Room Building Manual

Overview:

This document is designed to give an explanation and overview of the Room Component GUI as well as explain the object hierarchy for room objects. It will also contain an example of building a room from scratch. Figure 1 contains a screenshot of the room component GUI that the room builder will have to use to build the room. Table 1, following figure ,1 gives a description of each of the fields seen in figure 1. Figure 2 shows a screenshot of the room prefab object hierarchy that the room builder will place objects into.

Inspector:

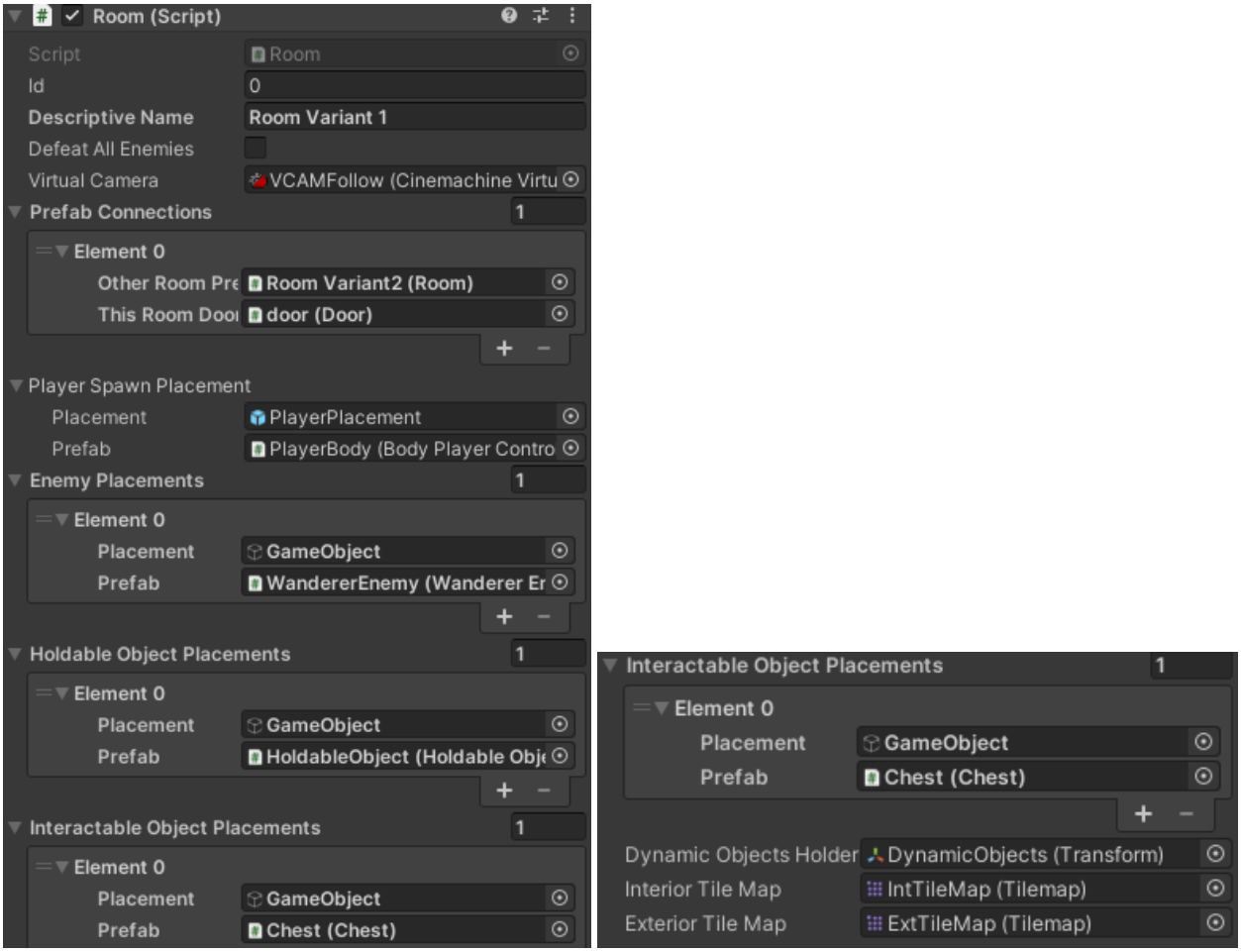

Figure 1A-B: Screen shot of the room component GUI with some field expanded. A) shows inspector fields from the top of the inspector to the interactable object Placement fields. B) shows inspector fields from the interactable objects placements field to the end of the inspector.

Table 1: Detailed description of each GUI element

GUI Field	Explanation
ID (Integer)	The UNIQUE room id. Every room needs a different id. This is used to get resources from disk. Can be set to any value greater than or equal to 0 Example Value: 0
Descriptive Name (String)	Name that the room designer can give to the room to help identify later, can be left blank Example Value: Bedroom
Defeat All Enemies (Bool)	A gameplay flag that if checked will make it so doors in the room remain locked until all enemies are defeated Example Value: false
Virtual Camera (Object Reference)	A reference to the cinemachine virtual camera for the room. It should already be set to reference the virtual camera in the object hierarchy so no need to change. Example Value: See figure 1
Prefab Connections (Custom Prefab Connection Type)	For every room that will be connected to the currently open room, there will need to be a prefab connection. Each prefab connection consists of a “Other Room Prefab” which is a reference to a different room object prefab variant and a “This Room Door” which is a reference to the door in the currently open room that leads to the “Other Room Prefab” To change the number of connections click the + or - symbols Example Value: See figure 1
Player Spawn Placement (Custom Object Placement Type)	Every room needs a player spawn placement setup. The player spawn placement field consists of a “Placement” value which is a reference to an empty GameObject within the room object hierarchy (Figure 2) which acts as the placement position and a “Prefab” reference which is the player controller prefab object to spawn at the “Placement” position. The “Placement” object should already be setup, the designer only needs to position the referenced object in the room hierarchy (See figure 2) in the desired location. The “Prefab” should also be set up and should not be changed. Example Value: See figure 1
Enemy Placements (Array of Custom Object Placement Type)	Every Enemy that belongs in a room needs an Object Placement setup in this array. It sets up very similarly to the Player Spawn Placement field. The only differences are that you use an Enemy Prefab object instead of a player controller prefab, and that you should place the “Placement” empty GameObject as a child of the Enemy Placements object in the room hierarchy (Figure 2). To add or remove use the + or - symbols Example Value: See Figure 1

Holdable Object Placements (Array of Custom Object Placement Type	Similar to the Enemy placements field except that you place your “Placement” GameObjects as children of the HoldablePlacements object in the room hierarchy (Figure 2) and you must use Holdable Object prefabs for the “Prefab” field. To add or remove use the + or - symbols Example Value: See Figure 1
Interactable Object Placements (Array of Custom Object Placement Type	Similar to the Enemy placements field except that you place your “Placement” GameObjects as children of the InteractablePlacements object in the room hierarchy (Figure 2) and you must use Interactable Object prefabs for the “Prefab” field. This field is for chests, NPCs, equipment pick ups. It is not for Doors, or holdable objects. To add or remove use the + or - symbols Example Value: See Figure 1
Dynamic Objects Holder (GameObject Type)	This is just a reference to an empty Game Object in the room hierarchy (See Figure 2). It is used to hold objects that get created during gameplay so that the room can keep track of them. The Room designer should not touch this.
Interior Tile Map (TileMap Type)	This is a reference to the interior tilemap object in the room hierarchy (See Figure 2). It is used to figure out spacing between rooms. The Room designer should not touch this.
Exterior Tile Map (TileMap Type)	This is a reference to the exterior tilemap object in the room hierarchy (See Figure 2). It is used to figure out spacing between rooms. The Room designer should not touch this.

Room Object Hierarchy:

Figure 2 shows an example room's object hierarchy with all of the objects transform hierarchies expanded. See Table 2 for an explanation of objects. Of note are the two tile maps. When drawing out the tiles for the rooms make sure you have selected the correct one. Additionally, there are the placement objects that are used as containers for other objects. Last, note the addition of the door prefab. This is how doors should be placed into the room's hierarchy. For more details see Table 2

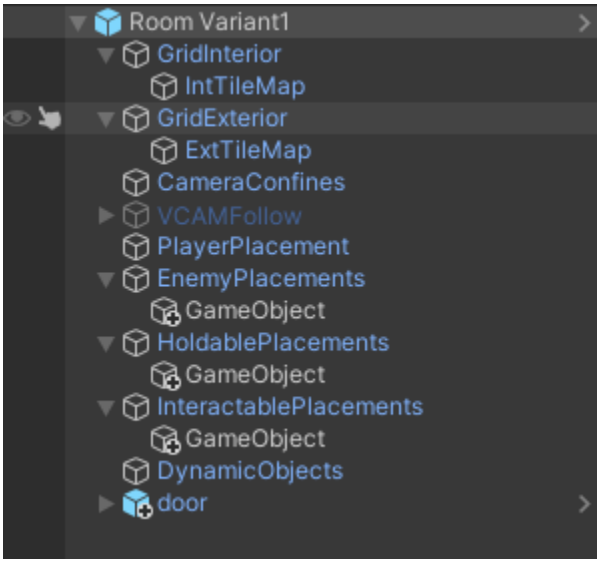

Figure 2: Screen shot of the default room object hierarchy with some changes made. Changes include adding some enemies, holdable, and interactable object placements as well as adding 1 door to the object.

Table 2: Description of Objects in the Room Prefab Variants Hierarchy

Object	Explanation
IntTileMap	Is used for drawing the tiles of the actual room. This includes floors and the tiles the have the wall sprites.
ExtTileMap	Is used to draw the tiles outside of the room that the player does not interact with like solid brick walls that act as tiles between rooms.
CameraConfines	This object has a polygon collider 2D component that is used to confine the rooms Virtual Camera. Edit the polygon on this component to adjust the confines. For small rooms it is advised the confines polygon forms a perfect rectangle
VCAMFollow	This is the room's virtual camera. It needs to be turned off in the prefab variant as seen in Figure 2.

PlayerPlacement	This is an empty gameobject that is used to determine the spawn in location of the player. Position is as desired within the room to change the initial spawn location of the player.
Enemy Placements	This is the root object for setting up enemy placements (See Table 1). Add empty game objects here to create initial positions for the room's enemies and reference them in the Room object's inspector (See Figure 1)
Holdable Placements	This is the root object for setting up holdable object placements (See Table 1). Add empty game objects here to create initial positions for the room's holdable objects and reference them in the Room object's inspector (See Figure 1)
Interactable Placements	This is the root object for setting up interactable object placements (See Table 1). Add empty game objects here to create initial positions for the room's interactable objects and reference them in the Room object's inspector (See Figure 1)
Dynamic Objects Holder	This is just an empty GameObject that acts as a container for objects that may spawn during game play. Can be left as is
Door	This is an example of a door added to a room. Doors are just placed in the RoomObjects hierarchy and then they are referenced in the inspector (See Figure and Table 1)