Drag those assets to your project.
Feel free to use your own assets if you'd like.

For the sprites in the Images folder, configure their Import Settings as following

Sprite	***Description***	***Import Settings***	
PanelBackground	A sprite with border, for inventory panel background	**Texture Type: Sprite(2D and UI)**	**Use Sprite Editor to set up border**
SlotBackground	A sprite with border, for item slot background		
CloseX	a sprite with the closing icon		
ItemPlaceHolder1	item 1's icon image		
ItemPlaceHolder2	item 2' icon image		

Unity Project Setup

Add *Inventory* to the Input Manager

Go to menu *Edit -> Project Setting -> **Input Manager***.
In the Input Manager, ***expand the Axes***, ***increase the Size by 1*** (you will get one more item **at the end of the list**).
Expand the new item (on the bottom of the list), Name it ***Inventory***, assign the Positive Button as ***i***, Alt Positive Button as ***b***

Basic In-Game Objects & Environment

Note that this is a point-and-click player control system. To control the Player, click somewhere in the game world and the Player will find path and walk toward it.

Follow the instructions below to setup the environment and control.

1. EventSystem

Make sure there is an EventSystem object in the Hierarchy. If there is not one, + -> UI -> EventSystem

2. Create Walkable Surface for NavMeshAgent

Create some surface such as terrain, plane, cube (flattened), etc. Make sure the surface objects are ***Navigation Static***. Set their **Layer** as **Ground**.
Then go to menu ***Window -> AI -> Navigation (…)***, in the Inspector

- Select the ***Bake*** tab, then click ***Bake***. If the baking is done successfully, you'll see blue tint appeared on walkable surface.
- If you need to specify which objects are walkable, do it in the Object tab, then **rebake**.

3. Layers

Add this new Layer: Ground
Set your ground object(s)'s Layer as Ground.

4. Player

Technically any player. If you need to create one, follow the steps below:

- Create an ***Empty*** and name it as ***Player*** – this serves as the parent of player graphic and whatever stuffs.
- Tag it as Player
- Create a shape (i.e., cube) and name it as ***PlayerGraphic*** – this can be substituted by anything. Set it as child of Player. Remove the collider component from it.
- Add the following **components** to the ***Player***:
    - **NavMeshAgent**
    - Any **Collider**, resize and position of the collider so it rough matches the graphic.
    - **Rigidbody**, **Is Kinematic**
    - **PlayerController** (script)
        - Movement Mask: Ground

- **PlayerMotor** (script)

5. Camera

- ***Using Cinemachine virtual camera***
  - ***Target: Player***

At this point if you play the game, LMB click somewhere in the game world, the player should walk toward the clicked point.

Inventory UI Hierarchy & Description

Note before you start:

- *Set the Game view as a fixed aspect ratio instead of Free Aspect*
- *Look at **Game view** while you're working on UI.*
- *Use Anchor Presets wherever you can.*
- *Rect Transforms are based on parent object.*
- *UI elements use **Draw Order** of the Hierarchy order.*

The UI elements hierarchy and their types and descriptions:

> **Canvas**

> **Inventory** — ***Panel***, use panel background image[1], Anchor Presets, adjust width and height

> **ItemsParent** — ***Empty*** game object, with ***GridLayoutGroup*** component[2]

> **InventorySlot** — ***Empty*** game object, serve as parent for one slot, `InventorySlot script`

> **ItemButton** — ***Panel, Button*** component[3], slot background image[1], ***OnClick***() link to InventorySlot-- ***UseItem***

> Icon — ***UI Image***, use an item icon such as ItemPlaceHolder1

> **RemoveButton** — ***UI Image,*** X sprite, ***Button*** component[3], ***OnClick***() link to InventorySlot -- ***OnRemoveButton***

> (more inventory slots from prefab)
> (more inventory slots from prefab)
> (more inventory slots from prefab)

> **Title** — ***Panel***

> **Text** — ***UI Legacy Text***

[1] Adjust the *Pixels Per Unit* to get proper border width
[2] Modify the GridLayoutGroup component attributes to get desired layout of multiple slots
[3] Set Navigation to None

Once you make UI elements for one InventorySlot, go selectthe InventorySlot object, in the Inspector for the Inventory Slot component, drag its grandchild Icon to the Icon field, and the child RemoveButton to the Remove Button field.
play test and make sure you can click the slot and the remove button (although nothing will happen yet).
Prefab the InventorySlot, and then duplicate it to get as many as you like. Don't align them manually though. The alignment will be controlled by the ***GridLayoutGroup*** component on the ***ItemsParent*** object. Configure that component to get your desired layout.
By default, we don't want to display any item or the remove button. To hide them by default:

- Find the ***Icon***, for the ***Image*** component, set the ***Source Image*** to ***None*** and **disable the Image component**.
- Find the ***RemoveButton***, for the ***Button component*** uncheck the Interactable.
- Select the InventorySlot and Overrides the Prefab.

Create Interactable Items

1. **First**, create items **in assets** (i.e., do this step in your Project panel):
   - Right-click in assets folder, ***-> Create -> Inventory -> Item***, you will get a new (scriptable) Item created
   - Give it a name (such as TestItem),
   - Assign a sprite for Icon.
2. **Then**, create ***real item in the Scene*** / Hierarchy (for example, you can create a shape like Sphere as a real item).
   - ***Attach the ItemPickup script***, then drag the corresponding item from your assets to the ***Item*** field of the ItemPickup component.
3. Prefab and duplicate as many items as you wish

Inventory Manager

Create an ***empty game object, name it as InventoryManager, add*** the following components:

- ***Inventory*** (script)
  - Set the Space as the number of Inventory Slots you have.
- ***InventoryUI*** (script),
  - Assign ***Items Parent*** by dragging the ItemsParent from Hierarchy to the field
  - Assign ***Inventory UI*** by dragging the Inventory (under the inventory canvas) from Hierarchy to the field.

Hooray! Enjoy it!

When playing, pressing I or B key can show / hide the inventory UI.

LMB click the game world, Player will walk to the clicked point.

When an item is on focus (i.e., the player is close to the item, closer than the Item Pickup's Radius)

Other Assets Credit

The PanelBackground.jpg was originally from rawpixel.com on Freepik
The SlotBackground.jpg was originally from Freepik and by nuart_design