# \*\*战利品系统说明文档\*\*

## \*\*概述\*\*

本战利品系统提供了一个简单、灵活的仓库管理和物品管理功能。通过一套 UI 系统，玩家可以查看、添加、丢弃仓库中的物品。物品生成和管理通过工厂模式（`LootFactory`）和接口（`ILoot`）来实现。整个系统的设计易于扩展，便于其他开发者添加新的物品类型或管理逻辑。

## \*\*系统结构\*\*

本战利品系统由多个类和 UI 组件组成，每个类和 UI 元素负责一个特定的功能。以下是系统的主要部分：

1. \*\*`ILoot` 接口\*\*：定义所有物品应遵循的接口。

2. \*\*`LootItem` 类\*\*：实现了 `ILoot` 接口的具体物品类，包含物品属性和行为。

3. \*\*`LootFactory` 类\*\*：工厂类，根据类型创建不同的战利品。

4. \*\*`LootSpawner` 类\*\*：用于生成物品的类，在游戏中触发物品掉落。

5. \*\*`InventoryUI` 类\*\*：用于管理并显示仓库界面的 UI 控制类。

6. \*\*`ItemUI` 类\*\*：显示每个物品的 UI 组件。

### \*\*1. `ILoot` 接口\*\*

\*\*文件路径\*\*：`Assets/Scripts/System/Loot/ILoot.cs`

```csharp

namespace LootSystem

{

public interface ILoot

{

string ItemName { get; set; } // 物品名称

string ItemType { get; set; } // 物品类型

int Quantity { get; set; } // 物品数量

bool IsPickedUp { get; set; } // 物品是否被拾取

void OnLootDropped(); // 物品掉落时的处理

void OnLootPickedUp(); // 物品被拾取时的处理

}

}

```

\*\*说明\*\*：

- `ILoot` 是所有物品类的基础接口，定义了物品名称、类型、数量、拾取状态及掉落与拾取时的行为。

- 所有具体的物品类（如 `LootItem`）必须实现该接口。

---

### \*\*2. `LootItem` 类\*\*

\*\*文件路径\*\*：`Assets/Scripts/System/Loot/LootItem.cs`

```csharp

using UnityEngine;

namespace LootSystem

{

public class LootItem : MonoBehaviour, ILoot

{

public string itemName; // 物品名称

public string itemType; // 物品类型

public int quantity; // 物品数量

public bool isPickedUp = false; // 物品是否已被拾取

public string ItemName { get => itemName; set => itemName = value; }

public string ItemType { get => itemType; set => itemType = value; }

public int Quantity { get => quantity; set => quantity = value; }

public bool IsPickedUp { get => isPickedUp; set => isPickedUp = value; }

public void OnLootDropped()

{

Debug.Log("Loot dropped: " + itemName);

}

public void OnLootPickedUp()

{

if (isPickedUp) return;

isPickedUp = true;

Debug.Log(itemName + " picked up!");

Destroy(gameObject);

}

void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Player"))

{

OnLootPickedUp();

}

}

}

}

```

\*\*说明\*\*：

- `LootItem` 类实现了 `ILoot` 接口，表示一个具体的物品。

- 该类处理物品的掉落和拾取逻辑，拾取后物品会被销毁。

---

### \*\*3. `LootFactory` 类\*\*

\*\*文件路径\*\*：`Assets/Scripts/System/Loot/LootFactory.cs`

```csharp

namespace LootSystem

{

public class LootFactory

{

public static ILoot CreateLoot(string itemType)

{

ILoot loot = null;

switch (itemType)

{

case "Potion":

loot = new PotionLoot();

break;

case "Weapon":

loot = new WeaponLoot();

break;

default:

loot = new GenericLoot();

break;

}

return loot;

}

}

public class PotionLoot : ILoot

{

public string ItemName { get; set; } = "Potion";

public string ItemType { get; set; } = "Consumable";

public int Quantity { get; set; } = 1;

public bool IsPickedUp { get; set; } = false;

public void OnLootDropped() { Debug.Log("Potion dropped!"); }

public void OnLootPickedUp() { Debug.Log("Potion picked up!"); }

}

public class WeaponLoot : ILoot

{

public string ItemName { get; set; } = "Sword";

public string ItemType { get; set; } = "Weapon";

public int Quantity { get; set; } = 1;

public bool IsPickedUp { get; set; } = false;

public void OnLootDropped() { Debug.Log("Weapon dropped!"); }

public void OnLootPickedUp() { Debug.Log("Weapon picked up!"); }

}

public class GenericLoot : ILoot

{

public string ItemName { get; set; } = "Generic Item";

public string ItemType { get; set; } = "Miscellaneous";

public int Quantity { get; set; } = 1;

public bool IsPickedUp { get; set; } = false;

public void OnLootDropped() { Debug.Log("Generic item dropped!"); }

public void OnLootPickedUp() { Debug.Log("Generic item picked up!"); }

}

}

```

\*\*说明\*\*：

- `LootFactory` 是一个工厂类，根据物品类型生成不同的 `ILoot` 类型的战利品。

- 目前支持生成药水、武器和通用物品。

---

### \*\*4. `LootSpawner` 类\*\*

\*\*文件路径\*\*：`Assets/Scripts/System/Loot/LootSpawner.cs`

```csharp

using UnityEngine;

namespace LootSystem

{

public class LootSpawner : MonoBehaviour

{

public GameObject lootPrefab;

public void SpawnLoot(Vector3 position, string itemName, string itemType, int quantity)

{

GameObject loot = Instantiate(lootPrefab, position, Quaternion.identity);

LootItem lootItem = loot.GetComponent<LootItem>();

lootItem.ItemName = itemName;

lootItem.ItemType = itemType;

lootItem.Quantity = quantity;

lootItem.OnLootDropped();

}

}

}

```

\*\*说明\*\*：

- `LootSpawner` 用于在游戏中生成战利品（例如怪物死亡后掉落的战利品）。

- `SpawnLoot` 方法根据指定的物品属性创建一个新的战利品。

---

### \*\*5. `InventoryUI` 类\*\*

\*\*文件路径\*\*：`Assets/Scripts/System/Loot/InventoryUI.cs`

```csharp

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

namespace LootSystem

{

public class InventoryUI : MonoBehaviour

{

public GameObject inventoryPanel;

public GameObject itemPrefab;

public Transform itemsParent;

private List<ILoot> playerInventory = new List<ILoot>();

public void OpenInventory()

{

inventoryPanel.SetActive(true);

UpdateInventoryUI();

}

public void CloseInventory()

{

inventoryPanel.SetActive(false);

}

private void UpdateInventoryUI()

{

foreach (Transform child in itemsParent)

{

Destroy(child.gameObject);

}

foreach (ILoot loot in playerInventory)

{

GameObject itemUI = Instantiate(itemPrefab, itemsParent);

ItemUI item = itemUI.GetComponent<ItemUI>();

item.SetItemInfo(loot);

}

}

public void AddItemToInventory(ILoot item)

{

playerInventory.Add(item);

UpdateInventoryUI();

}

public void DiscardItem(ILoot item)

{

playerInventory.Remove(item);

UpdateInventoryUI();

}

}

}

```

\*\*说明\*\*：

- `InventoryUI` 控制仓库界面的显示和物品的添加与丢弃。

- 物品通过 `AddItemToInventory` 添加到仓库，丢弃物品通过 `DiscardItem` 移除。

---

### \*\*6. `ItemUI` 类\*\*

\*\*文件路径\*\*：`Assets/Scripts/System/Loot/ItemUI.cs`

```csharp

using UnityEngine;

using UnityEngine.UI;

namespace LootSystem

{

public class ItemUI : MonoBehaviour

{

public Text itemNameText;

public Text itemQuantityText;

public Button discardButton;

private ILoot loot;

public void SetItemInfo(ILoot item)

{

loot = item;

itemNameText.text = loot.ItemName

;

itemQuantityText.text = "x" + loot.Quantity.ToString();

discardButton.onClick.AddListener(DiscardItem);

}

private void DiscardItem()

{

FindObjectOfType<InventoryUI>().DiscardItem(loot);

}

}

}

```

\*\*说明\*\*：

- `ItemUI` 负责单个物品的 UI 显示，包括名称、数量和丢弃按钮。

- 按钮点击时触发丢弃物品的操作。

---

## \*\*如何使用\*\*

### \*\*1. UI 设置\*\*

1. \*\*创建 Canvas\*\*：在 Unity 中创建一个 Canvas，用于容纳所有 UI 元素。

2. \*\*创建 InventoryPanel\*\*：创建一个 Panel 用作仓库的背景，并将其与 `InventoryUI` 脚本关联。

3. \*\*添加 ScrollView\*\*：将 `ScrollView` 用作物品列表显示区域，并将 `Content` 部分作为物品展示容器。

4. \*\*物品预制体\*\*：创建 `ItemPrefab` 作为每个物品的显示模板，包含物品名称、数量和丢弃按钮。

### \*\*2. 添加物品和丢弃物品\*\*

1. \*\*添加物品到仓库\*\*：使用 `InventoryUI.AddItemToInventory()` 添加物品，`LootSpawner` 生成物品后，可以将它添加到仓库中。

2. \*\*丢弃物品\*\*：每个物品的 `ItemUI` 脚本会自动为其绑定一个丢弃按钮，点击按钮时，调用 `InventoryUI.DiscardItem()` 丢弃物品。

---