

DOC MADE 10th August 2023.

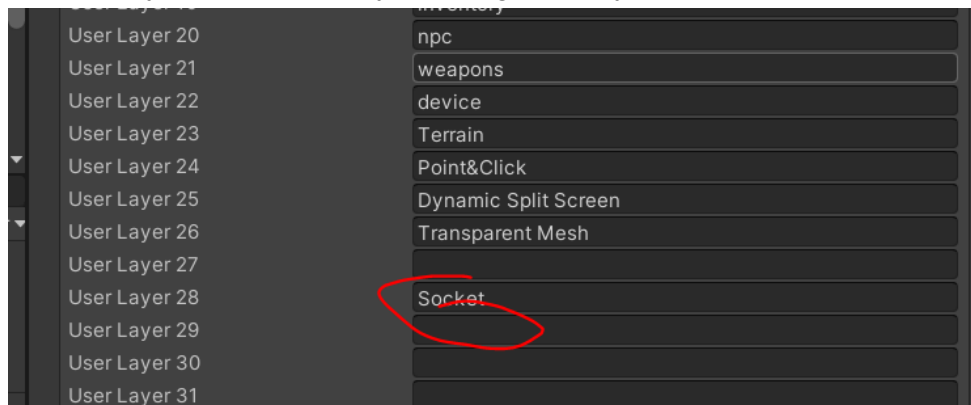
Follow these steps:

-You can import then in any other order or if you have already GKC in your project, to import EBS after, or viceversa (remember to make backups before anything else)

-Import EBS

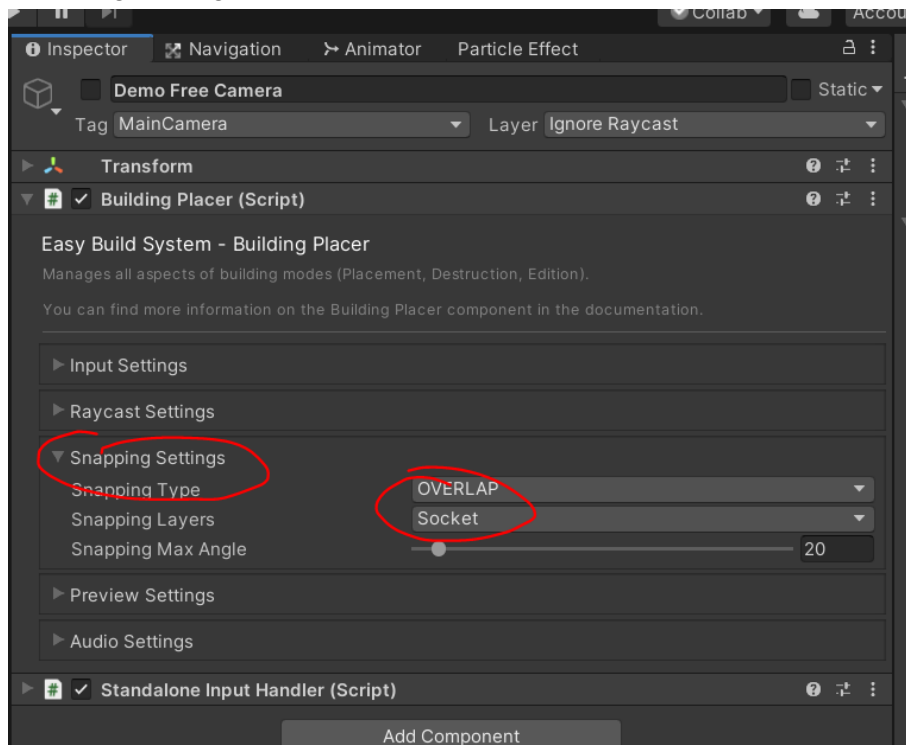
-Import GKC

-Add the layer "Socket" on layer settings of unity



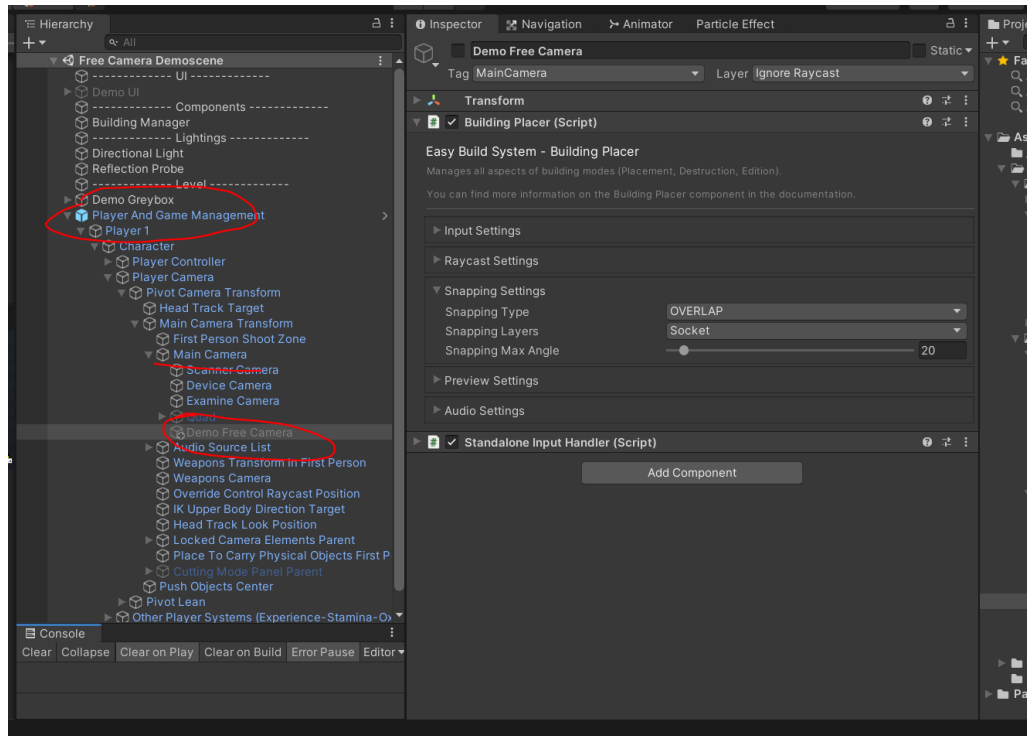
-Open the demo scene called "Free Camera DemoScene" from EBS

-Configure the Socket layer on the Building Placer of Demo Free Camera object, on its Snapping Setting section

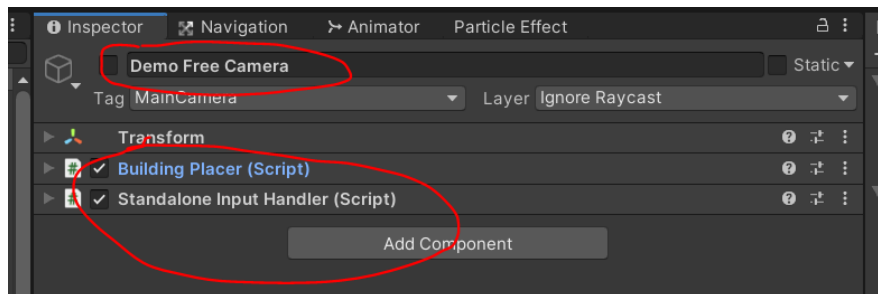


-Drop the Player and Game Management Prefab from GKC in scene

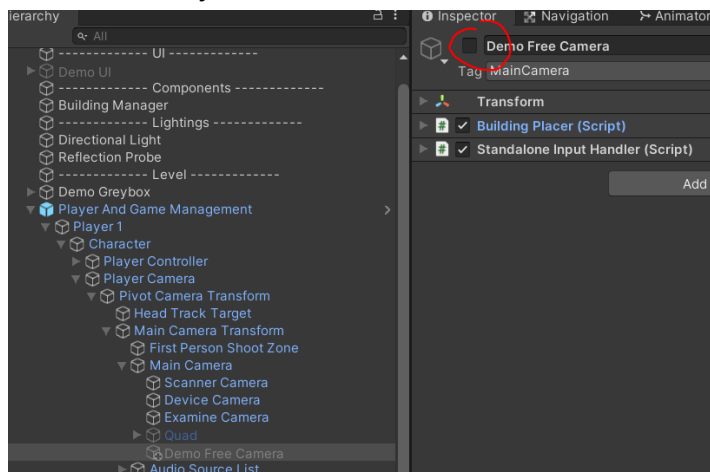
-Take the object of EBS “Demo Free Camera”, and move it inside the main camera of GKC, inside the Player Camera gameObject



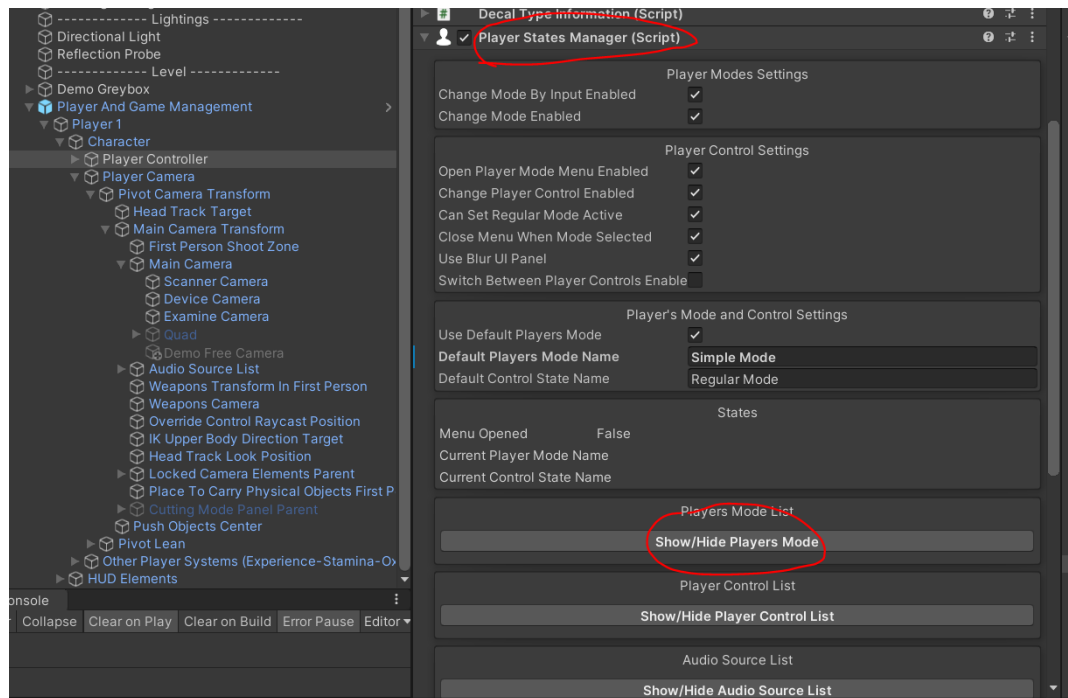
-Remove the component Camera and Audio Listener from Demo Free Camera object and leave only these attached



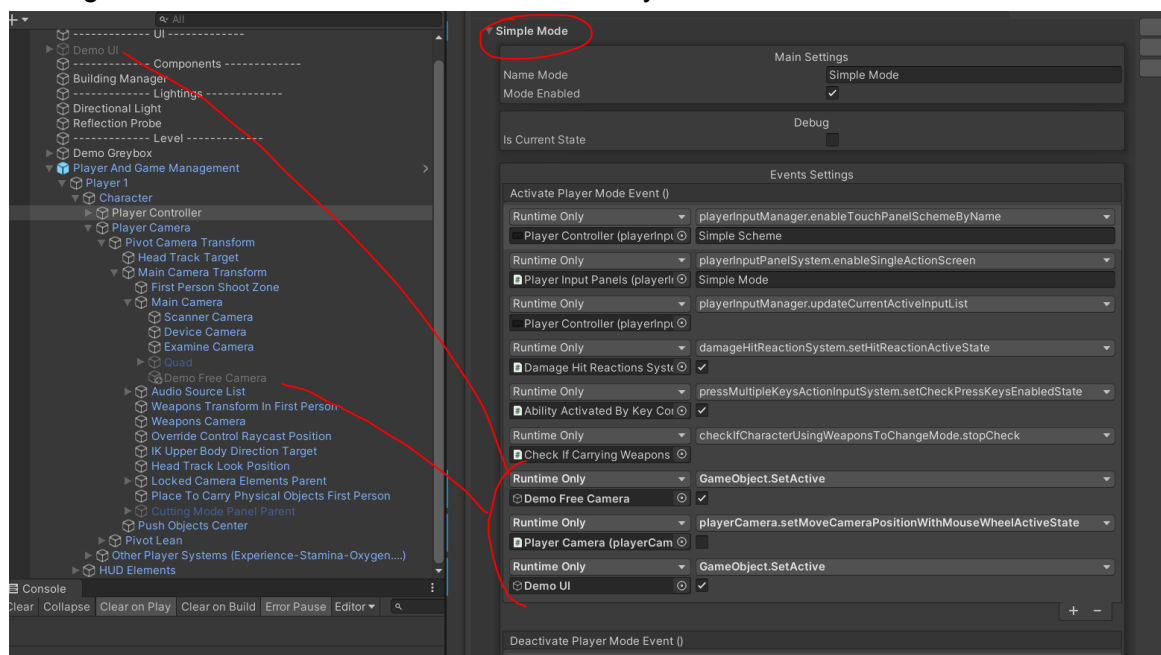
-Disable the object Demo Free Camera



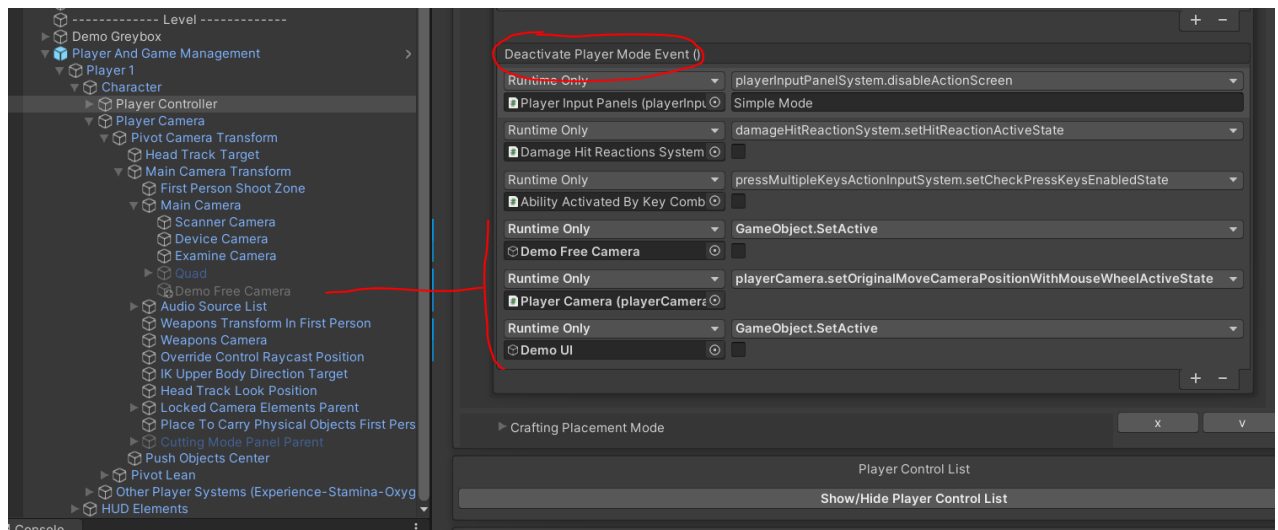
-Go to the Player States Manager on Player Controller gameObject and either add a new player's mode or use the Simple Mode one to configure these events to toggle the Demo Free Camera object, in order to use the placement mode of EBS



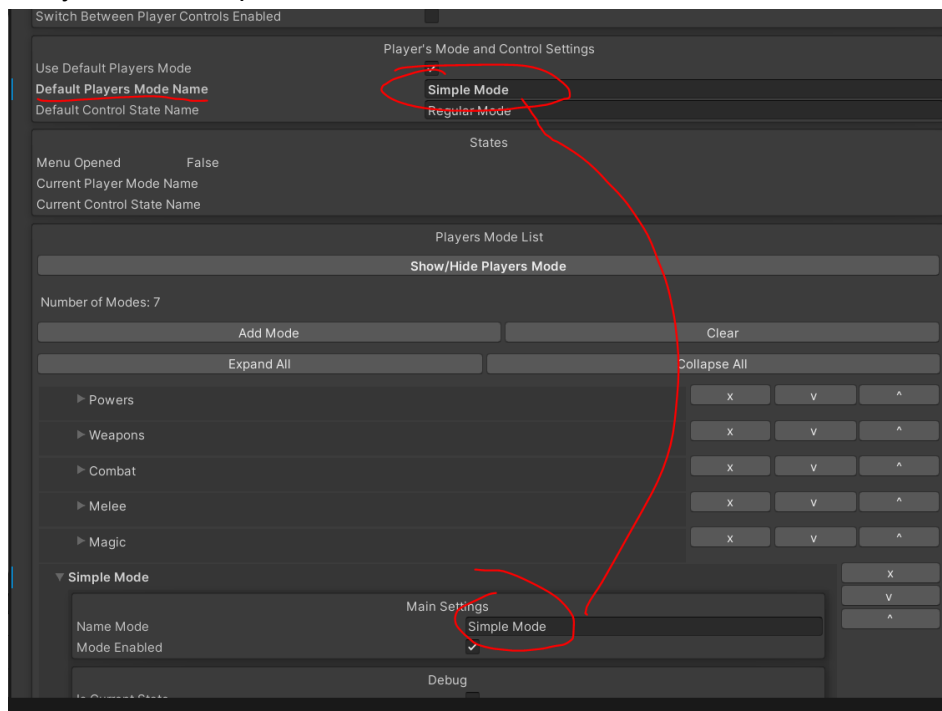
-Configure these extra events on the Activate Player Mode Event section



-Configure these extra events on the Deactivate Player Mode Event section



-Set the Players Mode to start on Default Players Mode Name for the one that you created or if you used Simple Mode



And that would be all to have the systems working in place.

You can use H key by default to change between the different player's modes

You can see more info about this player's mode system here:

<https://www.youtube.com/watch?v=bvd8zv73euA>

INTEGRATE GKC INVENTORY SYSTEM WITH EBS

A further integration to consume inventory objects from the player when placing objects on EBS and check if there are available inventory objects for it can be handled by adding the next code to the next scripts of GKC and EBS:

Code on GKC:

-Open getObjectFromInventorySystem.cs script:

-Inside this script, if the function called “useInventoryObjectAndGetResult” is not present, add it after the function “useInventoryObject” with the following code (if there is already a function called like that, ignore this step on the GKC script):

```
public bool useInventoryObjectAndGetResult (int amountToUse)
{
    if (mainInventoryManager == null && checkInventoryManager) {
        return false;
    }

    int remainAmount = 0;

    if (checkInventoryManager) {
        remainAmount = mainInventoryManager.getInventoryObjectAmountByName
(inventoryObjectName);
    }

    if (remainAmount < 0) {
        remainAmount = 0;
    }

    if (useInfiniteObjects) {
        remainAmount = 1;
    }

    remainAmount += extraInventoryObjectAmount;

    if (remainAmount >= amountToUse) {
        if (checkInventoryManager) {
            if (extraInventoryObjectAmount == 0) {

                mainInventoryManager.removeObjectAmountFromInventoryBy
Name (inventoryObjectName, amountToUse);
            }
        }

        eventOnAmountAvailable.Invoke ();
    }
}
```

```

        if (showDebugPrint) {
            print ("using event on amount available");
        }

        return true;
    } else {
        eventOnAmountNotAvailable.Invoke ();

        if (showDebugPrint) {
            print ("using event on amount not available");
        }

        return false;
    }
}

```

END OF MODIFICATIONS ON getObjectFromInventorySystem.cs SCRIPT

Code on EBS:

-Open BuildingPlacer.cs script:

-Before the line with “region Events” on that script, add the following fields:

```

public bool useExternalCheckConditionToPlaceBuildingPart;
public getObjectFromInventorySystem mainGetObjectFromInventorySystem;

```

```

public bool storeDestroyObjectOnInventory;
public inventoryManager mainInventoryManager;

```

```

public bool useEventIfNotInventoryObjectLocated;
public UnityEvent eventIfNotInventoryObjectLocated;

```

-Then, replace the function PlacingBuildingPart with the next code:

```

public virtual bool PlacingBuildingPart()
{
    if (!HasPreview())
    {
        return false;
    }
}

```

```

    }

    if (!CanPlacing)
    {
        return false;
    }

    if(useExternalCheckConditionToPlaceBuildingPart){
        string currentObjectName = m_CurrentPreview.GetGeneralSettings.Name;

        mainGetObjectFromInventorySystem.setNewInventoryObjectName(currentO
        bjectName);

        if(!mainGetObjectFromInventorySystem.useInventoryObjectAndGetResult(1))
        {
            if(useEventIfNotInventoryObjectLocated){
                eventIfNotInventoryObjectLocated.Invoke();
            }

            return false;
        }
    }

    return confirmPlaceBuildingPart ();
}

```

-After that function add the next one:

```

public bool confirmPlaceBuildingPart(){

    BuildingManager.Instance.PlaceBuildingPart(GetSelectedBuildingPart,
        m_CurrentPreview.transform.position,
        m_CurrentPreview.transform.eulerAngles,
        m_CurrentPreview.transform.localScale);

    if (m_LastBuildMode == BuildMode.EDIT)
    {
        ChangeBuildMode(BuildMode.EDIT, true);
    }
    else
    {
        CancelPreview();
    }

    if (m_AudioSettings.AudioSource != null)
    {

```

```

        if (m_AudioSettings.PlacingAudioClips.Length != 0)
        {
            m_AudioSettings.AudioSource.PlayOneShot(m_AudioSettings.PlacingAudioClips[Random.Range(0,
                m_AudioSettings.PlacingAudioClips.Length)]);
        }
    }

    return true;
}

```

-Also, to take into account that removed pieces of scene of EBS is handled, add inside the function DestroyBuildingPart these lines before the line
 “BuildingManager.Instance.DestroyBuildingPart(m_CurrentPreview);”:

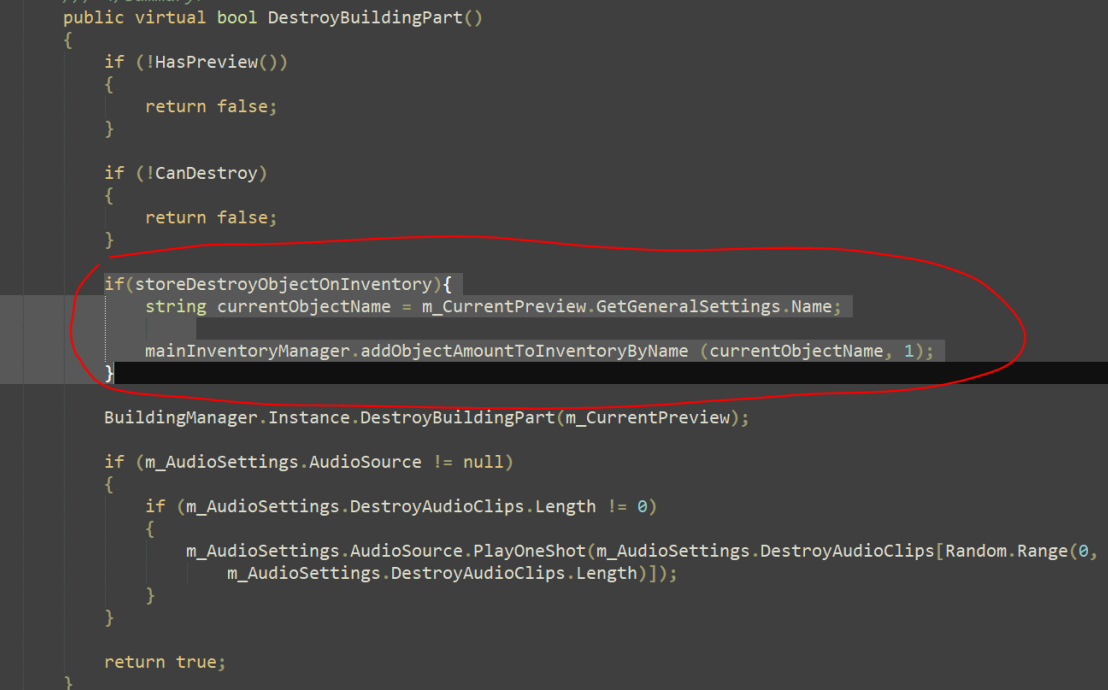
```

if(storeDestroyObjectOnInventory){
    string currentObjectName = m_CurrentPreview.GetGeneralSettings.Name;

    mainInventoryManager.addObjectAmountToInventoryByName (currentObjectName,
    1);
}

```

-So it looks like this:



```

// Summary
public virtual bool DestroyBuildingPart()
{
    if (!HasPreview())
    {
        return false;
    }

    if (!CanDestroy)
    {
        return false;
    }

    if(storeDestroyObjectOnInventory){
        string currentObjectName = m_CurrentPreview.GetGeneralSettings.Name;
        mainInventoryManager.addObjectAmountToInventoryByName (currentObjectName, 1);
    }

    BuildingManager.Instance.DestroyBuildingPart(m_CurrentPreview);

    if (m_AudioSettings.AudioSource != null)
    {
        if (m_AudioSettings.DestroyAudioClips.Length != 0)
        {
            m_AudioSettings.AudioSource.PlayOneShot(m_AudioSettings.DestroyAudioClips[Random.Range(0,
                m_AudioSettings.DestroyAudioClips.Length)]);
        }
    }

    return true;
}

```

END OF MODIFICATIONS ON BuildingPlacer.cs SCRIPT

-Open BuildingPlacerEditor.cs script:

-Before the first "#endregion" line, add these fields:

```
SerializedProperty useExternalCheckConditionToPlaceBuildingPart;  
SerializedProperty mainGetObjectFromInventorySystem;
```

```
SerializedProperty storeDestroyObjectOnInventory;  
SerializedProperty mainInventoryManager;
```

```
SerializedProperty useEventIfNotInventoryObjectLocated;  
SerializedProperty eventIfNotInventoryObjectLocated;
```

-At the end of the function OnEnable, add these lines:

```
useExternalCheckConditionToPlaceBuildingPart = serializedObject.FindProperty  
("useExternalCheckConditionToPlaceBuildingPart");  
mainGetObjectFromInventorySystem = serializedObject.FindProperty  
("mainGetObjectFromInventorySystem");
```

```
storeDestroyObjectOnInventory = serializedObject.FindProperty  
("storeDestroyObjectOnInventory");  
mainInventoryManager = serializedObject.FindProperty ("mainInventoryManager");
```

```
useEventIfNotInventoryObjectLocated = serializedObject.FindProperty  
("useEventIfNotInventoryObjectLocated");  
eventIfNotInventoryObjectLocated = serializedObject.FindProperty  
("eventIfNotInventoryObjectLocated");
```

-At the end of the script, before the line "if (GUI.changed)", paste these lines:

```
EditorGUILayout.Space ();
```

```
EditorGUILayout.PropertyField (useExternalCheckConditionToPlaceBuildingPart);  
if(useExternalCheckConditionToPlaceBuildingPart.boolValue){  
    EditorGUILayout.PropertyField (mainGetObjectFromInventorySystem);  
}
```

```
EditorGUILayout.Space ();
```

```
EditorGUILayout.PropertyField (storeDestroyObjectOnInventory);  
if(storeDestroyObjectOnInventory.boolValue){  
    EditorGUILayout.PropertyField (mainInventoryManager);  
}
```

```
EditorGUILayout.Space ();
```

```

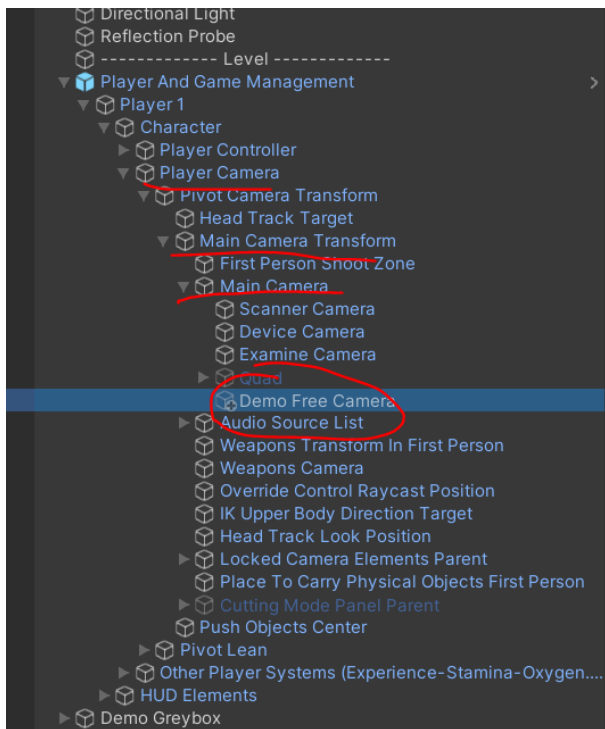
EditorGUILayout.PropertyField (useEventIfNotInventoryObjectLocated);
if(useEventIfNotInventoryObjectLocated.boolValue){
    EditorGUILayout.PropertyField (eventIfNotInventoryObjectLocated);
}

```

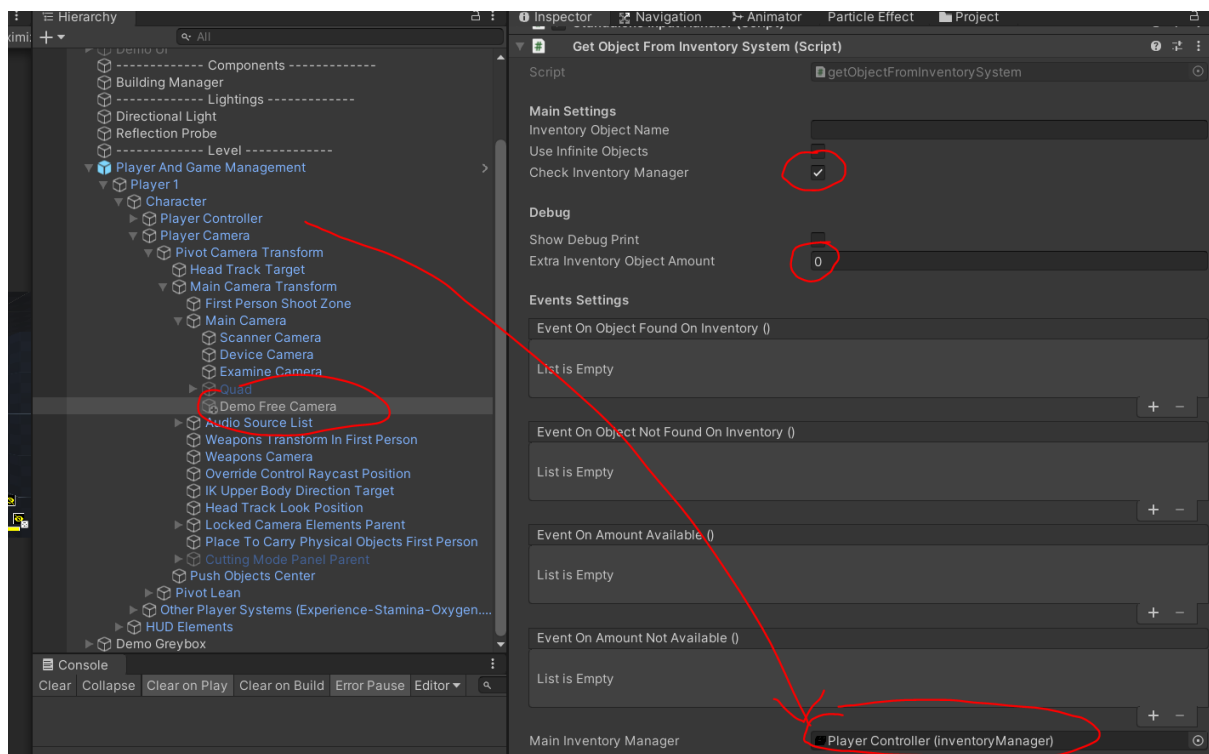
END OF MODIFICATIONS ON BuildingPlacerEditor.cs SCRIPT

And that is all on the code part. Now, it needs a couple of settings on the inspector to use this new option.

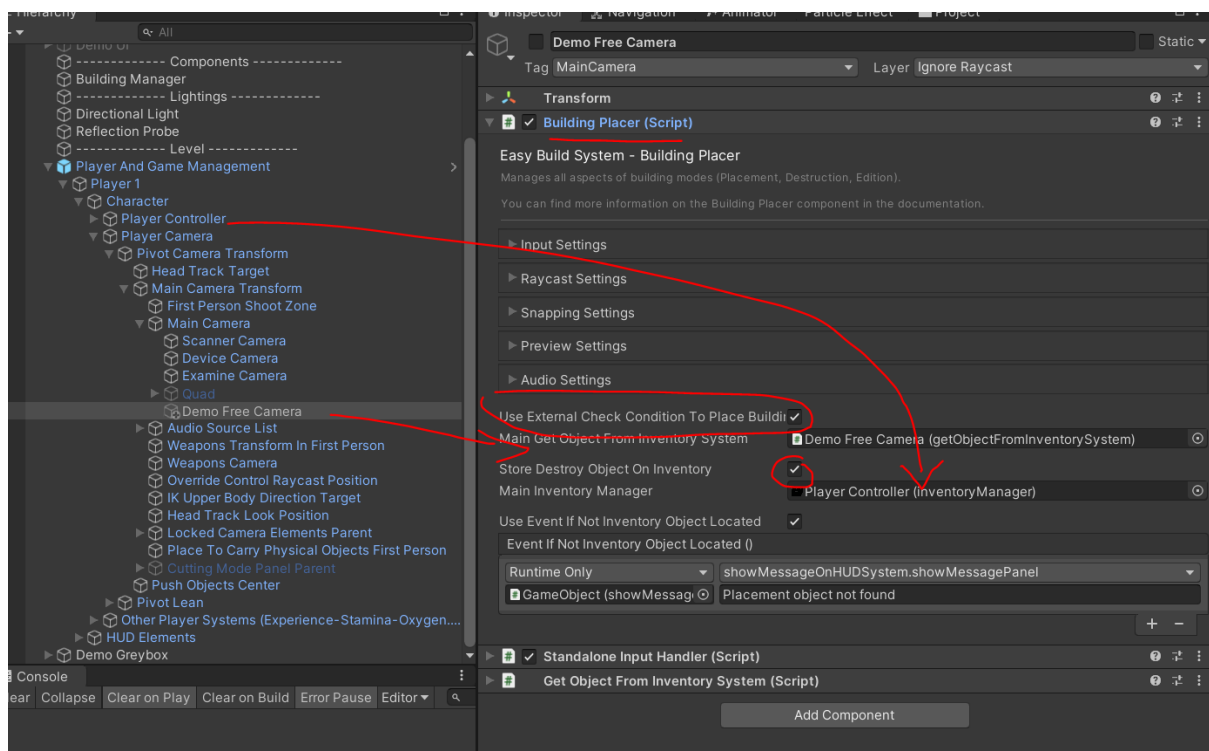
Go to the inspector of the object “Demo Free Camera” (the object from EBS on GKC player):



Attach the component getObjectFromInventorySystem on the object and set these values:



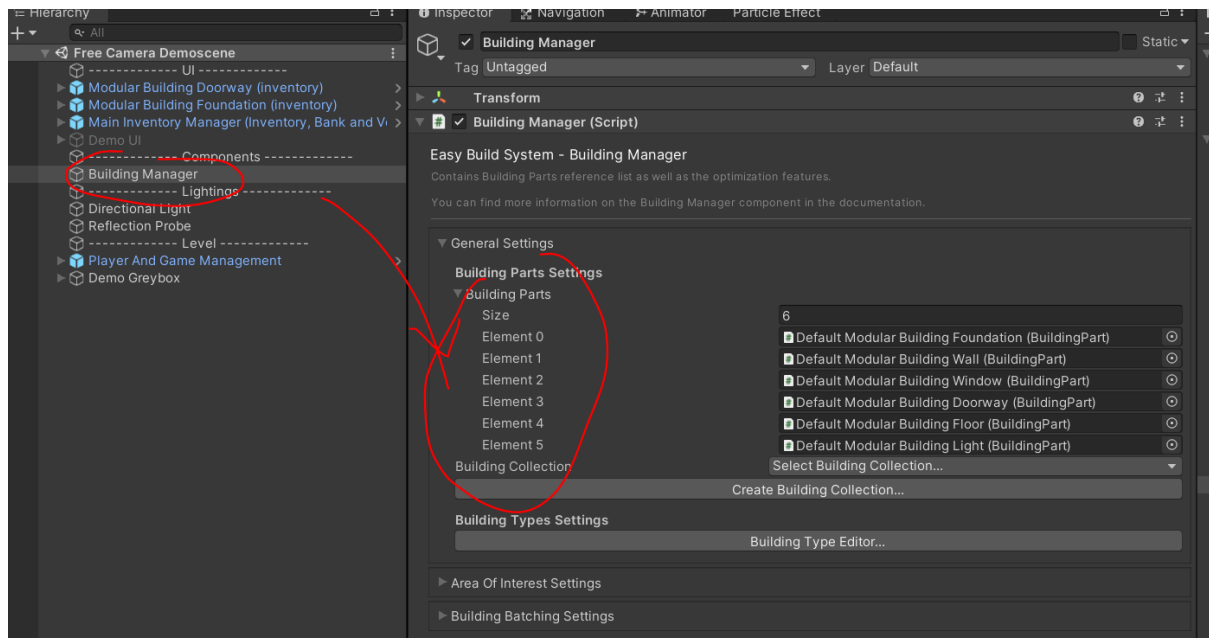
Now, on that same object, expand the component Building Placer, and set these options:



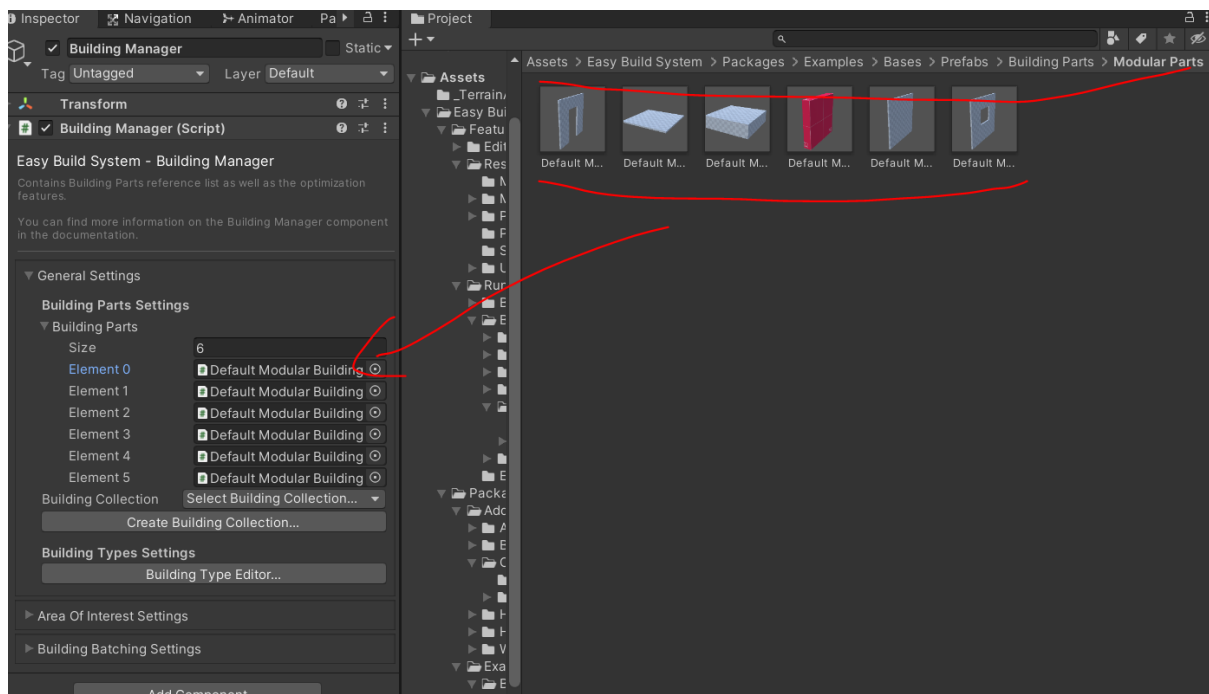
Finally, you just need to create new regular inventory objects for the EBS pieces that you will use, using the meshes and names from those EBS objects on the inventory creator of EBS, you can follow the regular steps of the inventory manager to create regular objects, as they don't need any special setting. You can see it on this video:

<https://www.youtube.com/watch?v=hilv79wDuDY>

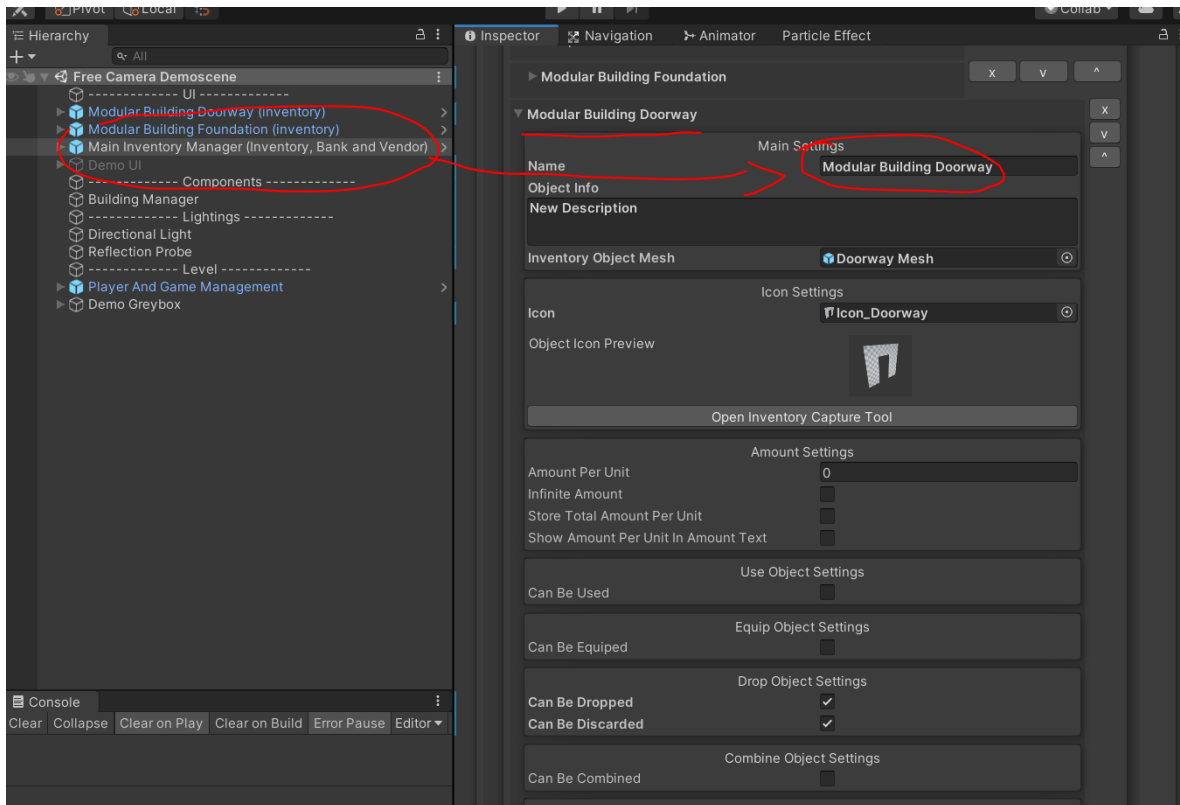
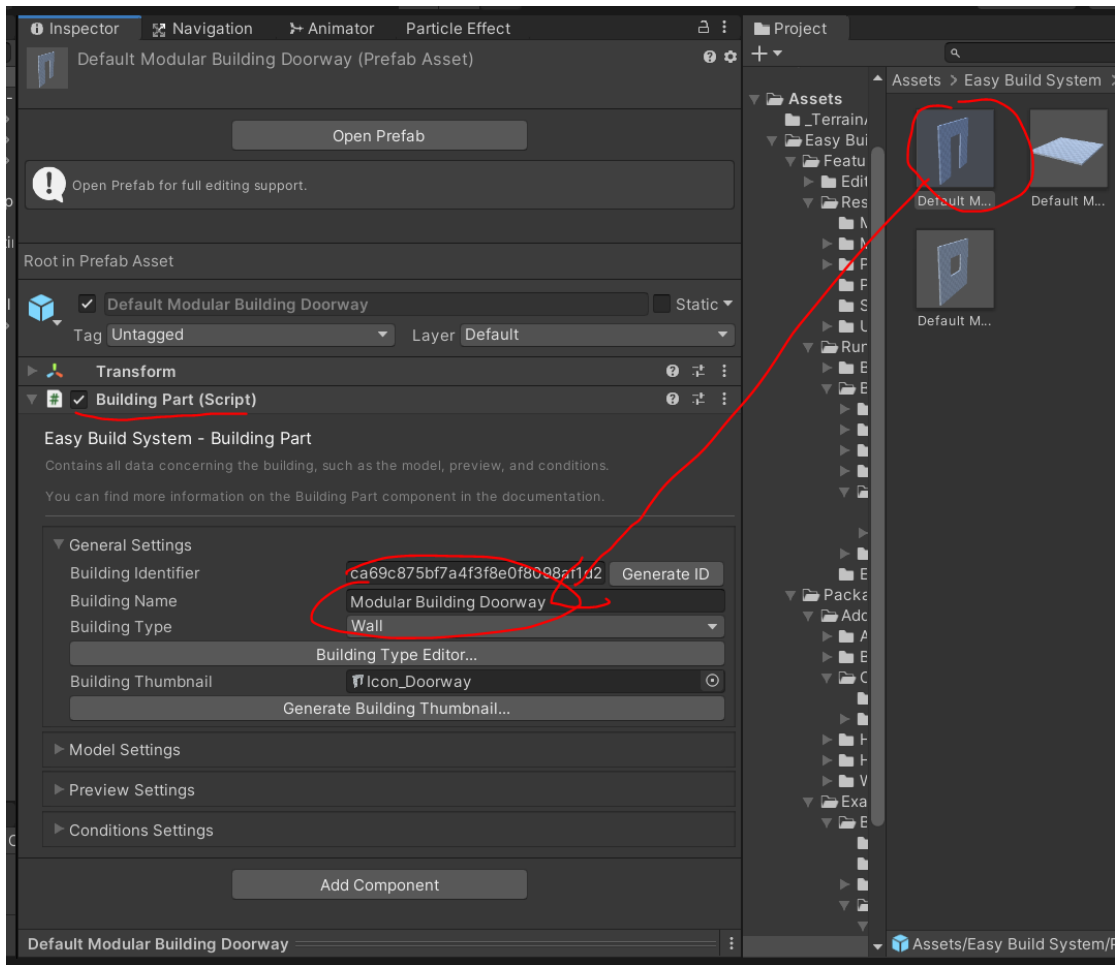
Remember that EBS has a list of pieces that are used on its manager to be placed on scene:



And these pieces has a name field for each one on their prefabs:



Make sure to use the EXACT SAME NAME when creating the GKC inventory object:



As this is the way each unique piece is identified in between both assets.

In that way, you can either drop the created inventory object pickups of GKC on scene (or set the pieces on the initial player's inventory) and the pieces will be taken from it when you use the EBS placement system, allowing or avoiding to place an object type according to if available on the player's inventory.

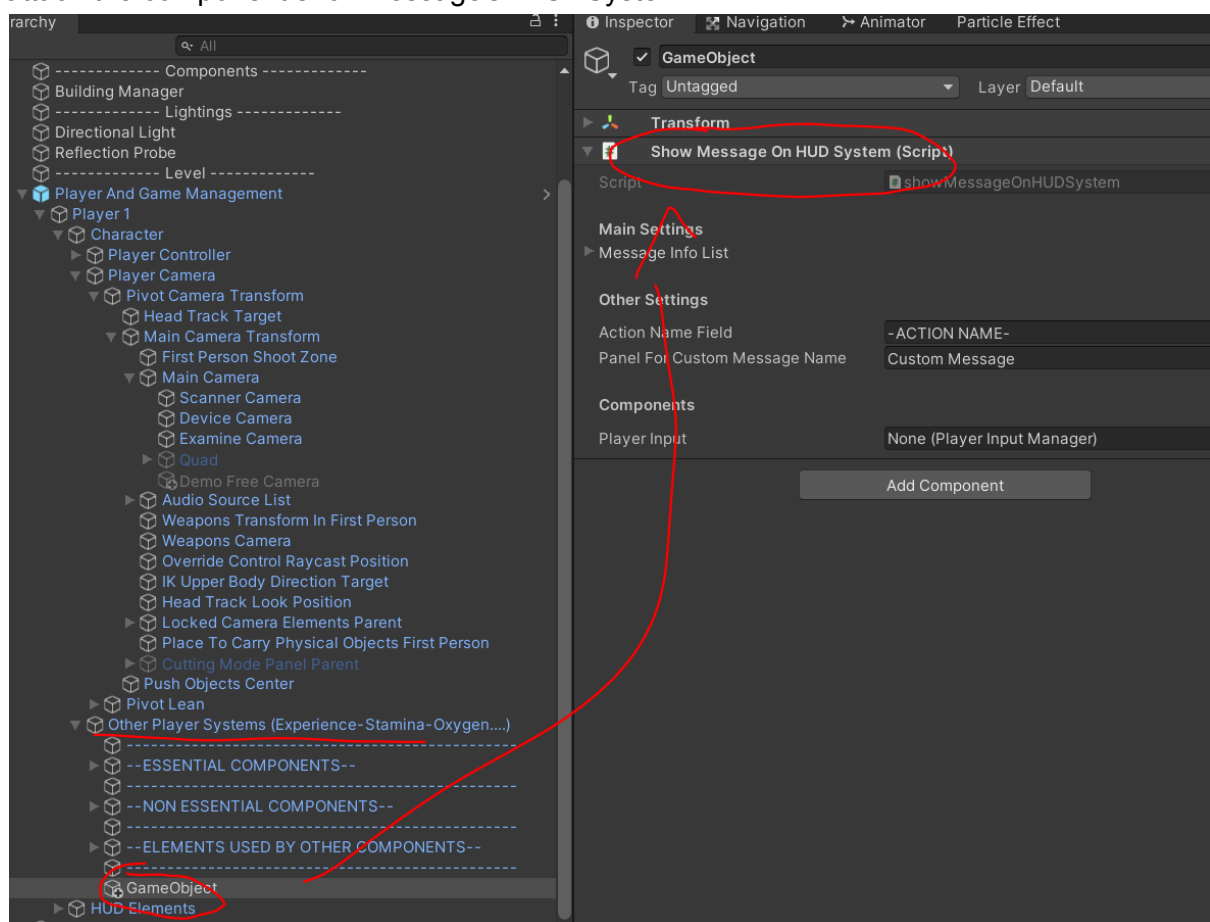
For example, I created the foundation and
You can see here an example:

<https://streamable.com/rv17t0>

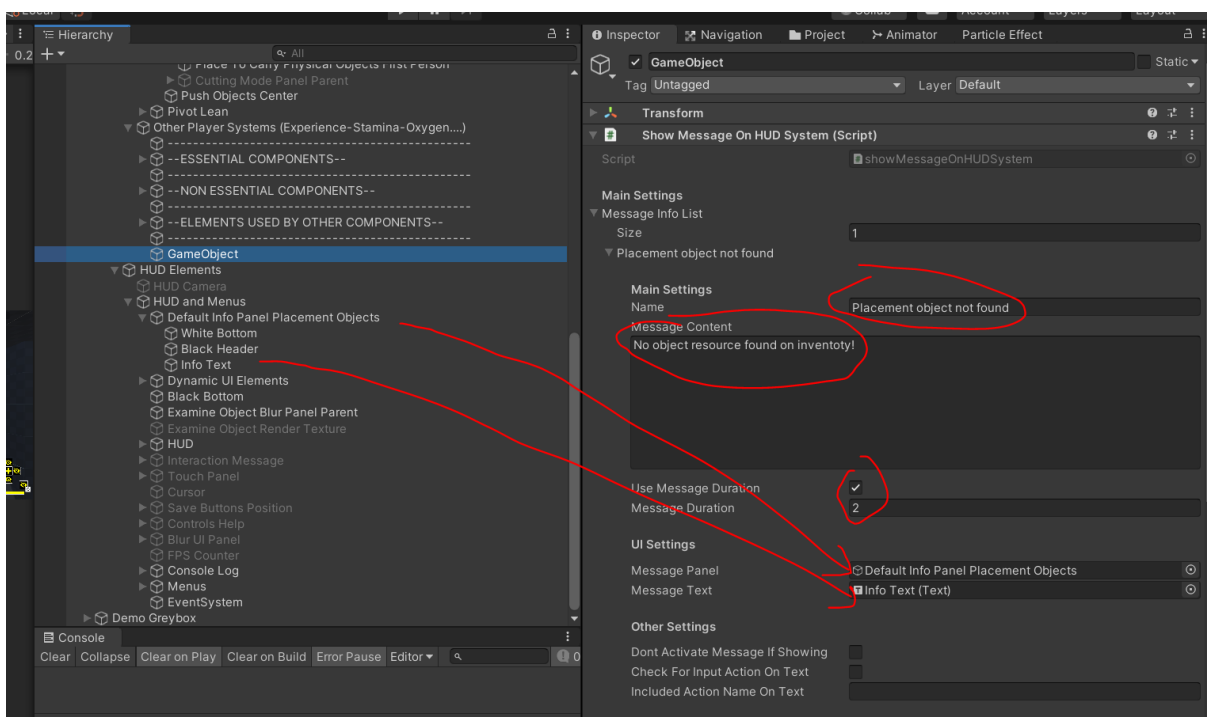
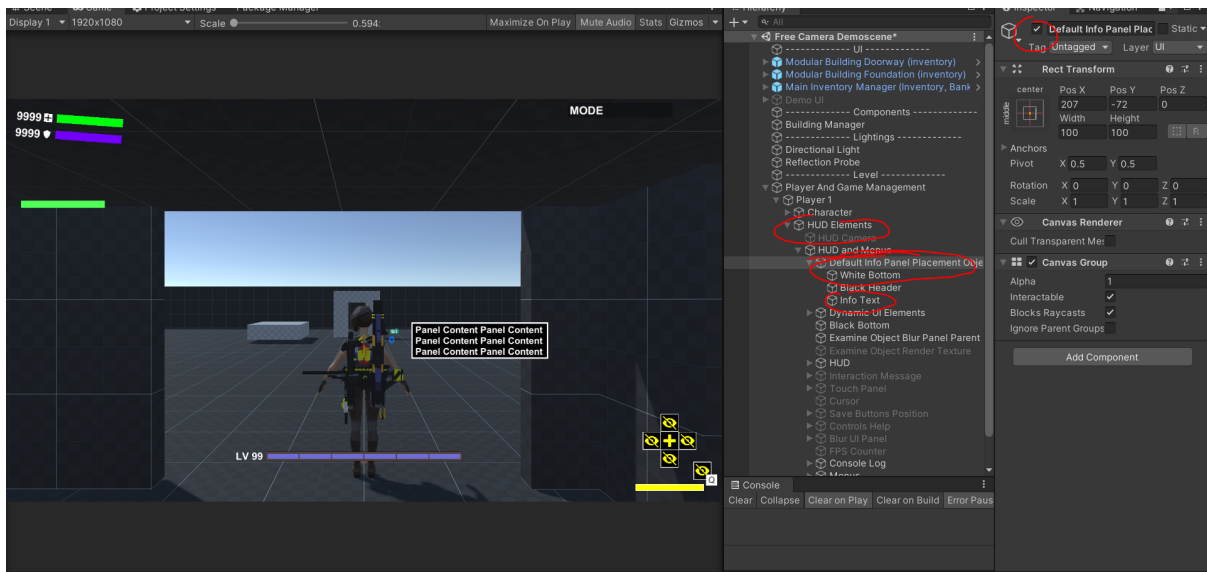
EXTRA: An extra element that can be used is an event to call if the player is trying to place an EBS object without carrying any unit of such object on his inventory. This can be used to show a message on screen for example.

For this follow the next steps:

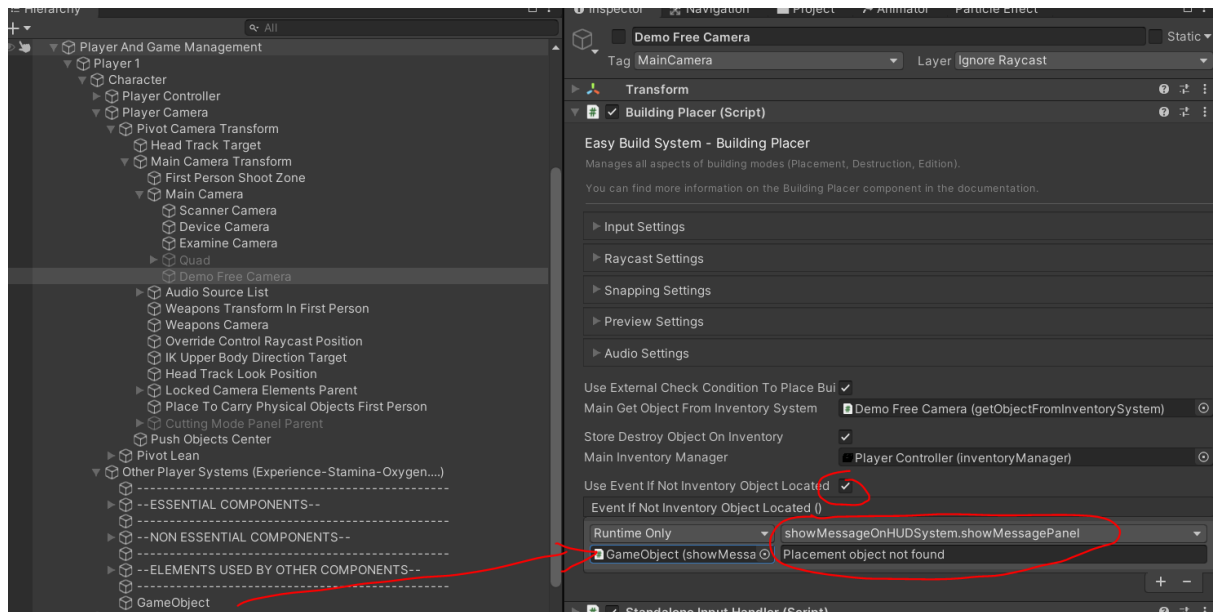
-Create a new object inside the Player, for example, on its Other Systems hierarchy and attach the component showMessageOnHUDSystem:



-On the player's hud, create a new panel with a text field on it and assign it on the above component, like this (leave that panel object as disabled on hierarchy):



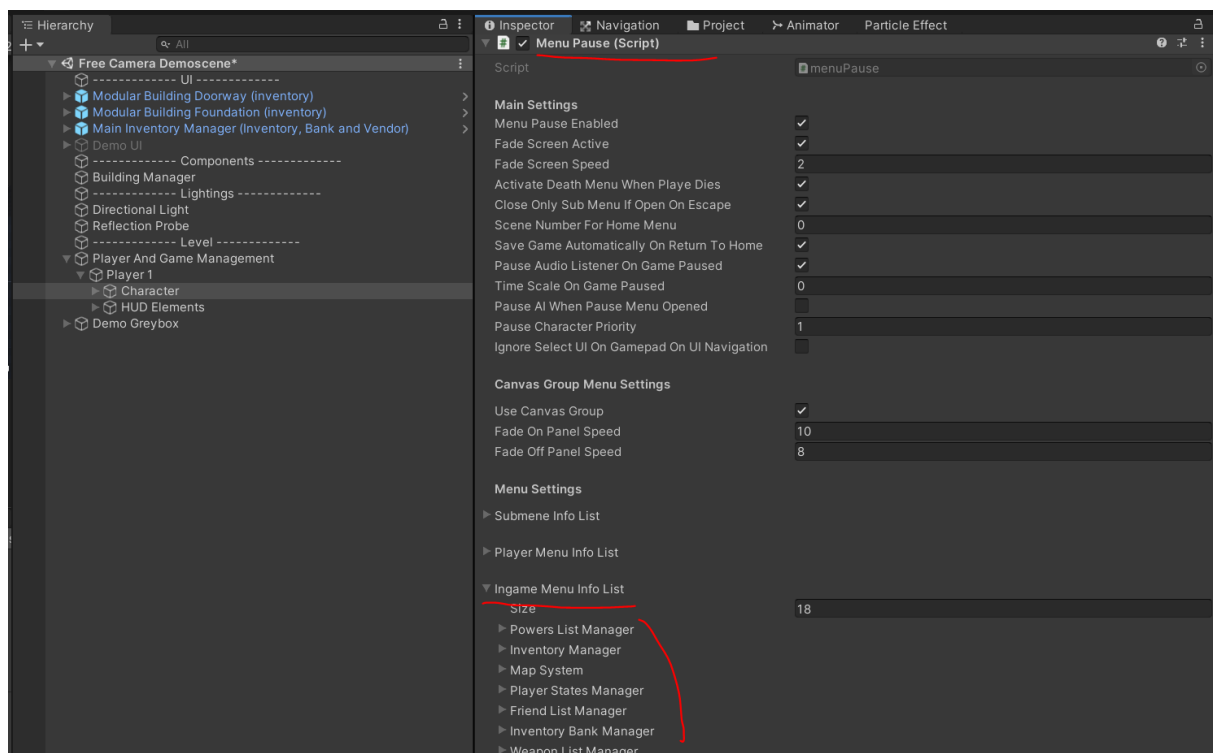
-Now, assign this new empty object on the option of the EBS Demo Free Camera gameObject, to be called, with the function showMessagePanel and setting the name of the panel configured on it that was configured on the previous step (in this example, the name used was "Placement object not found"):



Like this, when the player tries to put a piece not available, it will trigger that message panel, which you can customize as you prefer.

EXTRA: Disable EBS input when using a menu ingame of GKC:

If you want to disable the input of EBS while using a menu, go to the Character gameObject on GKC player, and on its Menu Pause component, locate the Ingame Menu Info List:



On each element of those ingame menus, there are events to call when each menus is opened and close, so you can use that to pause/resume EBS input component, StandAloneInputHandler, here:

