Room Management *in Unity Serializer*

You can use the features of the RoomManager class and its associated behaviours to manage “Rooms” in your game. Typically some scenes are effectively ***rooms*** which need their state saving when you move between scenes in your game, separate to the entire game being saved. The most likely difference is that you don’t want to save your player’s state at the time the room was left as this will of course be different when they come back later.

Unity Serializer RoomManager makes this process easy. It provides the following features:

- Handles the saving of all room states when you really save the game
- Marks objects as not being suitable for saving as a part of room state
- Sends a *OnRoomWasLoaded* message to all game objects that were saved – you can use this on your own components to place the player etc.
- Handles loading of scenes as rooms with automatic handling for cases where data is not present

Using RoomManager

To use the room manager you:

1. Create a new game object and attach some components to act as a Room Management Object
2. Attach scripts to the components you do not wish to save when they are a part of a room (like the Player).
3. Write a script to place the player in the loaded room
4. Use the RoomManager class to load your scenes.

Room Management Object

To each of your *rooms* you need to add a game object that will act as the Room Management Object. This automatically saves the state of the room when a new one is loaded.

Create a GameObject and call it Room Manager – attach a StoreInformation script to it. Then also attach the scripts Room (Components>Storage>Rooms>Room) and RoomDataSaveGameStorage (Components>Storage>Rooms>Room Data Save Game Storage).

Room handles saving the room when you leave.

RoomDataSaveGameStorage handles saving all of the room information when you save your game.

Flag objects for exclusion

To each object that should not be saved when this is a *room save* add a DontStoreObjectInRoom script (Components>Storage>Rooms>Dont Store Object In Room). You are likely to want to do this for your player.

Script the initialization of the player in the room

Create a script that will put the current player in the new room.

One good way to do this is to mark your player as DontDestroyOnLoad and then just put them at the right spawn point when they re-enter.

You could easily write a script that you attach to an empty game object with a box collider set to isTrigger that acts as a spawn point. Put these objects at the places the player might leave the room. When the player enters the trigger you flag that this is the current spawn point, this information is stored with the room and used in OnRoomWasLoaded to put the player at the right place.

Here is some example code for a basic player locator:

```
public class PlayerLocator : MonoBehaviour
{
        public static PlayerLocator Current;
        public static GameObject PlayerGameObject;

        void Awake()
        {
                DontDestroyOnLoad(gameObject);
                Current = this;
                PlayerGameObject = gameObject;
        }
}
```

This would be the code to be attached to a spawn point:

```
[RequireComponent(typeof(SphereCollider))]
[RequireComponent(typeof(StoreInformation))]
public class PlayerSpawnPoint : MonoBehaviour
{
        public static PlayerSpawnPoint currentSpawnPoint;

        public bool current
        {
                get
                {
                        return currentSpawnPoint == this;
                }
                set
                {
                        if(value)
                                currentSpawnPoint = this;
                        else if(currentSpawnPoint == this)
                                currentSpawnPoint = null;
                }
        }

        void Awake()
        {
                collider.isTrigger = true;
        }

        void OnTriggerEnter(Collider other)
        {
                if(other.gameObject == PlayerLocator.PlayerGameObject)
                {
                        current = true;
                }
        }
        void OnRoomWasLoaded()
        {
                if(current)
                {
                        PlayerLocator.Current.transform.position = transform.position;
                        PlayerLocator.Current.transform.rotation = transform.rotation;
                }
        }

}
```

These examples can be used and are included in Components>Storage>Rooms>Examples.

Loading Scenes with Room Manager

When working with the system you now use RoomManager to load your scenes. It performs the following tasks:

- Saves the current room
- Loads the new room as a vanilla room if there is no data; or
- Loads the room data back from storage
- Ensures that all components that were loaded receive an OnRoomWasLoaded message

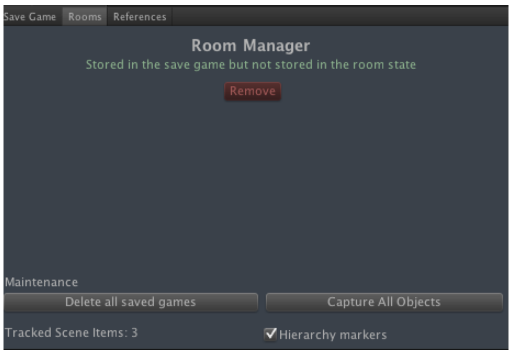
To load a scene using RoomManager you just do RoomManager.LoadRoom("yourRoomName").

You can also pass a bool parameter to suppress standard loading GUI.

RoomManager.LoadRoom("anotherRoomName", false);

Wizard

Unity Serializer Wizard now contains an extra page to help you configure the room status of objects – this works in a similar way to the standard configuration page for saving data. It can also automatically create the objects to be used as room managers etc.



Stored in the save game but not stored in the room state

Tracked Scene Items: 3