Лабораторна робота №12

Створення системи інвентарю у Unity

Мета: *ознайомити студентів з принципами розробки систем інвентарю в ігровому рушії Unity. Навчити студентів створювати базову систему інвентарю, включаючи додавання, видалення та відображення предметів. Розвинути навички роботи з Unity UI та C# скриптами.*

Література

Створення інвентарю та система створення предметів в Unity: https://uk.sharpcoderblog.com/blog/unity-3d-inventory-and-item-crafting-system

Кодування простої системи інвентаризації з інтерфейсом користувача Drag and Drop в Unity: https://uk.sharpcoderblog.com/blog/unity-3d-coding-a-simple-inventory-system-with-ui-drag-and-drop

Unity 6.1 User Manual: https://docs.unity3d.com/Manual/index.html

Programming concepts (C#): https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/

Зміст роботи

Завдання. Створити систему інвентаризації з інтерфейсом користувача Drag and Drop в Unity

Методичні рекомендації

Інвентар — це таблиця елементів, яка забезпечує швидкий доступ до предметів гравців і простий спосіб їх упорядкування.

Крок 1: Створіть скрипти. Для даної реалізації потрібні 3 скрипти:

SC_CharacterController.cs

```
using UnityEngine;

[RequireComponent(typeof(CharacterController))]

public class SC_CharacterController : MonoBehaviour
{
    public float speed = 7.5f;
    public float jumpSpeed = 8.0f;
    public float gravity = 20.0f;
    public Camera playerCamera;
    public float lookSpeed = 2.0f;
    public float lookXLimit = 60.0f;

    CharacterController characterController;
    Vector3 moveDirection = Vector3.zero;
    Vector2 rotation = Vector2.zero;

    [HideInInspector]
```

```
    public bool canMove = true;

    void Start()
    {
        characterController = GetComponent<CharacterController>();
        rotation.y = transform.eulerAngles.y;
    }

    void Update()
    {
        if (characterController.isGrounded)
        {
            Vector3 forward = transform.TransformDirection(Vector3.forward);
            Vector3 right = transform.TransformDirection(Vector3.right);
            float curSpeedX = speed * Input.GetAxis("Vertical");
            float curSpeedY = speed * Input.GetAxis("Horizontal");
            moveDirection = (forward * curSpeedX) + (right * curSpeedY);

            if (Input.GetButton("Jump"))
            {
                moveDirection.y = jumpSpeed;
            }
        }

        moveDirection.y -= gravity * Time.deltaTime;

        characterController.Move(moveDirection * Time.deltaTime);

        if (canMove)
        {
            rotation.y += Input.GetAxis("Mouse X") * lookSpeed;
            rotation.x += -Input.GetAxis("Mouse Y") * lookSpeed;
            rotation.x = Mathf.Clamp(rotation.x, -lookXLimit, lookXLimit);
            playerCamera.transform.localRotation = Quaternion.Euler(rotation.x,
0, 0);
            transform.eulerAngles = new Vector2(0, rotation.y);
        }
    }
}
```

SC_PickItem.cs

```
using UnityEngine;

public class SC_PickItem : MonoBehaviour
{
    public string itemName = "Some Item";
    public Texture itemPreview;

    void Start()
    {
        gameObject.tag = "Respawn";
    }
```

```
    public void PickItem()
    {
        Destroy(gameObject);
    }
}
```

SC_InventorySystem.cs

```
using UnityEngine;


public class SC_InventorySystem : MonoBehaviour
{
    public Texture crosshairTexture;
    public SC_CharacterController playerController;
    public SC_PickItem[] availableItems;


    int[] itemSlots = new int[12];
    bool showInventory = false;
    float windowAnimation = 1;
    float animationTimer = 0;


    int hoveringOverIndex = -1;
    int itemIndexToDrag = -1;
    Vector2 dragOffset = Vector2.zero;


    SC_PickItem detectedItem;
    int detectedItemIndex;


    void Start()
    {
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;


        for (int i = 0; i < itemSlots.Length; i++)
        {
            itemSlots[i] = -1;
        }
    }


    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Tab))
        {
            showInventory = !showInventory;
            animationTimer = 0;
```

```
            if (showInventory)
            {
                Cursor.visible = true;
                Cursor.lockState = CursorLockMode.None;
            }
            else
            {
                Cursor.visible = false;
                Cursor.lockState = CursorLockMode.Locked;
            }
        }


        if (animationTimer < 1)
        {
            animationTimer += Time.deltaTime;
        }


        if (showInventory)
        {
            windowAnimation = Mathf.Lerp(windowAnimation, 0, animationTimer);
            playerController.canMove = false;
        }
        else
        {
            windowAnimation = Mathf.Lerp(windowAnimation, 1f, animationTimer);
            playerController.canMove = true;
        }


        if (Input.GetMouseButtonDown(0) && hoveringOverIndex > -1 &&
itemSlots[hoveringOverIndex] > -1)
        {
            itemIndexToDrag = hoveringOverIndex;
        }


        if (Input.GetMouseButtonUp(0) && itemIndexToDrag > -1)
        {
            if (hoveringOverIndex < 0)
            {
                Instantiate(availableItems[itemSlots[itemIndexToDrag]],
playerController.playerCamera.transform.position +
(playerController.playerCamera.transform.forward), Quaternion.identity);
                itemSlots[itemIndexToDrag] = -1;
            }
            else
            {
                int itemIndexTmp = itemSlots[itemIndexToDrag];
                itemSlots[itemIndexToDrag] = itemSlots[hoveringOverIndex];
                itemSlots[hoveringOverIndex] = itemIndexTmp;


            }
```

```
            itemIndexToDrag = -1;
        }


        if (detectedItem && detectedItemIndex > -1)
        {
            if (Input.GetKeyDown(KeyCode.F))
            {
                int slotToAddTo = -1;
                for (int i = 0; i < itemSlots.Length; i++)
                {
                    if (itemSlots[i] == -1)
                    {
                        slotToAddTo = i;
                        break;
                    }
                }
                if (slotToAddTo > -1)
                {
                    itemSlots[slotToAddTo] = detectedItemIndex;
                    detectedItem.PickItem();
                }
            }
        }
    }


    void FixedUpdate()
    {
        RaycastHit hit;
        Ray ray = playerController.playerCamera.ViewportPointToRay(new
Vector3(0.5F, 0.5F, 0));


        if (Physics.Raycast(ray, out hit, 2.5f))
        {
            Transform objectHit = hit.transform;


            if (objectHit.CompareTag("Respawn"))
            {
                if ((detectedItem == null || detectedItem.transform !=
objectHit) && objectHit.GetComponent<SC_PickItem>() != null)
                {
                    SC_PickItem itemTmp = objectHit.GetComponent<SC_PickItem>();


                    for (int i = 0; i < availableItems.Length; i++)
                    {
                        if (availableItems[i].itemName == itemTmp.itemName)
                        {
                            detectedItem = itemTmp;
                            detectedItemIndex = i;
                        }
                    }
```

```
                }
            }
            else
            {
                detectedItem = null;
            }
        }
        else
        {
            detectedItem = null;
        }
    }


    void OnGUI()
    {
        GUI.Label(new Rect(5, 5, 200, 25), "Press 'Tab' to open Inventory");

        if (windowAnimation < 1)
        {
            GUILayout.BeginArea(new Rect(10 - (430 * windowAnimation),
Screen.height / 2 - 200, 302, 430), GUI.skin.GetStyle("box"));


            GUILayout.Label("Inventory", GUILayout.Height(25));


            GUILayout.BeginVertical();
            for (int i = 0; i < itemSlots.Length; i += 3)
            {
                GUILayout.BeginHorizontal();

                for (int a = 0; a < 3; a++)
                {
                    if (i + a < itemSlots.Length)
                    {
                        if (itemIndexToDrag == i + a || (itemIndexToDrag > -1 &&
hoveringOverIndex == i + a))
                        {
                            GUI.enabled = false;
                        }


                        if (itemSlots[i + a] > -1)
                        {
                            if (availableItems[itemSlots[i + a]].itemPreview)
                            {
                                GUILayout.Box(availableItems[itemSlots[i +
a]].itemPreview, GUILayout.Width(95), GUILayout.Height(95));
                            }
                            else
                            {
                                GUILayout.Box(availableItems[itemSlots[i +
a]].itemName, GUILayout.Width(95), GUILayout.Height(95));
                            }
```

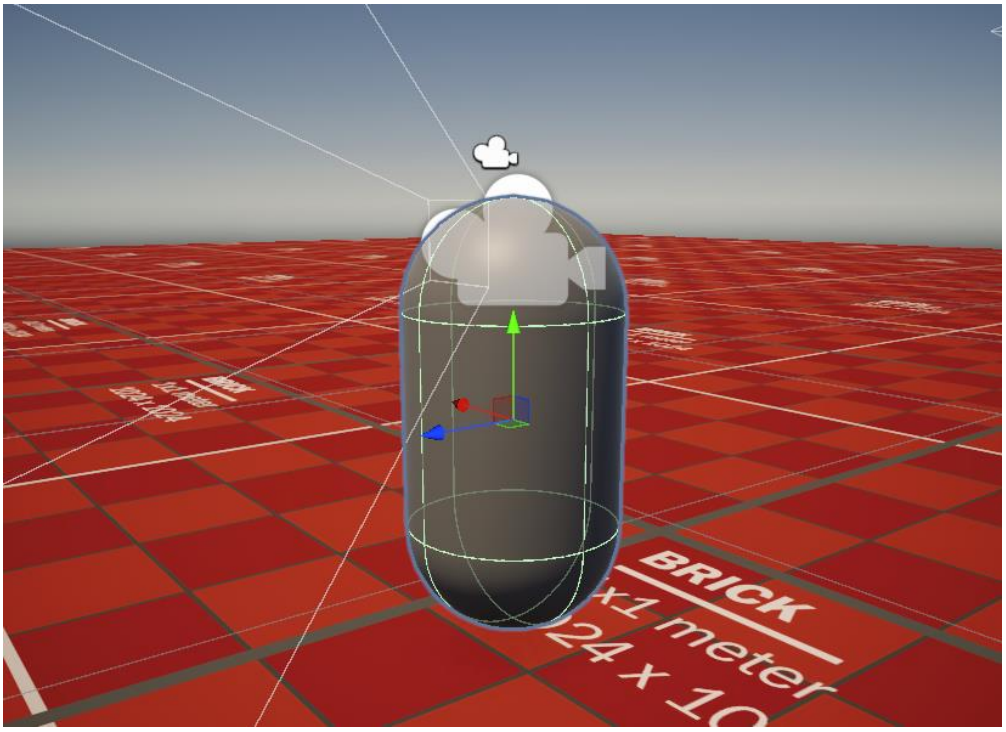
```
                        }
                        else
                        {
                            GUILayout.Box("", GUILayout.Width(95),
GUILayout.Height(95));
                        }


                        Rect lastRect = GUILayoutUtility.GetLastRect();
                        Vector2 eventMousePositon = Event.current.mousePosition;
                        if (Event.current.type == EventType.Repaint &&
lastRect.Contains(eventMousePositon))
                        {
                            hoveringOverIndex = i + a;
                            if (itemIndexToDrag < 0)
                            {
                                dragOffset = new Vector2(lastRect.x -
eventMousePositon.x, lastRect.y - eventMousePositon.y);
                            }
                        }


                        GUI.enabled = true;
                    }
                }
                GUILayout.EndHorizontal();
            }
            GUILayout.EndVertical();


            if (Event.current.type == EventType.Repaint &&
!GUILayoutUtility.GetLastRect().Contains(Event.current.mousePosition))
            {
                hoveringOverIndex = -1;
            }


            GUILayout.EndArea();
        }


        if (itemIndexToDrag > -1)
        {
            if (availableItems[itemSlots[itemIndexToDrag]].itemPreview)
            {
                GUI.Box(new Rect(Input.mousePosition.x + dragOffset.x,
Screen.height - Input.mousePosition.y + dragOffset.y, 95, 95),
availableItems[itemSlots[itemIndexToDrag]].itemPreview);
            }
            else
            {
                GUI.Box(new Rect(Input.mousePosition.x + dragOffset.x,
Screen.height - Input.mousePosition.y + dragOffset.y, 95, 95),
availableItems[itemSlots[itemIndexToDrag]].itemName);
            }
```

```
        }


        if (hoveringOverIndex > -1 && itemSlots[hoveringOverIndex] > -1 &&
itemIndexToDrag < 0)
        {
            GUI.Box(new Rect(Input.mousePosition.x, Screen.height -
Input.mousePosition.y - 30, 100, 25),
availableItems[itemSlots[hoveringOverIndex]].itemName);
        }


        if (!showInventory)
        {
            GUI.color = detectedItem ? Color.green : Color.white;
            GUI.DrawTexture(new Rect(Screen.width / 2 - 4, Screen.height / 2 -
4, 8, 8), crosshairTexture);
            GUI.color = Color.white;

            if (detectedItem)
            {
                GUI.color = new Color(0, 0, 0, 0.84f);
                GUI.Label(new Rect(Screen.width / 2 - 75 + 1, Screen.height / 2
- 50 + 1, 150, 20), "Press 'F' to pick '" + detectedItem.itemName + "'");
                GUI.color = Color.green;
                GUI.Label(new Rect(Screen.width / 2 - 75, Screen.height / 2 -
50, 150, 20), "Press 'F' to pick '" + detectedItem.itemName + "'");
            }
        }
    }
}
```

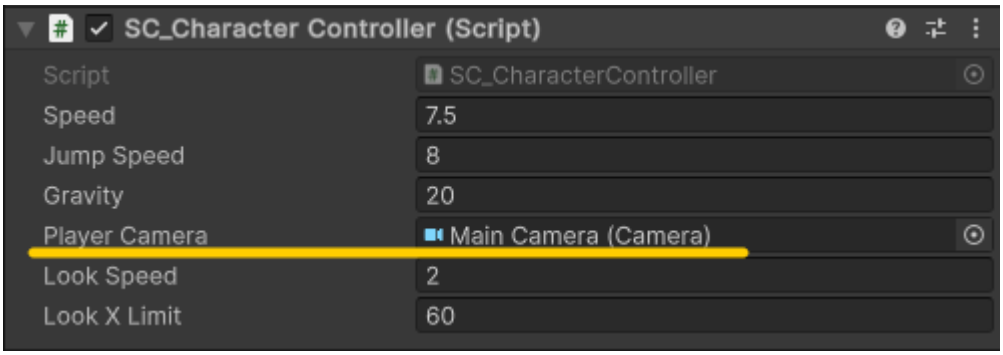
Крок 2: Налаштуйте систему гравця та інвентарю

Налаштування гравця:

- Створіть новий GameObject і назвіть його "Player"
- Створіть нову капсулу (GameObject -> 3D Object -> Capsule), видаліть компонент Capsule Collider, потім перемістіть капсулу всередину об'єкта "Player" і змініть її положення на (0, 1, 0)
- Перемістіть головну камеру всередину об’єкта "Player" і змініть її положення на (0, 1,64, 0)



- Додайте скрипт до об’єкта "Player" (він автоматично додасть інший компонент Character Controller, та змінить його центральне значення на (0, 1, 0))
- Призначте головну камеру змінній "Player Camera" у SC_CharacterController



Налаштуйте Pick Up items – це будуть префаби предметів, які можна вибрати в грі.

Для прикладу будемо використовувати прості форми (куб, циліндр і сфера), але можна додати різні моделі.

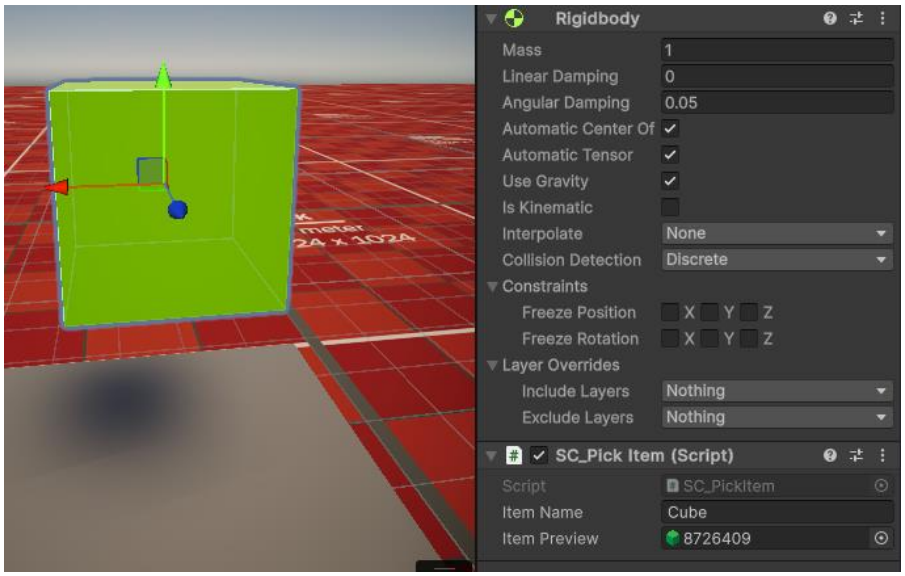
- Створіть новий GameObject і назвіть його "SimpleItem"
- Створіть новий куб (GameObject -> 3D Object -> Cube), зменшіть його масштаб до (0,4, 0,4, 0,4), а потім перемістіть його всередину "SimpleItem" GameObject

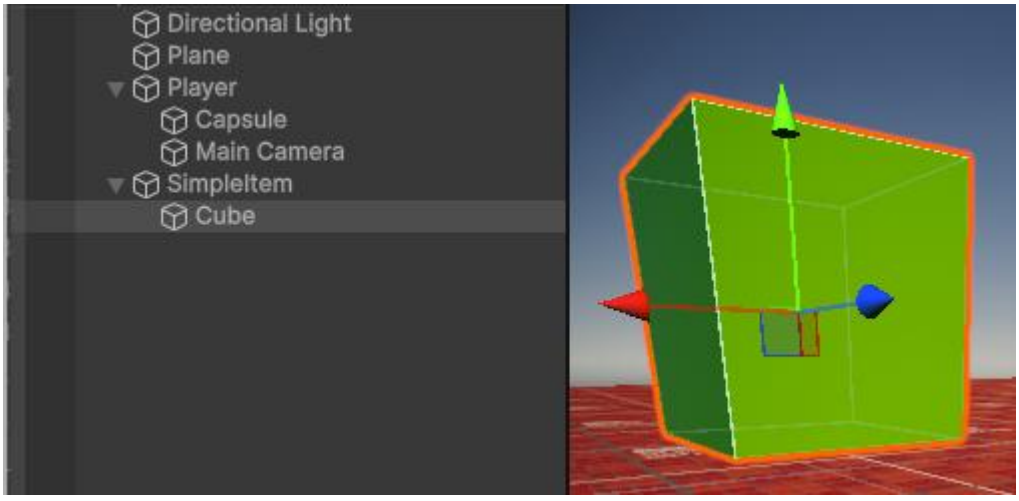
- Виберіть "SimpleItem" і додайте компонент Rigidbody і скрипт SC_PickItem

Можна помітити, що в SC_PickItem є 2 змінні:

Item name має містити унікальну назву.
Item Preview - текстура що буде відображатися в інвентарі, спробуйте знайти картинку що зображуватиме ваш предмет.

У прикладі назва елемента — "Cube", а попередній перегляд — зелений куб:





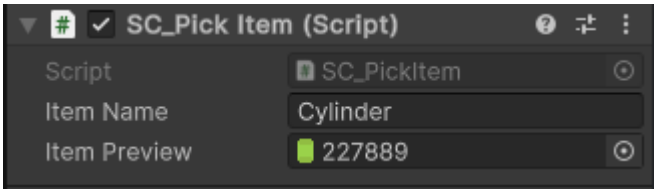
Повторіть ті самі кроки для інших 2 елементів.

Для елемента Циліндр:

- Скопіюйте об’єкт "SimpleItem" і назвіть його "SimpleItem 2"

- Видаліть дочірній куб і створіть новий циліндр (GameObject -> 3D Object -> Cylinder). Перемістіть його всередину "SimpleItem 2" і масштабуйте до (0,4, 0,4, 0,4).

- Змініть назву елемента в SC_PickItem на "Cylinder" і попередній перегляд елемента на зображення циліндра

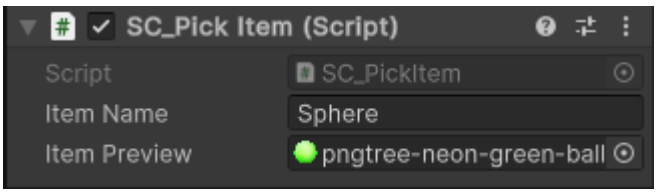


Для елемента Sphere:

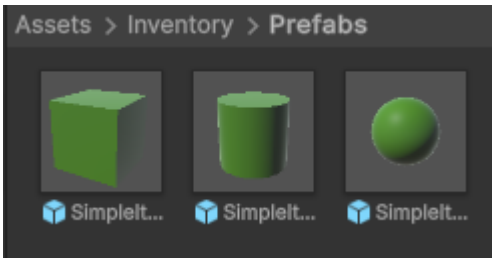
- Скопіюйте об’єкт "SimpleItem" і назвіть його "SimpleItem 3"

- Видаліть дочірній куб і створіть нову сферу (GameObject -> 3D Object -> Sphere). Перемістіть його всередину "SimpleItem 3" і змініть масштаб до (0,4, 0,4, 0,4).

- Змініть ім’я елемента в SC_PickItem на "Sphere" і пункт «Попередній перегляд» на зображення сфери



Тепер збережіть кожен елемент у Prefab:

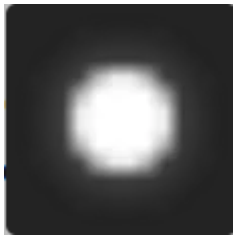


Предмети готові.

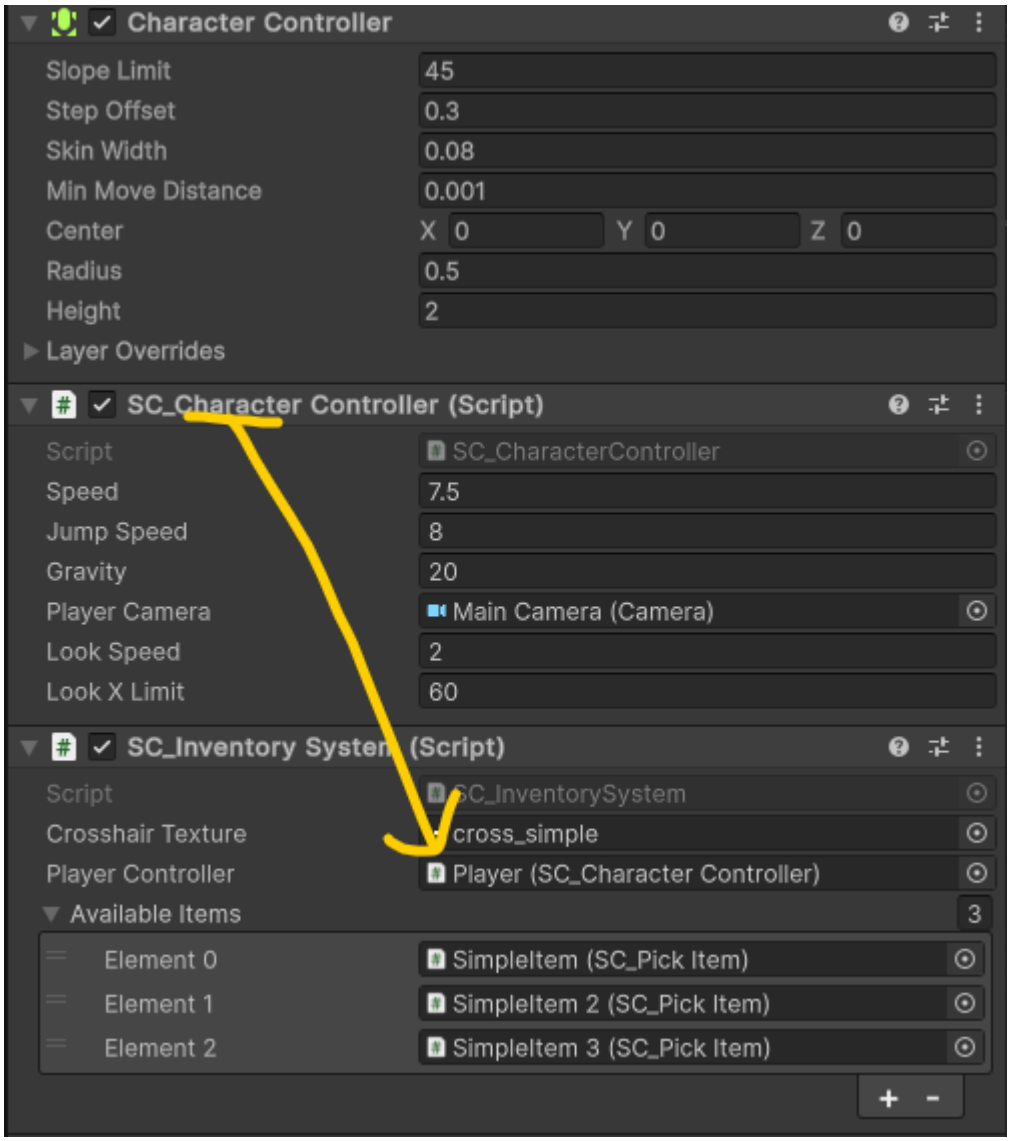
Останнім кроком є налаштування системи інвентаризації:

- Додайте SC_InventorySystem до об’єкта "Player"

- Призначте змінну текстури прицілу (ви можете використати зображення нижче, або створити власну текстуру):



- Призначте SC_CharacterController змінній "Player Controller" у SC_InventorySystem
- Для "Available Items" призначте попередньо створені префаби (Примітка: це мають бути екземпляри Prefab з перегляду проекту, а не об’єкти зі сцени):



Крок 3: Система інвентаризації готова, наступний етап тестування:

- Протестуйте всі функції системи інвентарю, включаючи додавання, видалення.
- Переконайтеся, що інвентар коректно відображається при зміні його вмісту.
- Перевірте, чи працюють обмеження на кількість предметів в інвентарі.

Контрольні питання:

1. Який тип даних найчастіше використовується для зберігання предметів в інвентарі у Unity?

- A) Array
- B) List
- C) Dictionary
- D) HashSet
- E) Queue

2. Який компонент Unity UI найкраще підходить для відображення слотів інвентарю?

- A) Button
- B) Image
- C) RawImage
- D) Panel
- E) ScrollView

3. Який метод слід використовувати для додавання предмета до інвентарю?

- A) inventory.Add(item)
- B) inventory.AddItem(item)
- C) inventory.Insert(item)
- D) inventory.Push(item)
- E) inventory.Append(item)

4. Що таке Prefab у Unity?

- A) Скрипт C#
- B) Компонент UI
- C) Попередньо створений і збережений GameObject
- D) Матеріал для 3D об'єкта
- E) Звуковий ефект

5. Який метод найкраще підходить для оновлення відображення інвентарю на UI?

- A) Update()
- B) LateUpdate()
- C) OnGUI()
- D) UpdateUI()
- E) Render()

6. Як правильно створити слот інвентарю з іконкою та лічильником кількості?

- A) Створити Image та Text як дочірні об'єкти Panel
- B) Створити Panel з дочірніми об'єктами Image та Text
- C) Створити Text з дочірнім об'єктом Image
- D) Створити Image з дочірнім об'єктом Text
- E) Створити RawImage з дочірнім об'єктом Text

7. Який тип скрипту зазвичай використовується для опису характеристик предмета (назва, опис, іконка)?

- A) MonoBehaviour
- B) Editor
- C) ScriptableObject
- D) статичний клас
- E) Інтерфейс

8. Який метод слід використовувати для видалення предмета з інвентарю?

- A) inventory.Remove(item)
- B) inventory.DeleteItem(item)
- C) inventory.Pop(item)
- D) inventory.Clear(item)
- E) inventory.Destroy(item)

9. Що таке "слот" інвентарю?

- A) Окремий предмет в інвентарі
- B) Місце для відображення одного предмета в UI інвентарю
- C) Категорія предметів в інвентарі
- D) Скрипт, що описує предмет
- E) Звуковий ефект при додаванні предмета

10. Який компонент Unity використовується для відображення тексту?

- A) Text
- B) Label
- C) TextField
- D) TextMesh
- E) InputField

Самостійна робота:

Доопрацювати лабораторну роботу.