GKC - ORK Framework Integration

Initial Setup

- Import and install ORK Framework 3 in your Unity project.
- Optional: Do or download one of ORK Framework's tutorial projects (Status System Setup, 3D RPG Playground, 3D Action RPG, etc.) Pick the project that is closer to your game.

  https://orkframework.com/guide/tutorials/

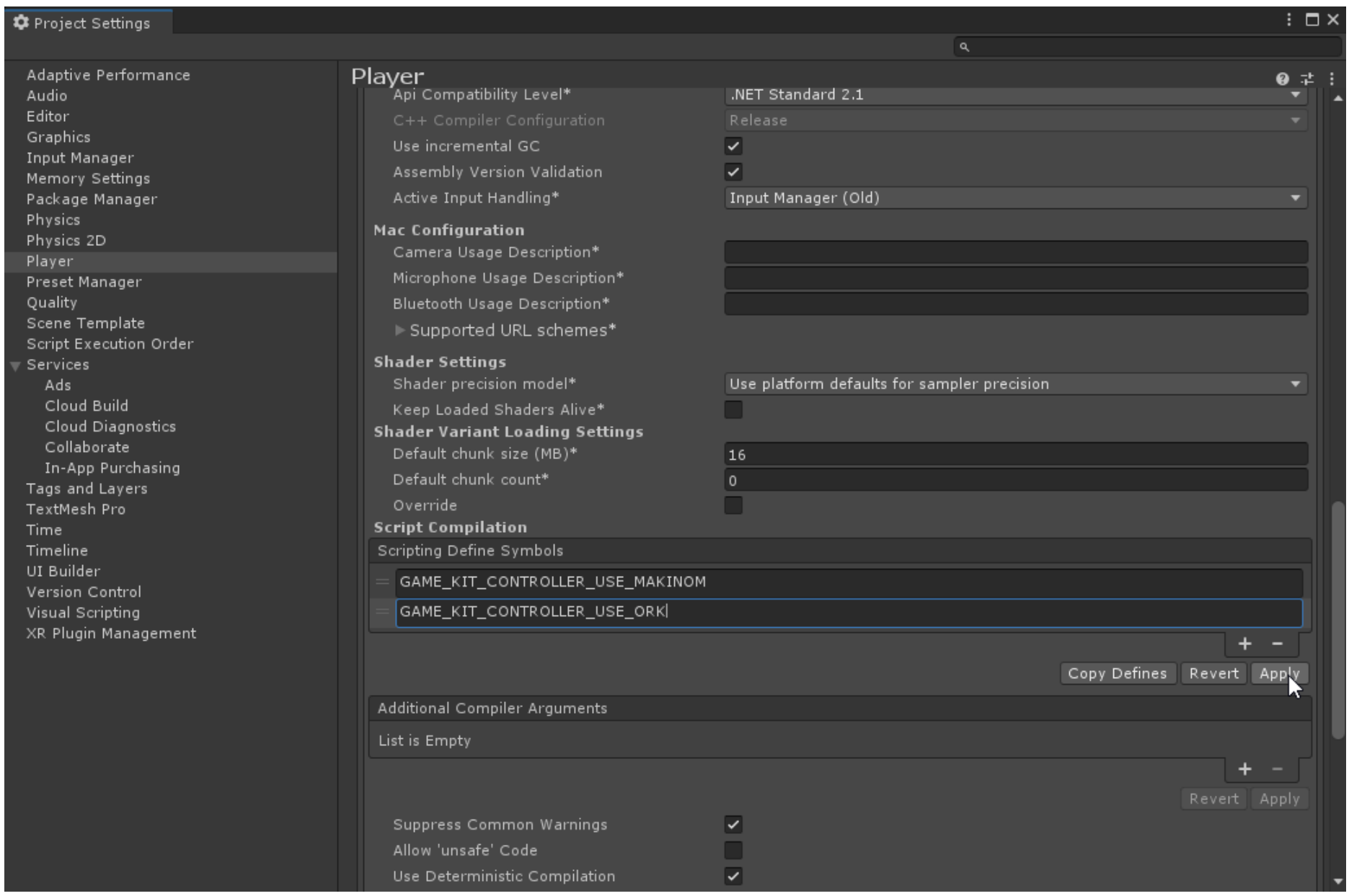
- Recommended: Follow this guide using the "3D Action RPG" completed tutorial project as a base for your own project.

  https://orkframework.com/guide/tutorials/3d-action-rpg/start-3d-action-rpg/#12-toc-title

- If you're starting an ORK project from scratch: Do the ORK Framework's initial setup, as described in the following pages. Alternatively, you can import one of the "Completed Setup" packages from ORK's tutorial projects mentioned above, to make this process faster.

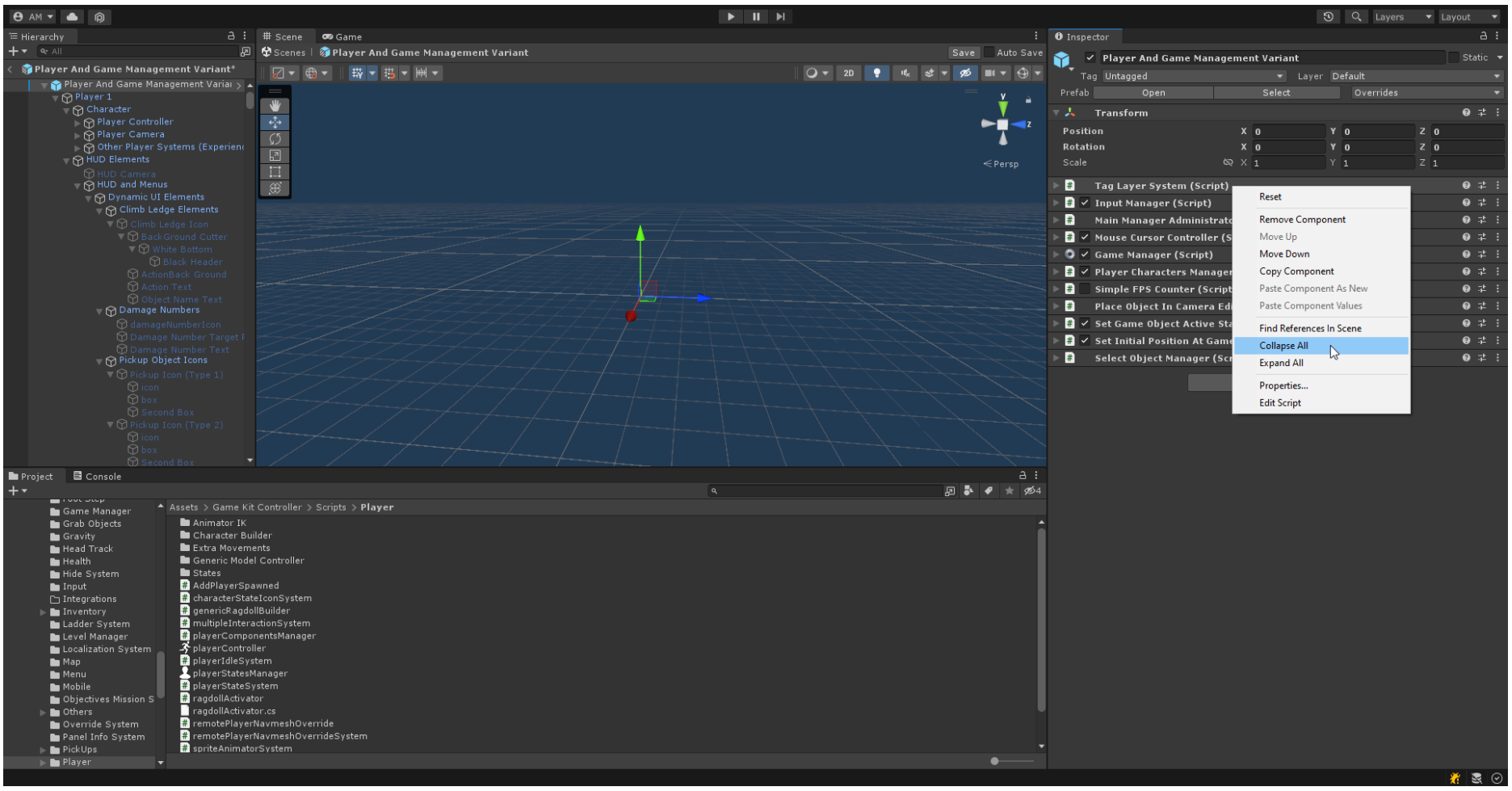
  https://orkframework.com/guide/tutorials/status-system-setup/start-status-system-setup/#5-toc-title
  https://orkframework.com/guide/tutorials/ui-setups/unity-ui-initial-setup/

- In Project Settings -> Player Settings, add the scripting define symbols "GAME_KIT_CONTROLLER_USE_MAKINOM" and "GAME_KIT_CONTROLLER_USE_ORK" (without quotes)

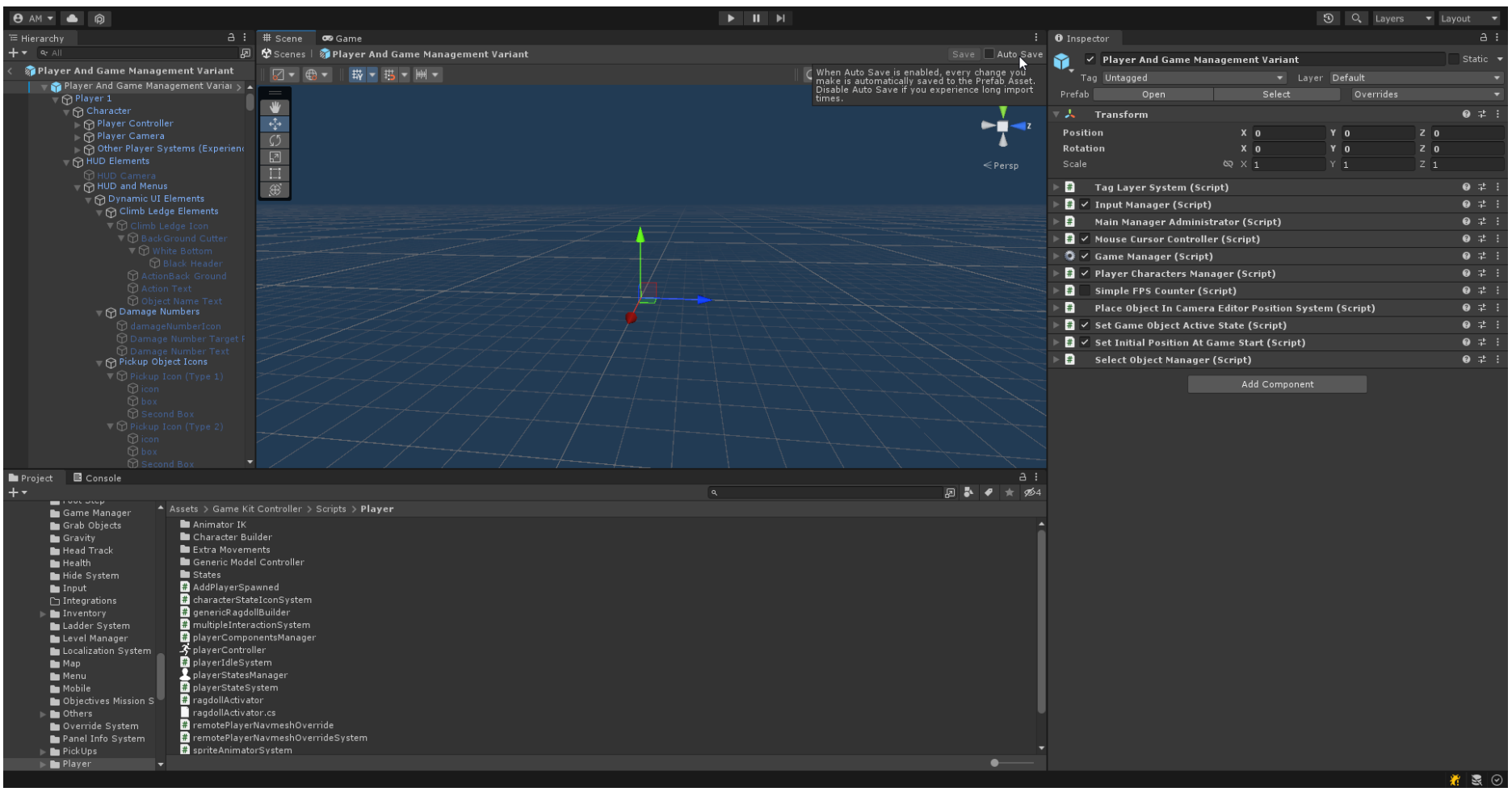


Prefab Setup

- TIP: Many gameobjects/prefabs of Game Kit Controller have a large number of components.
  To avoid these many expanded components slowing down the Unity Editor, you can right click on any component's header (the top area of the component, containing their name), then select "Collapse All". Then, expand only the components you're currently working on.
  This custom menu option is included with GKC.

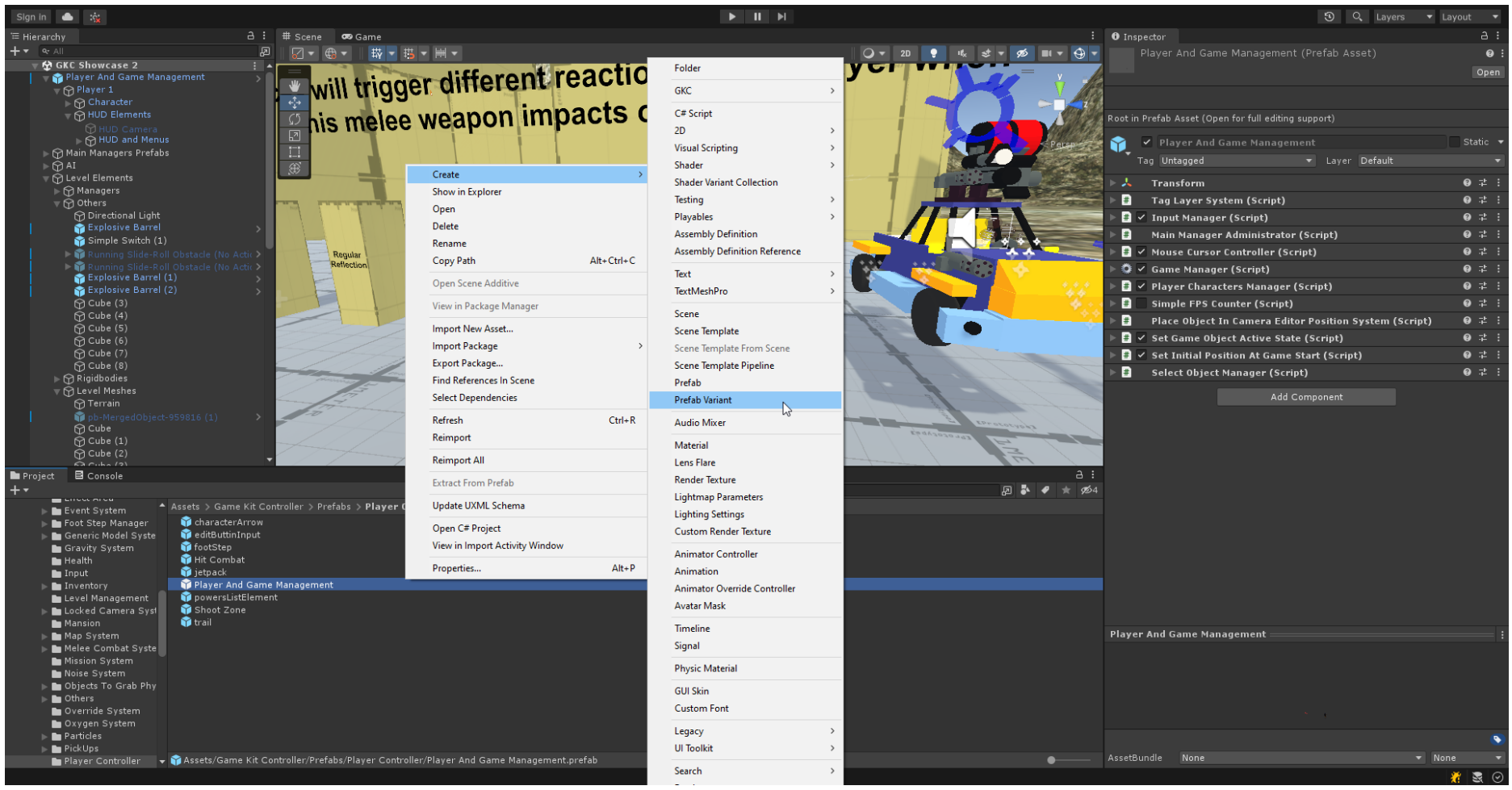


- TIP: Game Kit Controller has some prefabs with very large and complex structures. For this reason, we suggest disabling the "Auto Save" feature for prefabs, or else Unity Editor may hang/freeze for quite some time after changing *anything* on the prefab.



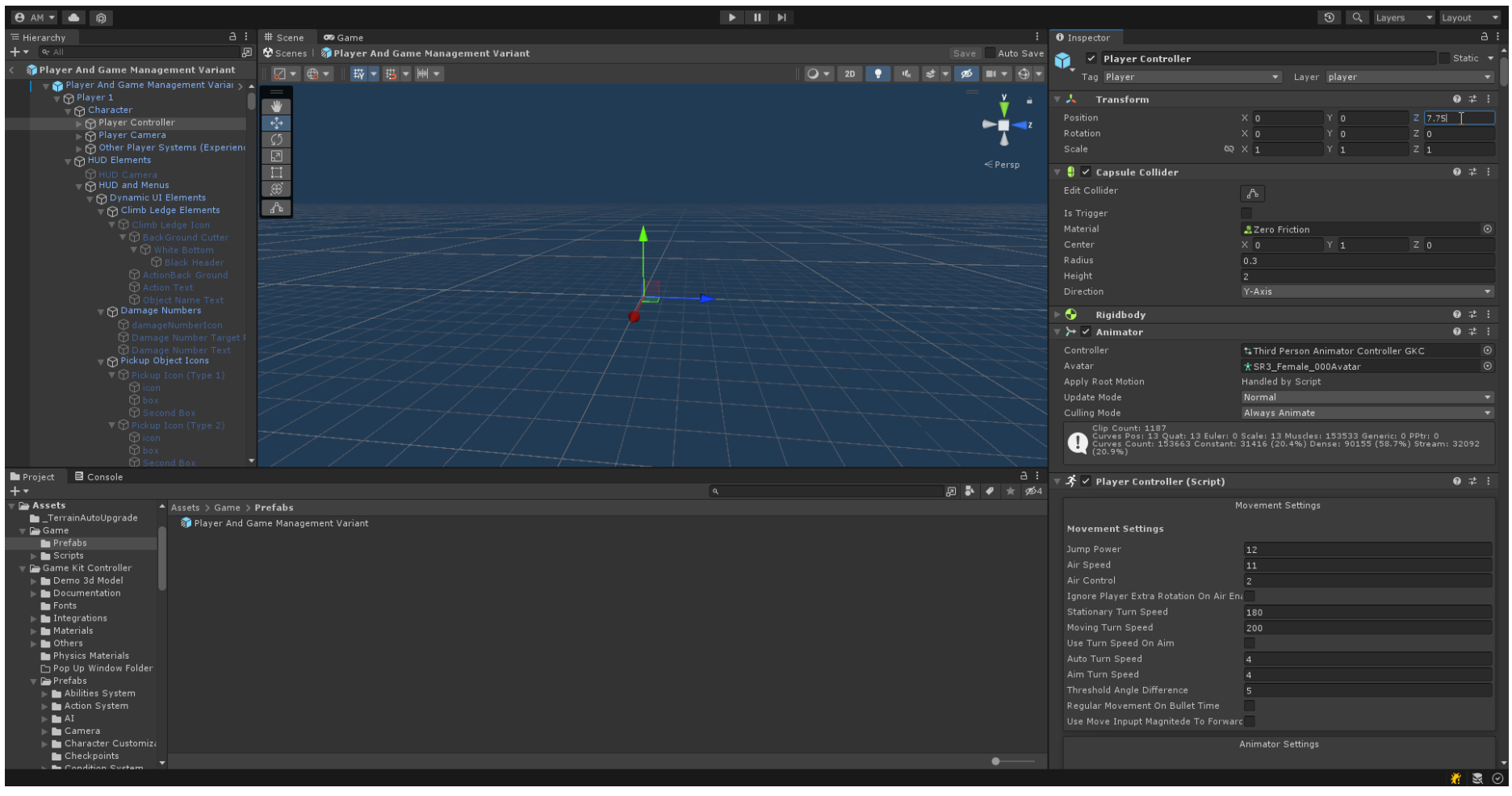
- Recommended: Create a prefab variant of [Player And Game Management] prefab.
  Save this prefab variant somewhere outside the “Assets\Game Kit Controller” folder.

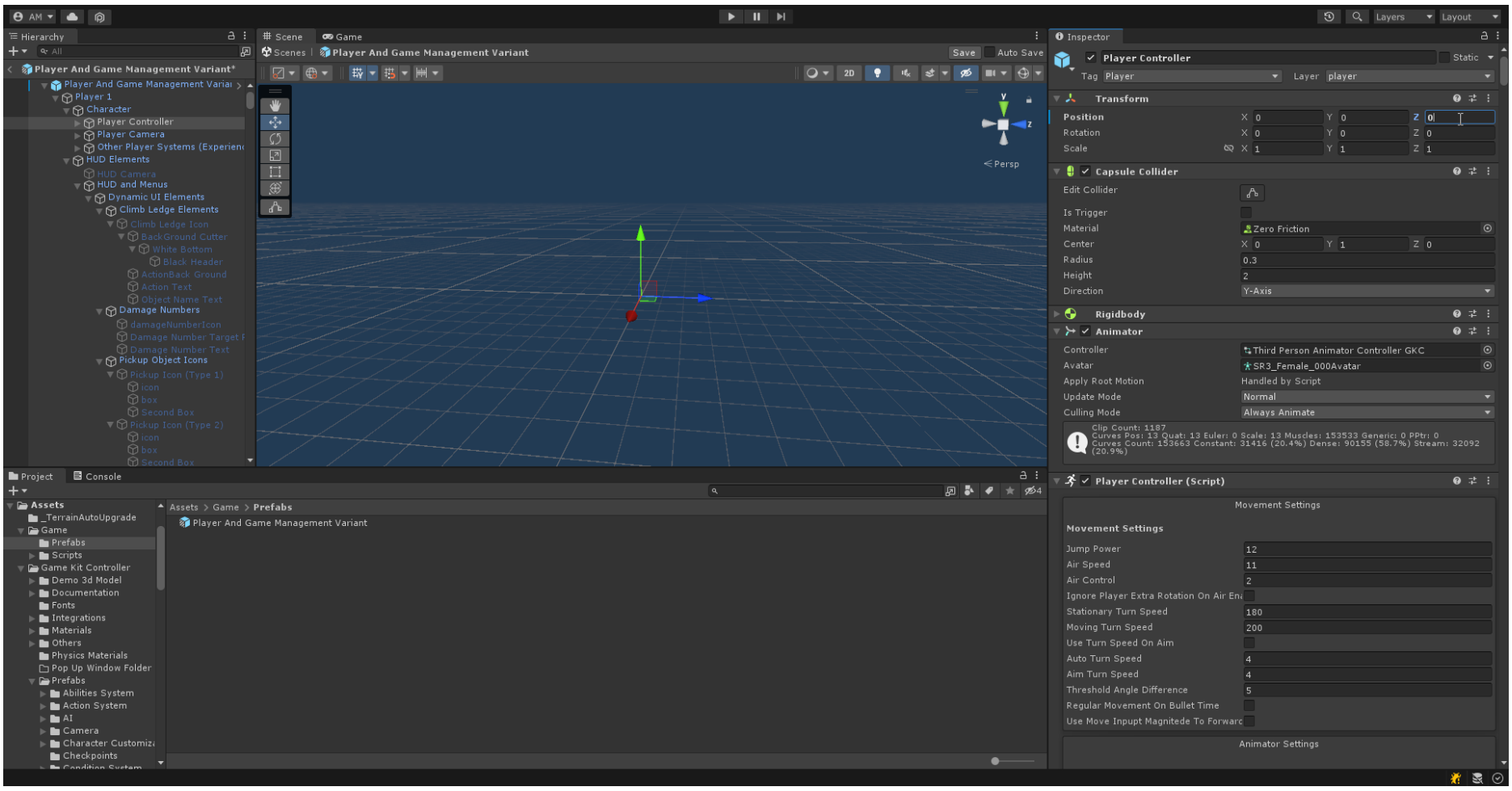
  This is so you can change stuff on the prefab variant without affecting the original prefab from GKC. Also, any prefab updates from future versions of GKC will be reflected on your new prefab variant, without overwriting your custom changes, whenever you update GKC to a newer version.

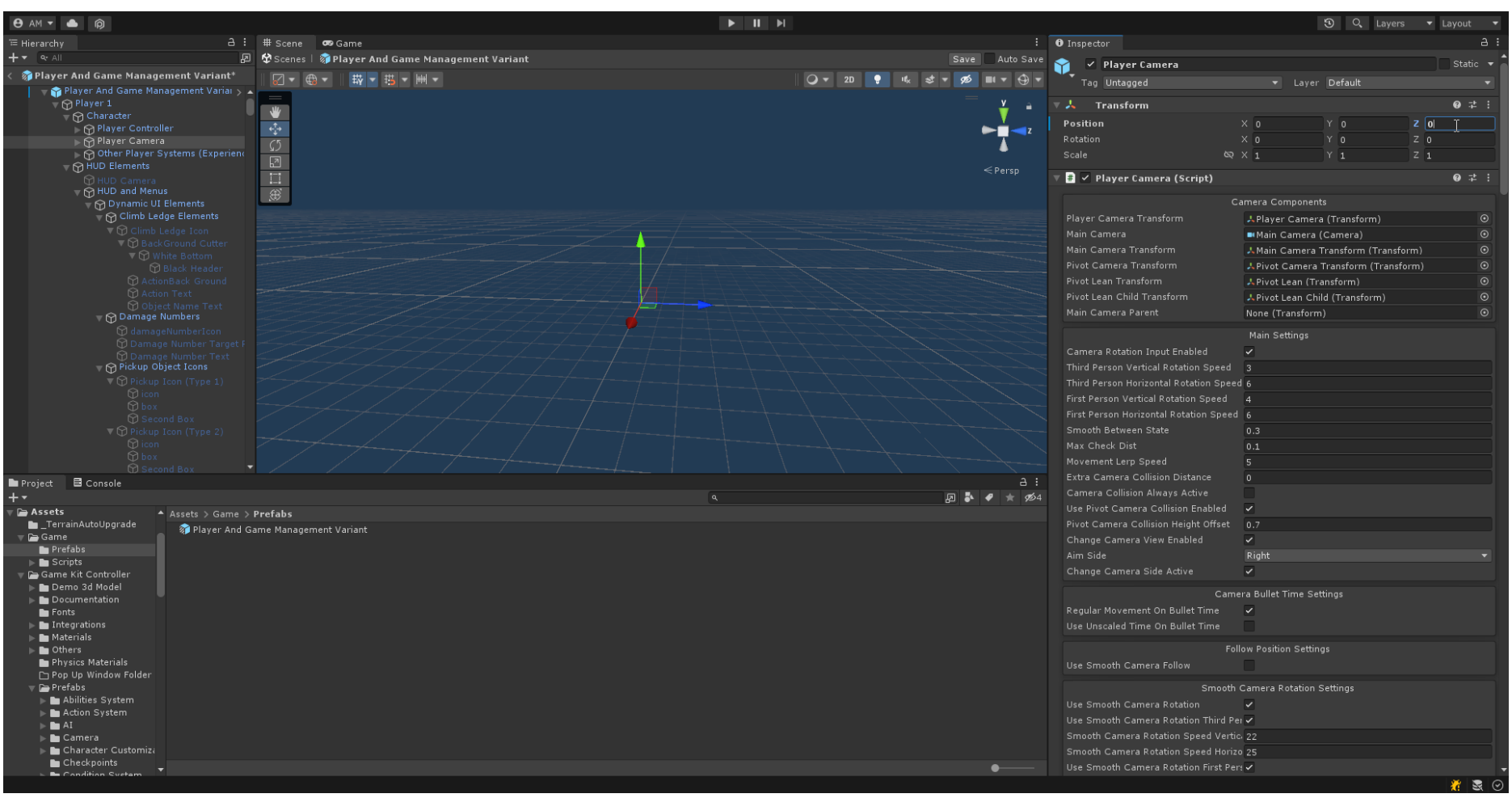


- Open [Player And Game Management Variant] in Prefab Mode.

- IMPORTANT: Check the Transform’s Z axis from the child objects “Player Controller” and “Player Camera”.
  If any of those contains a value different than 0, set their values to 0, for the ORK character spawning (described later) to work correctly (or else the character would be spawned at an offset of the desired spawn position).

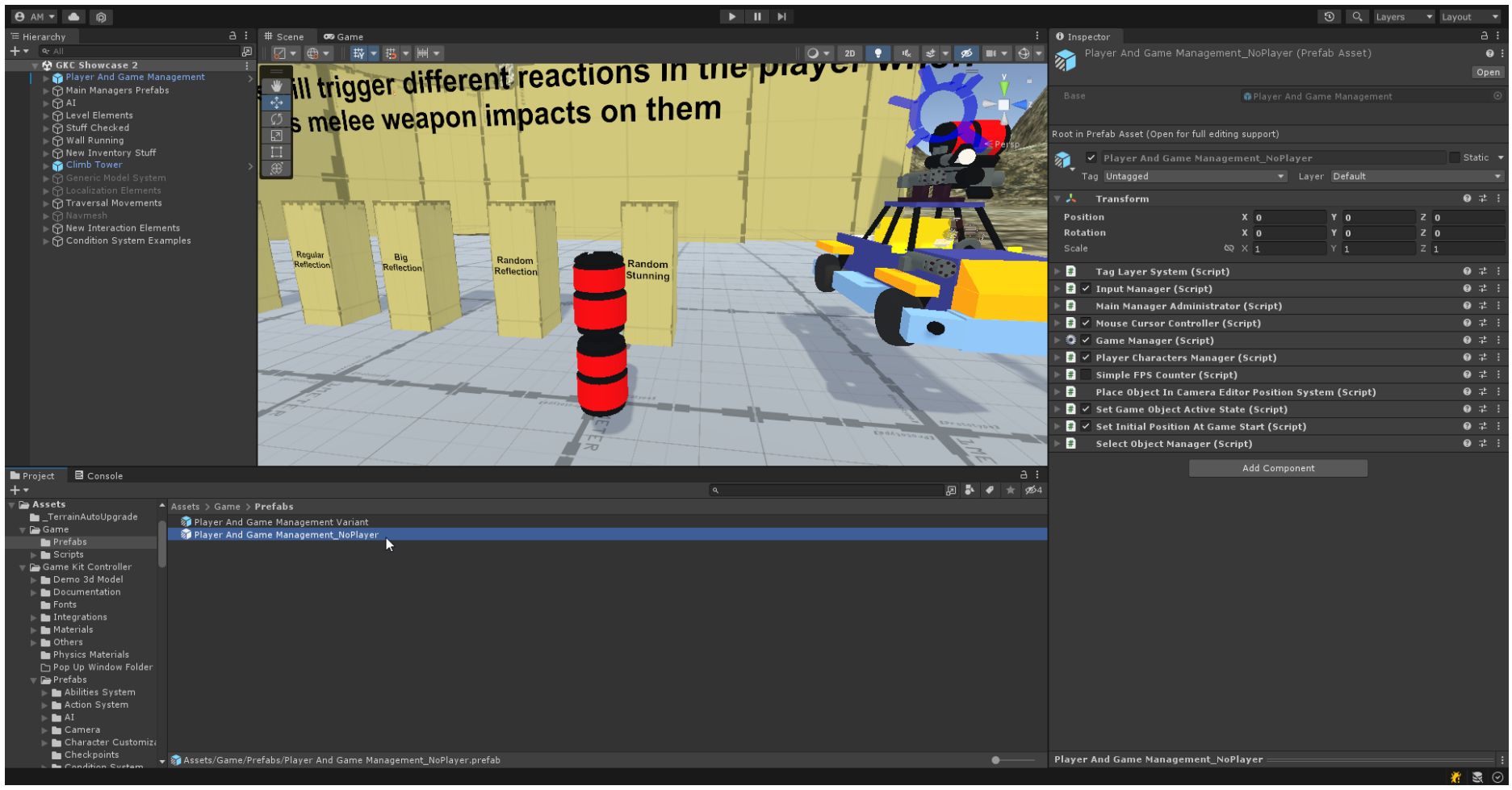




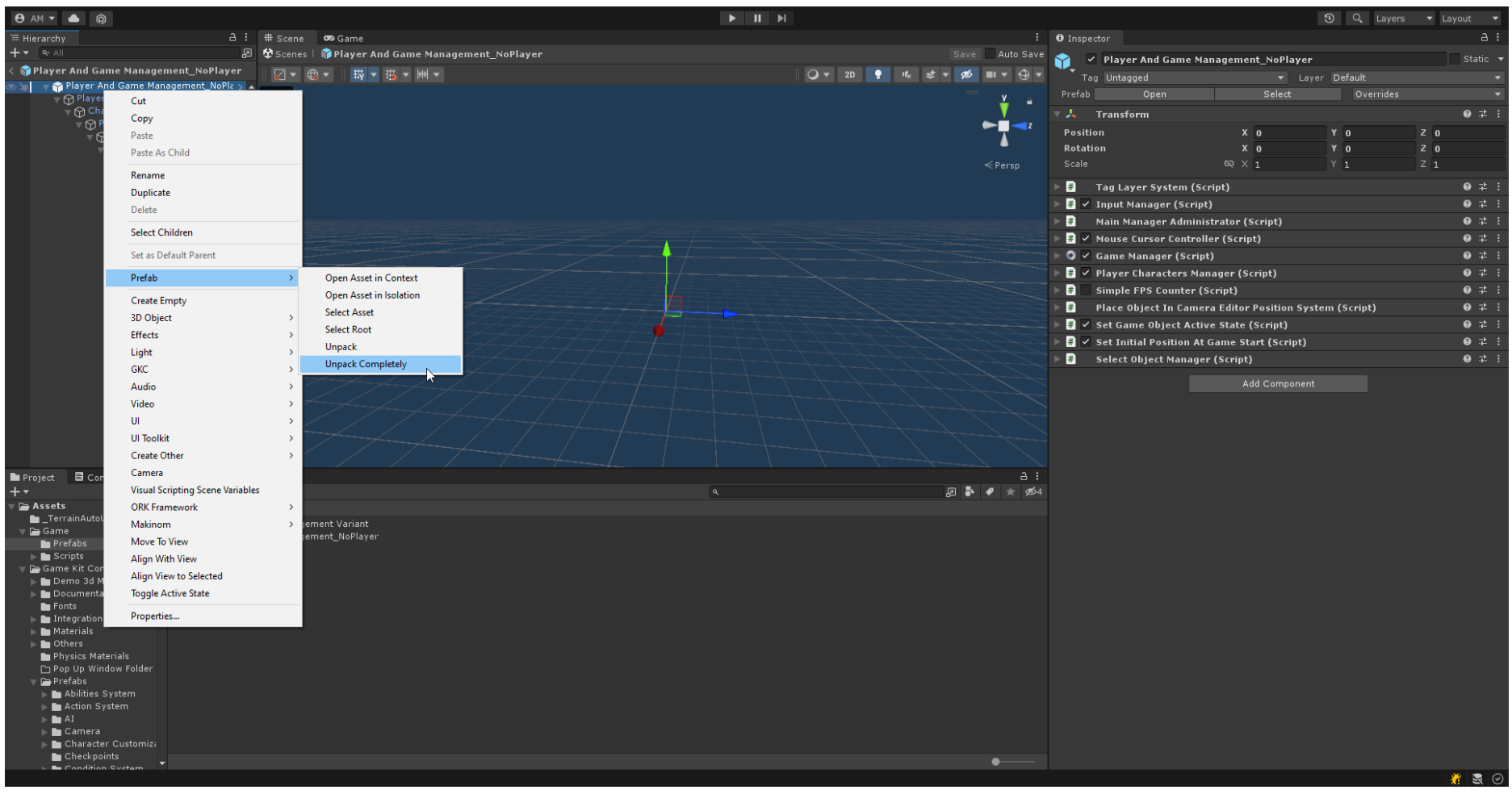


- Save the prefab. Exit Prefab Mode.

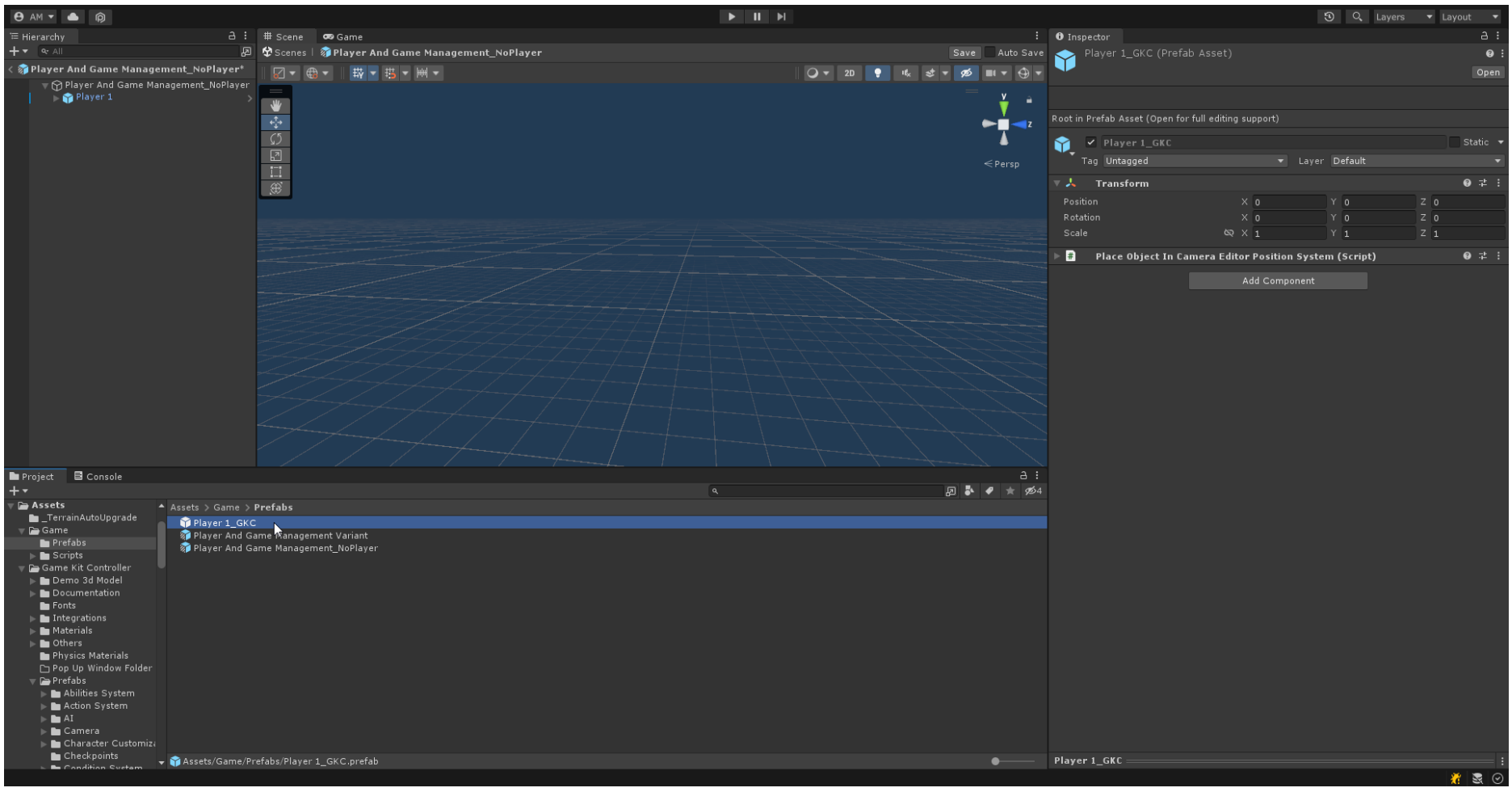
- Duplicate (Ctrl + D, or Command + D) the [Player And Game Management Variant] prefab we just made. Let’s call this [Player And Game Management_NoPlayer].



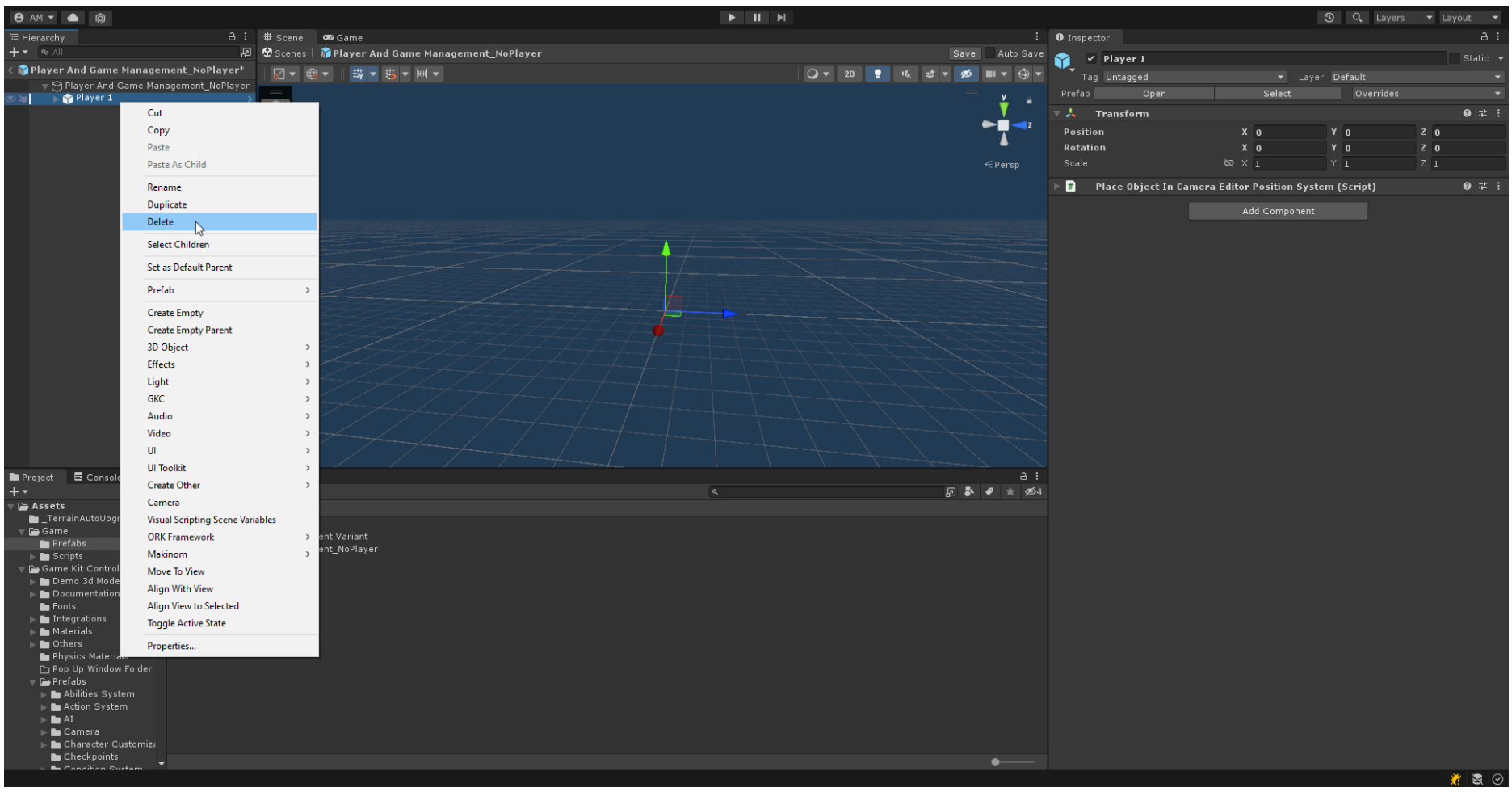
- Open [Player And Game Management_NoPlayer] in Prefab Mode.
- Right-click the root of the prefab, and select “Prefab -> Unpack Completely”



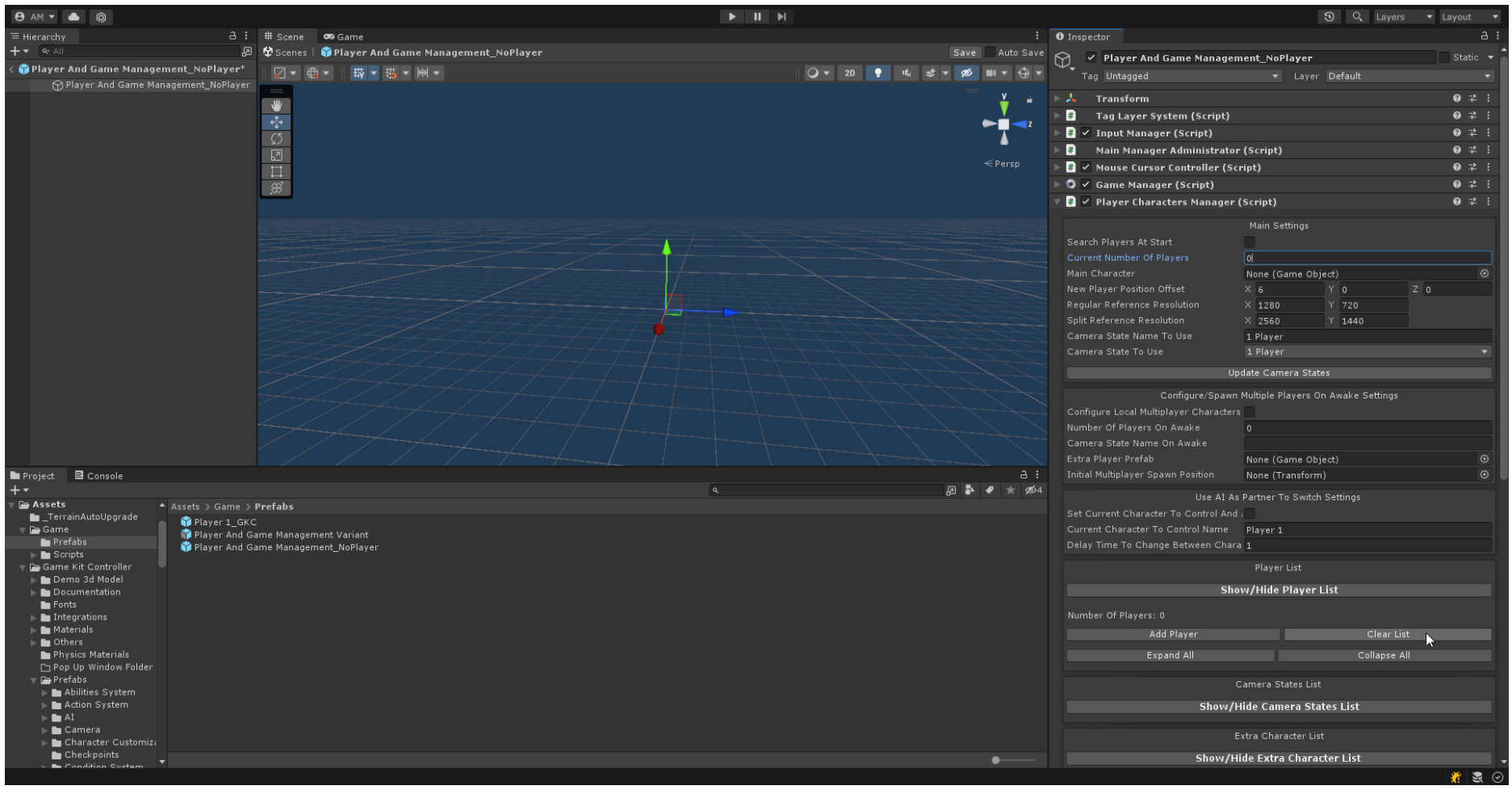
- Drag the “Player 1” child gameobject to the same folder as the earlier prefabs we created (in the Project view), to create a new prefab out of it. Let’s rename it to [Player 1_GKC].



- Delete the “Player 1” gameobject from [Player And Game Management_NoPlayer].

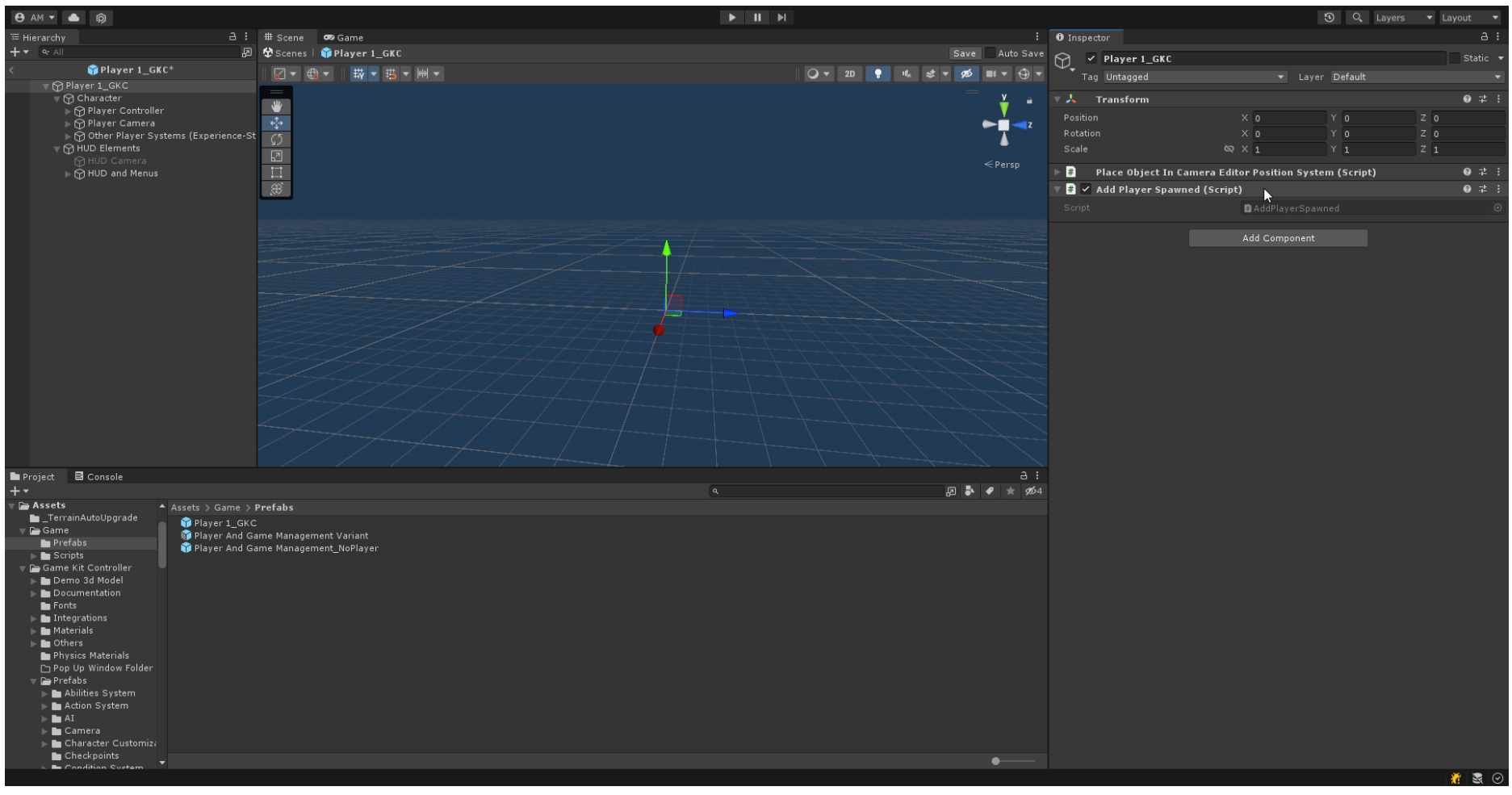


- Select the root gameobject.
  On the “Player Characters Manager” component, set “Search Players At Start” to false, and “Current Number of Players” to 0.
  Then, click the button “Show/Hide Player List”. Finally, click “Clear List”, to clear any existing players from the Player List’s data.

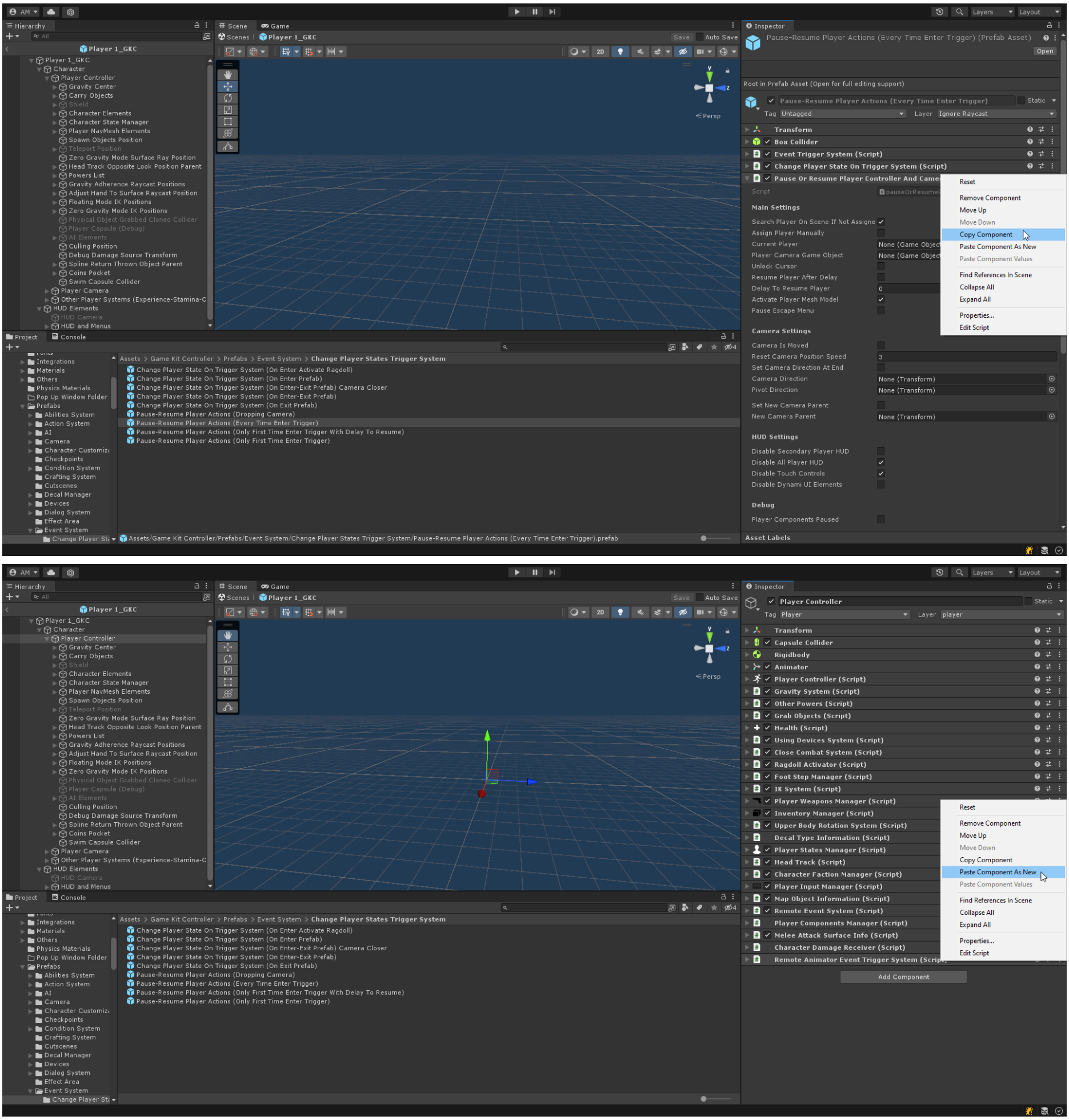


- Save the prefab. Exit Prefab Mode.

- Open [Player 1_GKC] in Prefab Mode.
  This prefab can be used for general dynamic spawning of GKC players, not only in the context of the ORK integration.
- Add the “Add Player Spawned” component to the root gameobject.

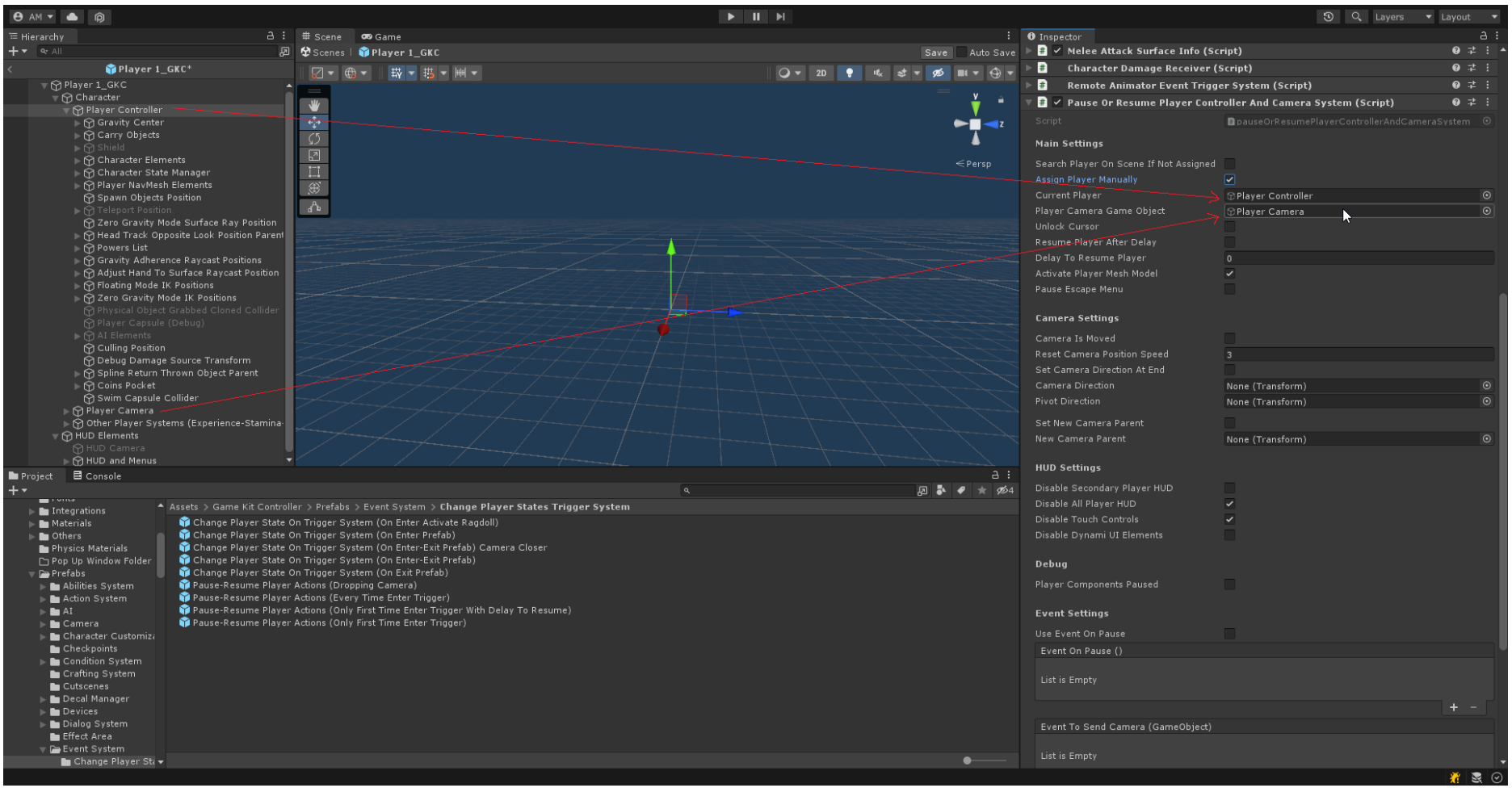


- On the “Player Controller” child gameobject, add the PauseOrResumePlayerControllerAndCameraSystem component.
  The easiest way to set it up is to copy the component from the “Pause-Resume Player Actions (Every Time Enter Trigger)” prefab (right-click that component’s header to show the dropdown options), then paste it as a new component on Player Controller (right-click any component’s header).



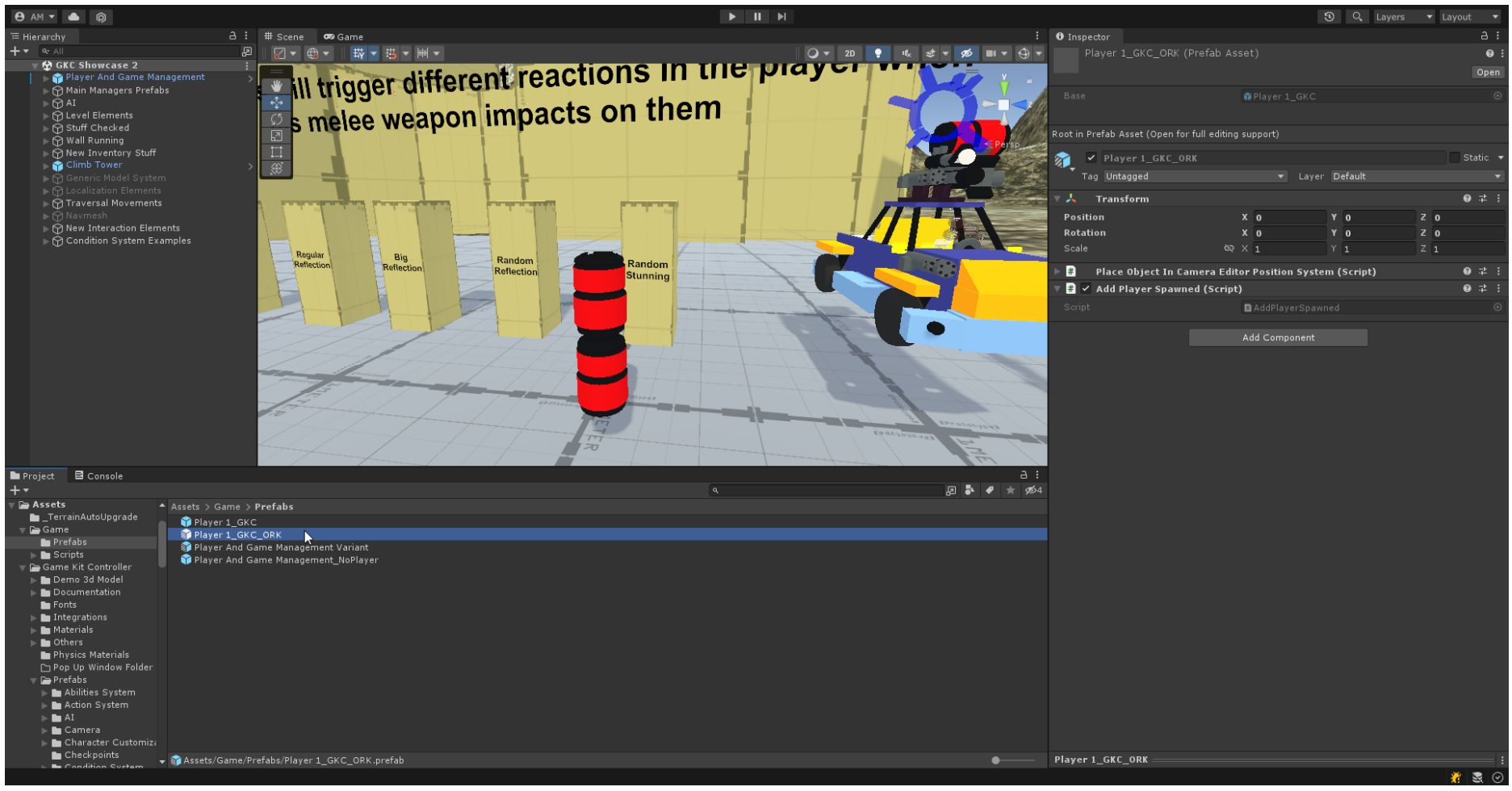
- On the newly added PauseOrResumePlayerControllerAndCameraSystem component, set up the following settings:

  Search Player On Scene If Not Assigned = false
  Assign Player Manually = true
  Current Player = drag the Player Controller gameobject from the prefab hierarchy
  Player Camera Game Object = drag the Player Camera gameobject from the prefab hierarchy



- Save the prefab. Exit Prefab Mode.

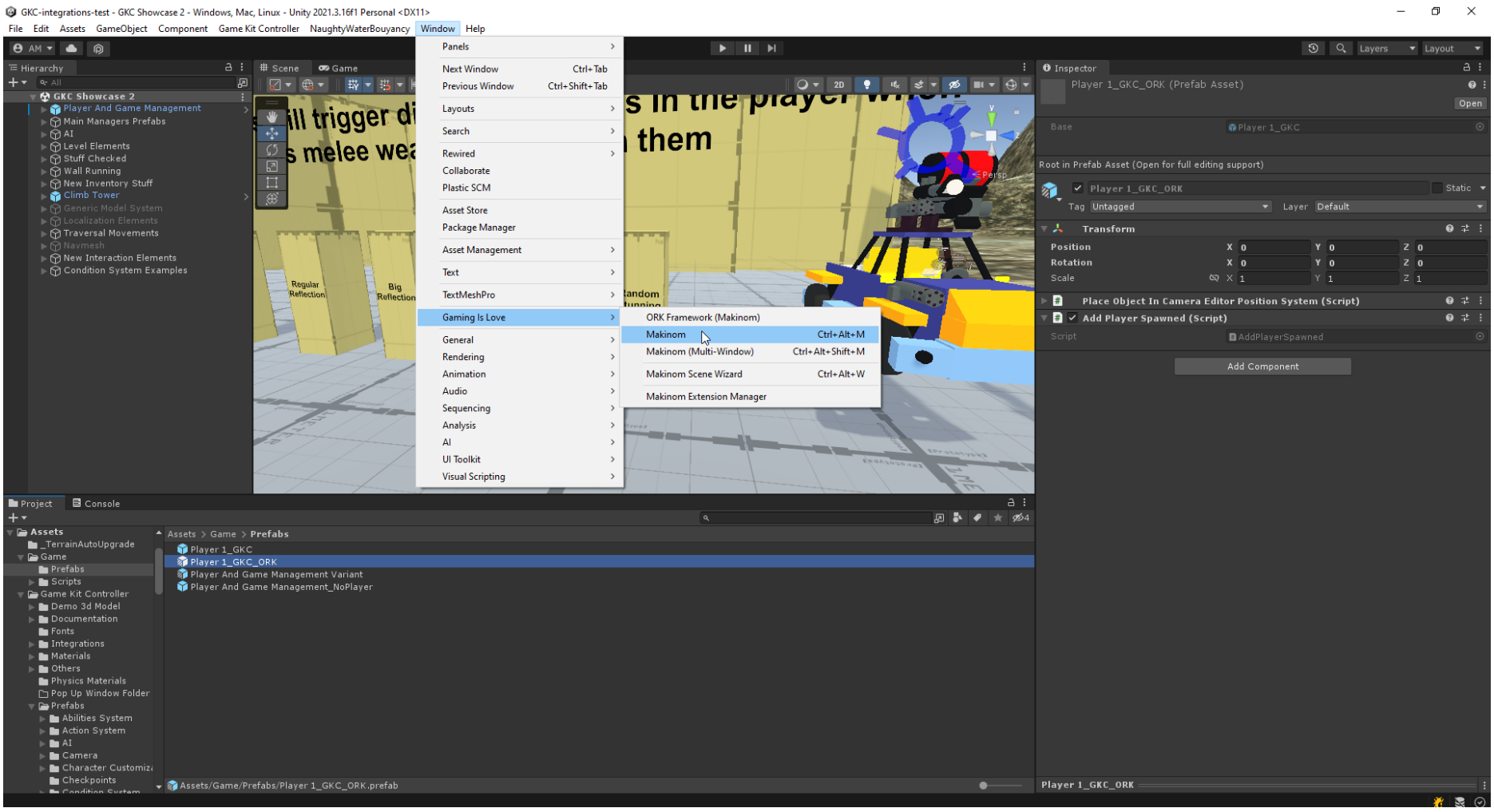
- Create a prefab variant of [Player 1_GKC] prefab.
  Let's call this [Player 1_GKC_ORK].



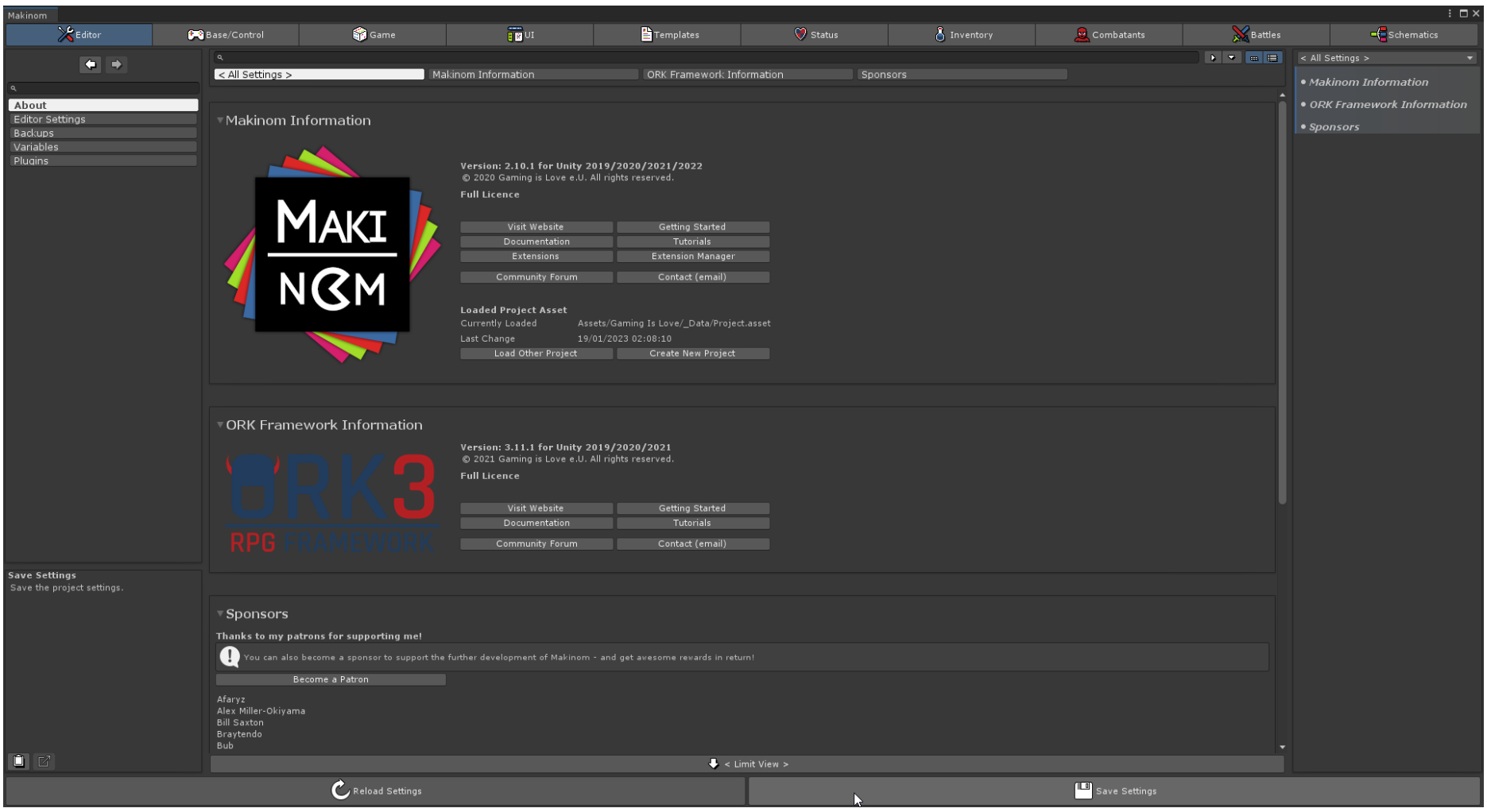
- Before resuming with the [Player 1_GKC_ORK] prefab, we need to do some setup in the Makinom/ORK database.

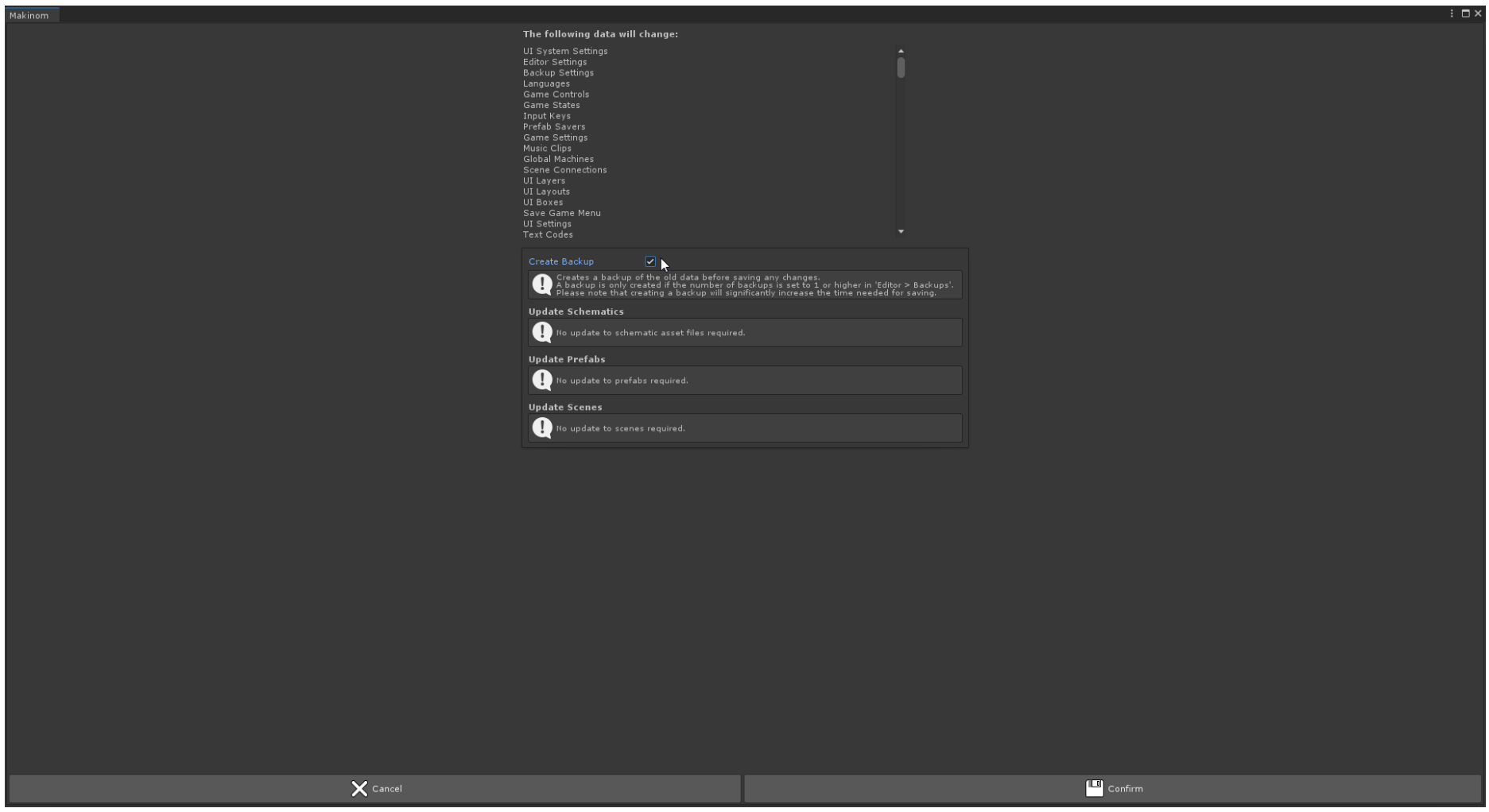
Makinom/ORK Database Setup

- Open Makinom Window: Window -> Gaming Is Love -> Makinom
  Or press Ctrl + Alt + M, or Command + Alt + M



- **If you're creating a new database from scratch:** At the bottom, click "Save Settings" and then "Confirm", to save the initial Makinom/ORK data (in case you haven't done this step earlier).
  You can disable "Create Backup" for faster saving (especially if your project is already on a Version Control System).

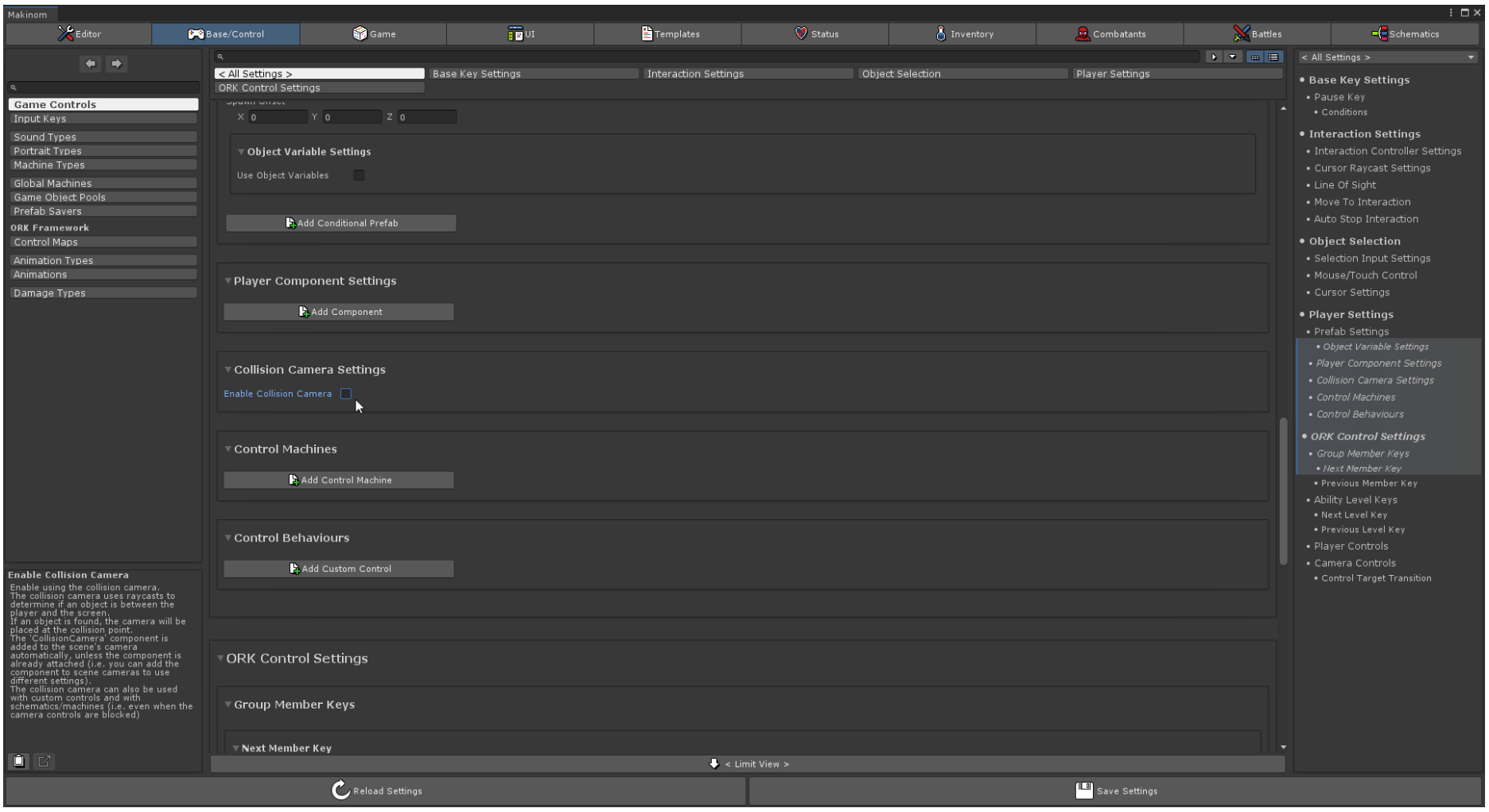




- **Recommended approach:** In case you don’t have an ORK project already set up. We suggest picking one of ORK’s tutorial projects that has its UI already set up (such as 3D RPG Playground, or 3D Action RPG), and using that project as a base for your own project, as setting up ORK’s UI (and some other elements) from scratch is outside the scope of this guide.

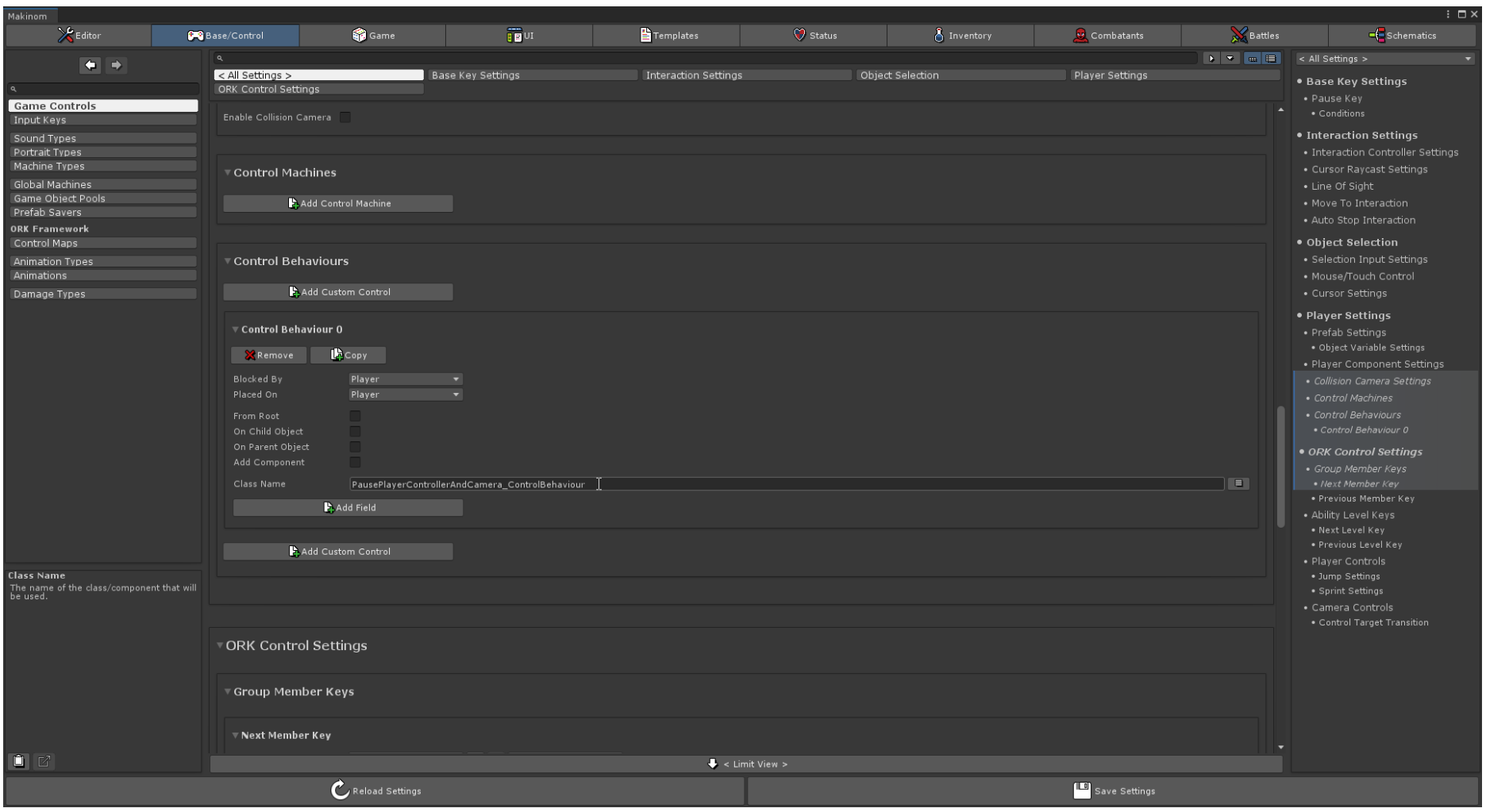
- IMPORTANT: The rest of this guide assumes we’re basing our ORK database on the completed “3D Action RPG” tutorial project.

- IMPORTANT: In [Base/Control] tab, [Game Controls] section, go to Collision Camera Settings.
  Make sure “Enable Collision Camera” is disabled, so ORK won’t add an additional Collision Camera component to the main GKC camera.

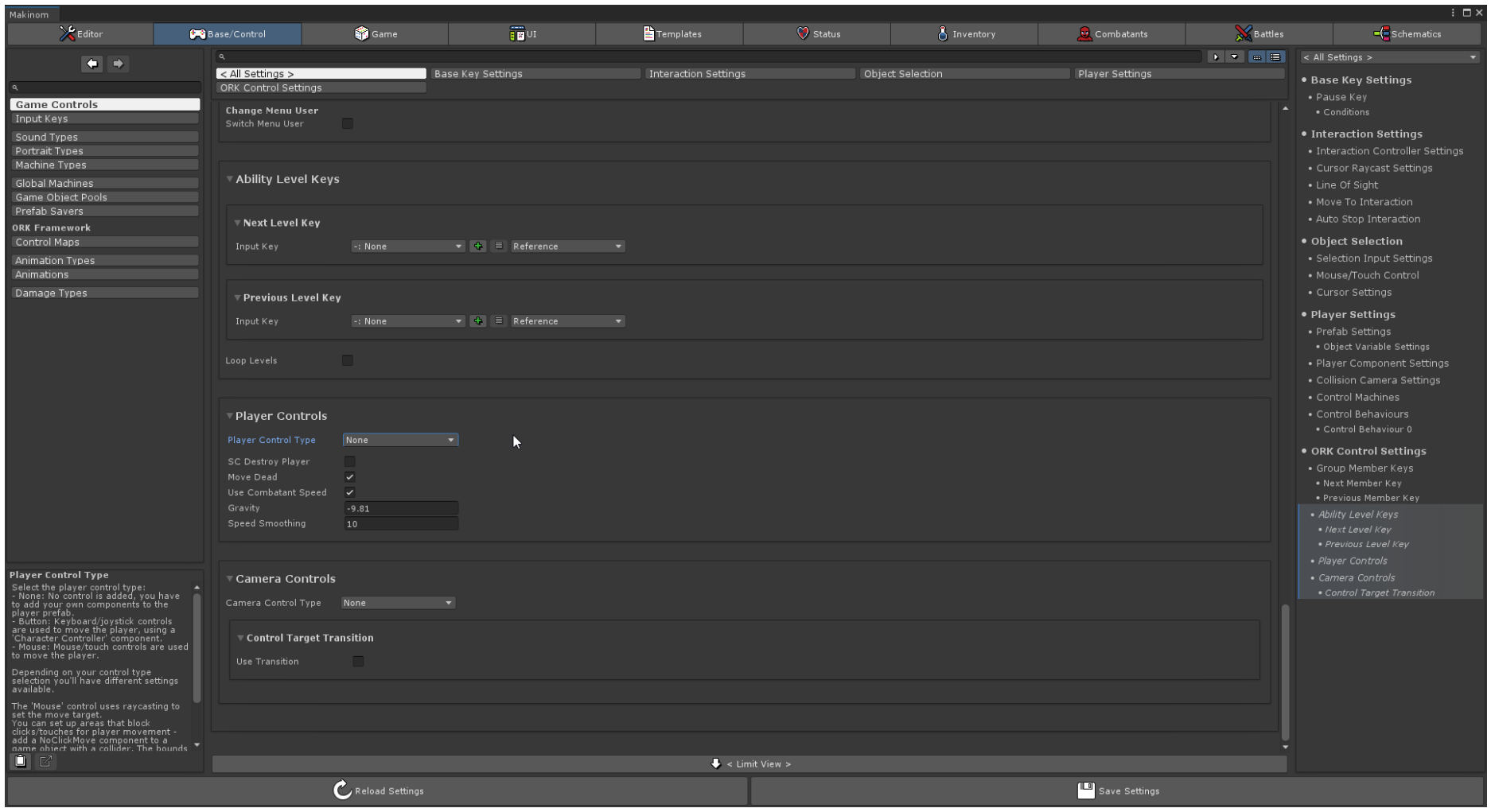


- IMPORTANT: Go to Control Behaviours, and add a new Control Behaviour with the info below.
  This will be used later on, to hide/show the GKC UI whenever ORK blocks/unblocks the player control (e.g.: While displaying an ORK NPC dialogue)

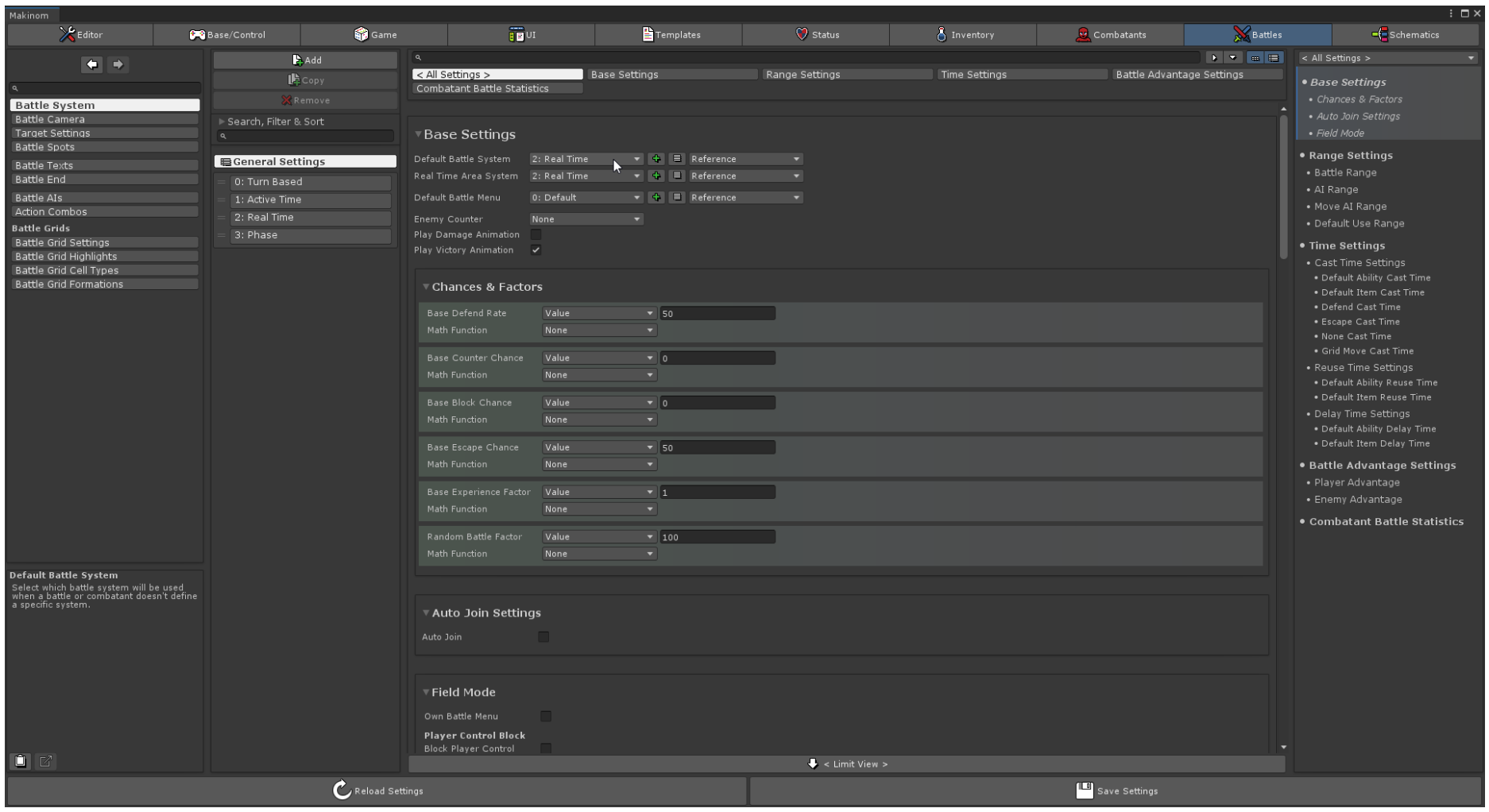
  Class Name = PausePlayerControllerAndCamera_ControlBehaviour
  Blocked By = Player



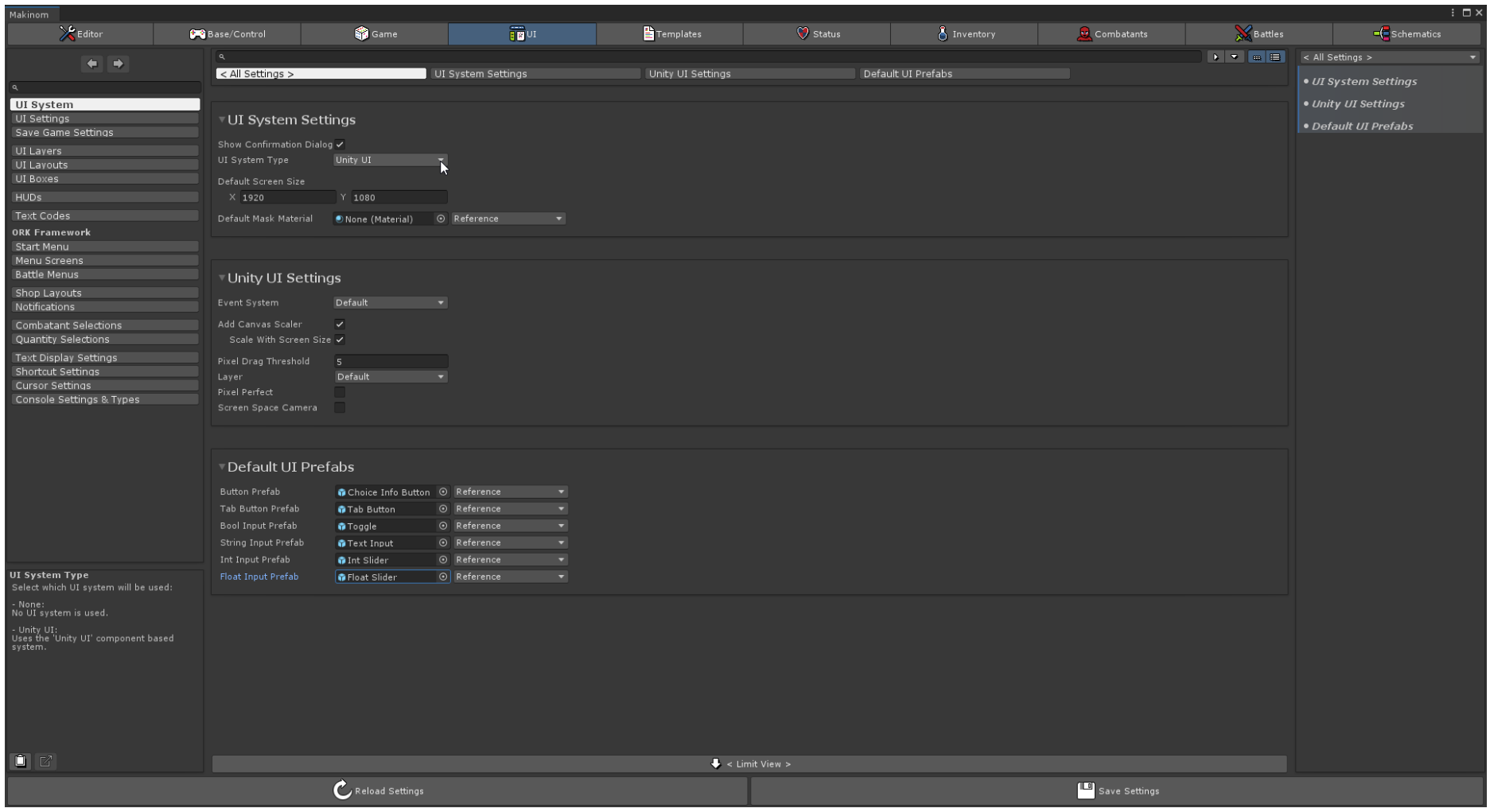
- On “Player Controls” and “Camera Controls” settings, set their respective Control Type to “None”.
- Don’t forget to click “Save Settings” frequently, as you make changes to the Makinom/ORK data.

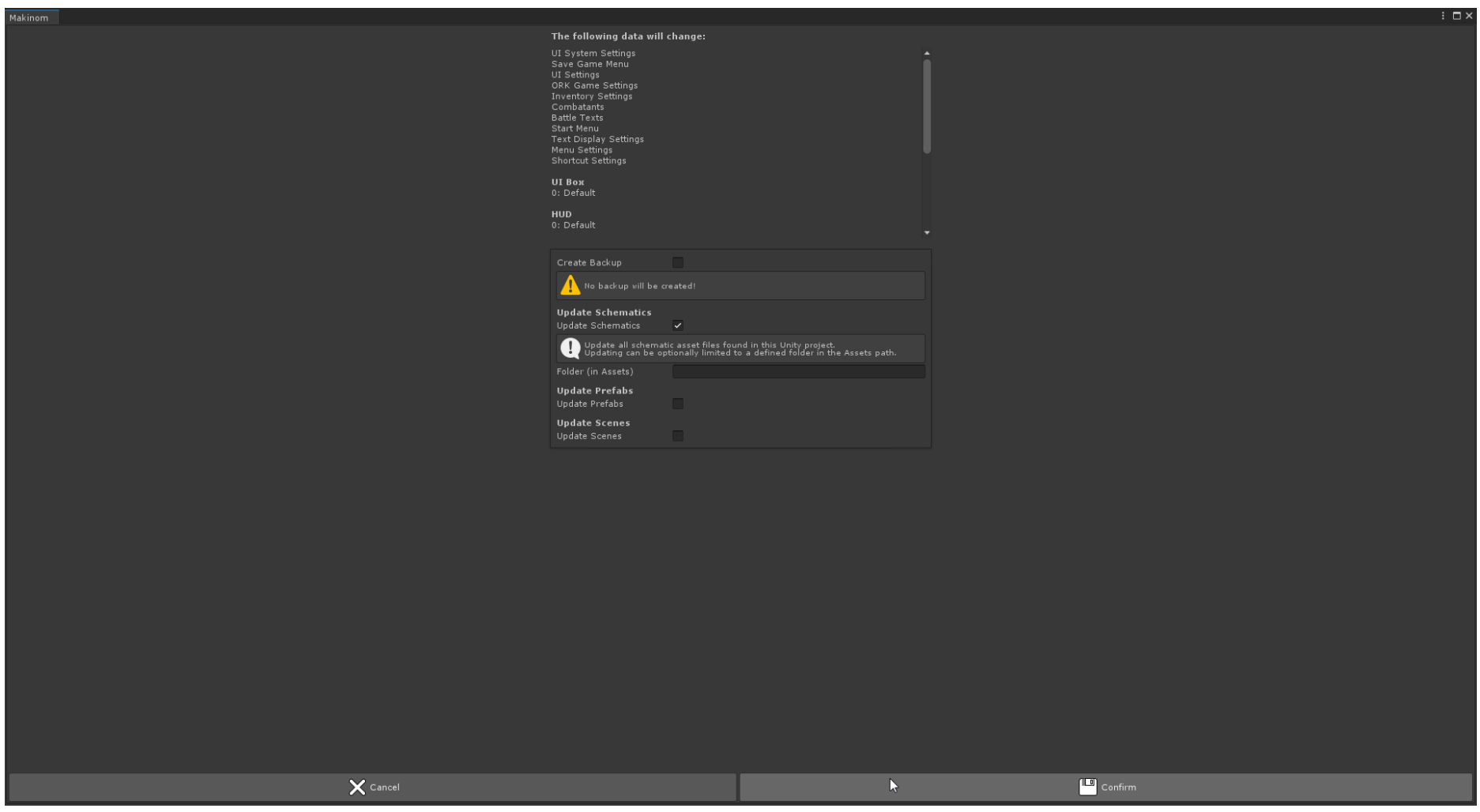


- IMPORTANT: In [Battles] tab, [Battle System] section, General Settings, make sure “Default Battle System” is set to your game’s intended battle system (Turn Based, Real Time, etc.), or else certain ORK features may not work correctly (Console Messages, Abilities, Rewards, etc.).
  For a simple GKC - ORK test, “Real Time” is recommended.

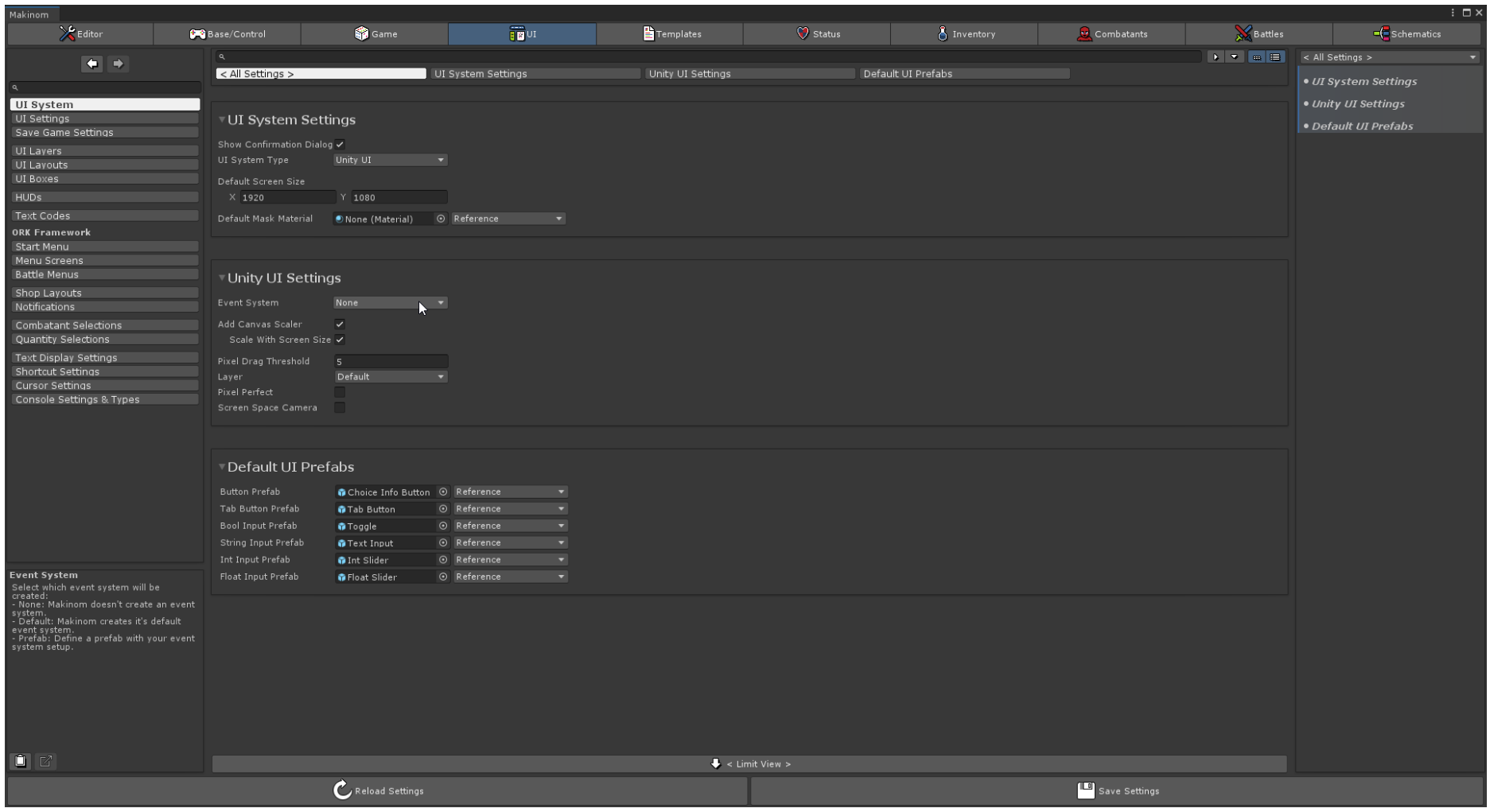


- IMPORTANT: In [UI] tab, [UI System] section, ensure UI System Type is set to “Unity UI”.
  Don’t forget to save your settings after changing the UI System Type.

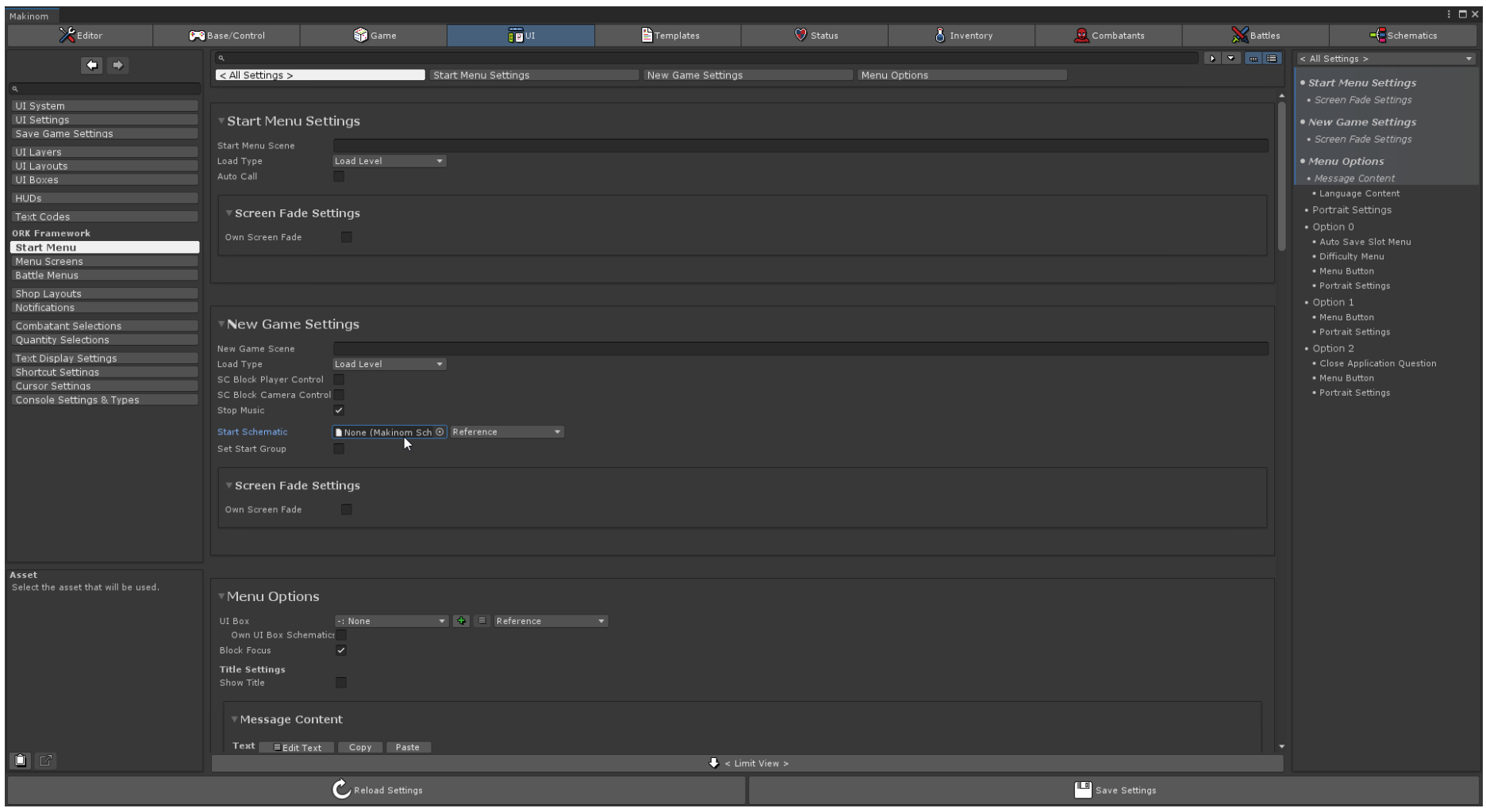




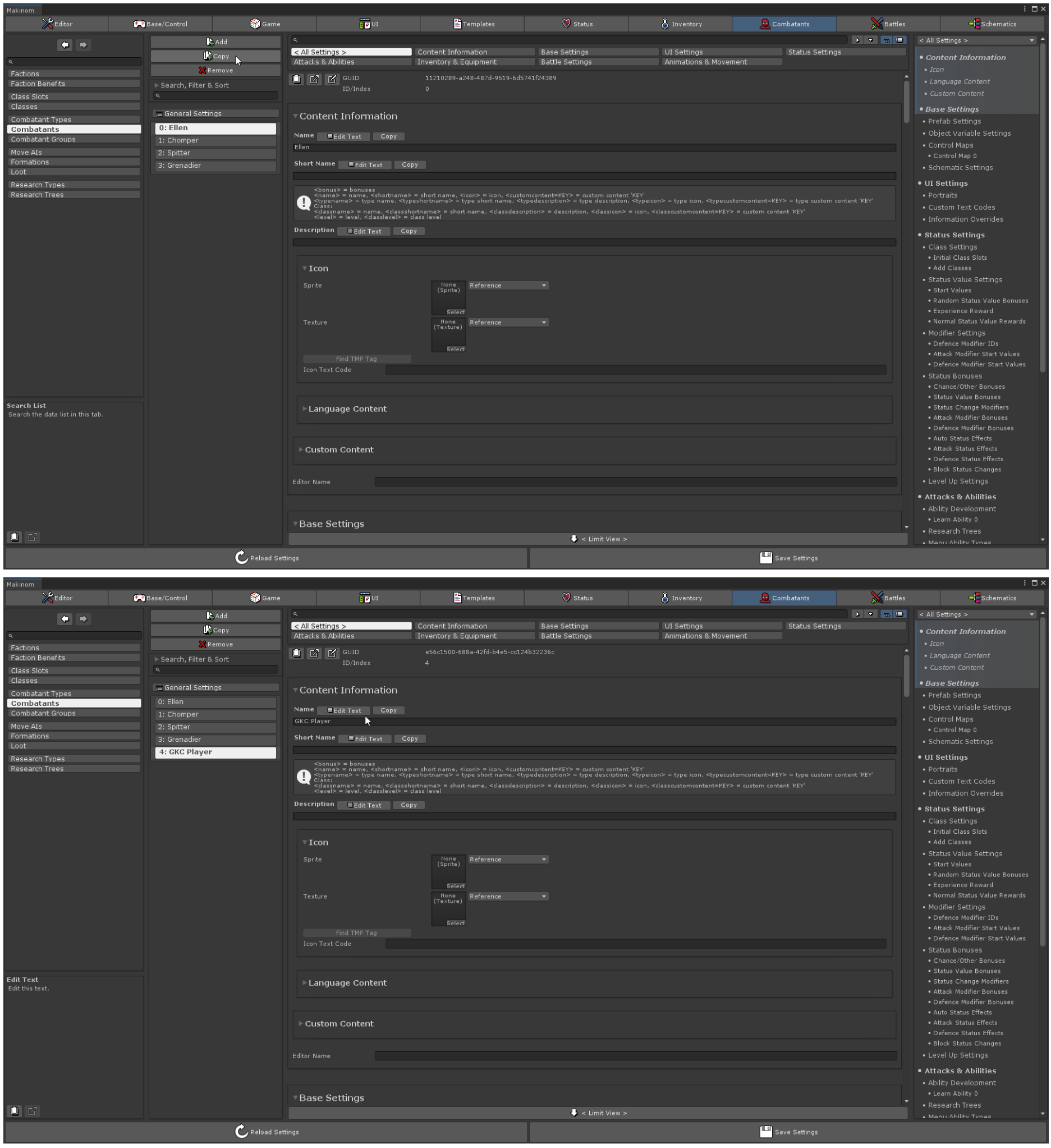
- IMPORTANT: Set the “Event System” field to None, to avoid creating a duplicate Event System gameobject when spawning our character (later on this guide).



- IMPORTANT: Open the [ORK Framework] -> [Start Menu] section.
  Go to New Game Settings, set the “Start Schematic” field to None (in case there’s any schematic previously assigned to the field).

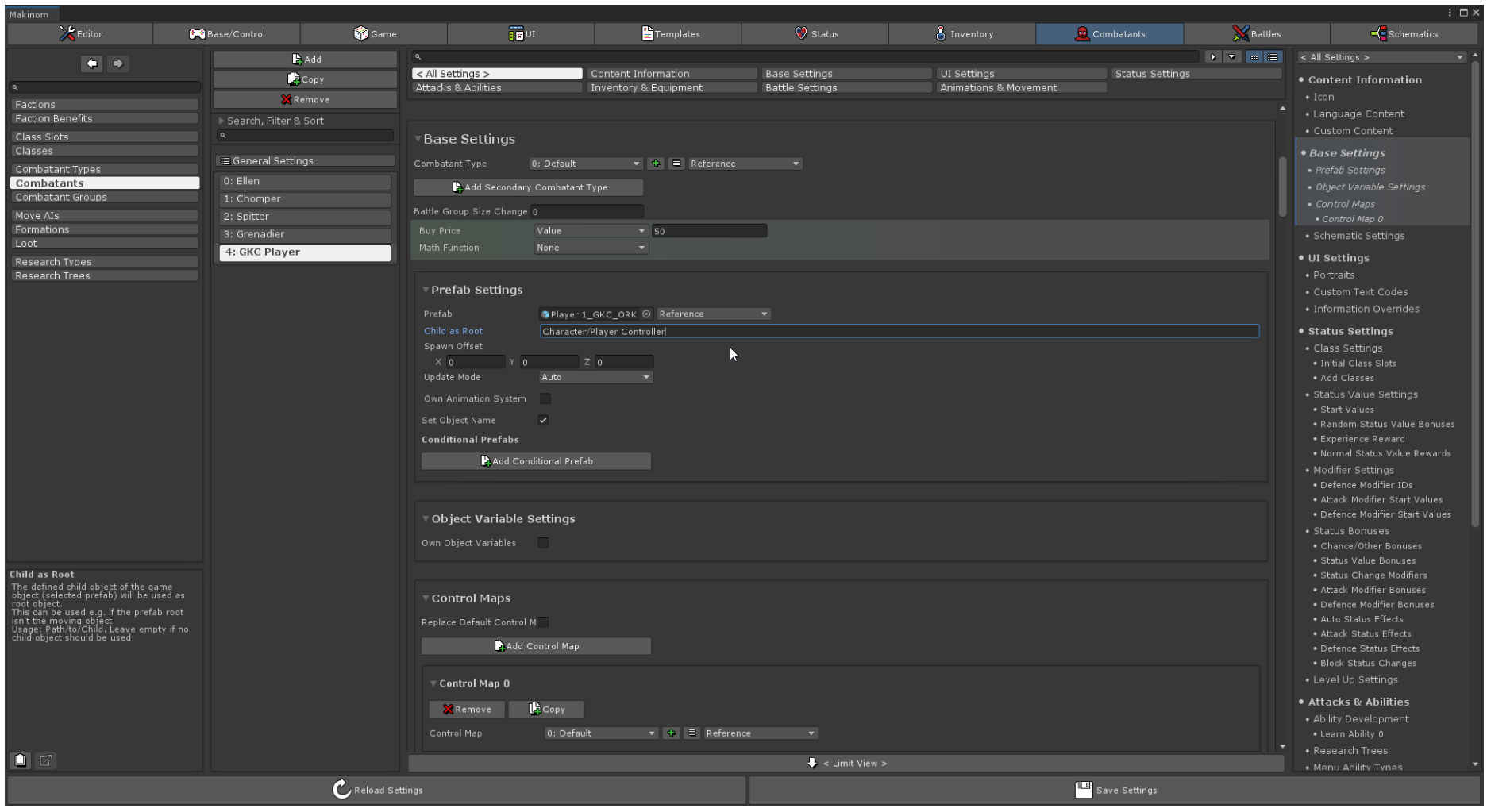


- In [Combatants] tab, [Combatants] section, add a new combatant to test the GKC integration. Let's call this combatant "GKC Player". You can either copy the existing Ellen combatant, or add a new combatant to the list. Or just edit the Ellen combatant directly. This guide assumes we're copying the Ellen combatant to a new entry.



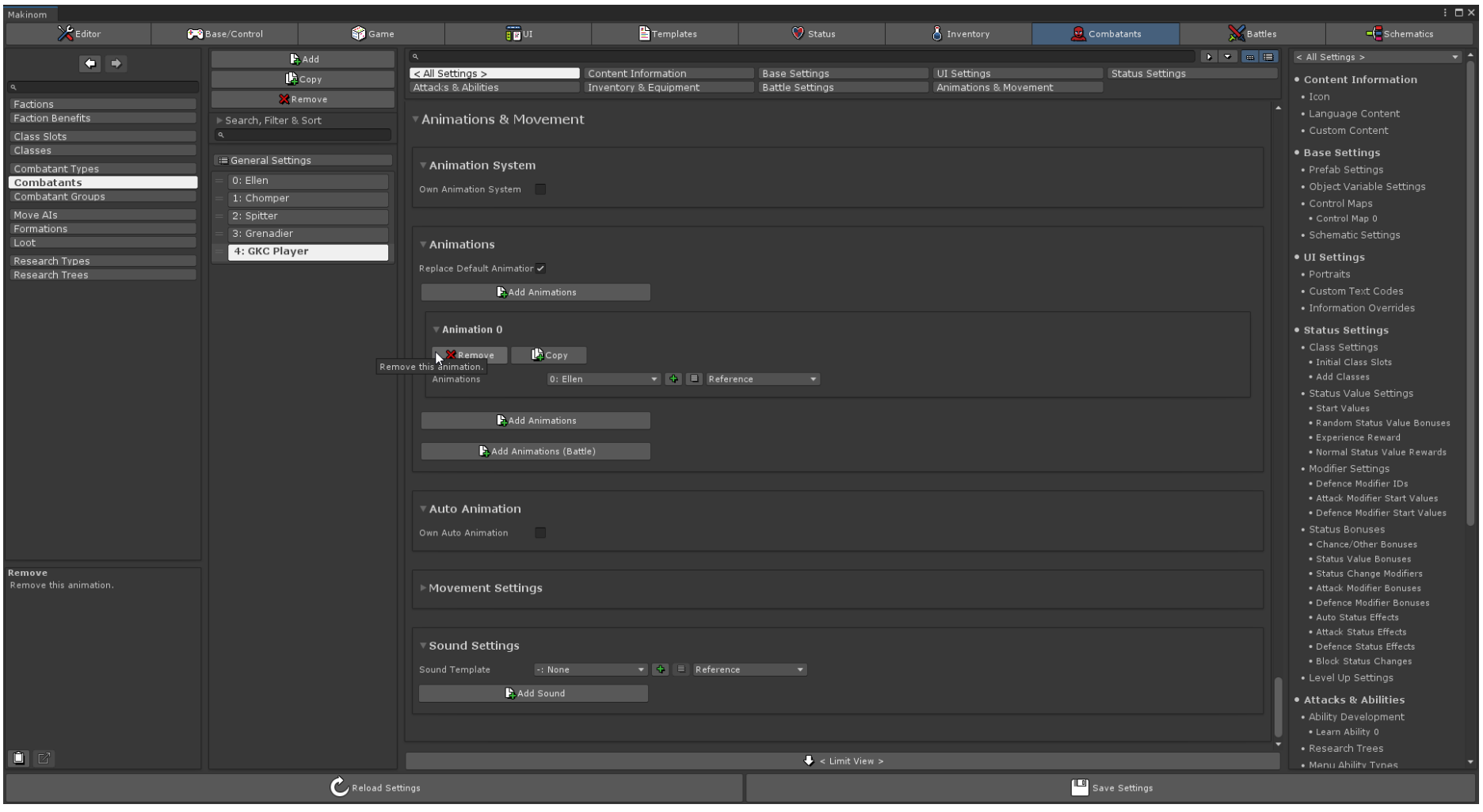
- Scroll down to Base Settings -> Prefab Settings. Change the Prefab field to the [Player 1_GKC_ORK] prefab we've created earlier. Set "Child as Root" field to "Character/Player Controller" (without quotes).

  Note: Player Controller is the actual moving object of the Player prefab. That's why this object should be used as the "root object" for Makinom/ORK, so certain operations will see/point to the moving object (e.g.: "Rotate To" schematic node).



- If you don't want ORK to rename your "Player Controller" gameobject to the combatant's name (in this case, GKC Player), then disable the "Set Object Name" setting.

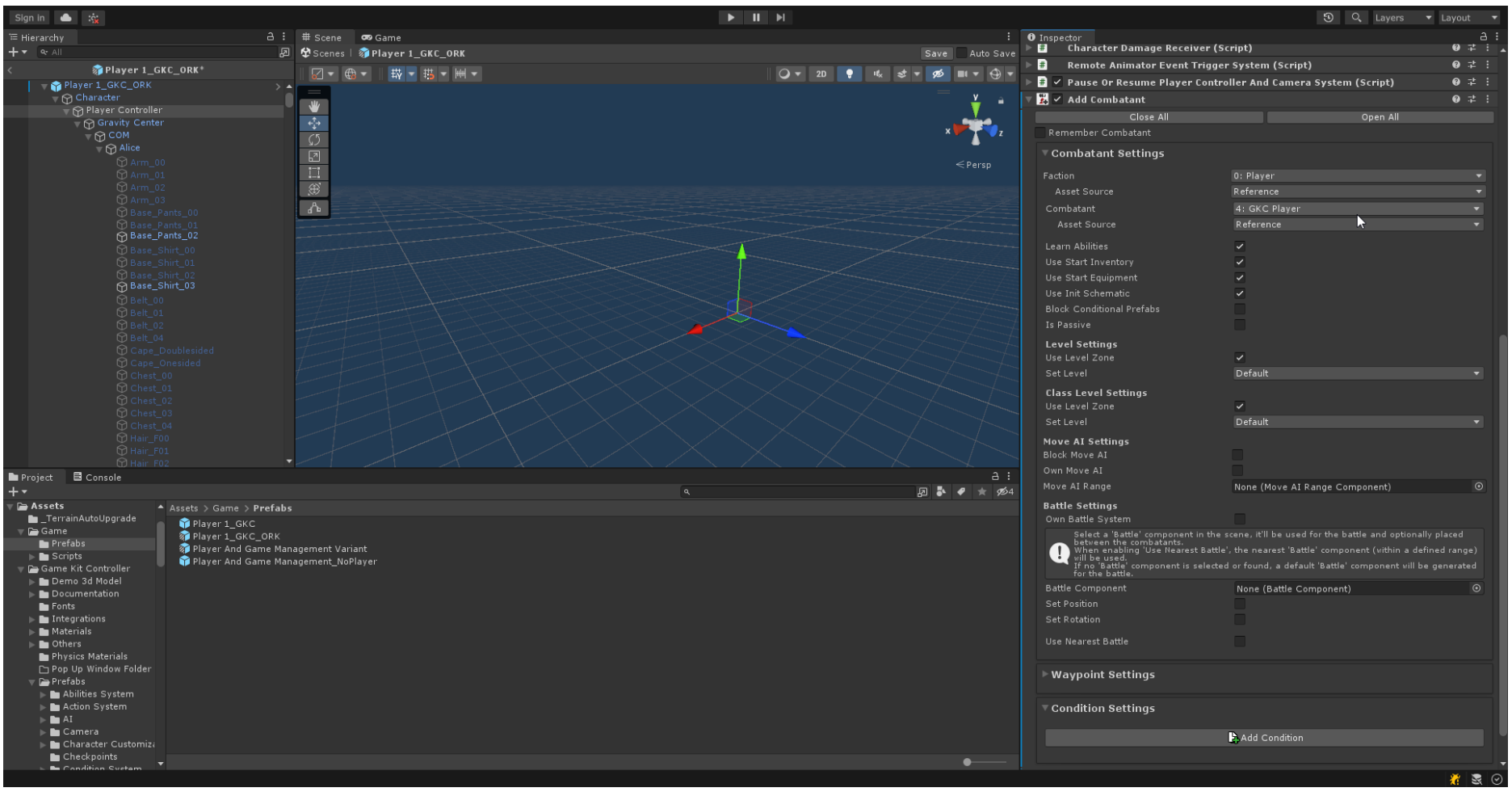
- Scroll down to Animations & Movement -> Animations.
  Remove the “Animation 0” entry, as we’ll use the GKC character’s Animator Controller for this test.



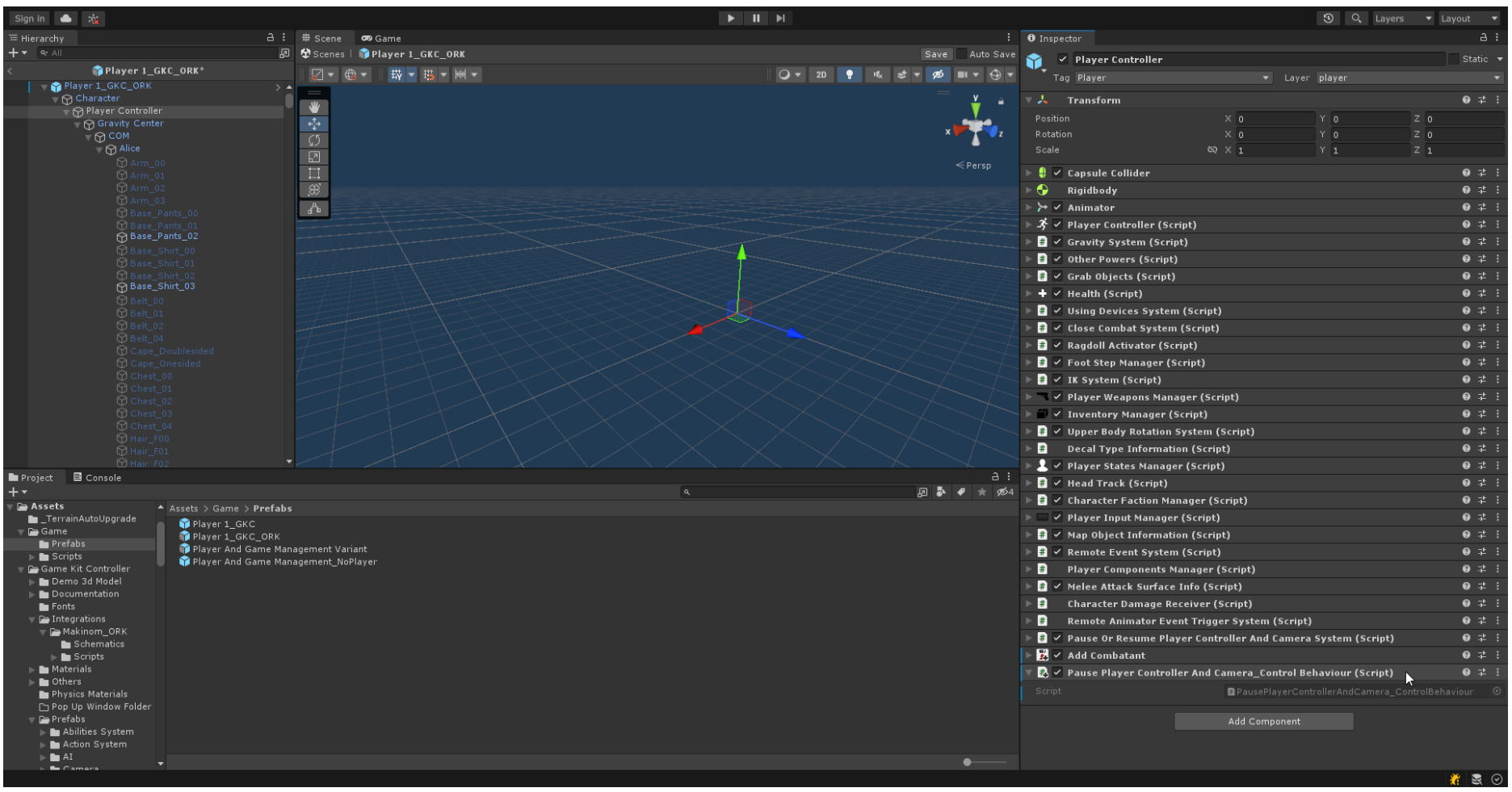
- IMPORTANT: After you make your desired changes to Makinom/ORK’s data, don’t forget to click the “Save Settings” button (at the bottom) before closing the Makinom window!
- Close the Makinom window.

Makinom/ORK Components Setup

- Open [Player 1_GKC_ORK] in Prefab Mode.
- In the “Player Controller” child gameobject, add an “Add Combatant” component.
  Set its Faction to “Player”, and its Combatant to “GKC Player”.

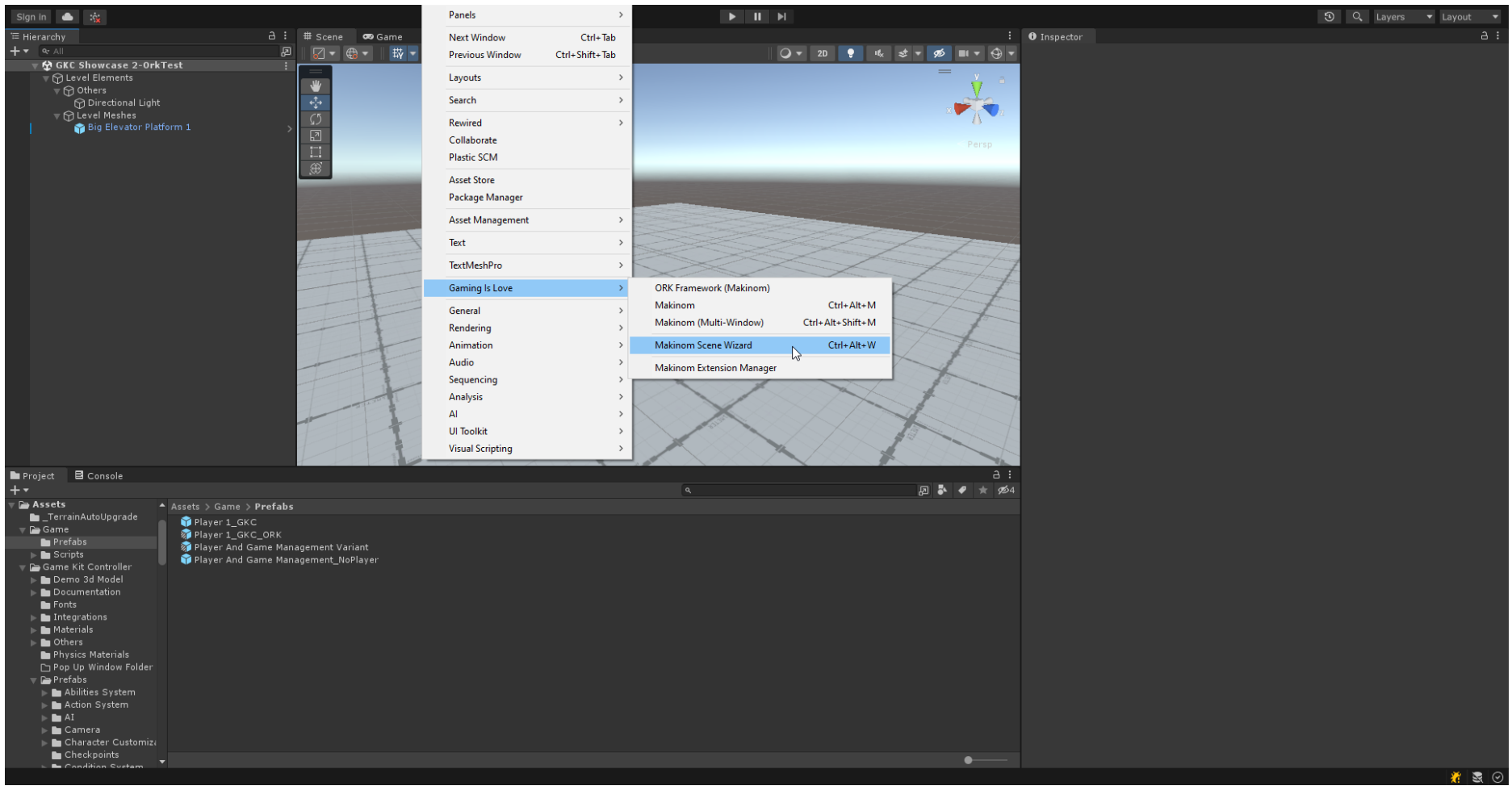


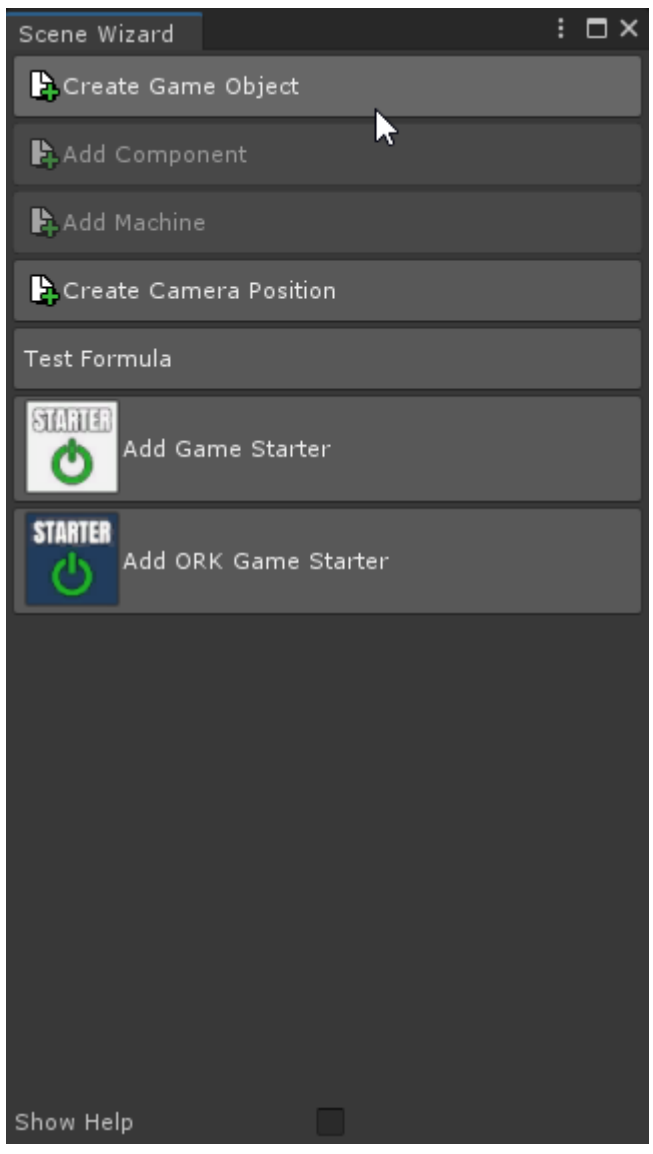
- In the same gameobject, add the component “PausePlayerControllerAndCamera_ControlBehaviour”.
  The Control Behaviour we’ve configured earlier will automatically disable/enable this script whenever ORK blocks/unblocks the player’s control (e.g.: While displaying an ORK NPC dialogue).
  This script, in turn, will call the PauseOrResumePlayerControllerAndCameraSystem component on the same gameobject, to hide/unhide the GKC UI.



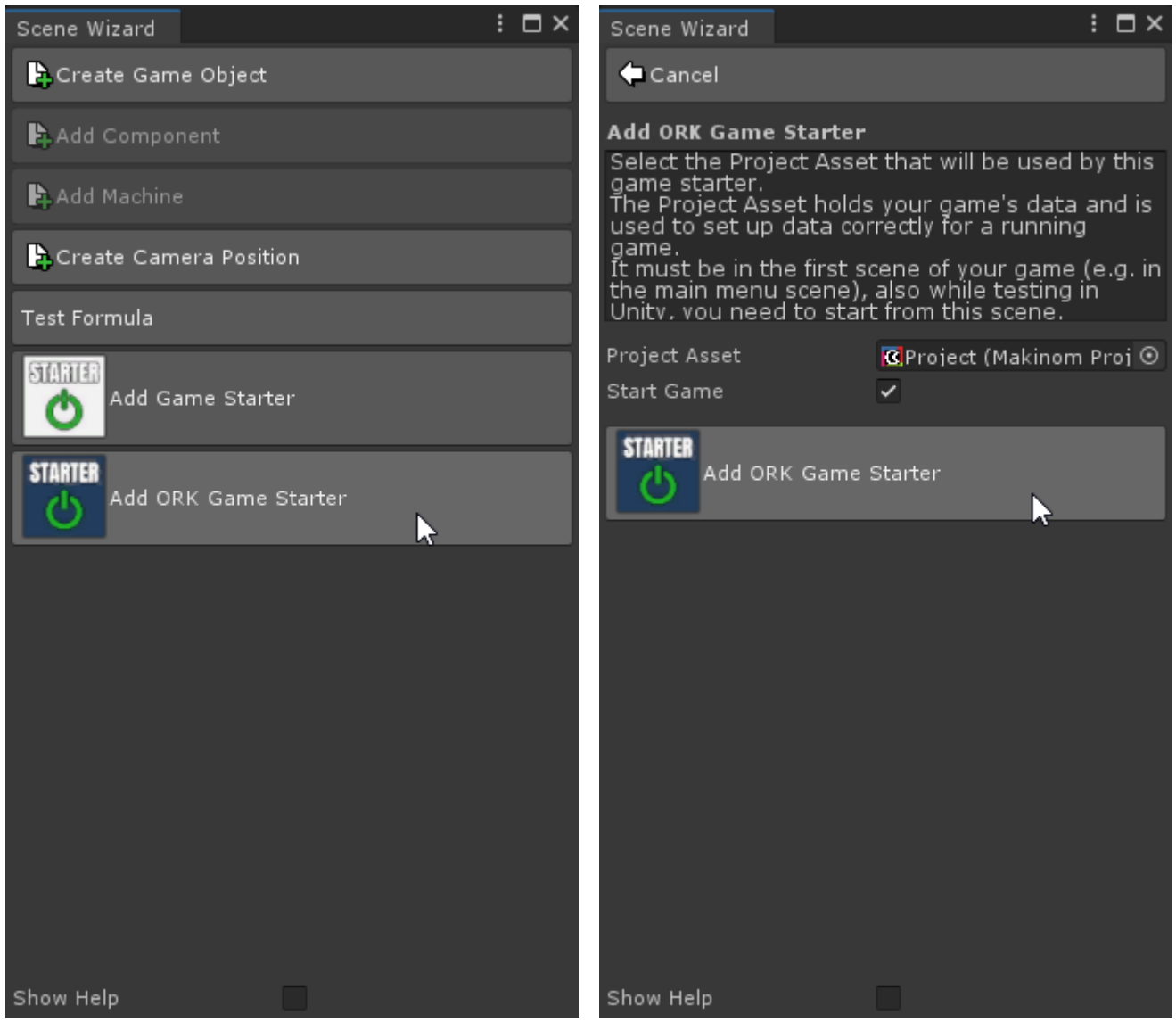
- Save the prefab. Exit Prefab Mode.

- Open Makinom Scene Wizard: Window -> Gaming Is Love -> Makinom Scene Wizard
  Or press Ctrl + Alt + W, or Command + Alt + W

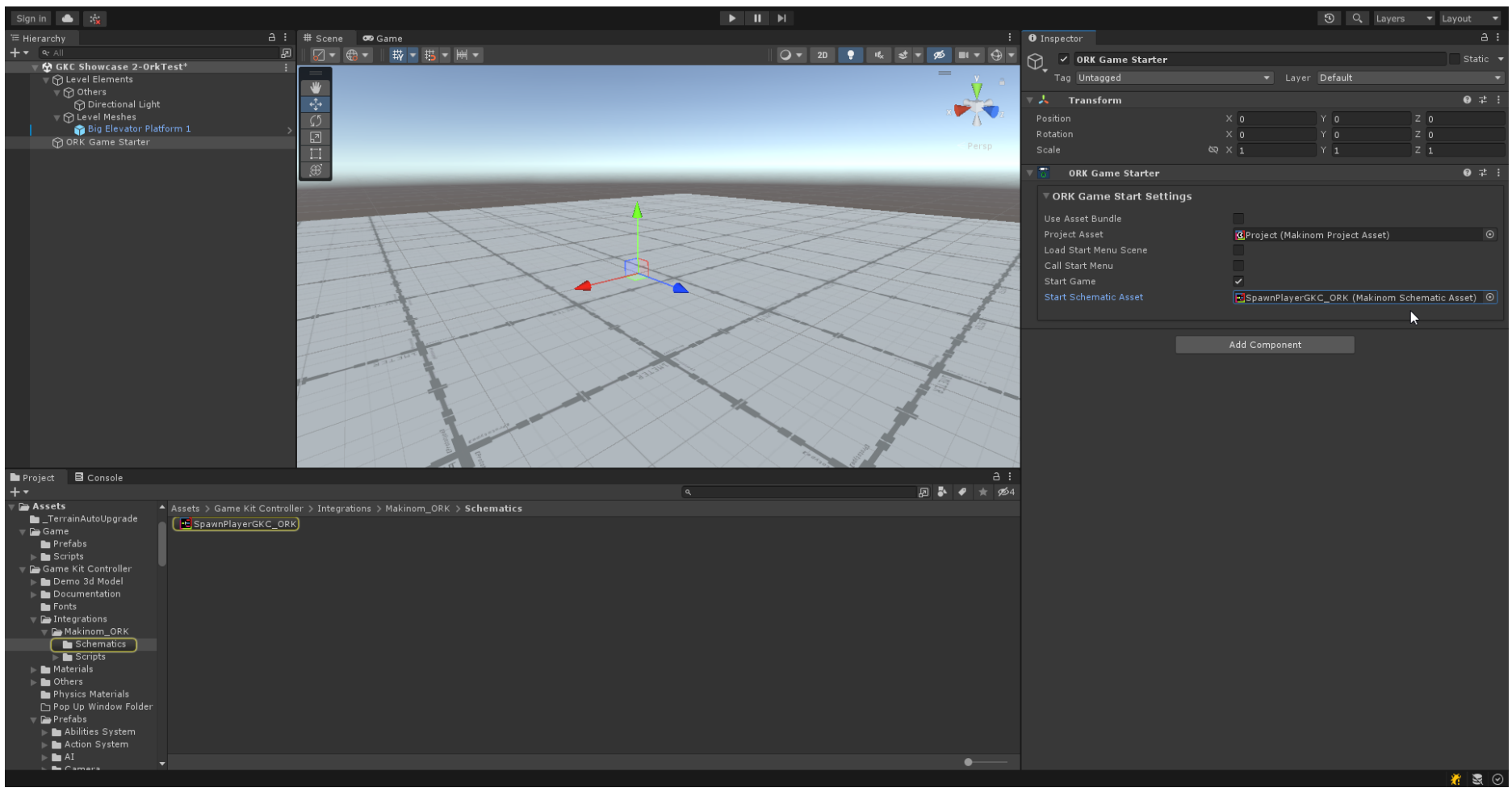




- Click “Add ORK Game Starter”, then click “Add ORK Game Starter” again on the next screen.

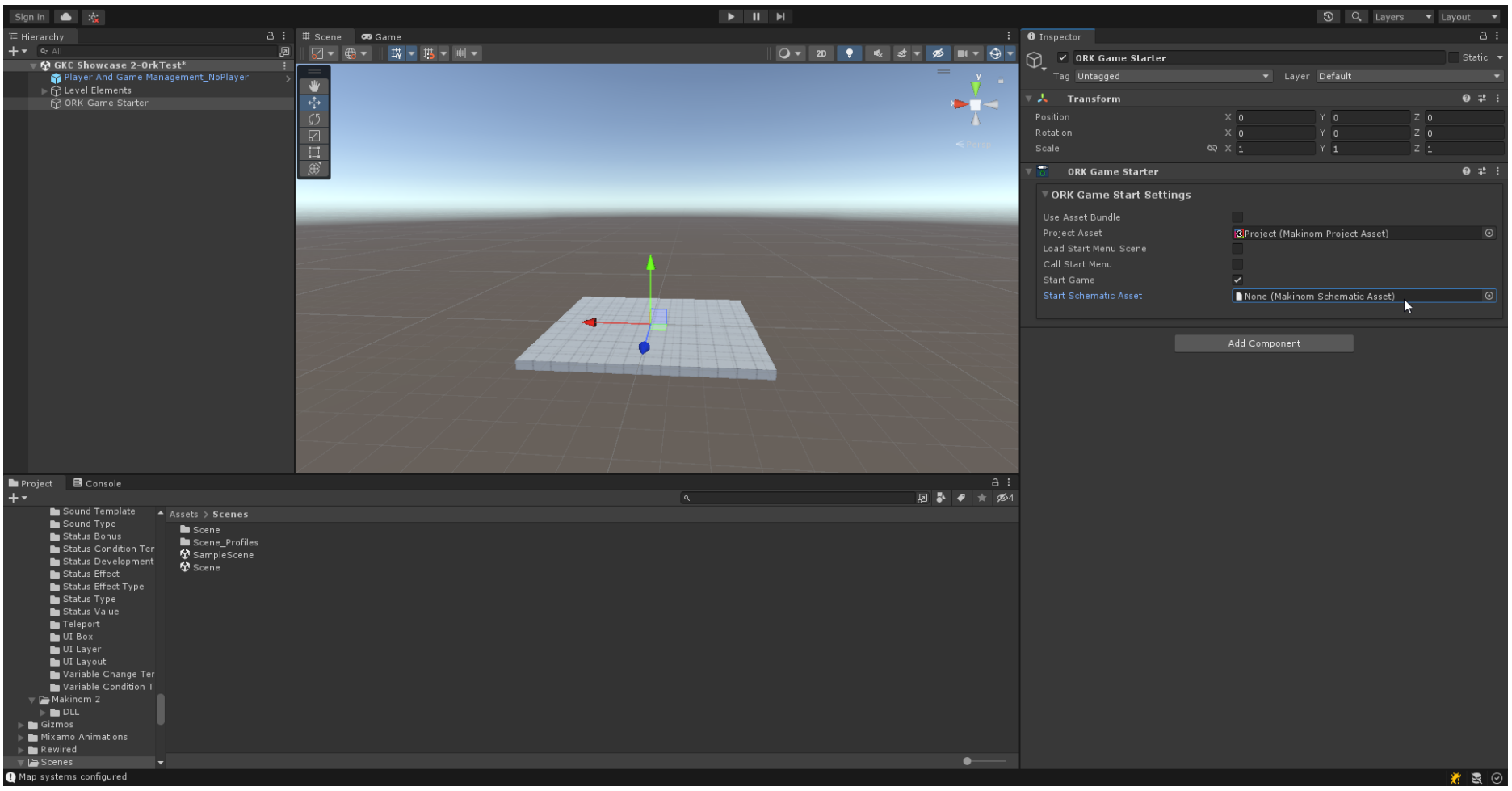


- Select the “ORK Game Starter” gameobject.
  On “Start Schematic Asset” field, assign the “SpawnPlayerGKC_ORK” schematic.

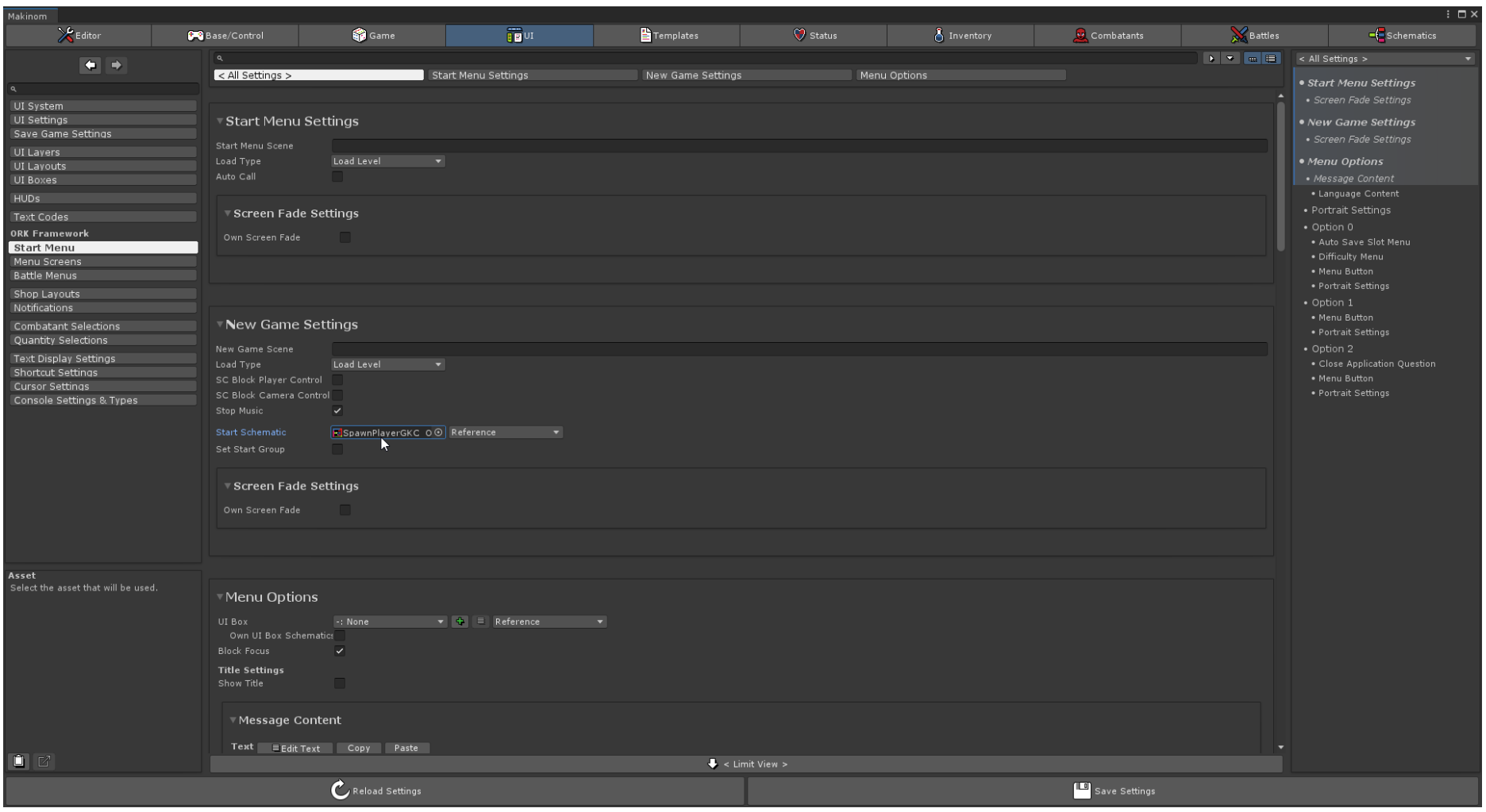


- The next steps are an alternate way of setting up the Start Schematic *globally*, if you're going to use **the same Start Schematic in *all* of your ORK game scenes** (the scenes that have an ORK Game Starter gameobject).

- Select the "ORK Game Starter" gameobject from the scene. Ensure its "Start Schematic Asset" field is set to None.

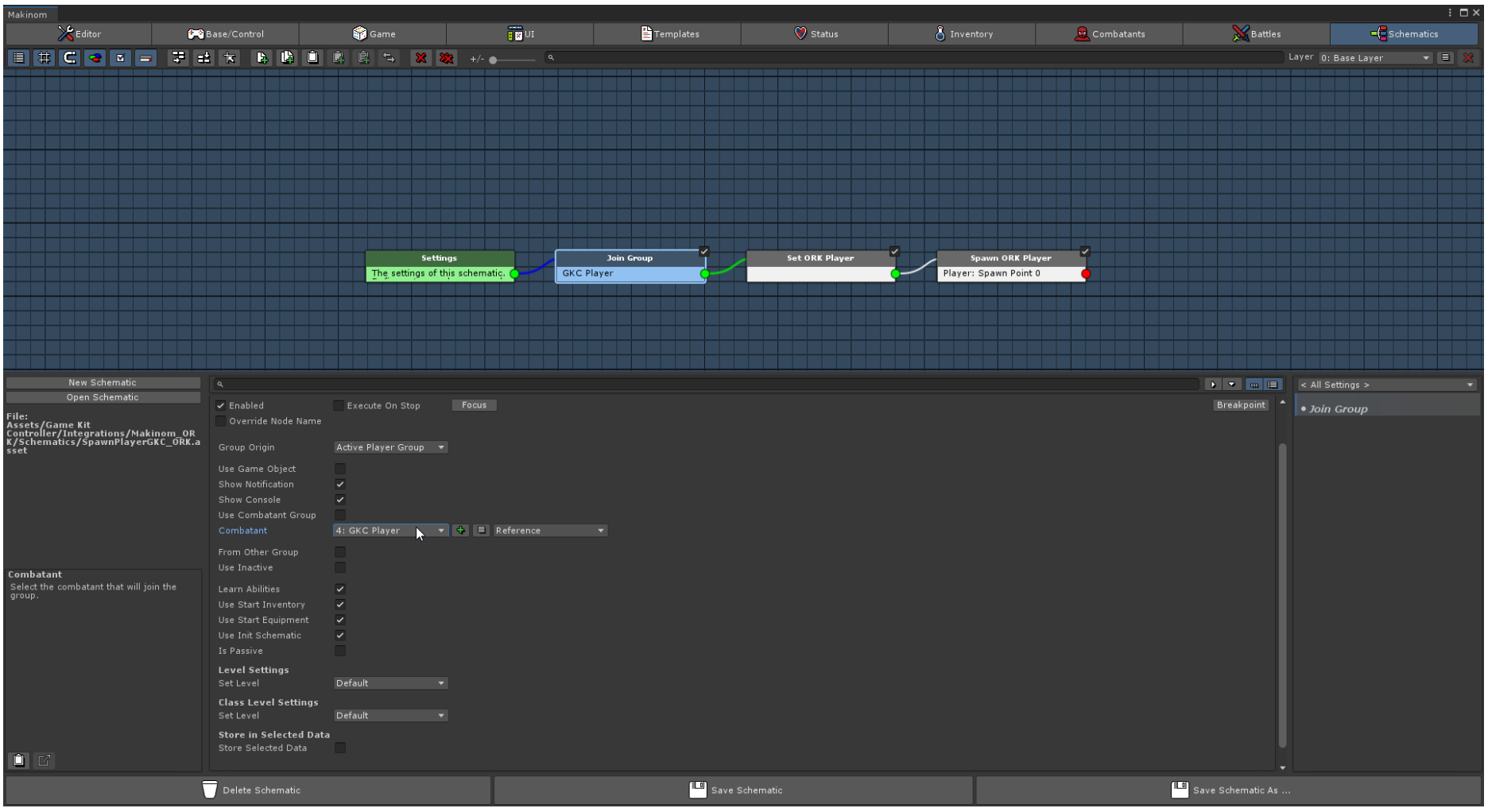


- Open Makinom window.
  In the [UI] tab, open the [ORK Framework] -> [Start Menu] section. Go to New Game Settings.
  On "Start Schematic" field, assign the "SpawnPlayerGKC_ORK" schematic.

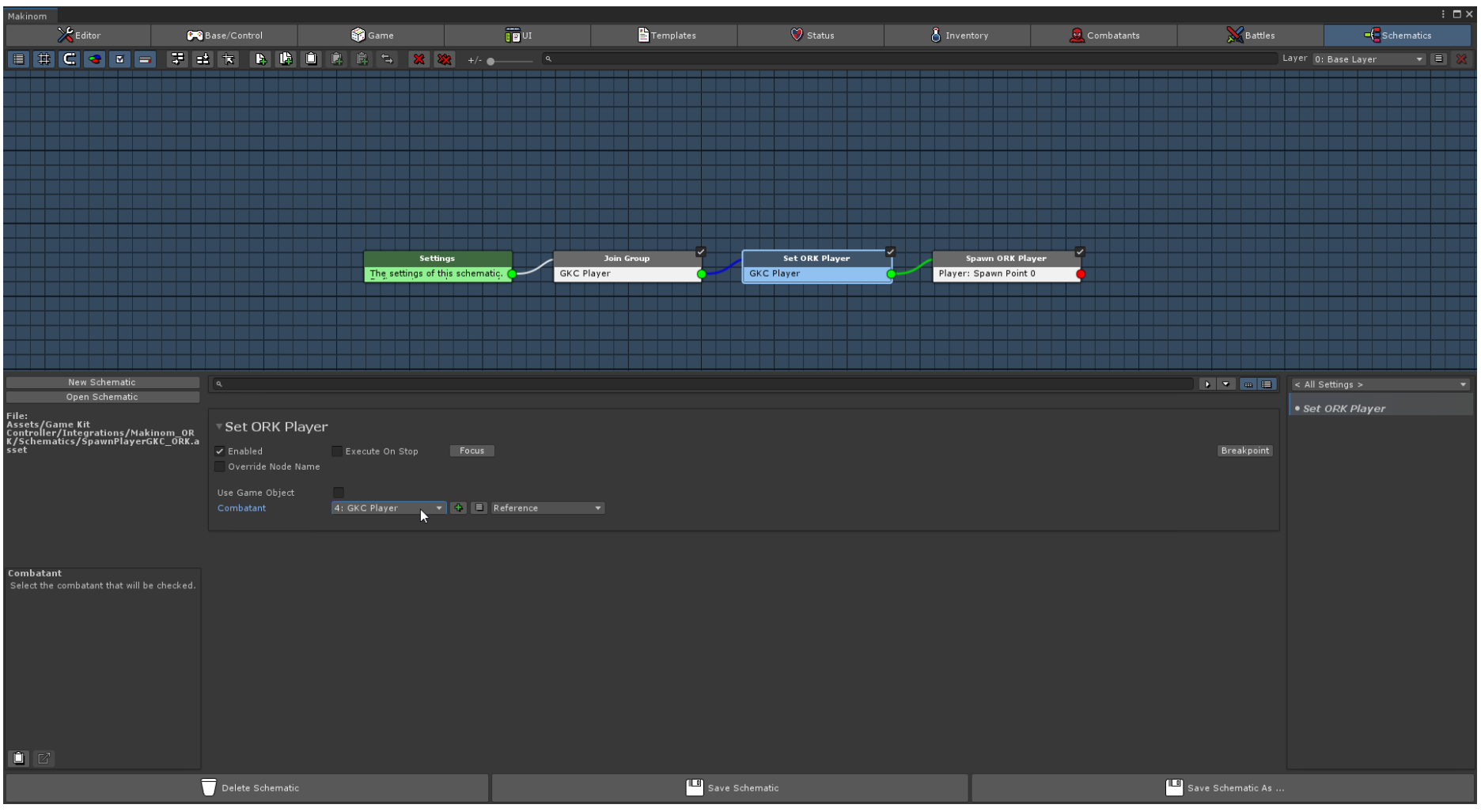


- This concludes the alternate way of setting up the Start Schematic.

- Open the “SpawnPlayerGKC_ORK” schematic. You can double-click on it.
- Select “Join Group” node. Set its Combatant field to GKC Player.

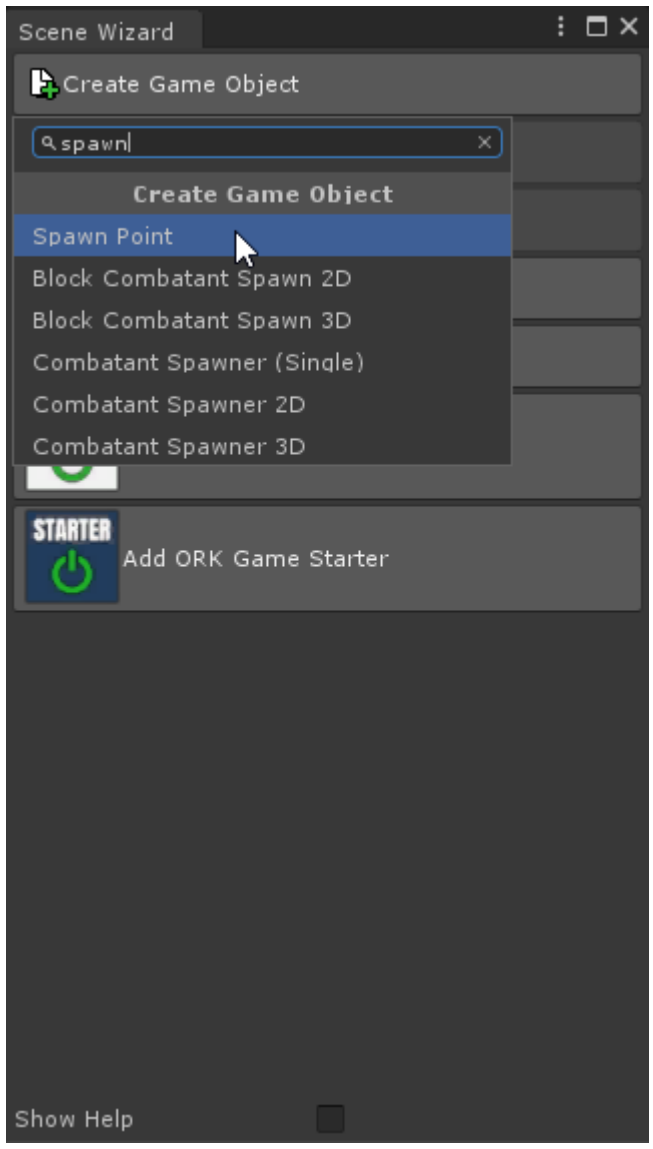


- Select “Set ORK Player” node. Set its Combatant field to GKC Player.

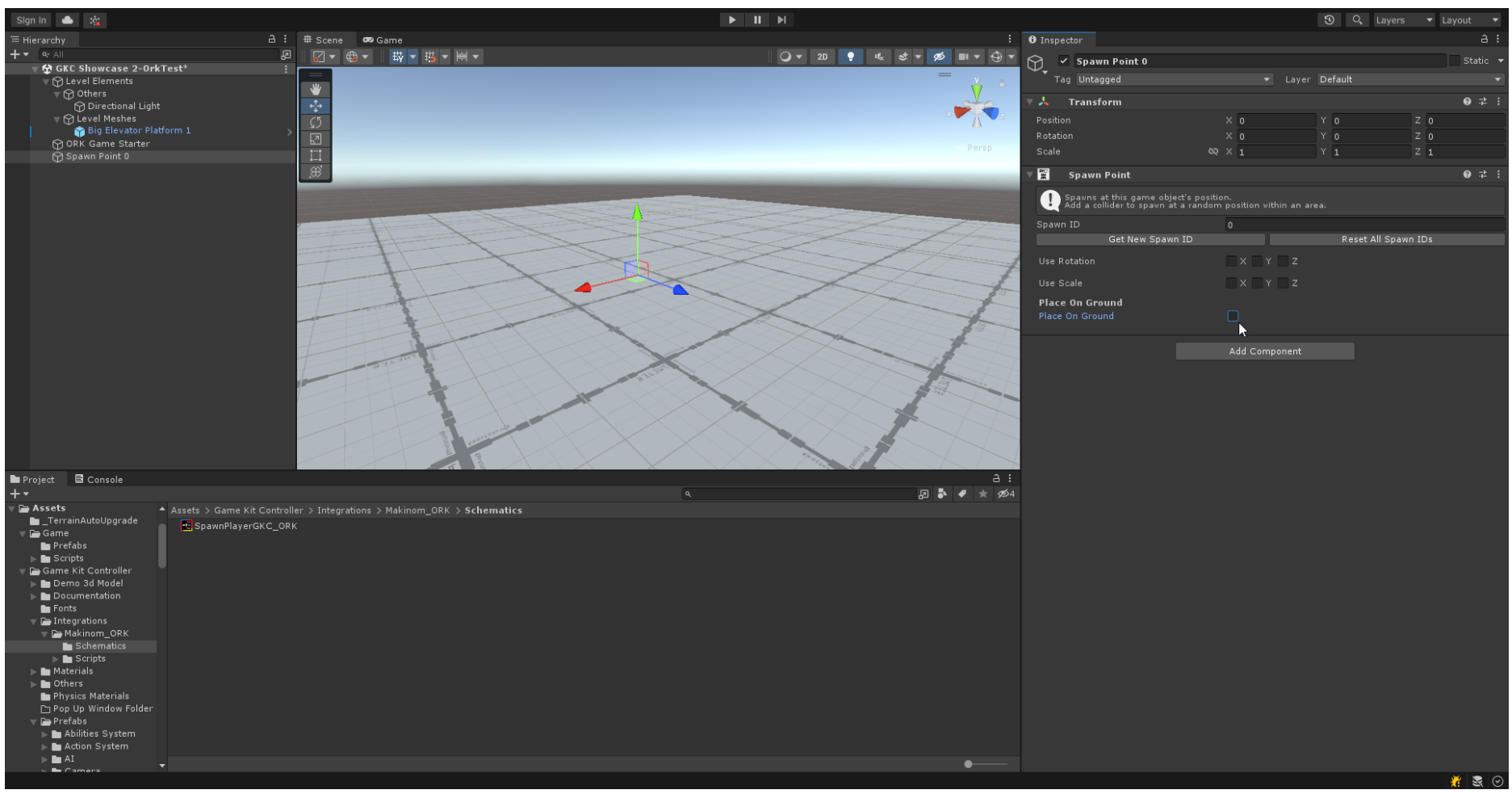


- IMPORTANT: After you make your desired changes to a schematic, don’t forget to click the “Save Schematic” button (at the bottom) before closing the Makinom window!
- Close the Makinom window.

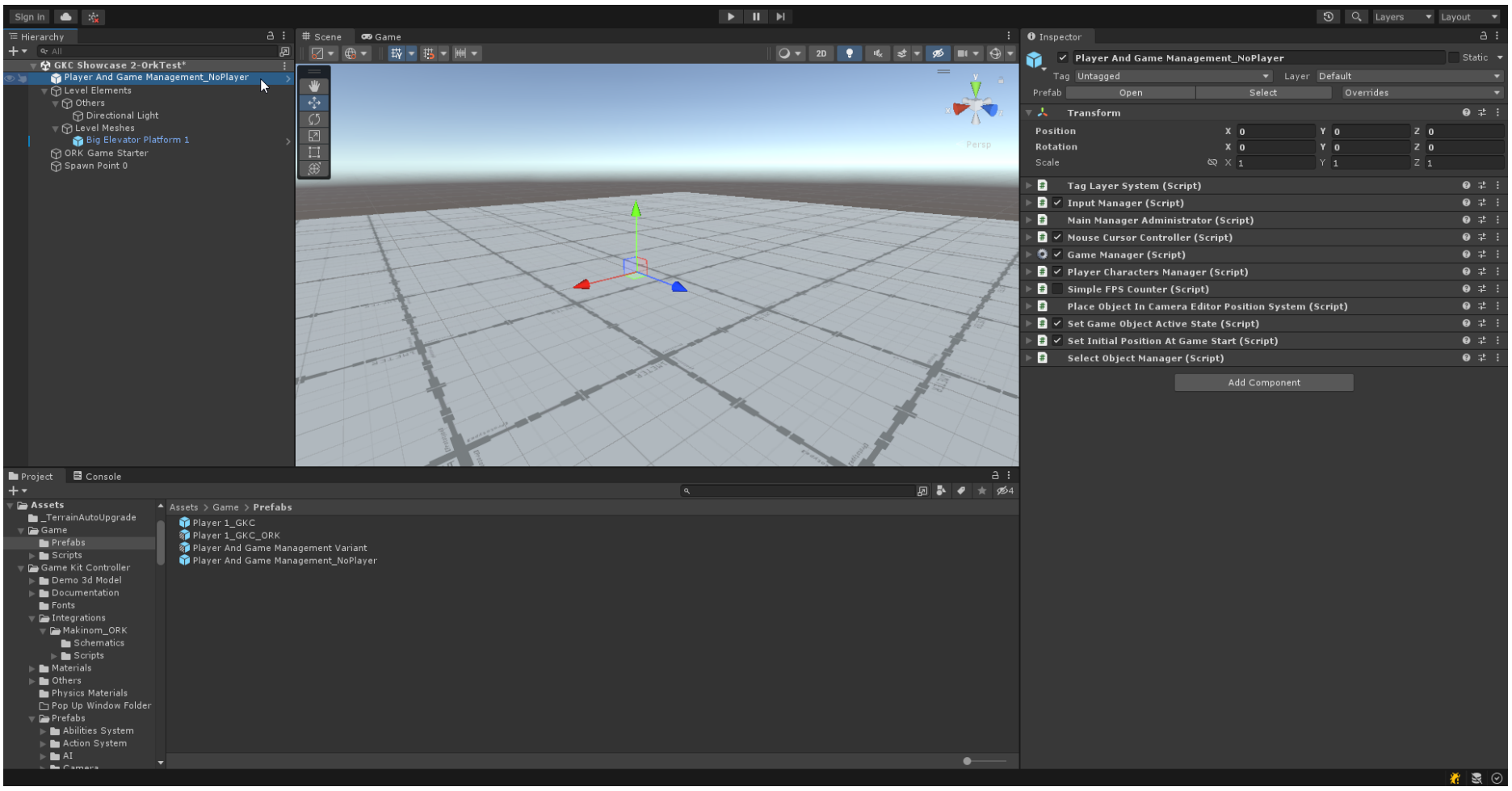
- Open Makinom Scene Wizard.
- Click “Create Game Object”, then create a “Spawn Point”.



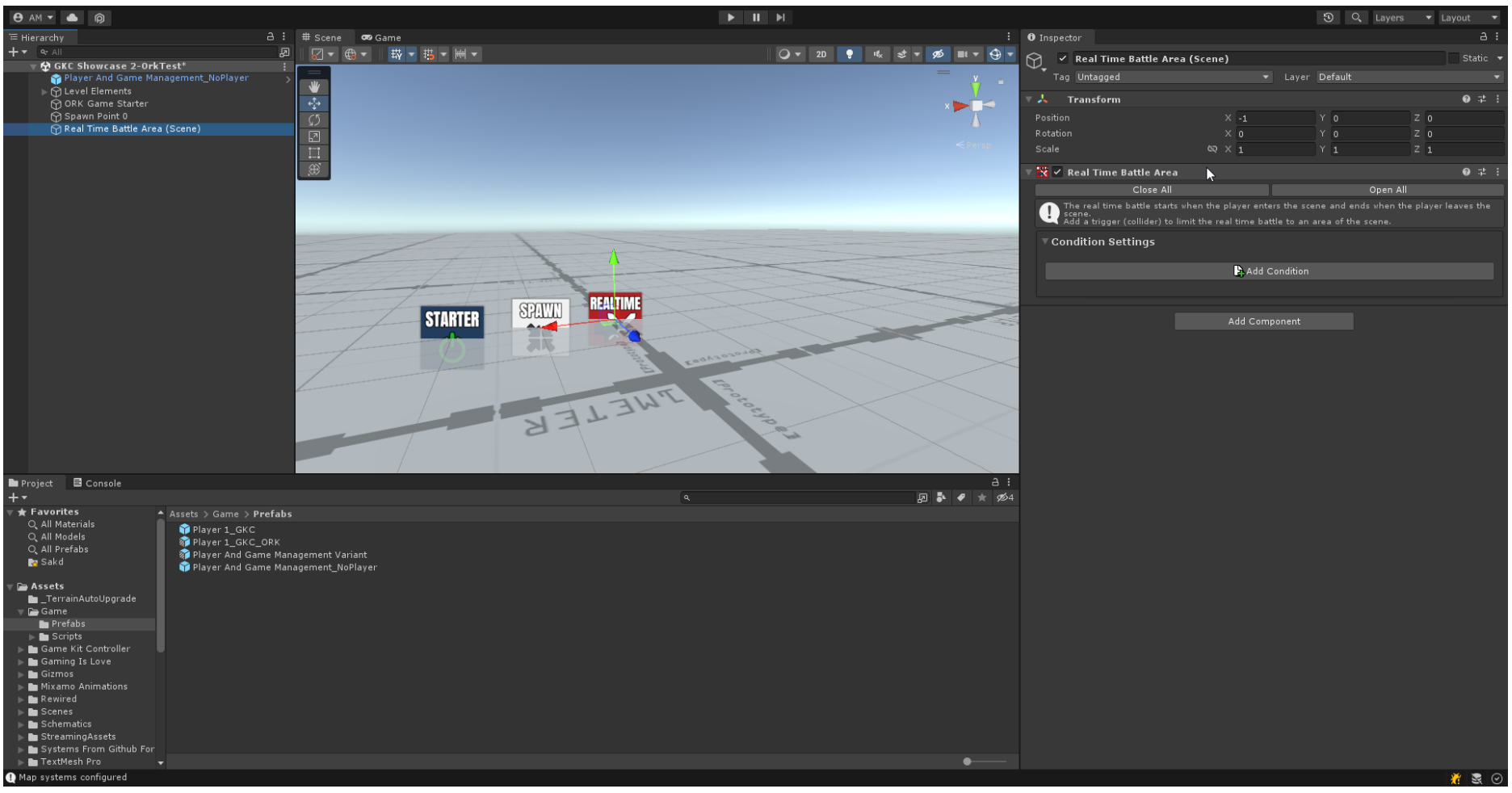
- Select the “Spawn Point 0” gameobject.
  Disable “Place On Ground”, as we want GKC to take care of the character’s ground placement.



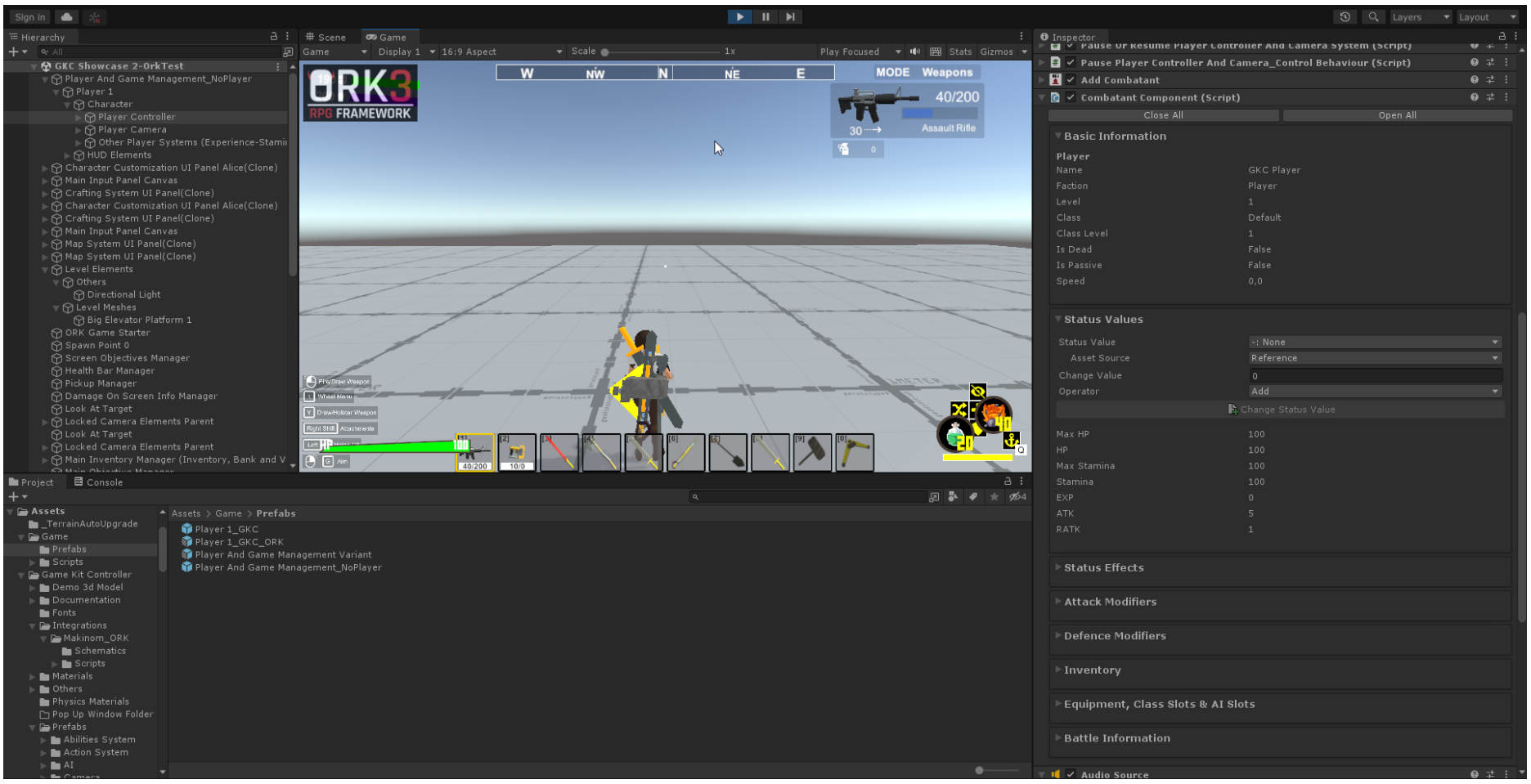
- IMPORTANT: Add the “Player And Game Management_NoPlayer” prefab to the scene.



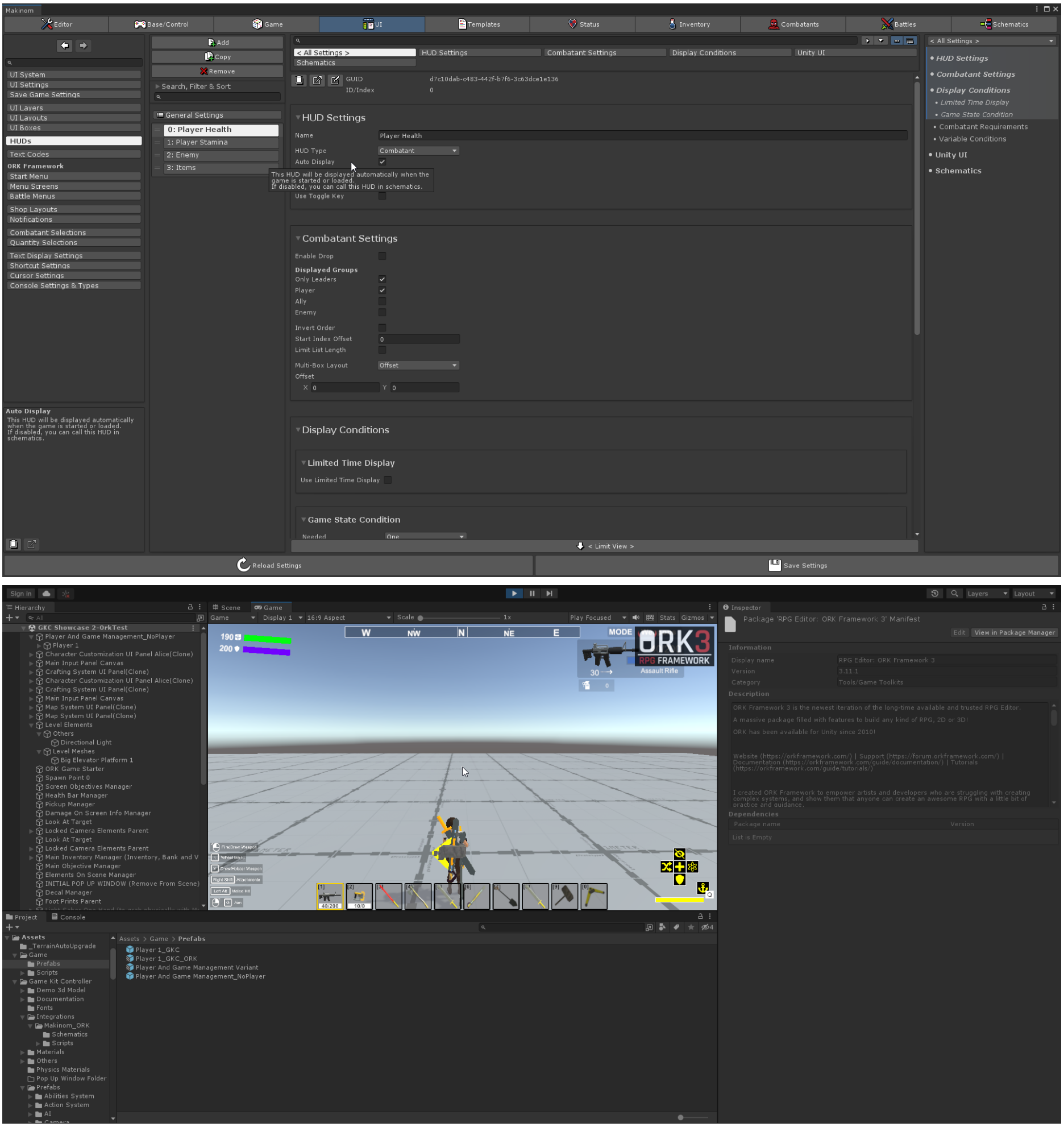
- IMPORTANT: If your game uses Real Time battle system.
  Open Makinom Scene Wizard, then create a “Real Time Battle Area (Scene)” gameobject.



- Save the scene, then press Play to run the game.
  The player character will be spawned in the scene (via the “SpawnPlayerGKC_ORK” schematic), on the position of the Spawn Point we’ve added to the scene (with Spawn ID “0”).
- On the player character’s “Player Controller” child gameobject, take a look at the data displayed in Combatant Component.



- To disable the “3D Action RPG” tutorial’s UI elements, open Makinom window, then go to [UI] tab, [HUDs] section.
  Select each HUD element from the list (Player Health, Player Stamina, etc.).
  On their HUD Settings, disable the “Auto Display” checkbox.



- To remove the “ORK 3 RPG Framework” watermark, you need to purchase the full version of ORK Framework 3, then import it into your project. The ORK game data (located in “Assets\Gaming Is Love_Data” folder) **won’t be overwritten/lost** when importing the full version of ORK.

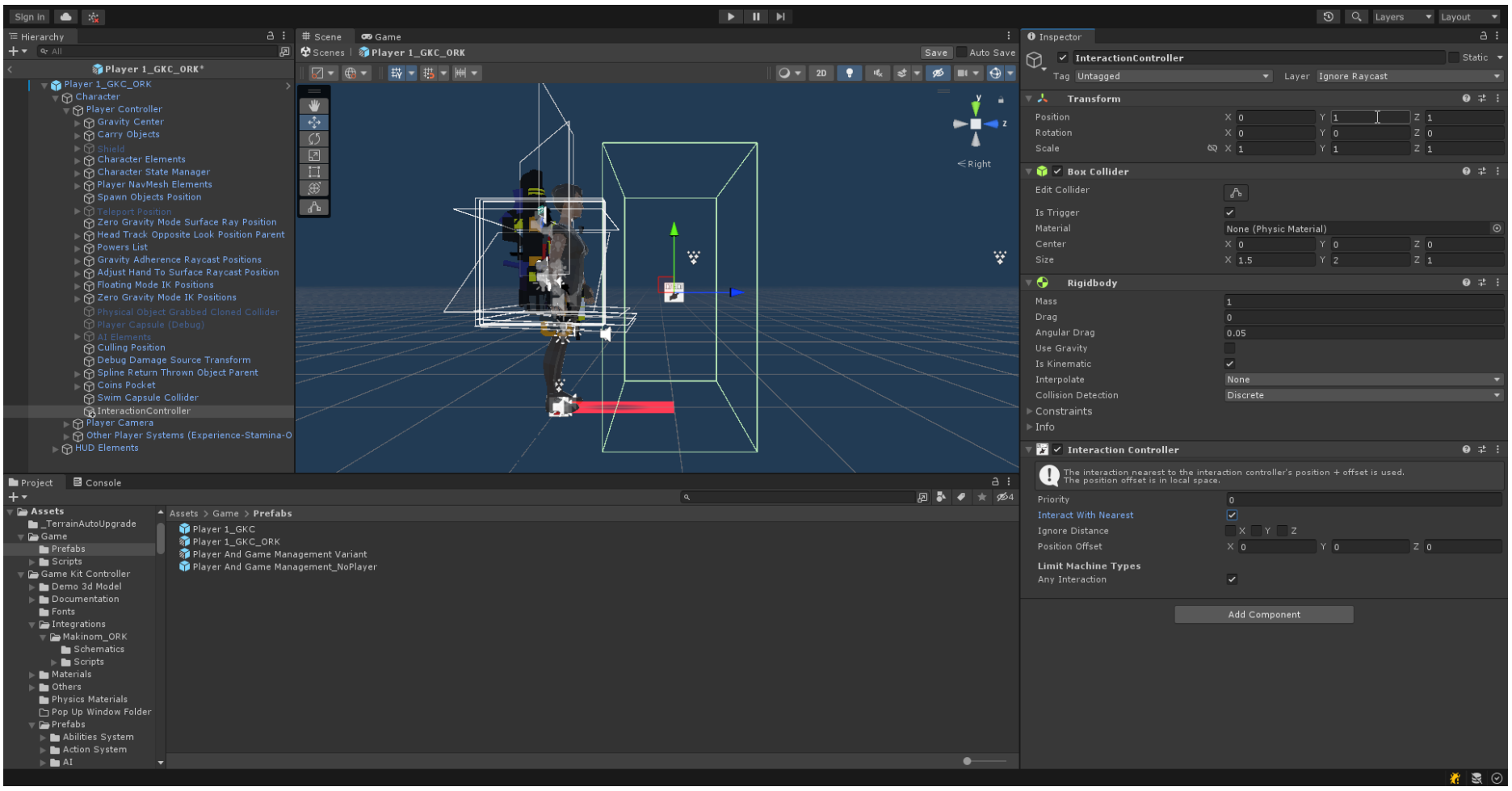
- In the next section of this guide, we’ll create an “NPC dialogue interaction” to further test our current setup.

Interaction Test - NPC Dialogue

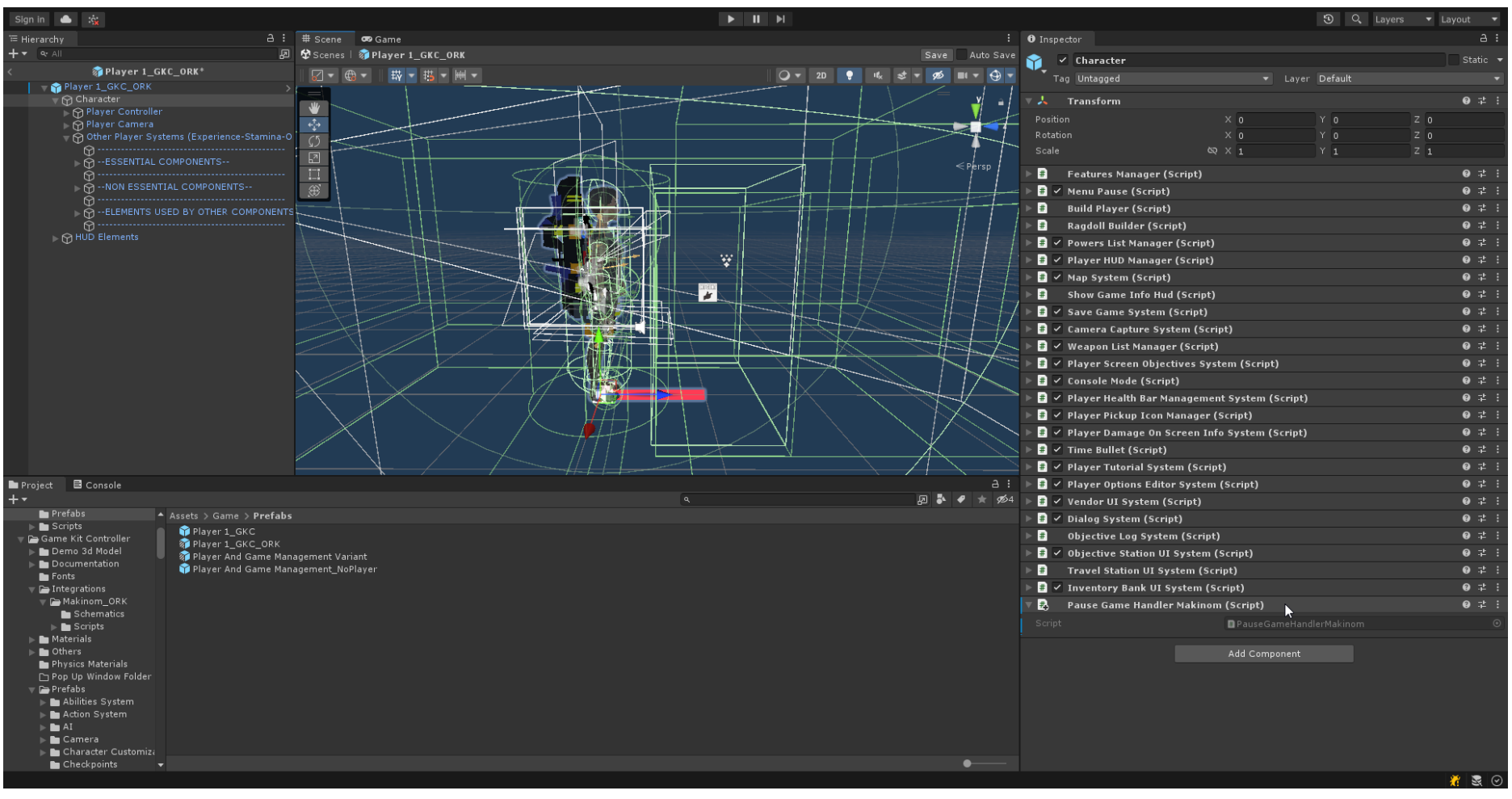
- Open the [Player 1_GKC_ORK] prefab.
- Right-click the “Player Controller” gameobject. Select “Create Empty” to create a new child gameobject.
- Name the new gameobject “InteractionController”.

- Open the Makinom Scene Wizard.
- Click the “Add Component” button. Select “Interaction Controller 3D”.
  This will add a Box Collider, Rigidbody, and Interaction Controller components to the InteractionController gameobject.
  The gameobject’s Layer will also be set to “Ignore Raycast”.

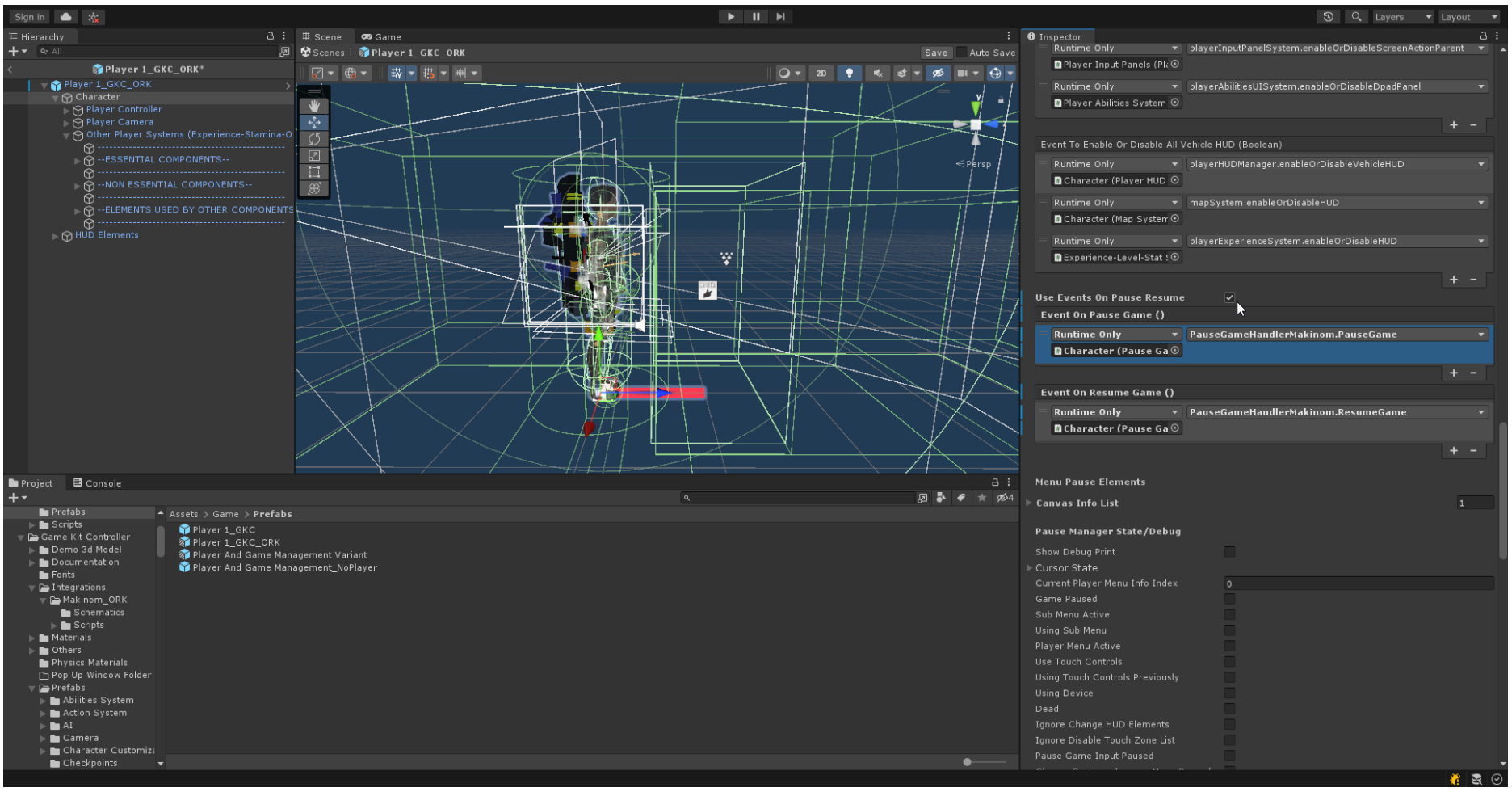
- Set the Transform component’s position to (0, 1, 1). This will move the Box Collider to the front of the character.
- In the Interaction Controller component, enable the “Interact With Nearest” checkbox.



- Select the “Character” gameobject.
- Add the “Pause Game Handler Makinom” component.



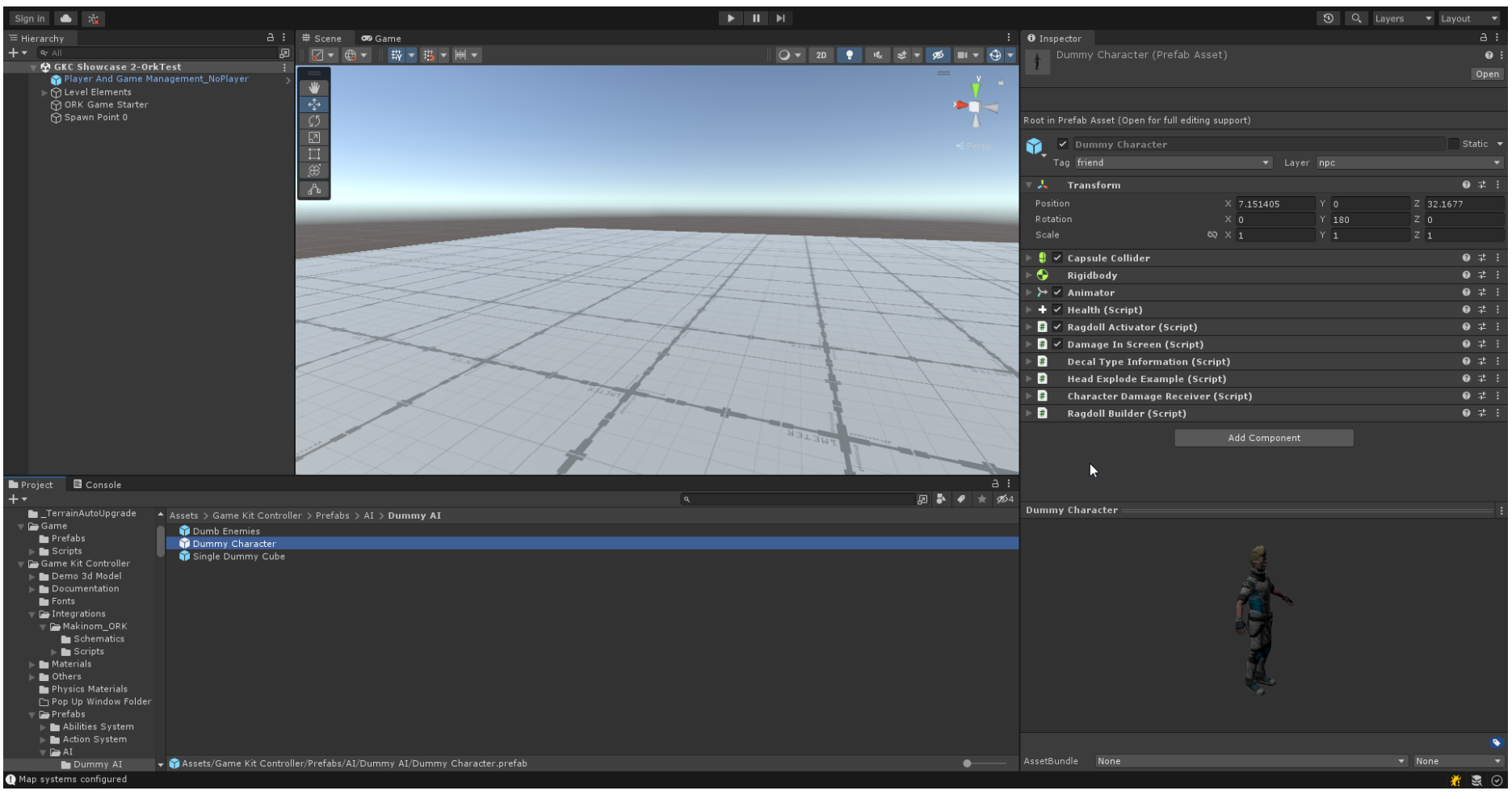
- In the “Menu Pause” component, scroll down to the “Use Events On Pause Resume” checkbox.
  Enable the checkbox, to activate the two events below.
- Add one listener to both events (click the “+” button). Drag the Character gameobject to both events.
- In the event “Event On Pause Game”, call PauseGameHandlerMakinom.PauseGame ()
- In the event “Event On Resume Game”, call PauseGameHandlerMakinom.ResumeGame ()



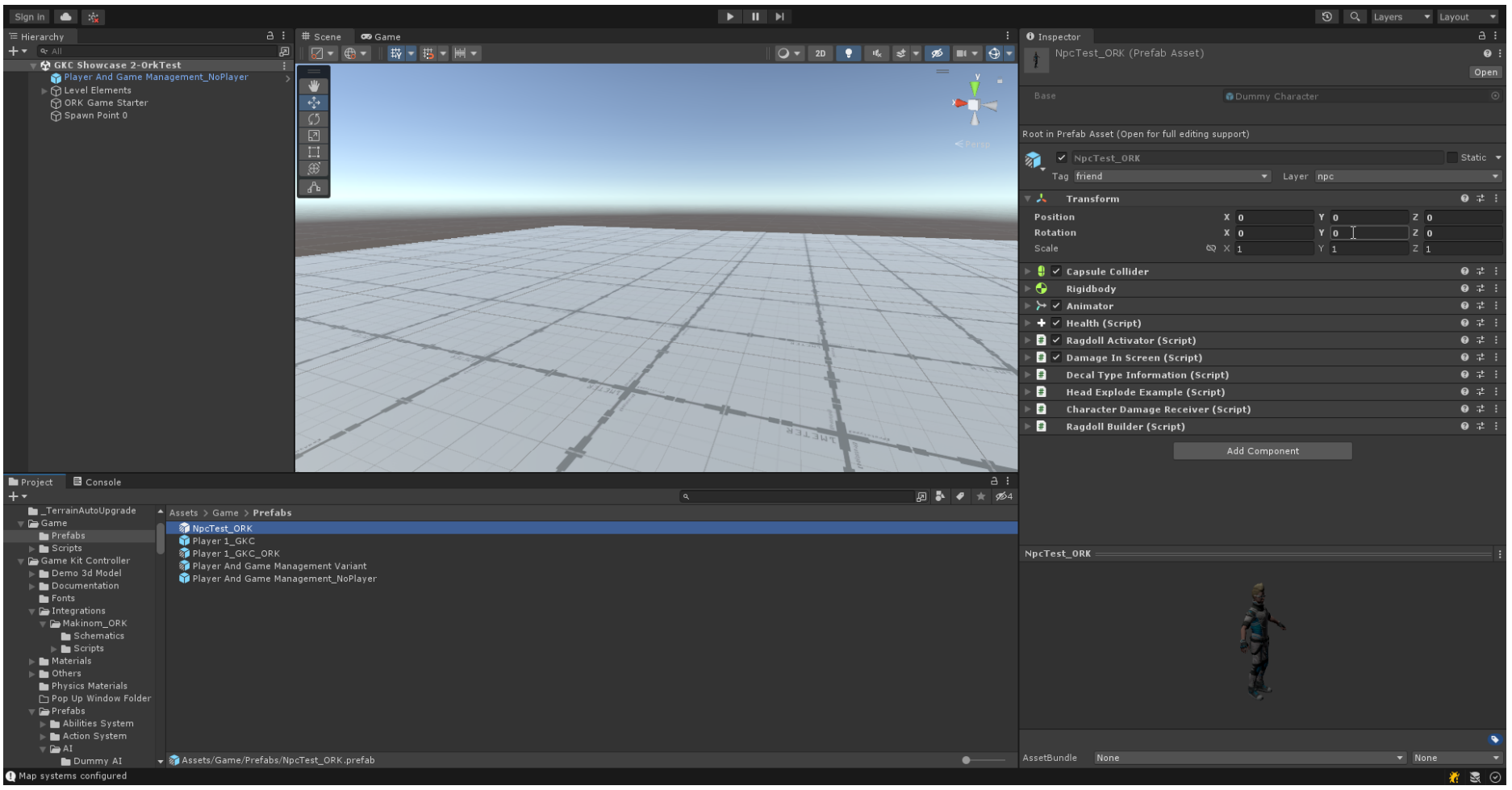
- After all this setup, whenever we pause/unpause the game via GKC and enter/exit GKC’s pause menu, the events will also pause Makinom. Makinom, in turn, will block/unblock the player’s control in its logic (this is different from GKC’s input system).
  This is needed so certain Makinom/ORK events (e.g.: Interactions; Starting an NPC dialogue) won’t be available while GKC is paused.

- Save the prefab. Close Prefab Mode.

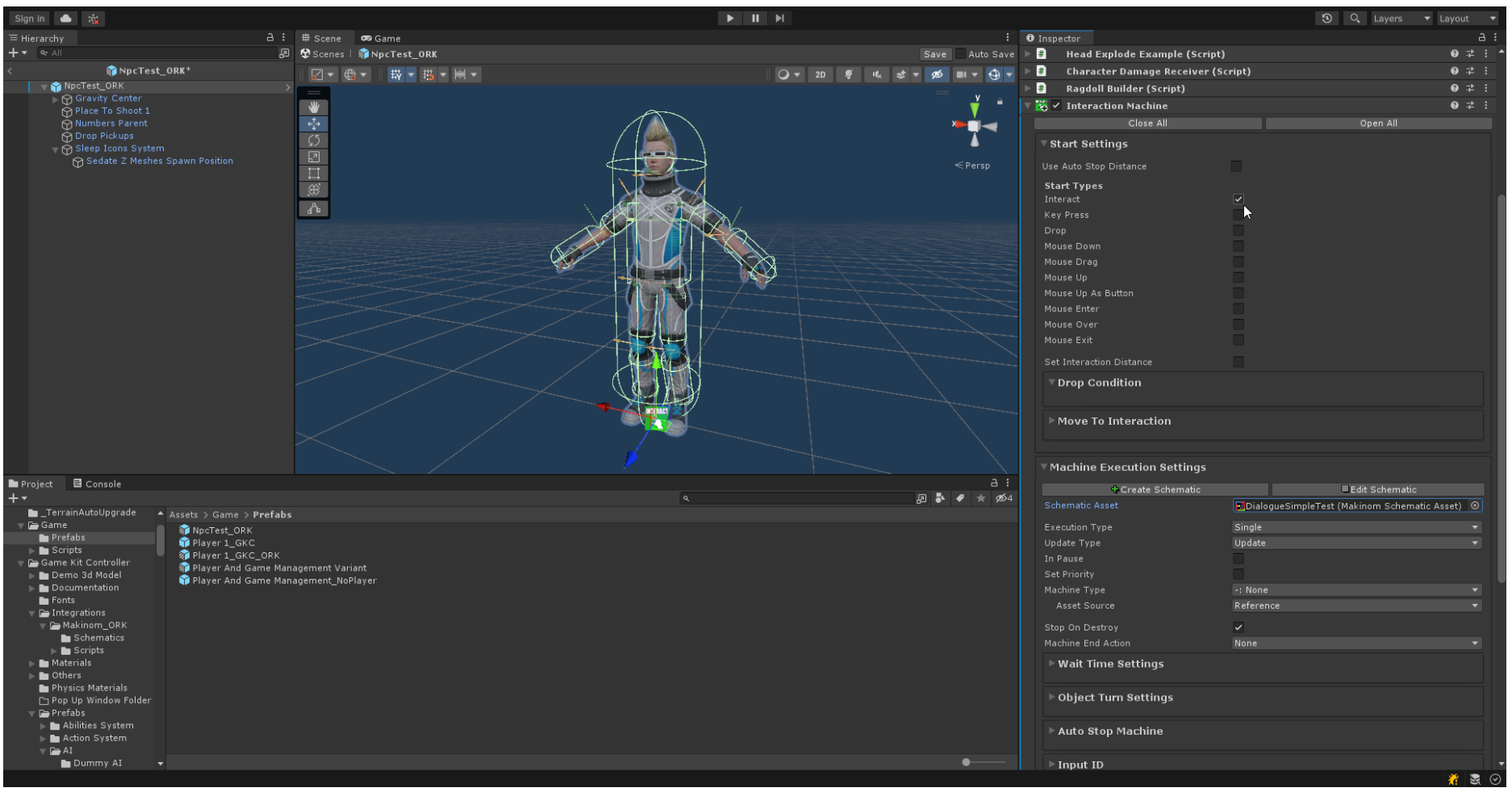
- Let’s add a simple NPC to the scene, to ensure that the pause behavior described above works correctly.
- Search for the [Dummy Character] prefab from GKC.
  The path is “Assets/Game Kit Controller/Prefabs/AI/Dummy AI/Dummy Character.prefab”



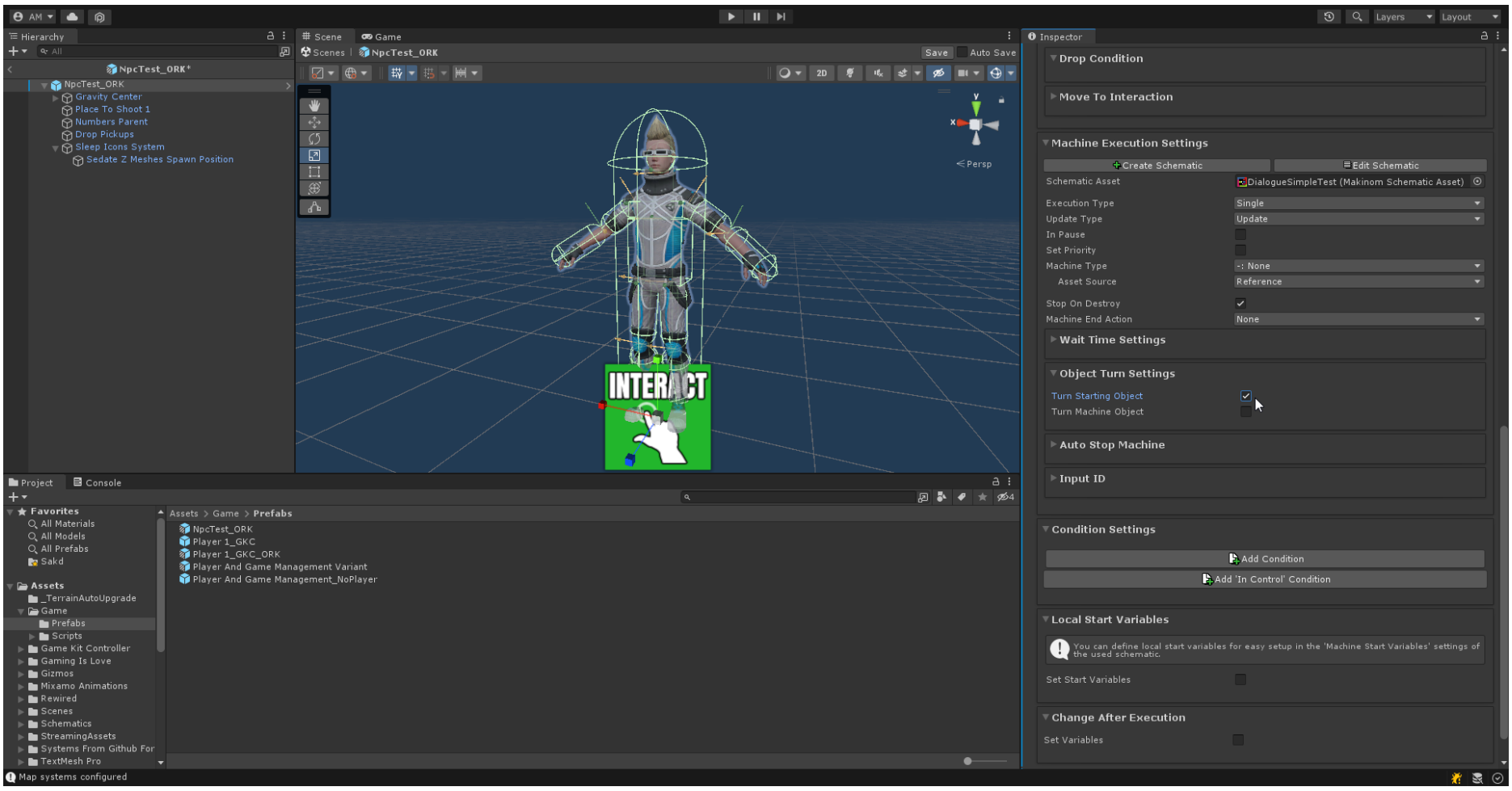
- Create a prefab variant of [Dummy Character] prefab.
  Save this prefab variant somewhere outside the “Assets\Game Kit Controller” folder.
- Rename the prefab variant to [NpcTest_ORK]
- Set its Transform position and rotation to (0, 0, 0)



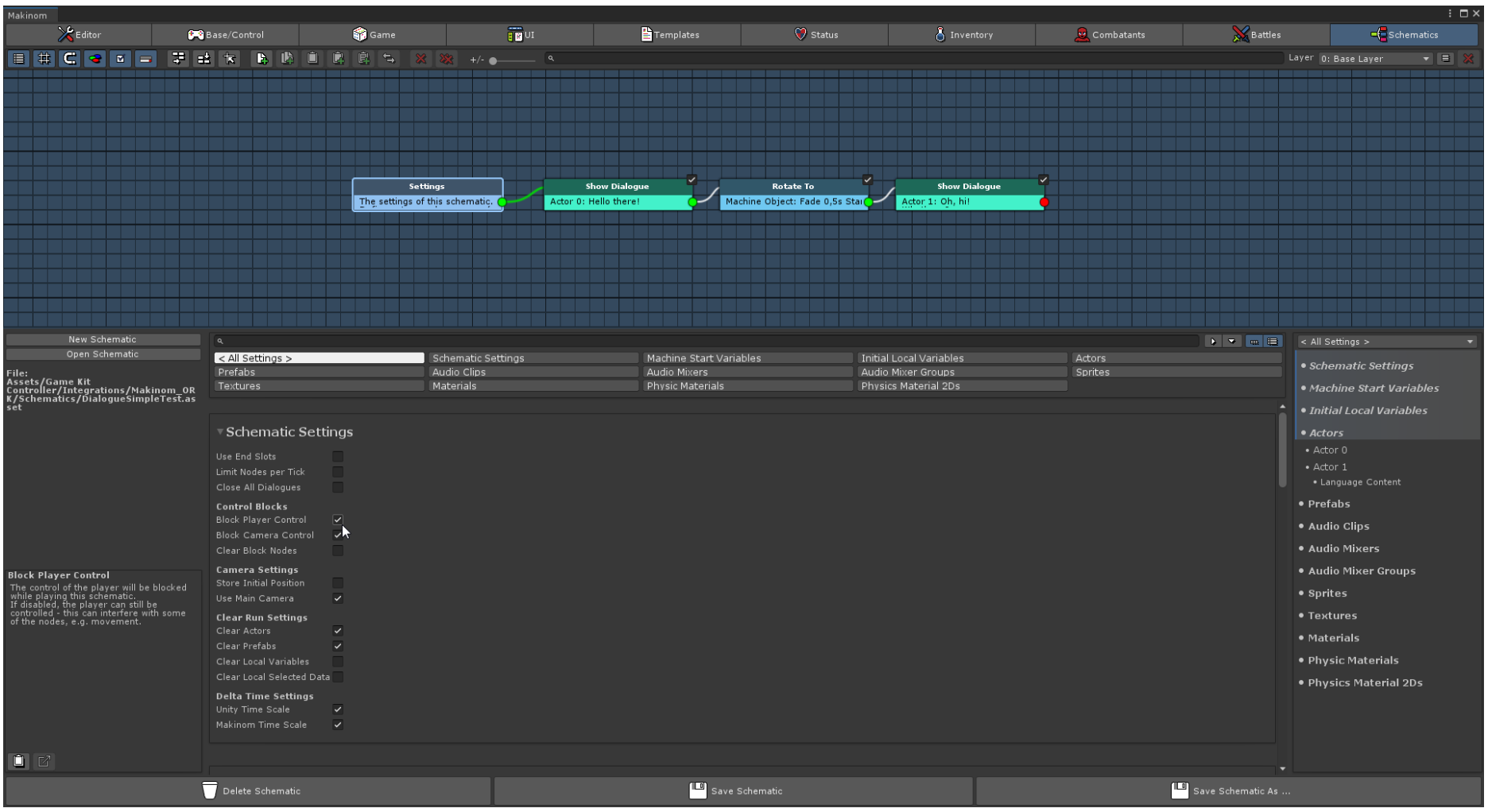
- Open [NpcTest_ORK] in Prefab Mode.
- On its root gameobject, add an Interaction Machine component.
- On the Interaction Machine’s Start Settings -> Start Types, enable the “Interact” checkbox.
- On Machine Execution Settings -> “Schematic Asset” field, assign the “DialogueSimpleTest” schematic.



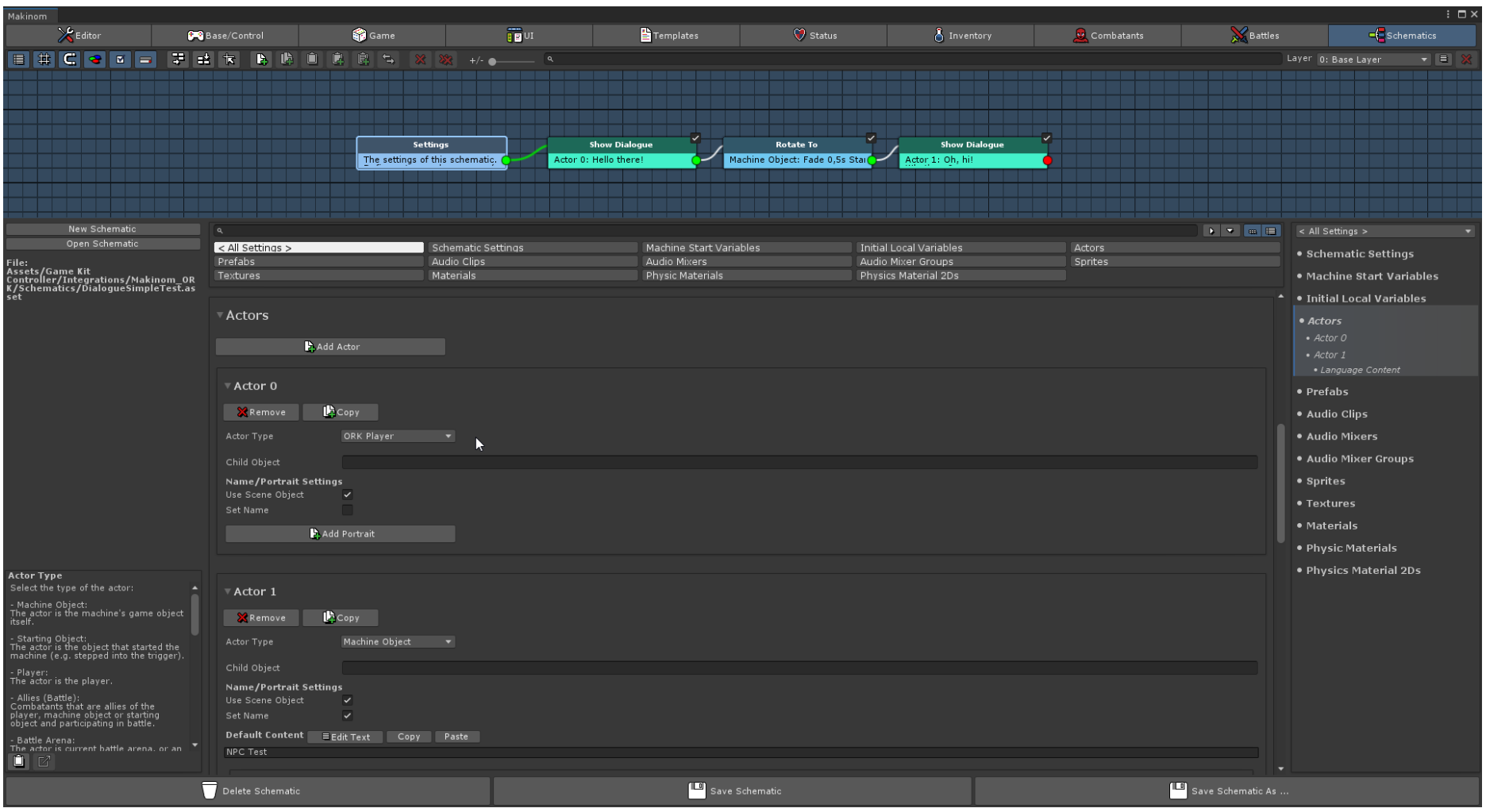
- Expand “Object Turn Settings”, then enable “Turn Starting Object”.
  By enabling this, the player character (which will be the Starting Object in this case) will automatically rotate to the NPCs direction when starting the dialogue.



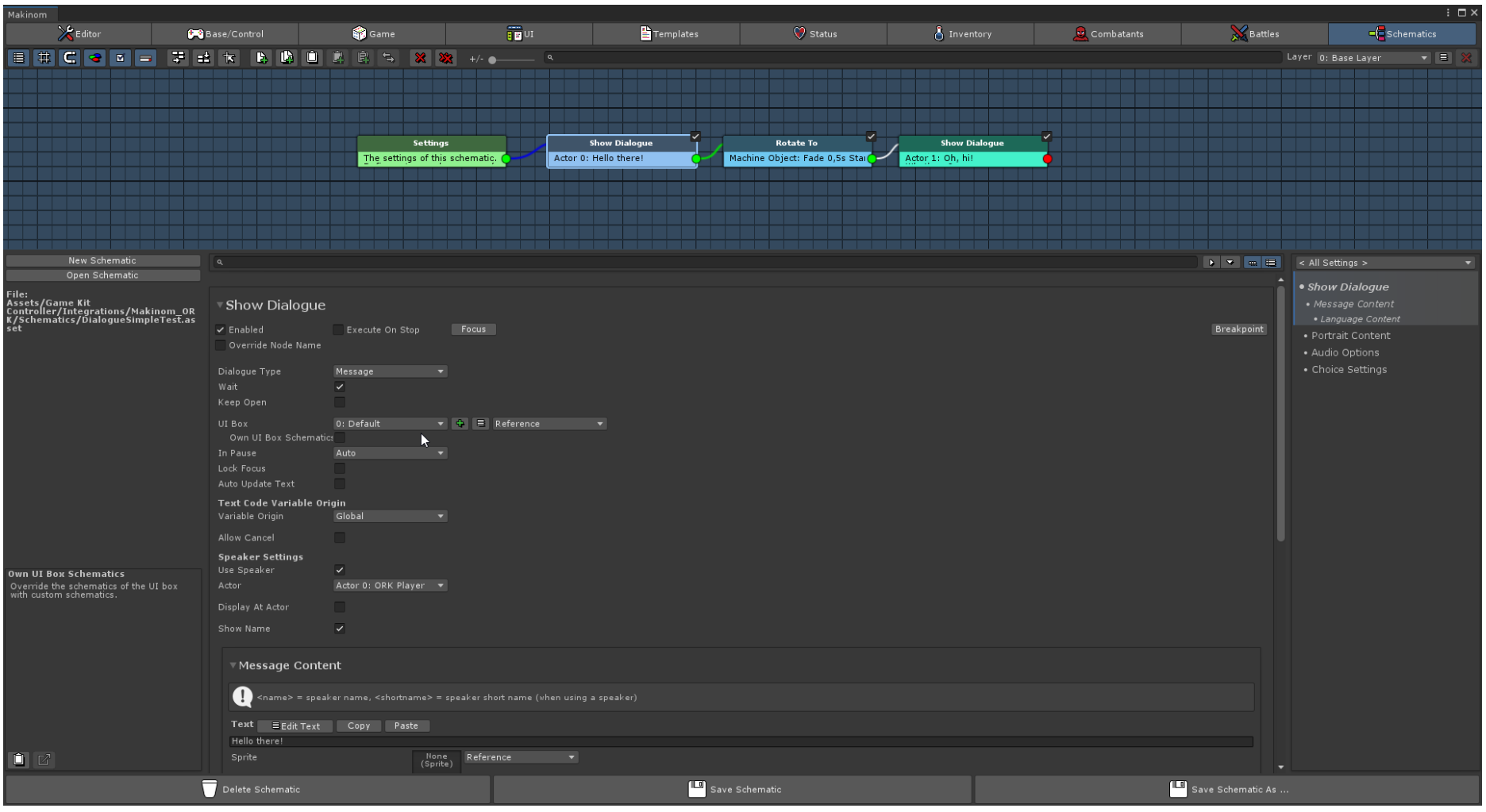
- Open the DialogueSimpleTest schematic.
  Click the “Edit Schematic” button on the Interaction Machine, or double-click the schematic asset.
- On the Settings node, enable “Block Player Control” and “Block Camera Control”.



- On the Settings node, set up the Actors like in the image below.
- Actor 0: Actor Type = ORK Player
- Actor 1: Actor Type = Machine Object; Set Name = true (optional)

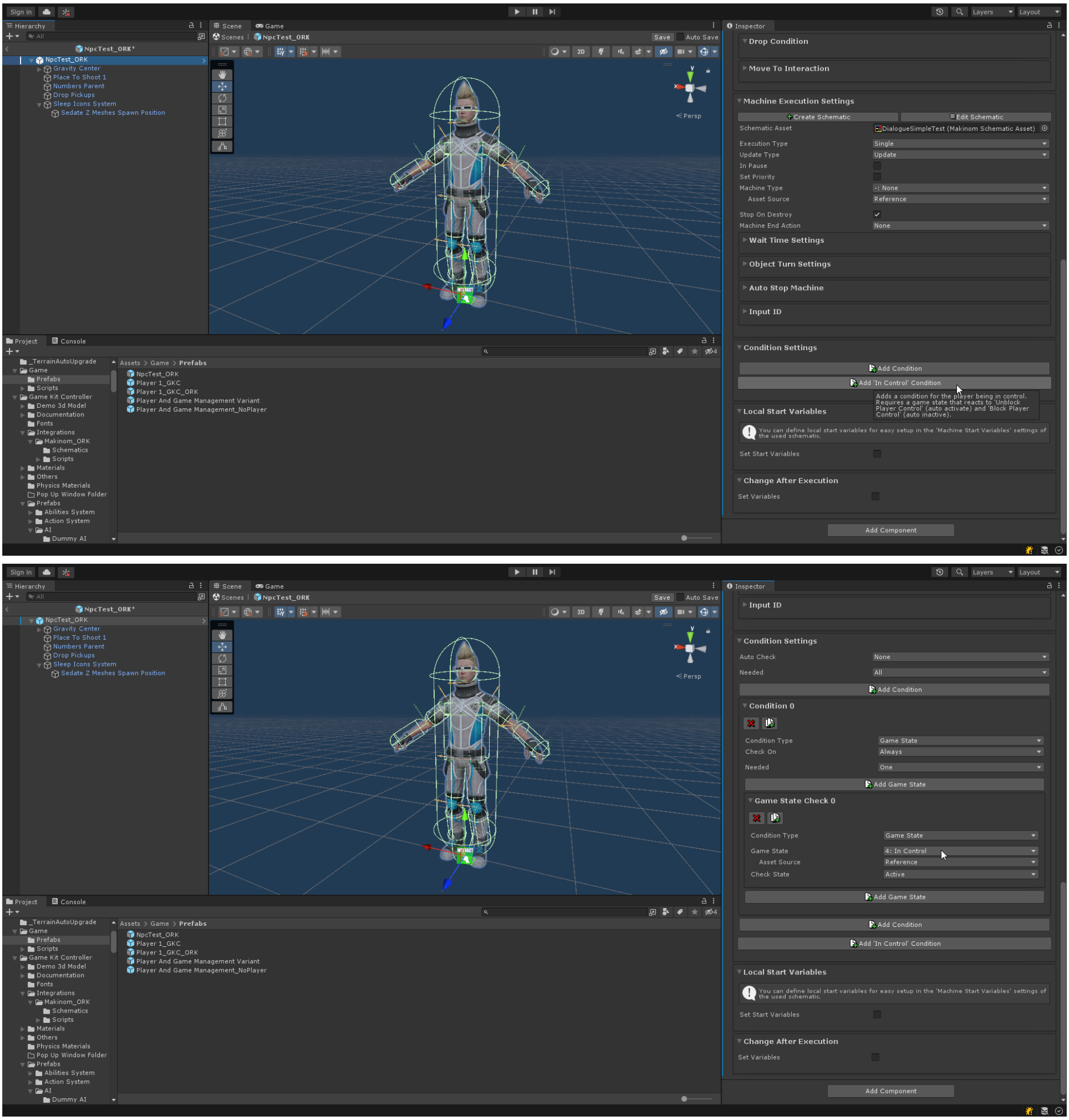


- On both “Show Dialogue” nodes, make sure “UI Box” is set to Default (or to any other UI Box you wish to use).
- Make sure “Dialogue Type” is set to Message, and “Wait” checkbox is enabled.



- Save the schematic.
- Close the Makinom window.

- Back to the [NpcTest_ORK] prefab, on the Interaction Machine component, scroll down to Condition Settings.
- Click the [Add ‘In Control’ Condition] button.
  This will automatically add a condition to this Interaction Machine, where the interaction will only be available if Makinom is currently in the “In Control” game state.
  In other words, whenever we pause GKC (and by extension, Makinom, thanks to the events we’ve set up previously), the player’s control will be blocked by Makinom. So, the interaction won’t be available if the game is currently paused, even if the player character is on the interaction trigger’s range while the game is paused.

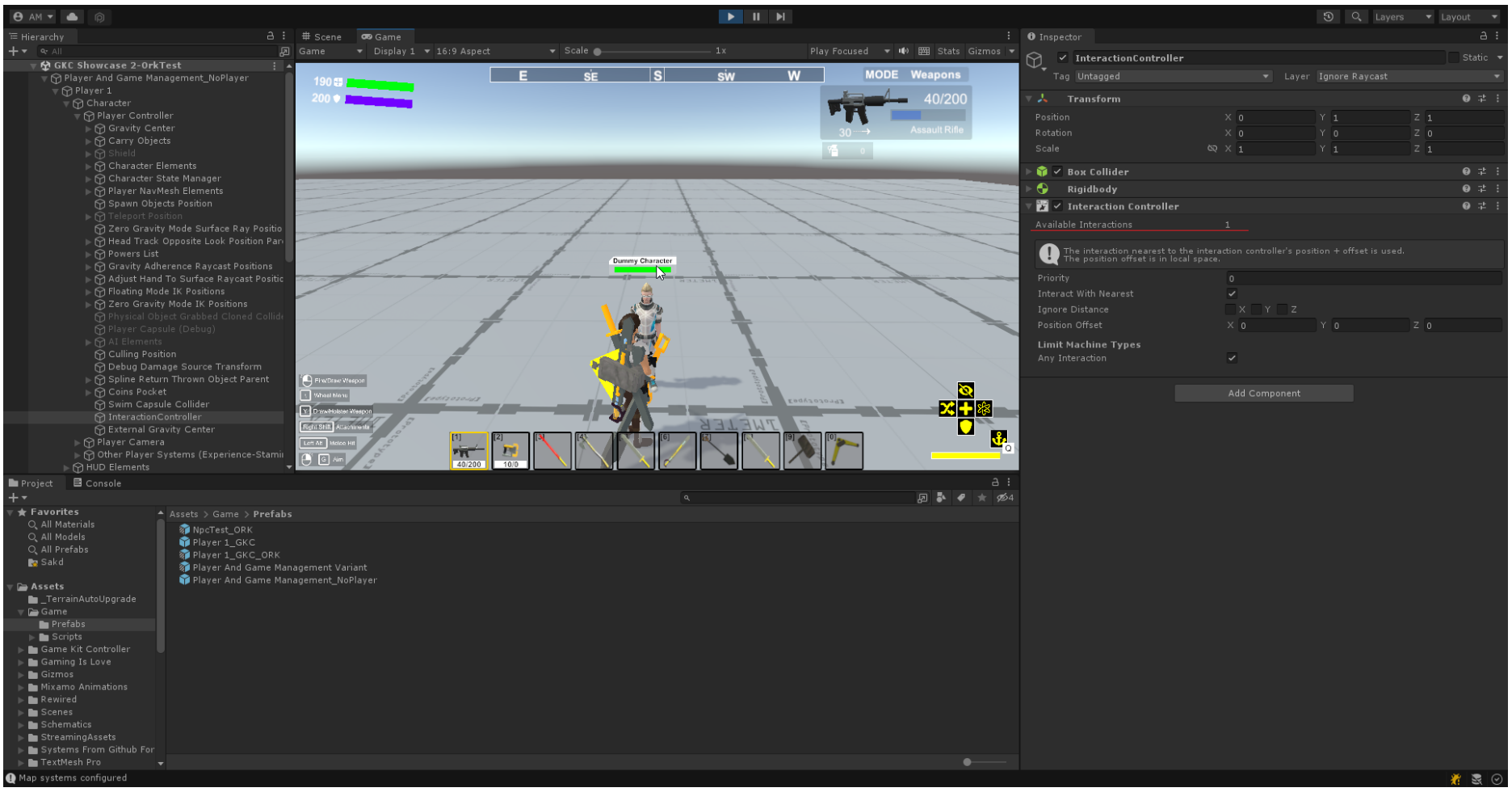


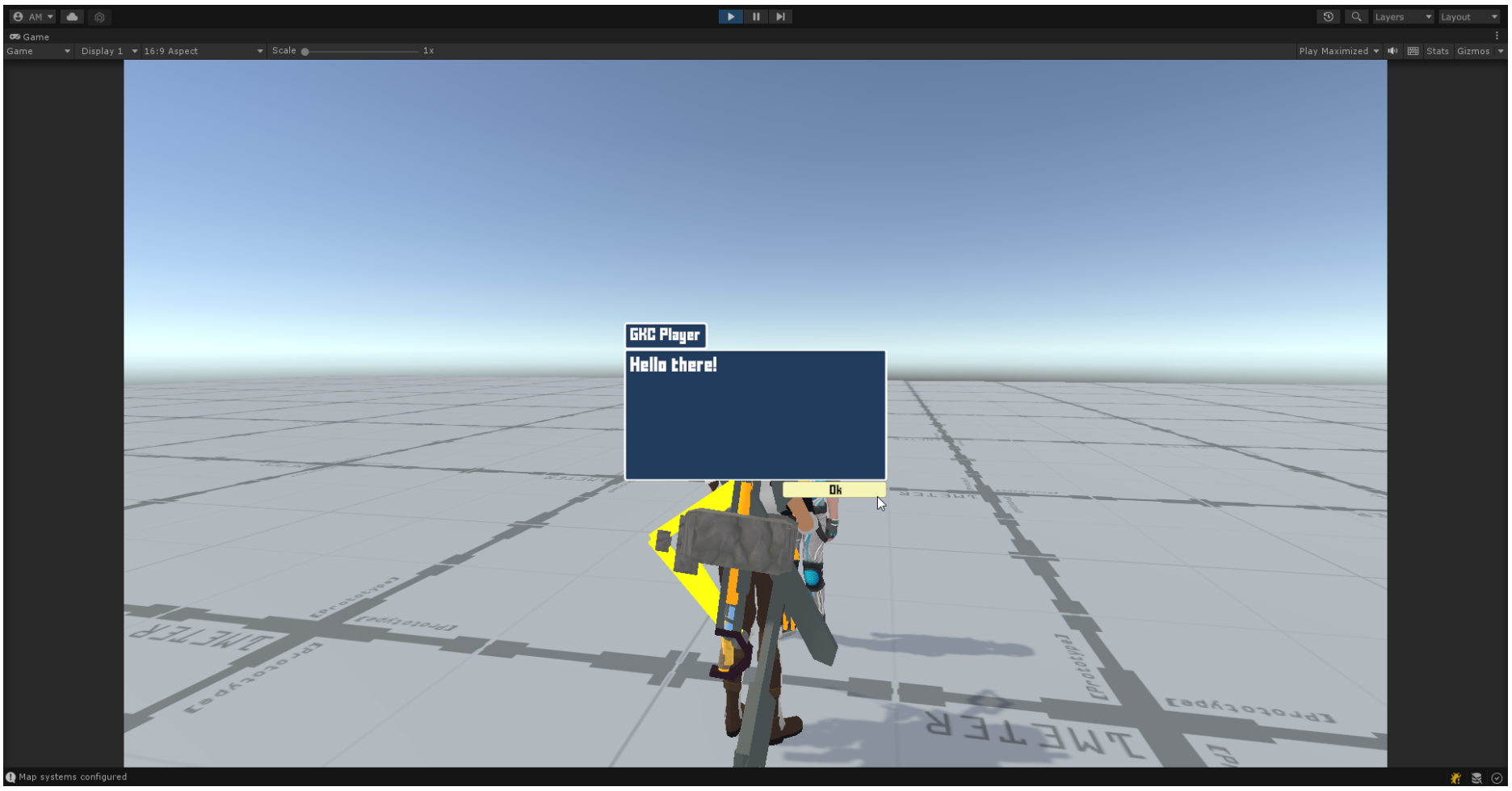
- Save the prefab and exit Prefab Mode.

- Add to the scene an instance of [NpcTest_ORK] prefab.

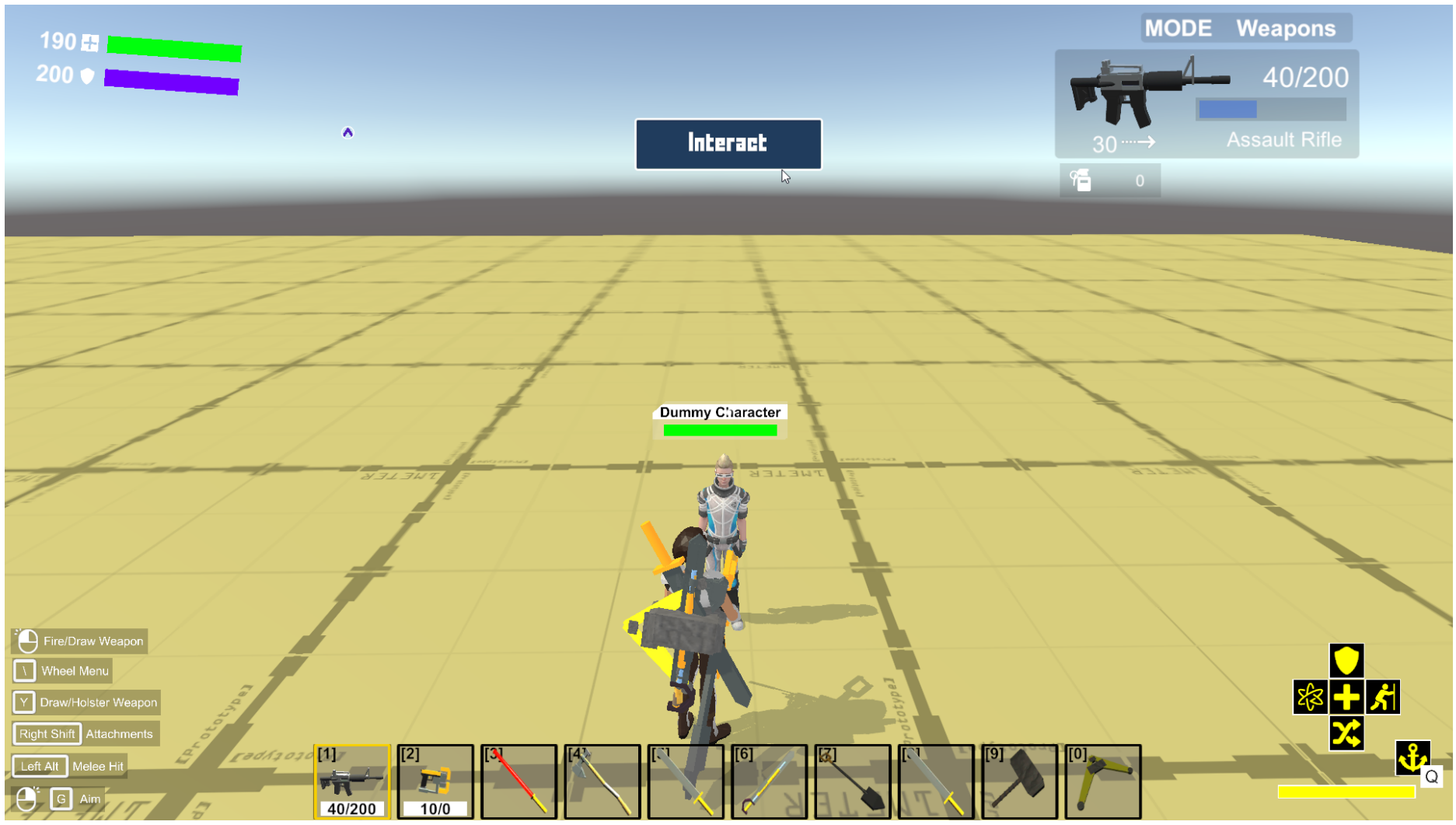
- Save the scene.

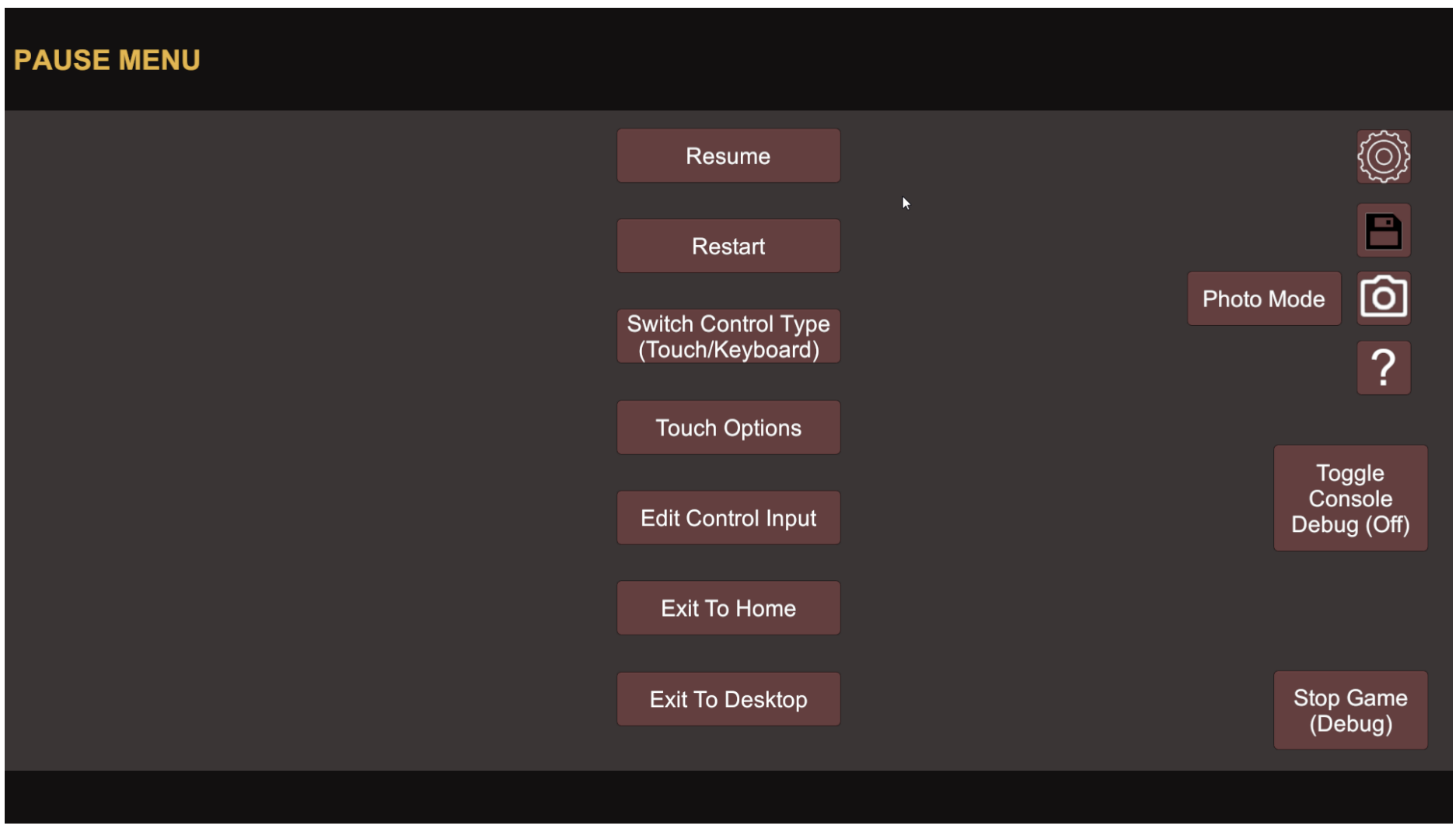
- Let’s test the dialogue interaction. Press Play to run the game.
- Move the player character to the front of the NPC. Press Enter/Return to start the dialogue.
  The player won’t be able to move the character/camera while the dialogue is open.
- Advance the dialogue lines, until the dialogue finishes.
  The player will be able to move the character/camera again.



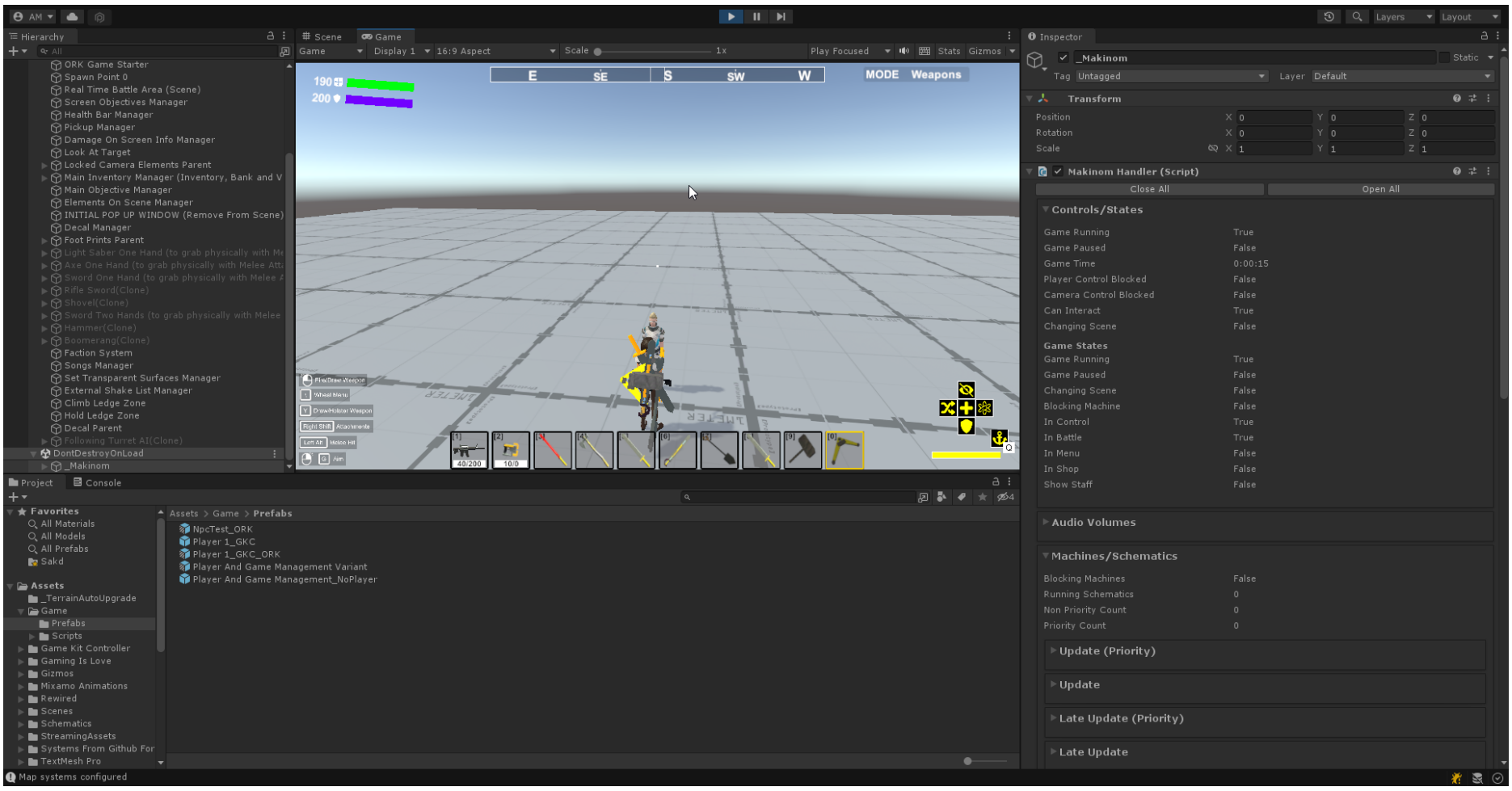


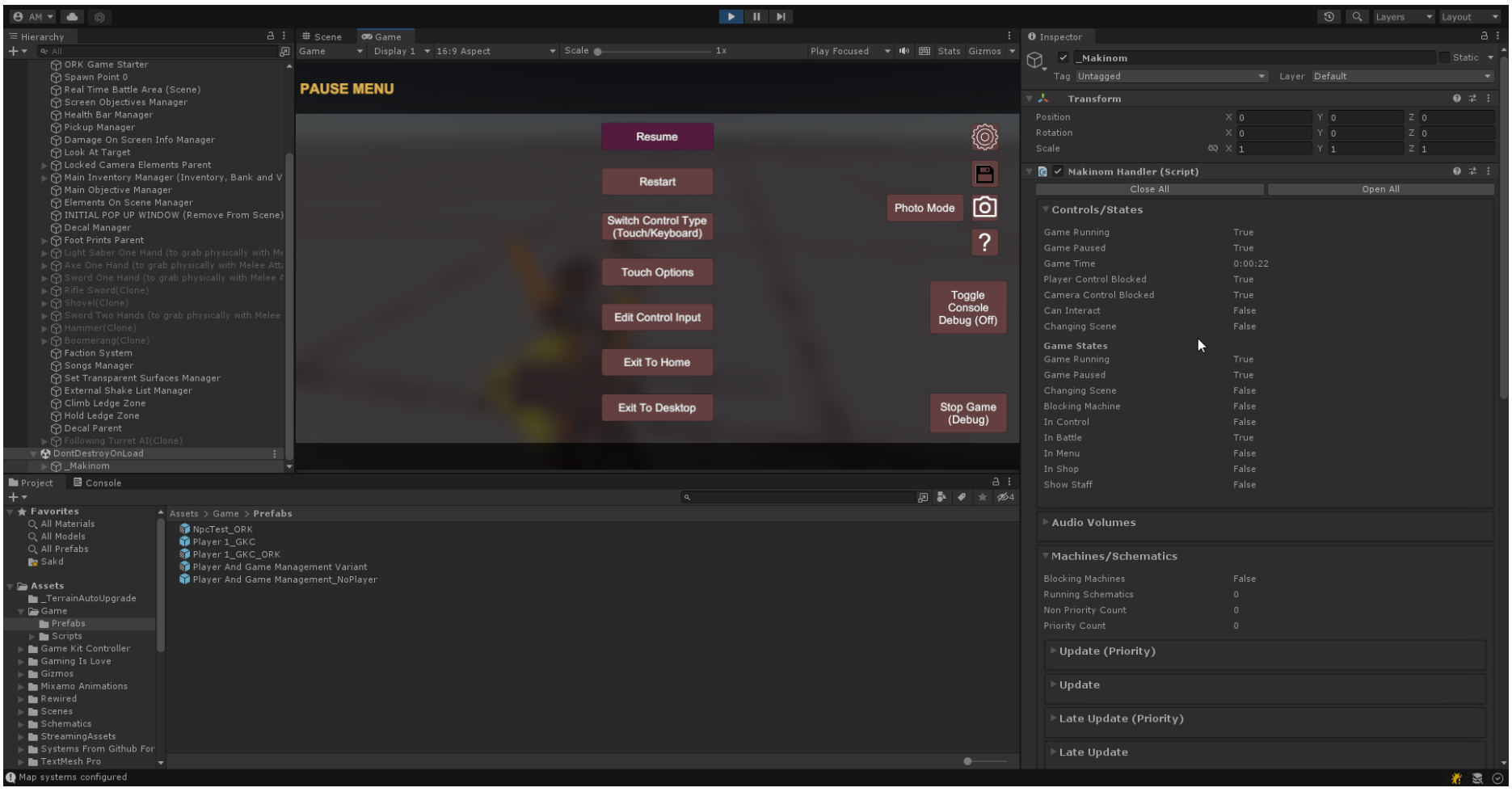
- While the player character is inside the interaction's range and the "Interact" box is being displayed (in case your game has its Interaction HUD set up), press Esc to pause the game and open the GKC pause menu.
  The "Interact" box should disappear while the game is paused.
  And if you press Enter/Return while the game is paused, the dialogue interaction won't be triggered.

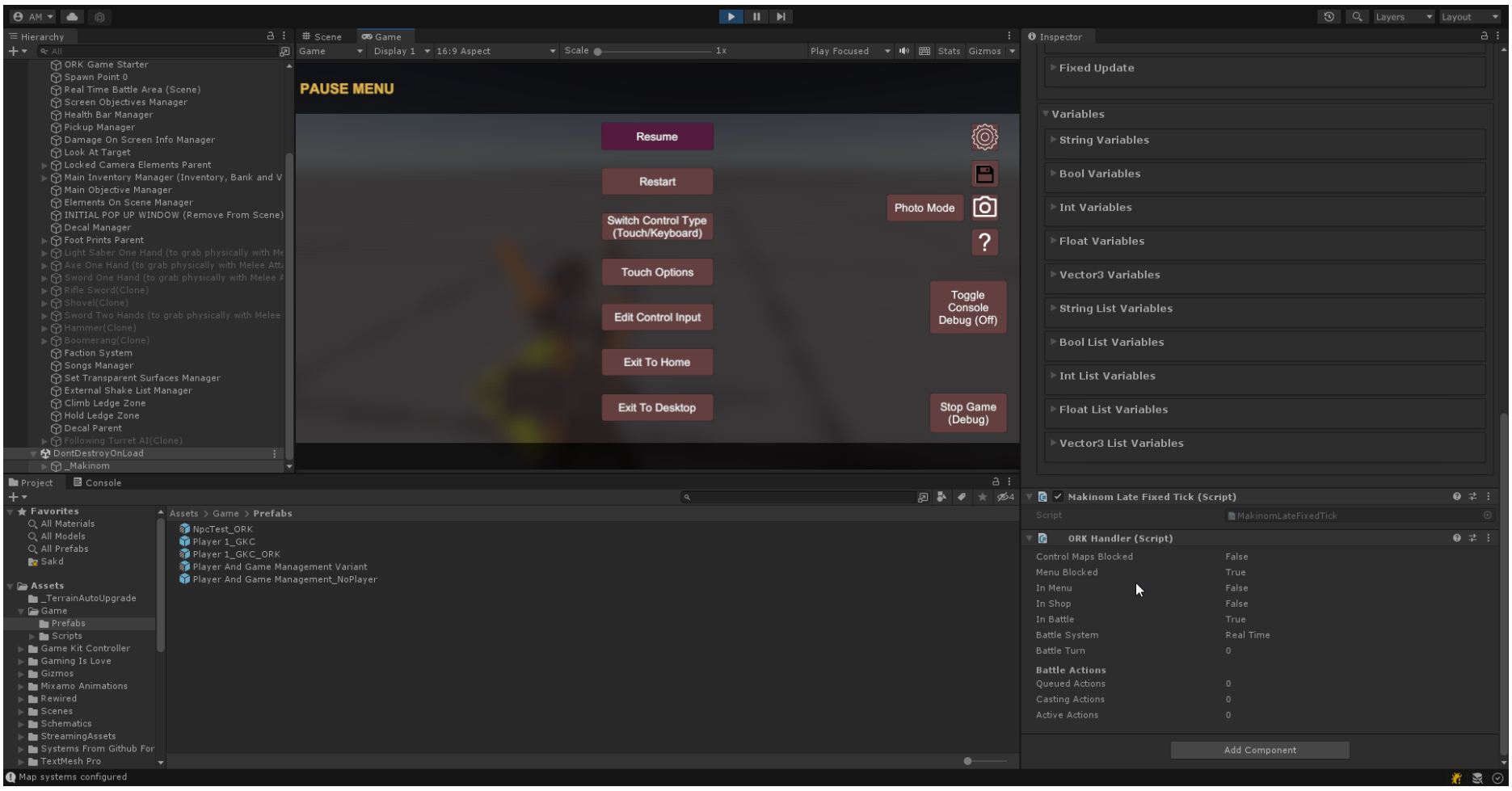




- TIP: While the game is running, you can debug the current Makinom/ORK states by expanding the DontDestroyOnLoad scene, then selecting the “_Makinom” gameobject.
- Check the “Makinom Handler” and “ORK Handler” components.

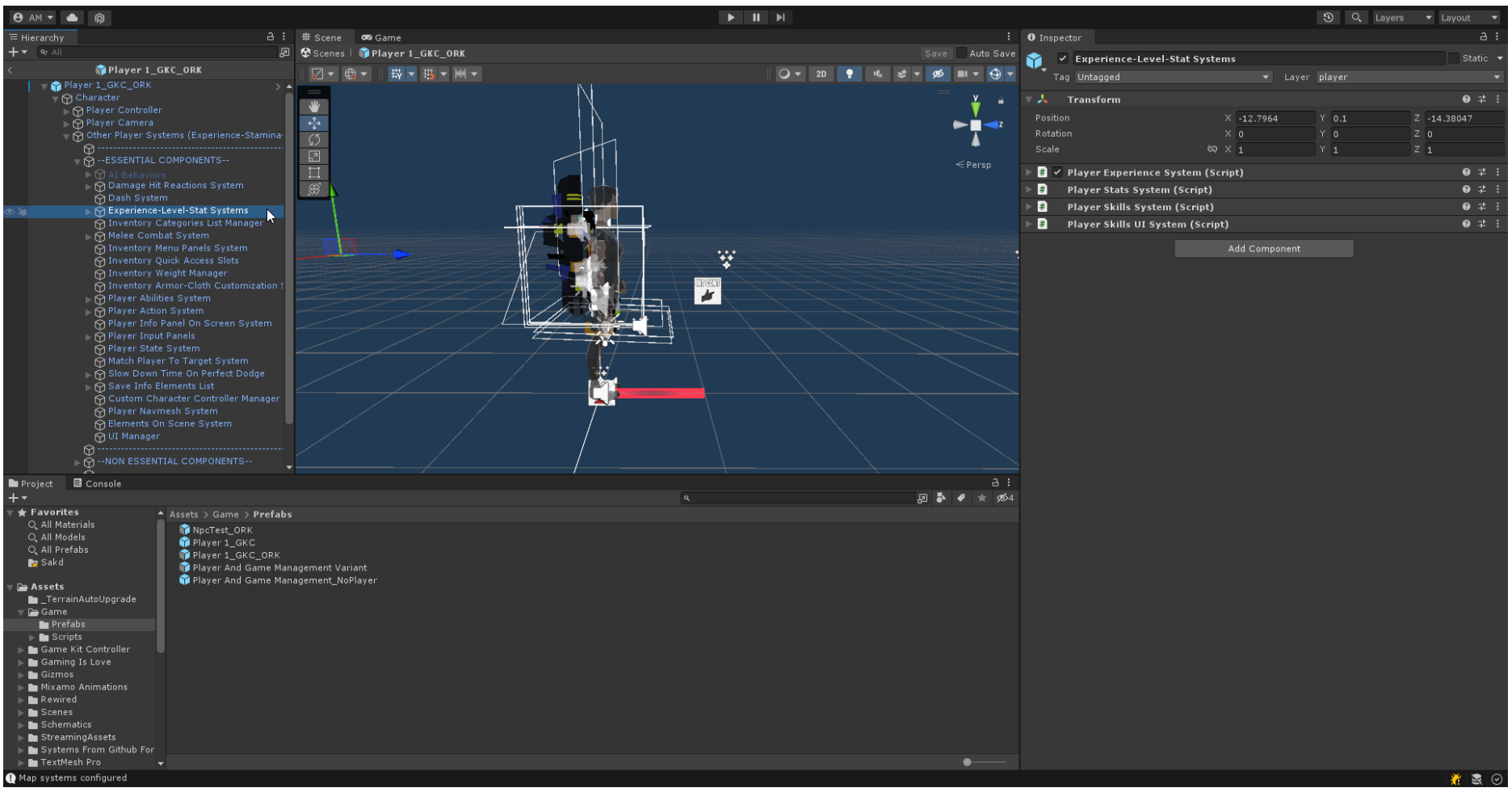




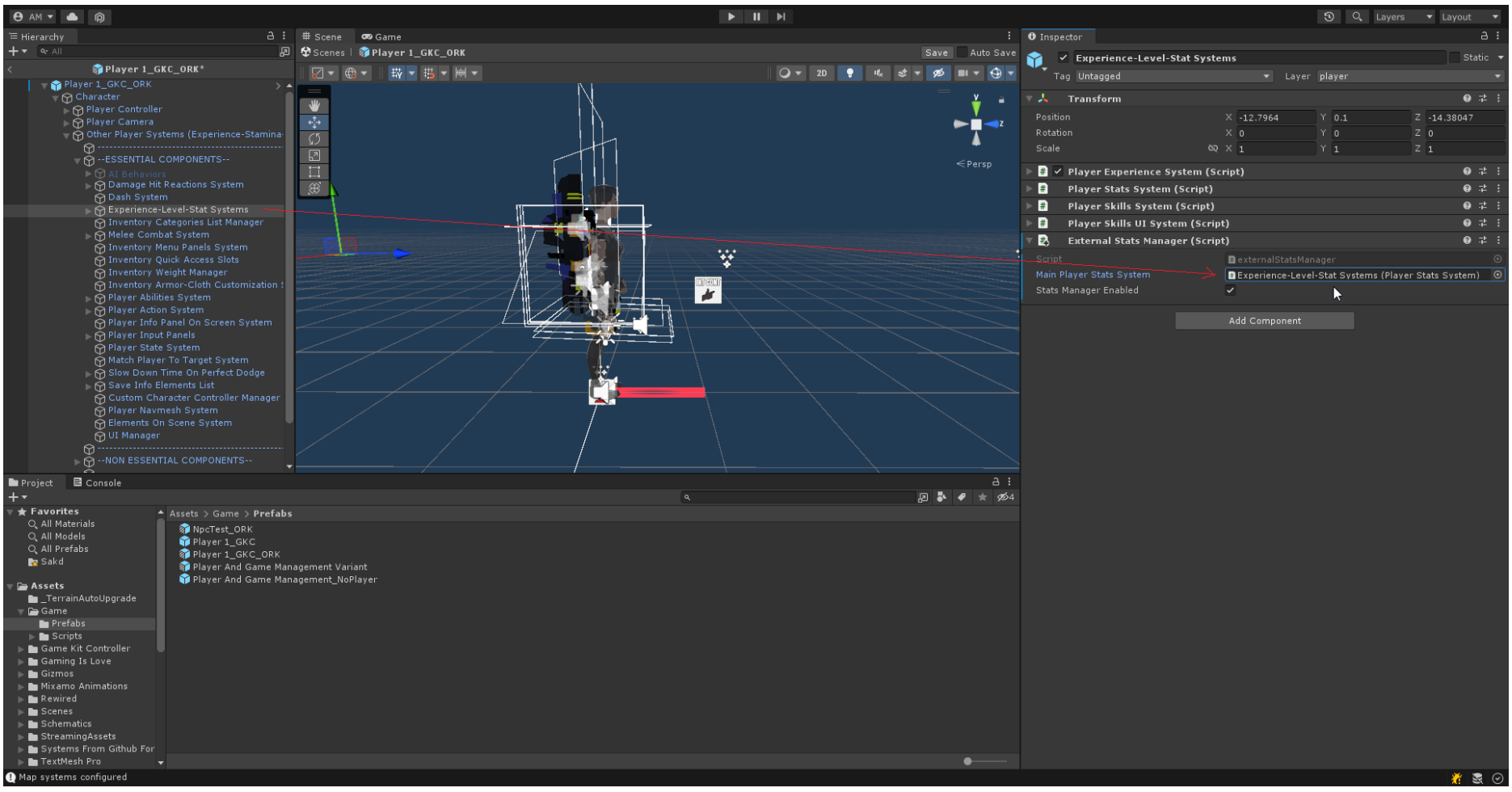

PAUSE MENU
Resume
Restart
Switch Control Type (Touch/Keyboard)
Touch Options
Edit Control Input
Exit To Home
Exit To Desktop
Photo Mode
Toggle Console Debug (Off)
Stop Game (Debug)
Hierarchy
Scene
Game
Inspector
Project
Console
Fixed Update
Variables
String Variables
Bool Variables
Int Variables
Float Variables
Vector3 Variables
String List Variables
Bool List Variables
Int List Variables
Float List Variables
Vector3 List Variables
Makinom Late Fixed Tick (Script)
ORK Handler (Script)
Control Maps Blocked False
Menu Blocked True
In Menu False
In Shop False
In Battle True
Battle System Real Time
Battle Turn 0
Battle Actions
Queued Actions 0
Casting Actions 0
Active Actions 0
Add Component
Map systems configured

GKC - ORK Stats Bridge

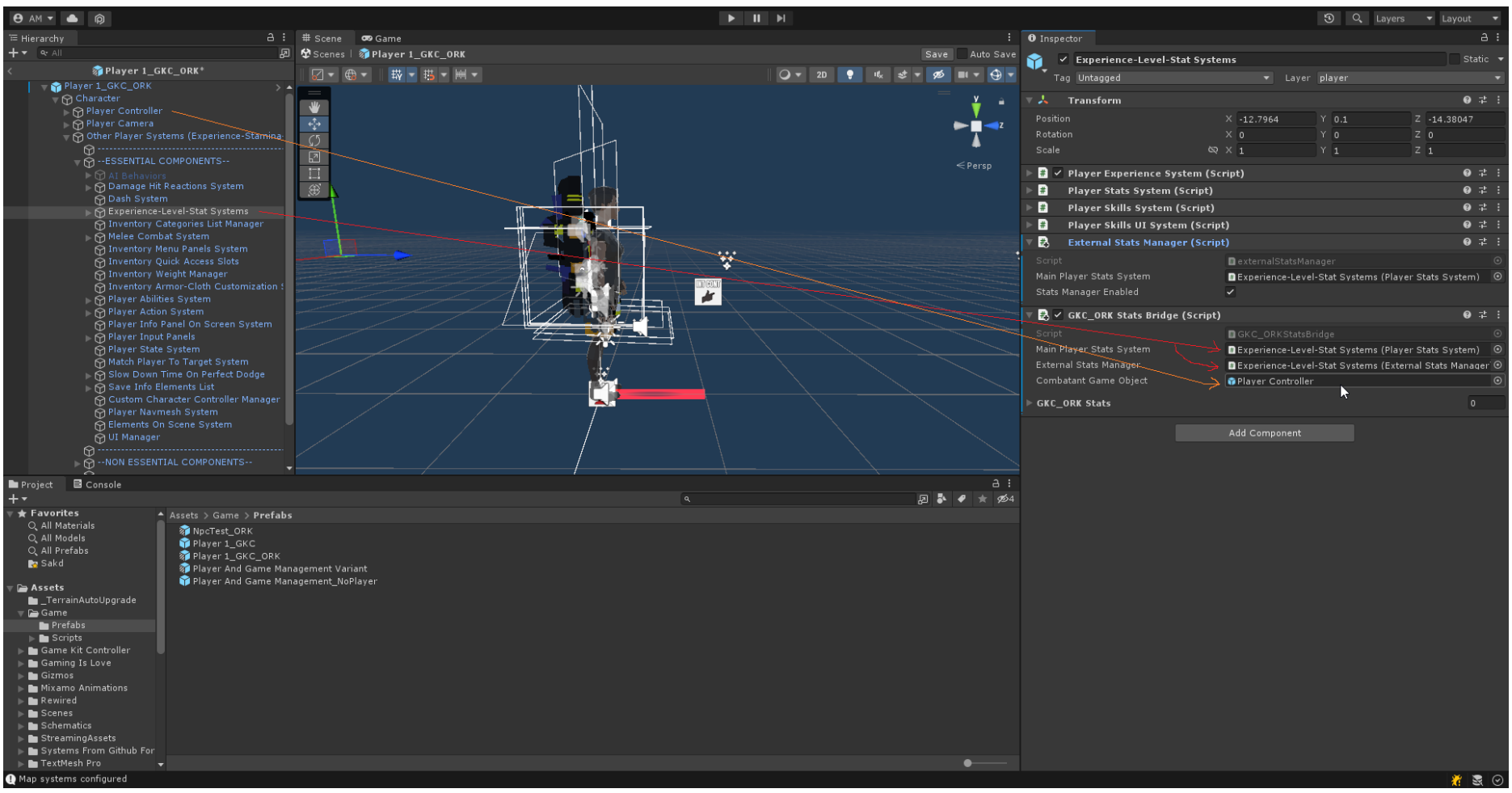
- Open the [Player 1_GKC_ORK] prefab.
- Search for the “Experience-Level-Stat Systems” child gameobject.
  The path is “Player 1-GKC-ORK/Character/Other Player Systems (Experience-Stamina-Oxygen....)/--ESSENTIAL COMPONENTS--/Experience-Level-Stat Systems”



- Select the “Experience-Level-Stat Systems” gameobject.
- Add the “External Stats Manager” component to it.
- Drag the “Experience-Level-Stat Systems” gameobject to the “Main Player Stats System” field.



- Add the [GKC_ORK Stats Bridge] component.
- Drag the “Experience-Level-Stat Systems” gameobject to the “Main Player Stats System” and “External Stats Manager” fields.
- Drag the “Player Controller” gameobject to the “Combatant Game Object” field.



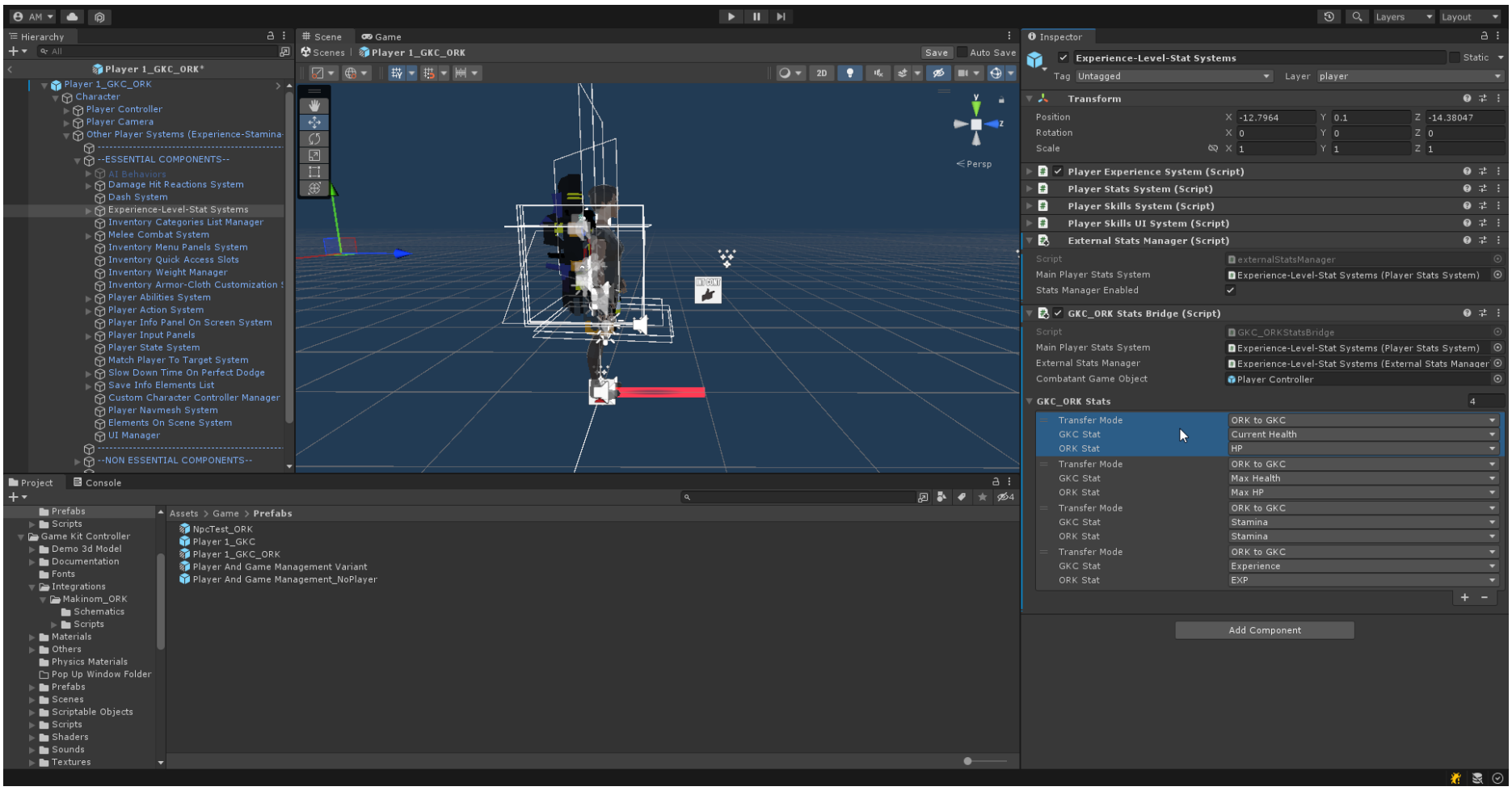
- Add new entries to the “GKC_ORK Stats” list.
- Set up the “GKC Stat” and its equivalent “ORK Stat” for each entry.
- Set up the “Transfer Mode” for each entry. Below is an explanation of each Transfer Mode:

    - None: Does nothing. Disables the entry.

    - ORK to GKC: Whenever the ORK StatusValue is updated, the stats bridge will automatically send the ORK StatusValue’s value to its respective GKC stat.

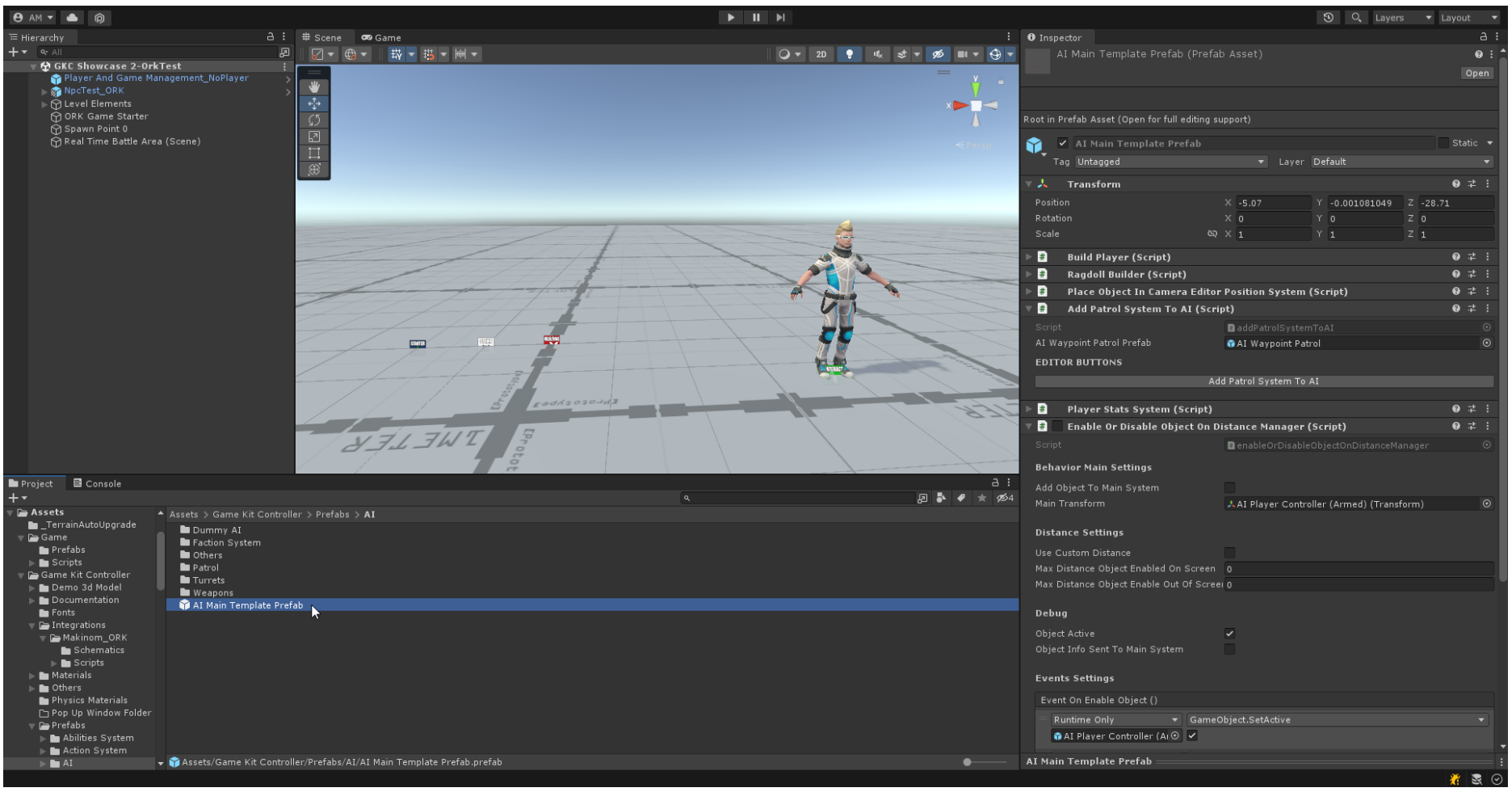
    - GKC to ORK: Whenever the GKC stat is updated, the stats bridge will automatically send the GKC stat’s value to its respective ORK StatusValue.

    - Both: Updated stats are automatically sent from both “ORK to GKC” and “GKC to ORK”. You can change the entry’s stats from any of the two systems (ORK, GKC), and the updated value will be sent to the other system.
      The “sending stat to the other system” logic is interrupted when it detects that the entry’s stats have the same value in both ORK and GKC’s stat systems.

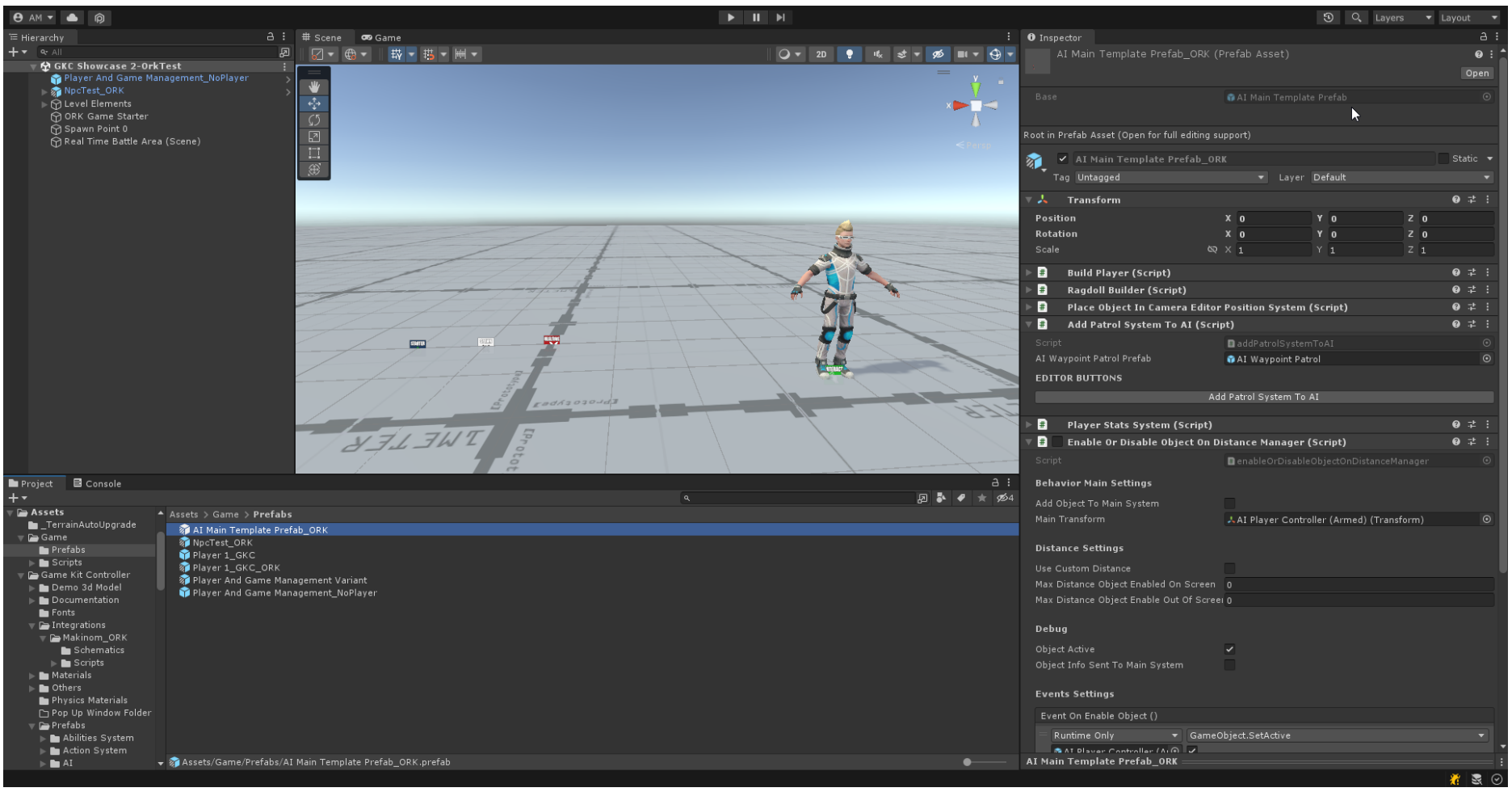


- Save the prefab.

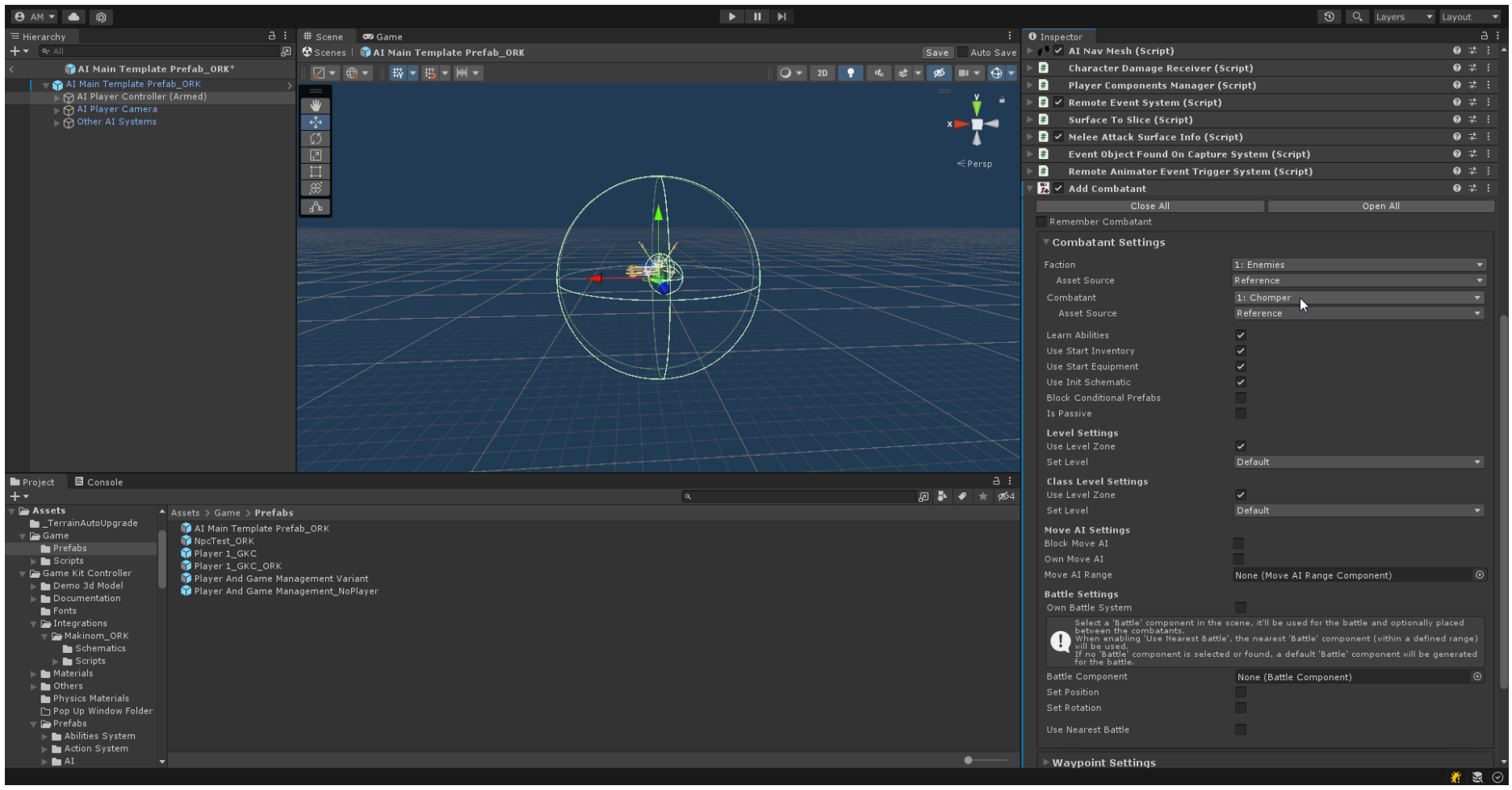
- To ensure that the stats bridge is working, let's add an enemy NPC combatant to the scene, for a quick test.
- Search for the [AI Main Template Prefab] prefab from GKC.
  The path is "Assets/Game Kit Controller/Prefabs/AI/AI Main Template Prefab.prefab"



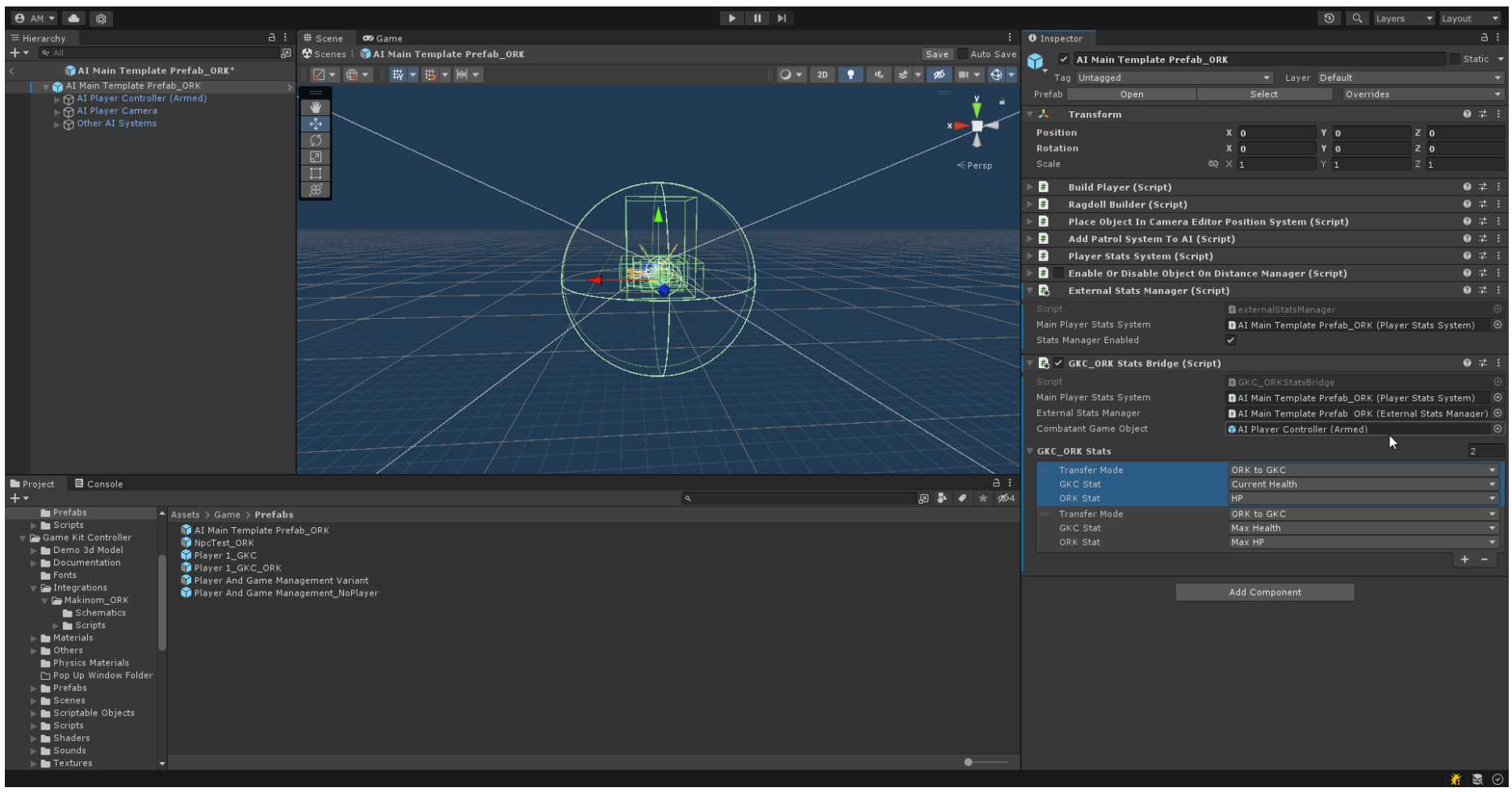
- Create a prefab variant of [AI Main Template Prefab] prefab.
  Save this prefab variant somewhere outside the "Assets\Game Kit Controller" folder.
- Rename the prefab variant to [AI Main Template Prefab_ORK]



- Select the “AI Player Controller (Armed)” gameobject.
- Add an “Add Combatant” component to it.
- Set its Faction to “Enemies”.
- Set its Combatant to “Chomper” (for a quick test), or to any other Combatant you wish to use.

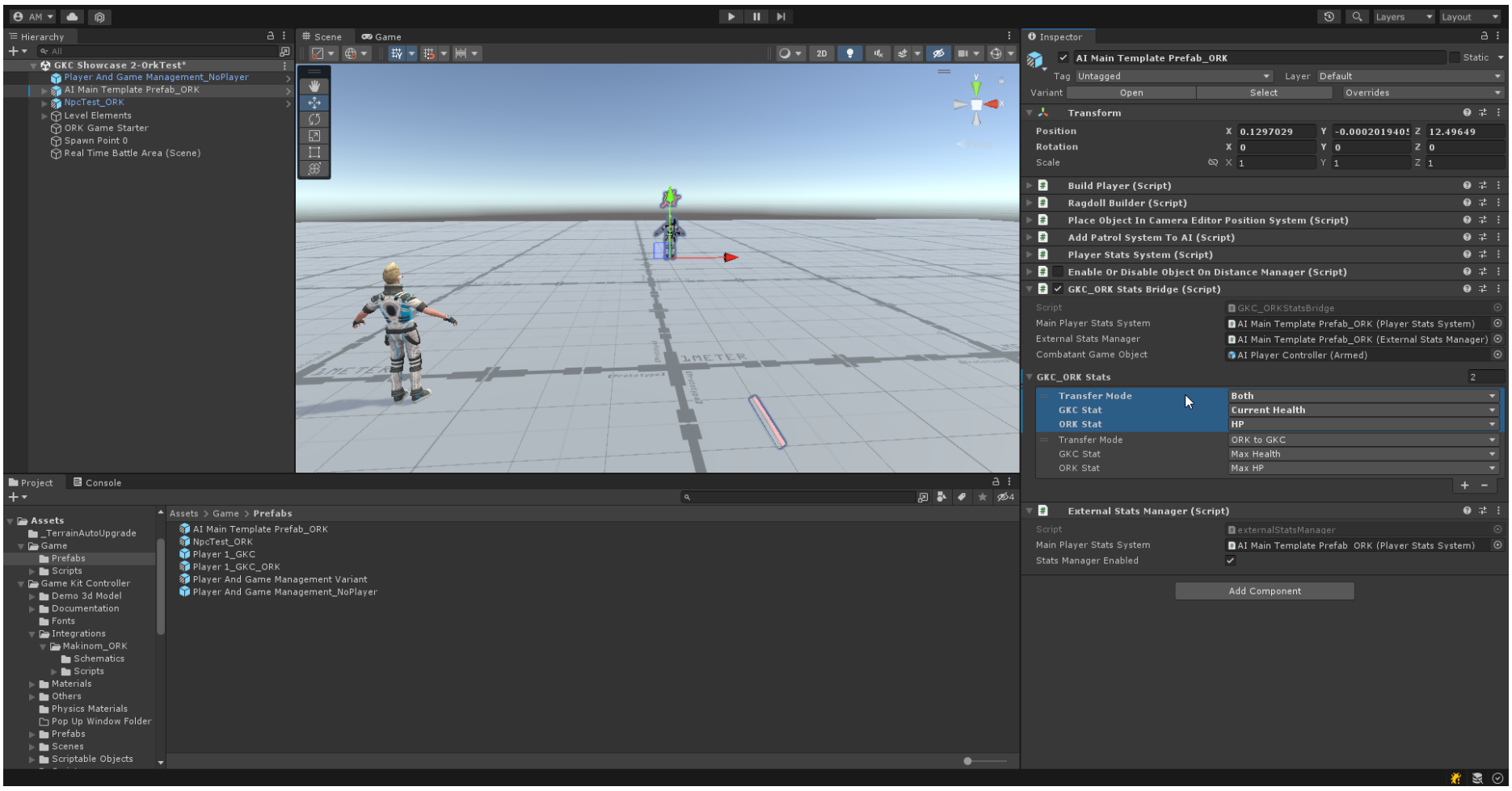


- Select the root gameobject. Add the “External Stats Manager” and “GKC_ORK Stats Bridge” components to it. Assign the components’ references, in a similar way as we did to our player.
- Set up the enemy’s GKC/ORK stats in the bridge component.



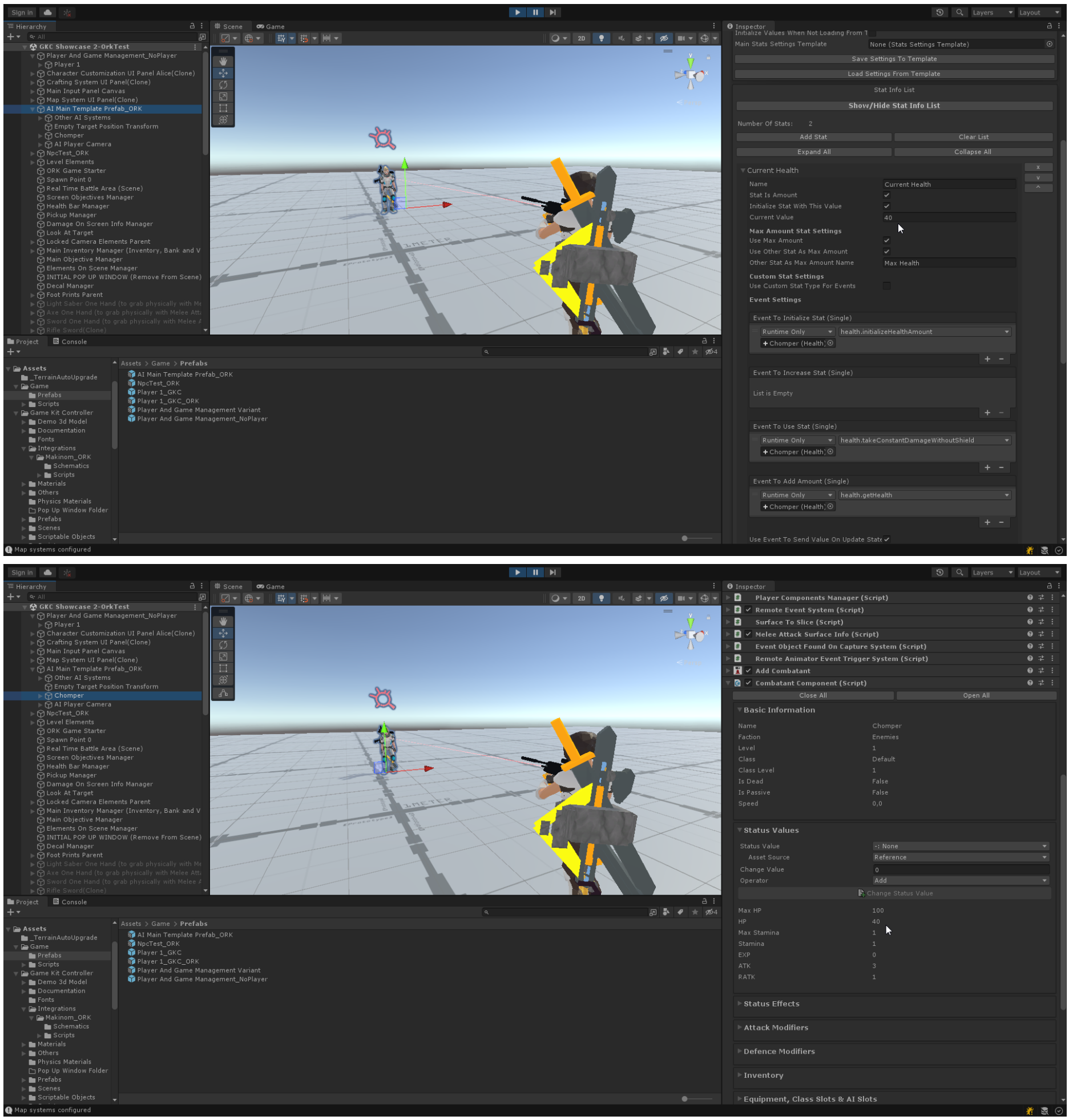
- Save the prefab.

- Add to the scene an instance of [AI Main Template Prefab_ORK] prefab.
  Place it a little far away from the player spawn point, so the enemy won't immediately chase and attack the player when the game starts.
- On the enemy's GKC_ORK Stats Bridge component, set the Current Health/HP stat Transfer Mode to "Both".

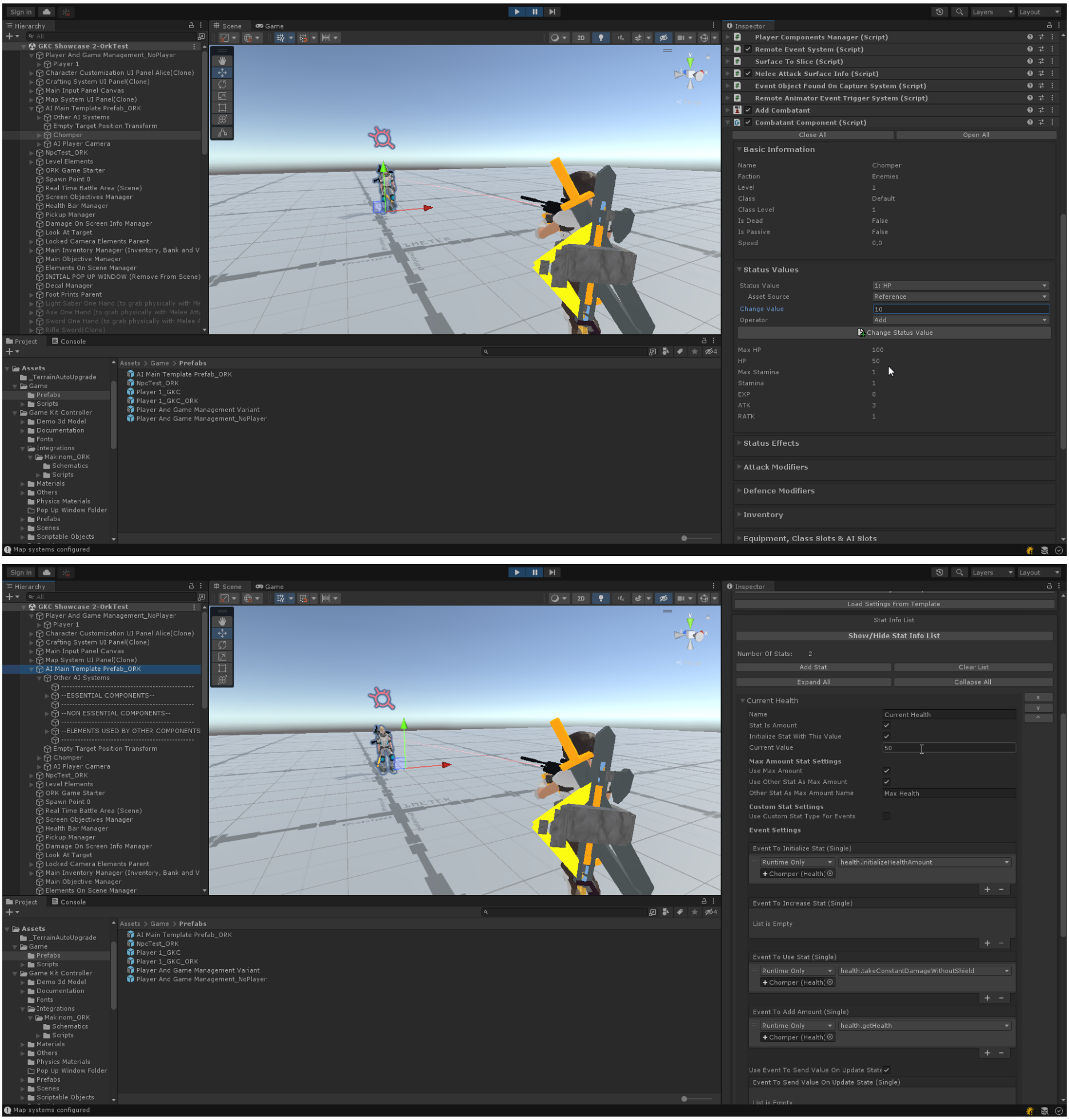


- Save the scene, then press Play to run the game.

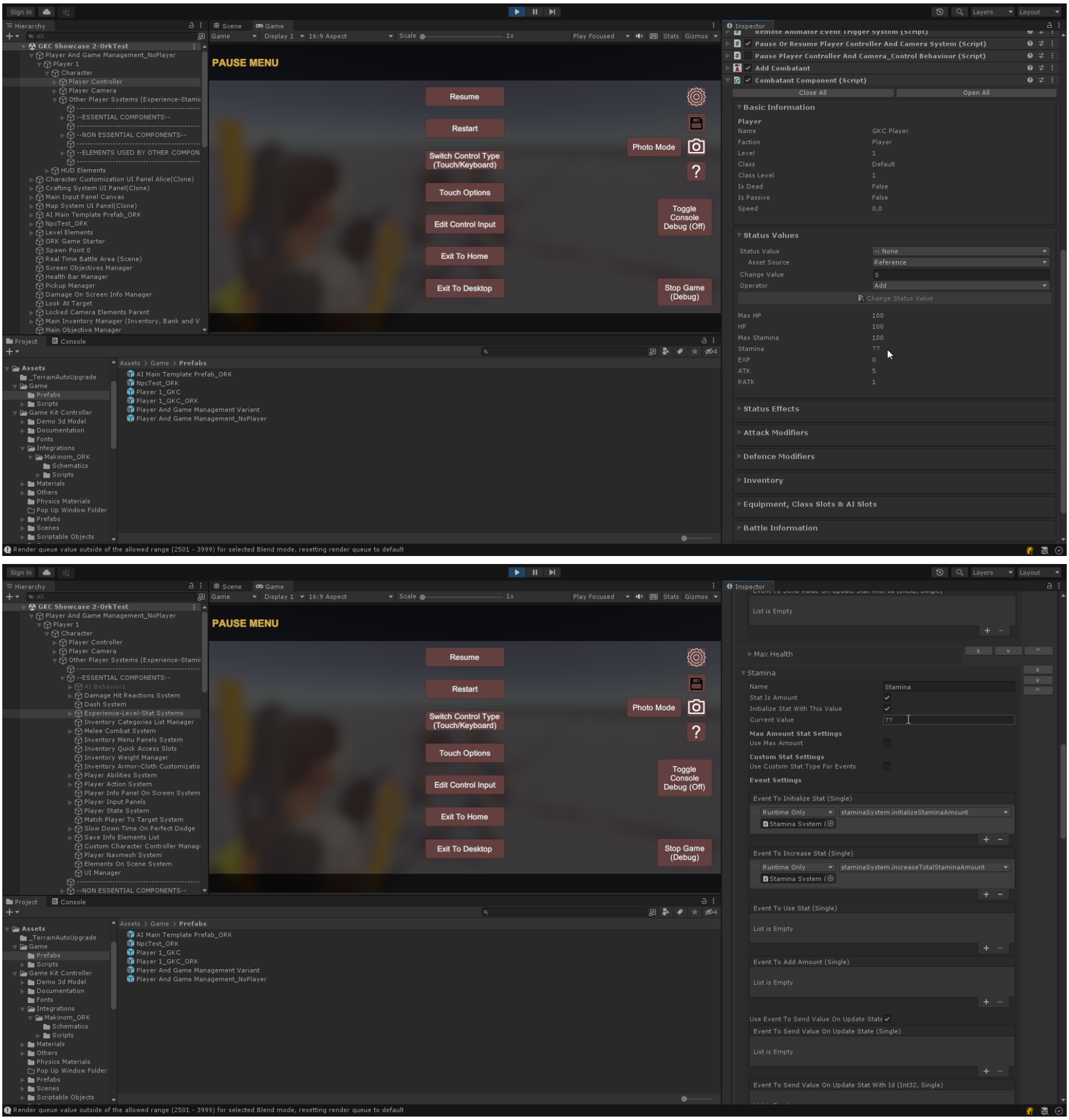
- Run towards the enemy, and shoot him a few times.
  The “Current Health” stat from his playerStatsStstem component will drop its value.
- The updated value is automatically sent from the playerStatsSystem to the enemy’s CombatantComponent’s “HP” stat (Current Health/HP stat Transfer Mode = Both).
- Inspect the enemy’s CombatantComponent to see its current HP stat value.



- Add 10 HP to the enemy, using the CombatantComponent's Status Values options and the "Change Status Value" button.
- The updated value will be automatically sent from the CombatantComponent back to the playerStatsSystem's "Current Health" stat (Current Health/HP stat Transfer Mode = Both).



- When the player attacks using the “left mouse click” input, this spends a bit of the player’s ORK “Stamina” stat (this is the default behavior from the “3D Action RPG” ORK example project).
- The updated value is automatically sent from CombatantComponent to the playerStatSystem’s “Stamina” stat (Stamina/Stamina stat Transfer Mode = ORK to GKC).



- Subtract 10 HP from the player, using the CombatantComponent's Status Values options and the "Change Status Value" button.
- The updated value will be automatically sent from the CombatantComponent to the playerStatsSystem's "Current Health" stat (Current Health/HP stat Transfer Mode = ORK to GKC).