RPG starter kit - Save / Load

Save/load part of the RPG starter kit is located in the folder "SaveLoad". You can modify that script to save additional information.

How to update save/load depends what type of information you want to save. There is two type of information we want to save.

1. Player information like inventory, quest log, current hit points or global information like game time, current weather etc....

2. Other GameObjects - enemies, containers, items, unlocked doors etc...

1. Player Information

You need to modify script "PlayerInformation" (Scripts\RPG\Hero\Stats) and there you can add any field or whole class. This class is automatically stored in saved position.

2. Other Game objects

You want to save other game objects with attached script that contains usefull information.

In this example we want to store this class.

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class Container : MonoBehaviour {
	public List<string> Items;
	// Use this for initialization
	void Start () {

		//items initialization
		.....
		....
	}
	// Update is called once per frame
	void Update () {
		//playet takes some item
		.....
	}
}
```

Important information for us is collection of Items (bold font).

a) First Change MonoBehaviour to BaseGameObject. The result will be like this

```
public class Container : BaseGameObject {
```

```
        public List<string> Items;
        // Use this for initialization
        void Start () {

                //items initialization
                .....
                ....
        }
        // Update is called once per frame
        void Update () {
                //playet takes some item
                .....
        }
}
```

b) Create class "SavedContainer" in folder "Scripts\SaveLoad\SaveGameObjects". Recommended name will be "SavedContainer". Our new class should look like this.

```
using UnityEngine;

using System.Collections;

using System.Collections.Generic;



public class SavedContainer : BasicSaveEntity {

        public List<string> Items;

}
```

b) Now we have to update script "SavedScene". Add List<SavedChest> to this class.

new line: **public List<SavedContainer> Containers;**

and update constructor: **Containers= new List<SavedContainer>();**

so SavedScene should look like this now:

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class SavedScene
{
        public List<SavedCharacter> Characters;
        public List<SavedItem> Items;
        public List<SavedContainer> Containers;
        public int SceneId;
```

```
        public SavedScene()
        {
                Characters = new List<SavedCharacter>();
                Items = new List<SavedItem>();
                Containers= new List<SavedContainer>();
        }
}
```

c) Now we have to update script "SaveLoadObjects". Add declaration

We need to create new method "StoreContainer"

```
private void StoreContainer(Container item)
{
        SavedContainer si = new SavedContainer ();
        si.FillBasicEntity(item.transform, item.startPosition, item.startRotation);
        si.Items = item.Items;
        scene.Containers.Add(si);
}
```

e) We need to use this method in public method "StoreContent". You have to put in this cycle.

```
foreach(GameObject go in objects)
{
        Character character = (Character)go.GetComponent<Character>();
        if (character != null)
        {
                StoreCharacter(character);
        }
        Item item = (Item)go.GetComponent<Item>();
        if (item != null)
        {
                StoreItem(item);
        }
}
```

So result should be

```
foreach(GameObject go in objects)
{
        Character character = (Character)go.GetComponent<Character>();
        if (character != null)
        {
                StoreCharacter(character);
        }
        Item item = (Item)go.GetComponent<Item>();
        if (item != null)
        {
                StoreItem(item);
        }
```

```
        Container container = go.GetComponent<Container>();
        if (chest != null)
        {
            StoreContainer(chest);
        }
}
```

f) Create new method "RestoreChest". This method is very similar to RestoreItem or RestoreWeapon.

```
private bool RestoreContainer(GameObject go, SavedScene selected)
        {
            Container item = go.GetComponent<Container>();
            //items
            if (item != null)
            {
                bool founded = false;
                foreach(SavedContainer saveItem in selected.Containers)
                {
                    if (saveItem.CheckPosition(item.startPosition,
item.startRotation))
                    {
                        founded = true;
                        //adjust settings for founded character
                        go.transform.position =
saveItem.Position.GetPosition();
                        go.transform.rotation =
saveItem.Position.GetRotation();
                        item.container.Items = new List<ItemInWorld>();
                        item.container.Items = saveItem.Items;
                        selected.Containers.Remove(saveItem);
                        break;
                    }
                }
                if (founded == false)
                {
                    objectsToRemove.Add(go);
                }
                return true;
            }
            return false;
        }
```

g) Add calling this method LoadObjectsin cycle.

Thats all. Now containers are stored and also can be removed from the world after load

position.

Save / load in different game (not using RPG kit)

Necessary files for save / load system

1. Folder Save / Load except folder "SaveGameObjects". These classes you have to create alone.

2. MainMenuGUI for loading games (GUI folder)

3. Player class for loading / saving game

4. BaseGameObject for easy manipulation with saved objects (RPG/Base folder)