Simulation Game Creator – Unity Asset

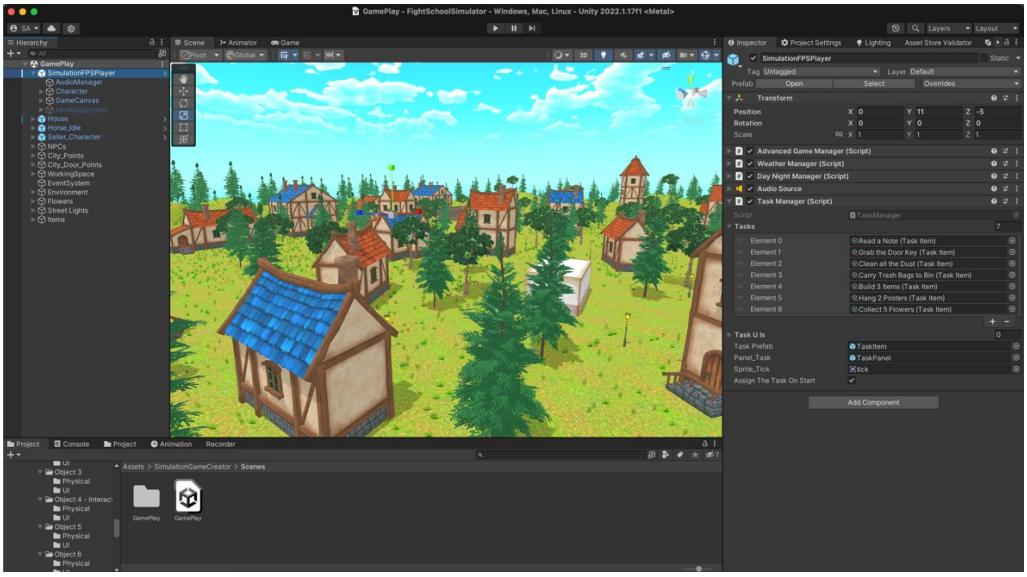
- Documentation -

1. Installation

You can purchase the package from Unity Asset Store and download it by using Package Manager in Unity. After downloading the package, import it into your project and "SimulationGameCreator" folder will appear in your project.

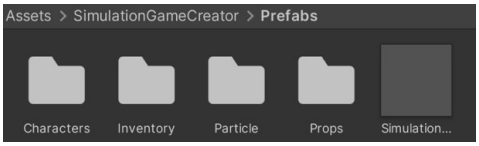
You can find the GamePlay Demo Scene in the Scenes folder. Everything is prepared as Prefab in the Prefabs folder. It is very easy to use it. If you have further questions, you can contact with me via queendeveloper95@gmail.com.

I advice you to explore "**GamePlay**" scene. Run and Play it. While you are playing, you will be able to explore all the components and capabilities of the package and imagine your own game.



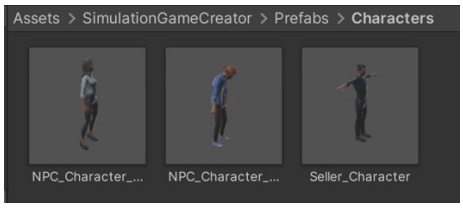
2. Prefabs to Use

The package has got lots of prefabs ready to use. You can find all of them archived carefully in the Assets\ SimulationGameCreator\Prefabs directory. You only need to drag and drop it from directory into your scene or you can find all of them prepared in the GamePlay scene and copy-paste it from there to your scene (That's what I prefer ☺).



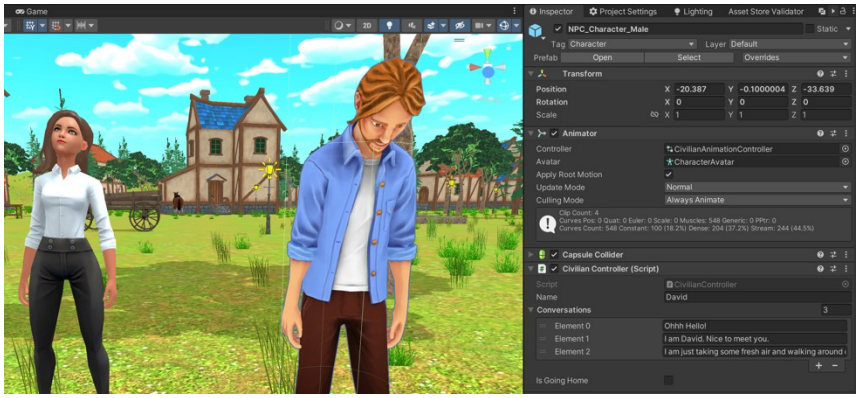
2.1. Prefabs \ Characters

There are three main character prefabs in the package. They are ready to use and they need baked Navmesh in order to move.



How can you bake your nav mesh? It is easy to do. Window > AI > Navigation. Go to Bake tab on the opened window. Click to Bake and Unity will figure out all walkable area on your Scene and bake it. Anyway, let's explore character types:

- **NPC_Character_Female and NPC_Character_Male**: These are our citizens who can wait idle, can walk around and talk with you. They have their own animations. You don't need to do anything. Just spread them to your scene and determine their conversation texts on their inspector. Let's explore their components:



Civilian Controller:

Name: Your NPC's name.
Conversations: They will pick a random conversation from this list and talk with you.

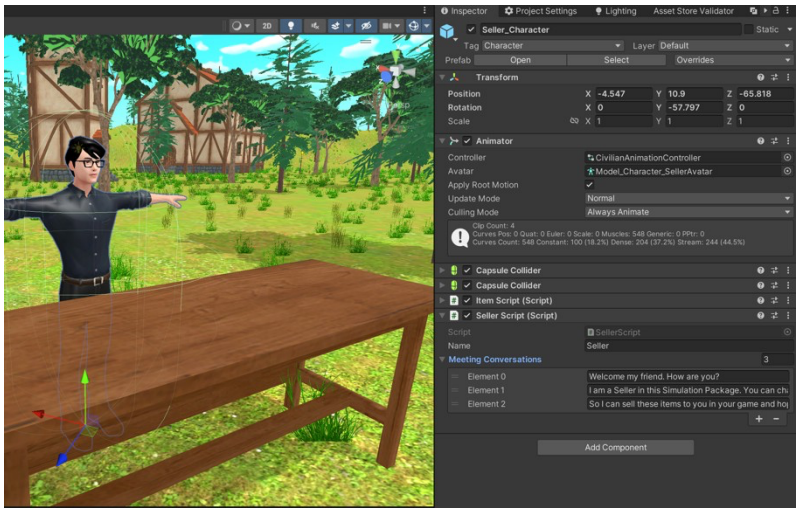
Replacing the NPC Models:

You can replace these NPC models with your own models easily. You can follow these steps:

1. Import your own NPC Model into Unity.
2. Make the model's rig system Humanoid.
3. Make sure it's rig is imported without any warning.
4. Open the NPC prefab.
5. Drag and Drop your own Model into the Prefab. Position it correctly.
6. Change the Avatar of Animator with your new model's avatar.
7. Remove the old model from Prefab and it is ready to go. ☺

We will talk about NPC Manager later in the documentation. You can determine your NPCs' walking targets, what hour they will appear or they will go home. You can even determine their houses' doors. So they will go there when it is getting dark. Later.

- **Seller_Character:** This is our merchant (item seller). You can place them where your market is in your scene. You can talk with him and check his items in order to buy. After your first conversation, he won't have the same conversation again and again with you. Instead of that, he will show his items (by UI) directly next time:



Seller Script:

Name: Your Seller's name.
Conversations: He will pick conversations one by one from this list and talk with you in order.

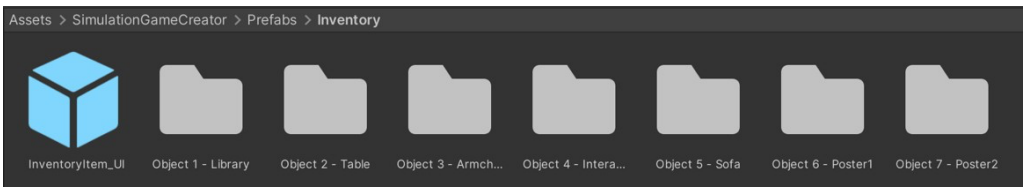
Replacing the Seller Models:

You can replace these Seller model with your own models easily. You can follow these steps:

1. Import your own Seller Model into Unity.
2. Make the model's rig system Humanoid.
3. Make sure it's rig is imported without any warning.
4. Open the Seller_Character prefab.
5. Drag and Drop your own Model into the Prefab. Position it correctly.
6. Change the Avatar of Animator with your new model's avatar.
7. Remove the old model from Prefab and it is ready to go. ☺

2.2. Prefabs \ Inventory

The package has got 7 sample Inventory objects. You can copy them and replace the meshes, the names and properties in order to create your own buyable – buildable – repairable objects in your game.



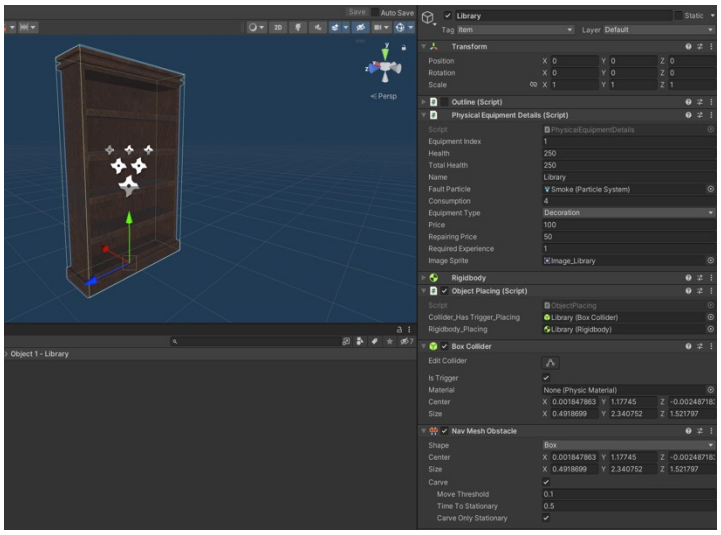
Before learning how to create our own objects, let's explore some of them. What we have:

1. Library
2. Table

3. Armchair
4. Decorable Drawer (This is important one because you can build and also interact with this object).
5. Sofa
6. Poster1
7. Poster2

These are Object prefabs for building, repairing and interacting (It will also appear on Seller's Item List and your Inventory). And we need this object's sprite in order to visualize it in the UI:

We have got couple of components on it and let's talk about all of them one by one:

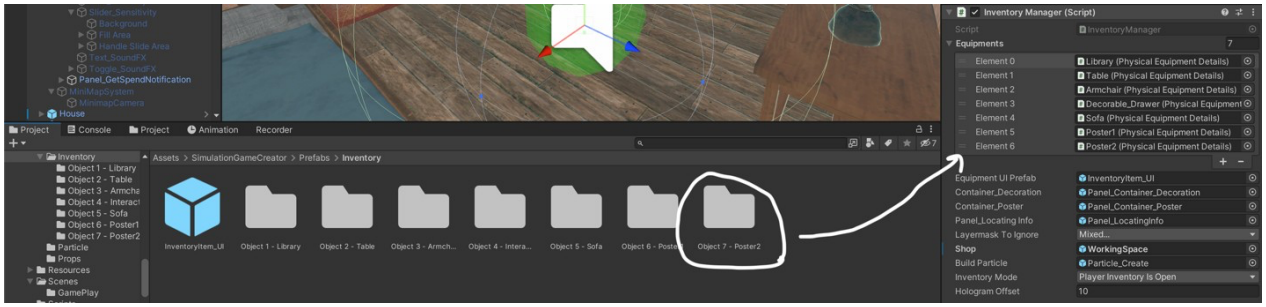


Outline: When you are in front of this object and looking to it (Crosshair), A nice outline around it will appear.
Rigidbody: We need it in order to collide with it.
Object Placing: This component give you ability to move and rotate it while you are building it. It is essential that this script must have Collider and Rigidbody attached.
Box Collider: We need for colliding.
NavMesh Obstacle: After we build this object on our map, we don't want to NPCs walk through it, right? By this component, they will realize it as obstacle and it will walk around it.
And the most importantly, **Physical Equipment Detail** component is the most important one. Let's explore its properties one by one:

- *Equipment Index:* Unique ID of the equipment. It is important for saving the current positions and rotations of the built objects on the scene. So when the player starts to play the game next day, he/she will find all previous built objects in the game.
- *Health:* Health of the Object. If it is less then Total Health, you can repair it. Your game mechanics can use the object by using **UseTheEquipment()** method.
- *Total Health:* Total Health of the Object.
- *Name:* Name of the Object. It is very important to give unique name for your each object.

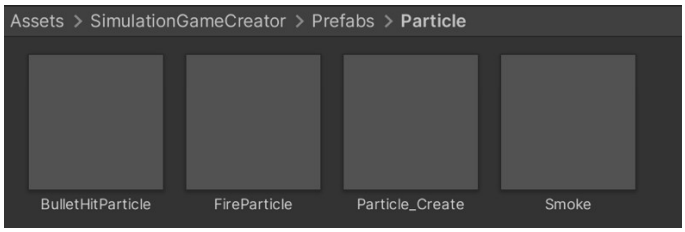
- *Fault Particle:* When Object's health is zero, this fault particle appears in loop (smoke). it means Object needs to be repaired and player will see this fault until it is repaired.
- *Consumption:* How much health will be consumed by each **UseTheEquipment()** method call.
- *Equipment Type:* You can define as Decoration (Can be built on ground) or Poster (Can be built on wall)
- *Price:* The Object's price to buy.
- *Repairing Price:* Repairing price of the Object.
- *Required Experience:* Minimum Experience Level of the Player in order to buy this Object from Seller.
- *Image Sprite:* The image of the Object for Inventory and Seller UI.

When you completed to create a new Inventory Object, please don't forget to add it's prefab to Equipments list in the Inventory Manager (SimulationFPSPlayer > Character object).



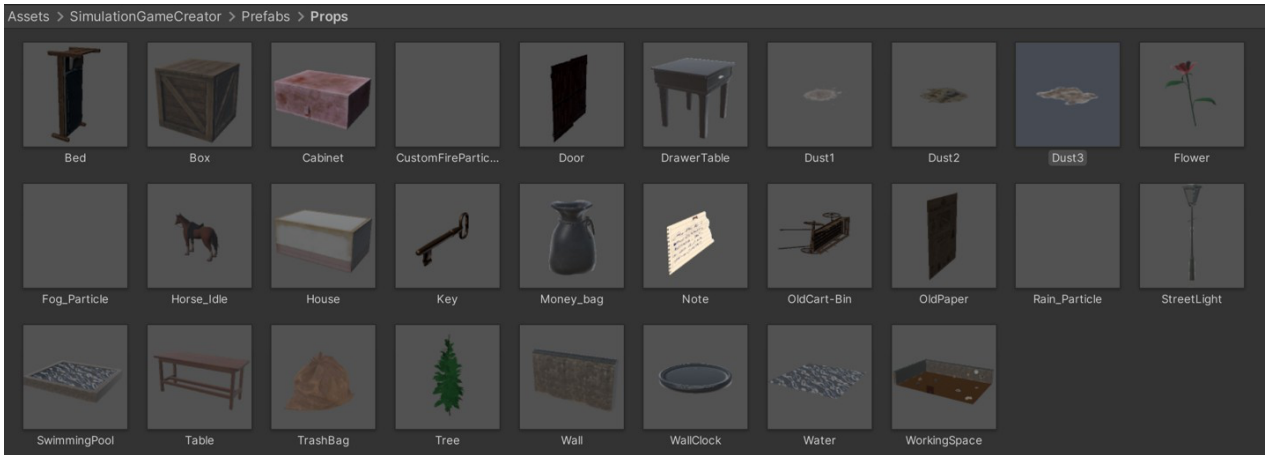
2.3. Prefabs \ Particles

You can find some ready to use particle prefabs here. They are also being used by some of the prefabs (like fault particle or building particle).

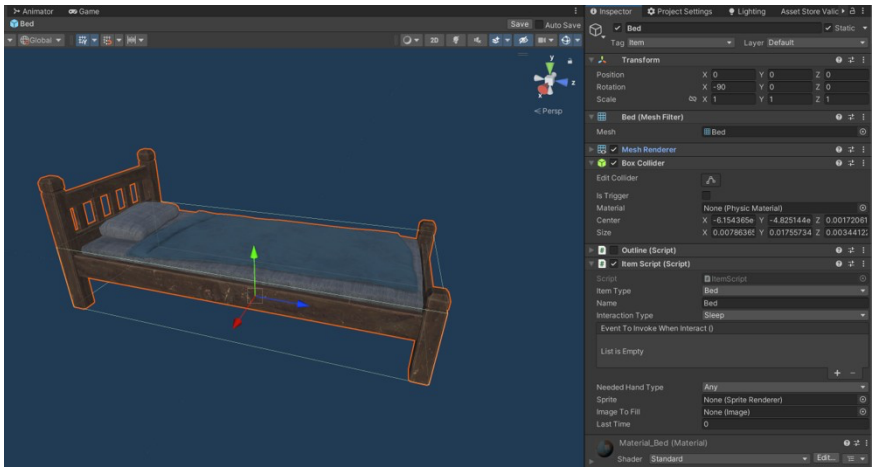


2.4. Prefabs \ Props

These props are ready to use and you can see many of them already in the GamePlay Demo scene. While you are designing your own Scene, you can use them by simply Drag and Drop. Please explore them one by one.



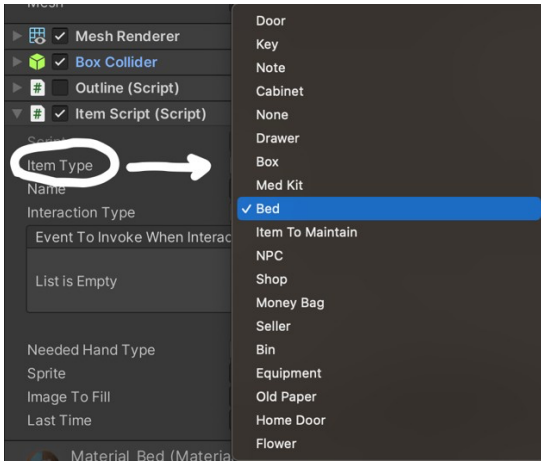
All these prefabs have got **ONE COMMON COMPONENT** which is called **ItemScript**. This script mostly decide how to be interacted by Player. Let's explore one them together:

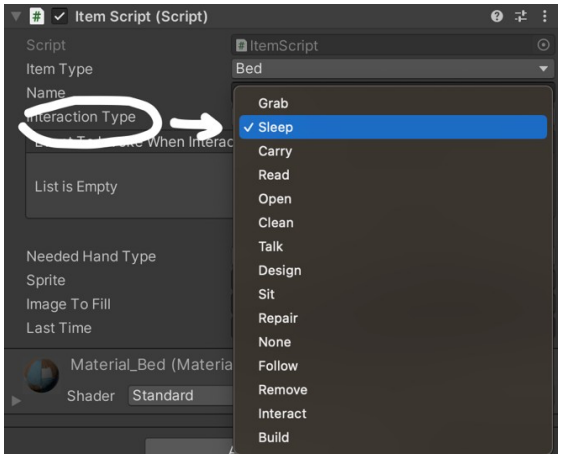


Item Script:

Item Type: You can select the item type or define new ones for your own game.
Interaction Type: How the player will interact with this object and what he/she will see as a text on the scene.

We have got various ready to use Item Types and Interaction Types. They are listed below:

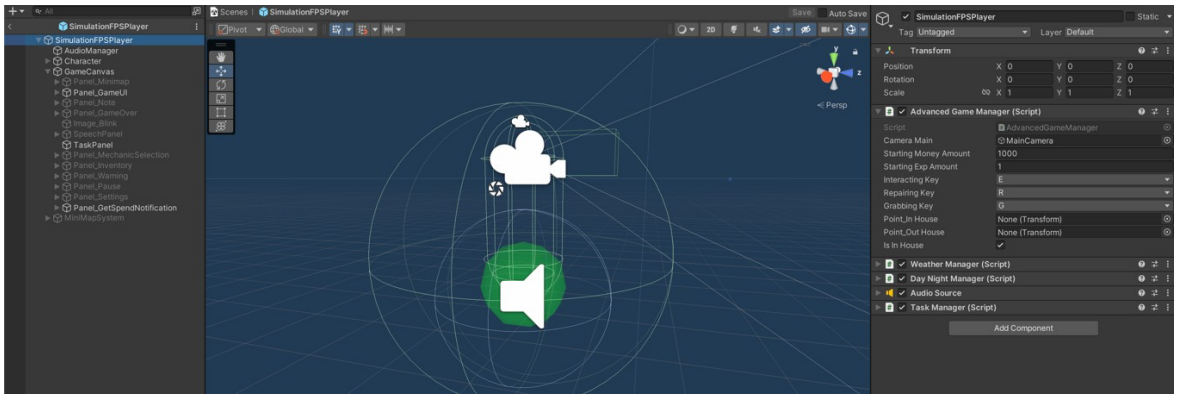




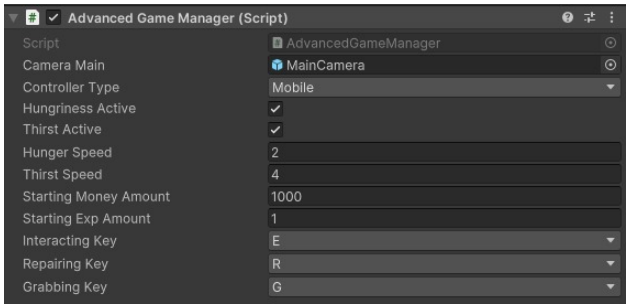
2.5. Prefabs \ SimulationFPSPlayer

Guys! This is the most important prefab in the game. It is the center of everything. This prefab determines the Key Inputs, Manages the Canvas UI Panels, Game Conditions, Weather, Tasks, Day – Night Cycle , Mini Map System.

This prefab manages the audio operations. It contains our FPS Character and all the mechanics like Repairing, Building, Cleaning. Let's check together.

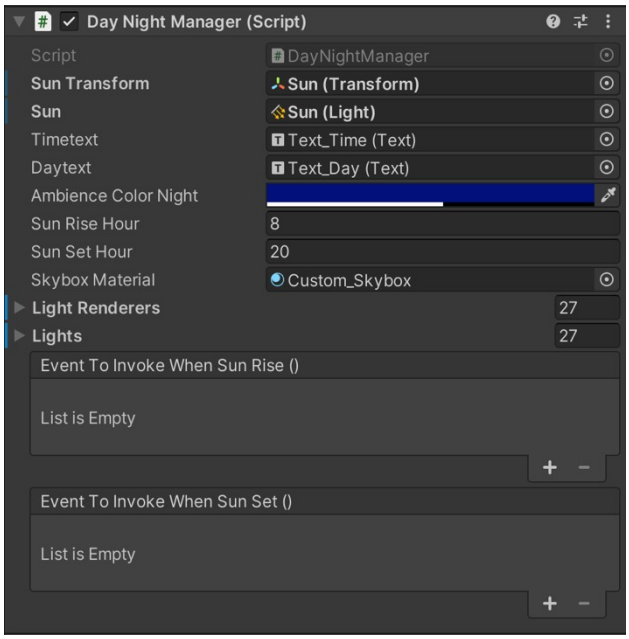


You should always have this prefab in your Scene. We will go through all the components in detailed so you can configure them easily for your own game.

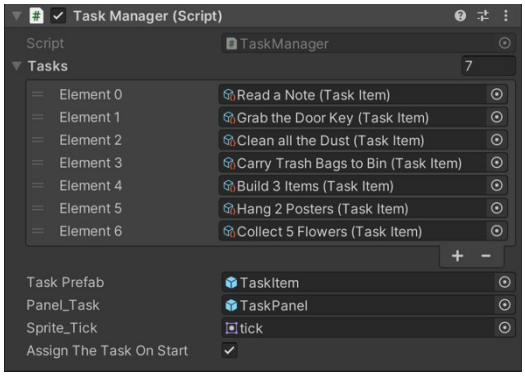


Starting Money Amount: When the player starts the game for the first time, how much money he/she will have.
Interacting Key: Key code for Interaction
Repairing Key: Key code for Repairing objects.
Grabbing Key: Key code for Grabbing objects.
Point_InHouse: The position of the player when he enters his house.
Point_OutHouse: The position of the player when he go outs from his house.
Controller Type: You can select PC or Mobile as controller type. If you select PC, your game will be playable by using Mouse and Keyboard. If you select Mobile, Touch Buttons, Joystick and Touchpad will appear during the game and player will be able to play the game by using these controls.
Hungeriess Active and Thirst Active: You can activate or deactivate these features. If you activate, UI will appear and player needs to eat or drink in order to stay alive in the game. There are also Hunger Speed and Thirst Speed parameters. These parameters are determining how fast the player will get hungry or thirsty in the game.

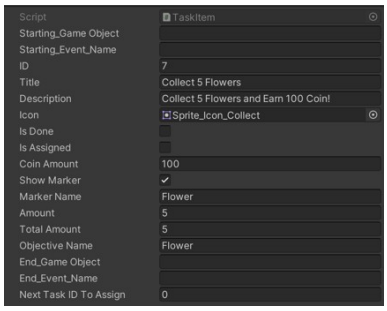
Rain Particle System: Attach your Rain particle object on the Scene.
Weather Change Interval: Every x Period, Weather Manager'll select a condition (Sun or Rain) randomly and executes it.



Sun Transform and Sun: Attach your Directional Light on the Scene to both. Light Intensity will be decreased or increased during Day-Night Cycle to make its dark or light.
TimeText: Text UI for showing the exact time in the game (Exp 14:53)
Daytext: Text UI for showing the current Day in the game (Exp: Day 4)
Ambience Color Night: When it gets dark, the ambience color of the environment.
Sun Rise Hour: What time the sun will rise in the cycle.
Sun Set Hour: What time the sun will set in the cycle.
Skybox Material: You have to assign your Skybox of the Scene in order to decrease or increase the exposure of the skybox for night.
Light Renderers: Attach your street lights' materials.
Lights: Attach your point or spot lights of your scene to this list. When it gets dark, Day Night Manager will light up your lights. When it is day time, Day Night Manager will switch off your lights automatically ☺



Task Manager manages all the tasks you defined in the game. You can create your basic tasks by simple right click to blank area on project tab > Create > ScriptableObject > TaskItem:

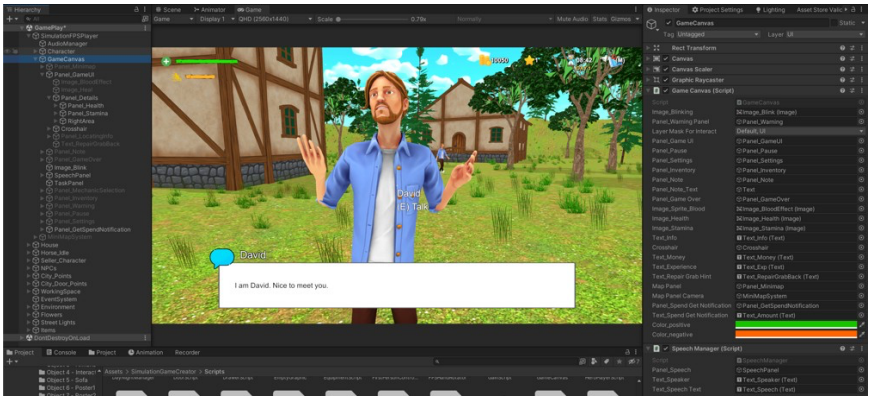


Please Explore the sample tasks. It is easy to create. When you create new tasks, don't forget to assign them to Tasks list on the Task Manager.
- Unique Id is important

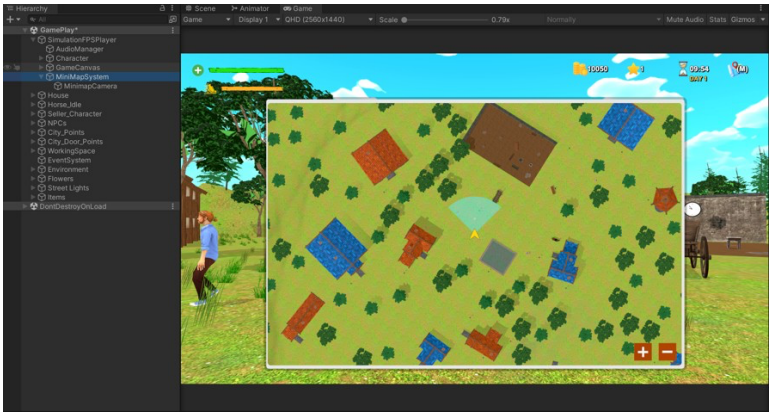
Assign The task on Start: This configuration is important. You can determine the Task Manager will assign the tasks on the start of the scene or you can assign any of them manually by calling **TaskManager.Instance. AssignTask(int taskID)** method from anywhere ☺

AudioManager has got all the required Audioclips and you can call any of them by easily calling **AudioManager.Instance.PlayXXX()** method:

GameCanvas has got Game Canvas Script and it manages all the UI Elements, opening – closing UI Panels, Interaction UI elements, Speech Management, etc. You can explore them easily.



Minimap System is the map of the scene. It follows our player from top. There are two buttons: Zoom In and Zoom Out. The camera's orthographic camera size is being increased or decreased by these buttons. You can change the MiniMap System's following object by simple Selecting MinimapCamera Object > Assign your Player to Player attribute of Minimap Script.



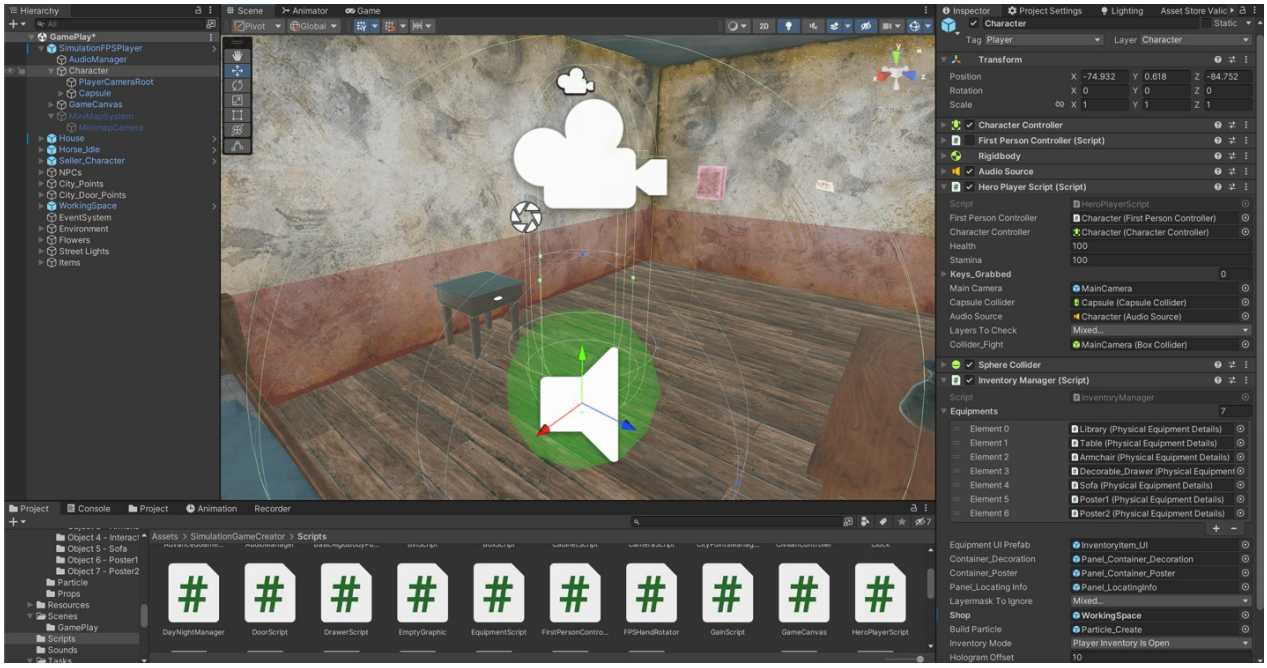
Character Object is our player character. It has got 3 main components. Our main camera is also attached in this object.

1. *First Person Controller Component:* You can configure our character's speed, jump height, sprint speed here.
2. *HeroPlayerScript Component:* You can configure our character's health and stamina here. When you are developing your own game, these helper methods can help you. You can call them from anywhere:
    a. HeroPlayerScript .Instance.GetDamage(int Damage)

b. HeroPlayerScript .Instance.Heal(int Damage)
c. HeroPlayerScript .Instance. ActivatePlayer()
d. HeroPlayerScript .Instance. DeactivatePlayer()

3. *Inventory Manager:* Our Seller Character's Items and our Inventory system is being managed by Inventory Manager. There are 2 important properties to focus here:
   a. Equipments: When you create your Objects to buy, build and repair, please don't forget to add their prefabs here Equipments list.
   b. Shop: When you buy an item from Seller and build it to your Working Space, this Shop transform is being its parent. So, All your built objects are being contained in this Shop transform.

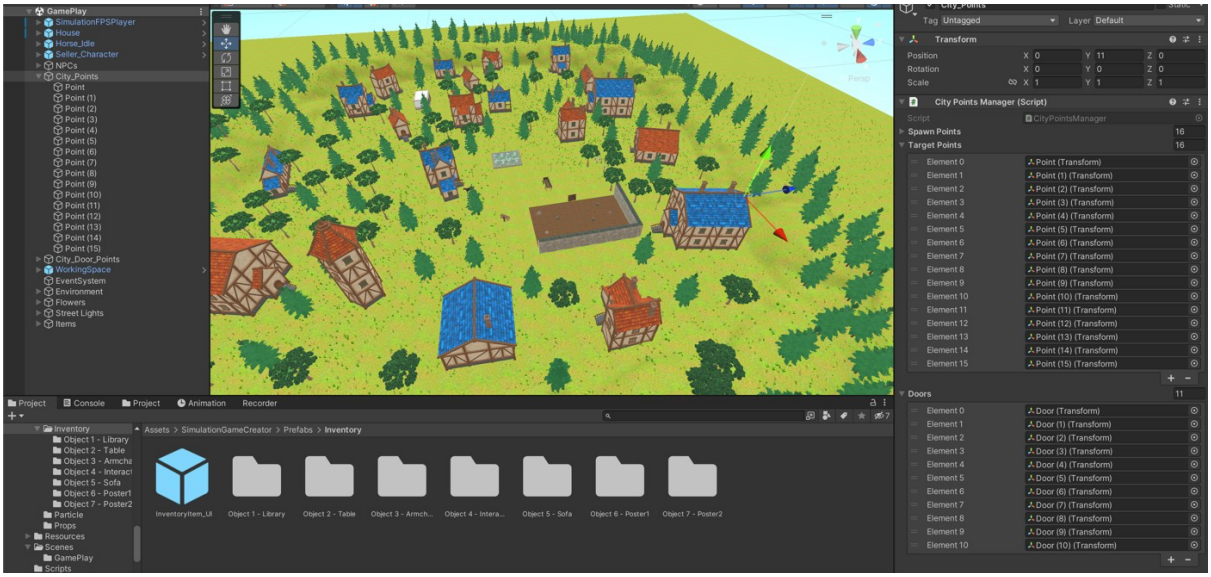
   IMPORTANT NOTE: You should know that you can only decorate the area triggering with "Collider_DecorableArea" object in the Scene. This collider has got "Shop" tag and you can scale, rotate and reposition it anywhere you want.



3. Other Helper Managers:

There are two important Components help to make your town more live and dynamic. These are City Points Manager and NPC Manager.

Let's start with **NPC City Points Manager**. These Manager contains NPCs' Spawn Points when the scene is started, their random Target points to walk and Doors list for going there and disappear when it gets dark. They don't appear until sun rise and when the sun rises, they appear from the same location.



Secondly, we have **NPC Manager** which manages the NPC characters moves and contains all of them as a parent. There are some important configurations you can configure for your own game:

Civilians list contains all the NPC characters on the map. Going out Hour is to determine the Hour when NPC characters will appear (for example 8 am in the morning) and Going Home Hour is to determine the Hour when NPC characters will take a random door as a target and go there to disappear until Going out Hour.

I hope you like my asset and create successful Simulation games with it. ☺
If you need any help or if you have any questions, please don't hesitate to ask me guys. You can send me an email anytime (queendeveloper95@gmail.com)

Kind Regards,
Queen Developer