Colony Simulator

Starter kit of this asset v1.01

Thank you for downloading Colony Simulator!

This document is not a tutorial, but more of a reference of the different parameters and list of available scripts and features. If you are looking for a tutorial, have a look at the Youtube Videos, or join the Discord and ask questions.
If you still need help, you may contact me at: contact@indiemarc.com

Important Concepts:
- After Importing the asset, try the demo scenes in the Scenes folder to understand how the engine works. Everything should work right away.
- The Resource folder contain all the data files for resources, building, crafting and more.
- Prefabs in the Prefabs folder can be dragged n dropped into the scene.
- The UI can be edited by opening the UICanvas prefab in Prefabs folder. Use the alpha parameter on the CanvasGroup components to show or hide a specific panel.
- This asset is Using the new Unity Input System, so make sure this package is also installed.

All the code in this asset has been structured in a way that makes it easy to customize (if you know how to code in C# of course). But the asset can also be used without coding knowledge if you're not planning on adding new features.

All items, constructions and characters are using scriptable object files, so you can add new objects or edit existing ones directly from the Unity editor. Those data files are in the Resources folder.

Colony Simulator is also compatible with other external and optional assets.

Integration with Map and Minimap:
1) Import Colony Simulator first
2) Import Map and Minimaps
3) If there are no compile errors, the "Scripting Define Symbol" MAP_MINIMAP should be added automatically to the player settings. If not add it manually.
4) Make sure the MapManager prefab is in the scene.
5) You may either disable fog in the Resources Settings, or add the FogReveal script to your buildings and characters.
5) Add a MapZone to your scene, and setup the zone to contain the map area, then use MapCapture to generate a template map sprite.
6) Add the MapLevelSettings script to your scene, and set it's properties. Then everything should work!

Scripts

Here is a summary of the main scripts available in the engine. For details on each public parameters, please check directly in the script, next to the variable declaration.

Manager Scripts

TheGame.cs TheControls.cs TheData.cs TheAudio.cs
ColonistManager.cs EventManager.cs TechManager.cs

All managers can be accessed with Get() example TheGame.Get()

TheGame.cs

Core manager of the gameplay, contain function for loading and save the game. Will spawn saved objects when the scene starts, and will make the time progress and manage time speed.

TheControls.cs

Everything related to keyboard controls, or mouse controls are in this script. Controls are never called directly in this engine, they always go through this script.

TheData.cs

Loads all data contained in the Resources folder. Its possible to put a subfolder path if you have other assets that use the Resources folder. Also spawn the UI and the Audio manager.

TheAudio.cs

Use this to play SFX or music. It works by channel so 2 sfx in different channel will play at the same time, but sfx in the same channel will replace each other. This prevent sounds accumulation if many actions happen at the same time, it also allow to control volume globally.

ColonistManager.cs

Manage the schedule of colonists, will send them to work automatically to the highest priority work location that is unassigned. And will make the colonist start/stop working when a condition is met.

EventManager.cs

Manages events and triggers them. Events are things that can happen during the game and affect the gameplay. There is also an Event Dialogue Box with text can be displayed to the player. See later for more info on the event system.

TechManager.cs

The tech manager is responsible for upgrading techs that are being researched, it has useful functions to know if a tech is completed or uncompleted, and to pay cost for tech or check their requirements.

Gameplay Scripts

Selectable.cs
One of the most basic components added to almost all interactable objects. Allow the object to be selected and be interacted with. This script is used by almost all other gameplay scripts.

UniqueID.cs
Important script that allow to generate a unique ID linked to the object. This ID is used by the save system. This script is used by almost all other gameplay scripts. You can generate a new random ID by clicking the button, or you can generate all IDS in the scene by going to ColonySimulator -> Generate IDS

Destructible.cs
Anything that can be attacked and destroyed. It has hp, armor, regen speed, and can be killed.

Character.cs
Anything that can move, attack, and execute actions, including Colonists.

CharacterCombat.cs
Allow a character to attack other characters and destructibles.

CharacterDestructible.cs
Updated version of the Destructible script, adapted for characters and colonists.

Colonist.cs
Colonists are characters controlled by the player, and by the ColonistManager, they can be assigned to actions automatically or manually.

ColonistAttribute.cs
Allow a colonist to have attributes such as health, energy, hunger. You can change which attributes you want to use, but I recommend to keep the "health" attribute if you use CharacterCombat or CharacterDestructible on the character.

NPC.cs
Use for any kind of non-player character, enemies or allies. NPC will save the current action of the character, and also allow spawning new characters with the Create method.

Item.cs
Items are resources that can be picked directly without spending time to harvest it.

Gatherable.cs
Gatherables are resources that can be harvested, such as a tree or a rock. Colonist will spend some time at this resources to collect it, before bringing it back to the base.

Buildable.cs
An object that can be placed on the map manually by the player, selecting the position with the mouse. It will check for valid terrain and obstacles before allowing to be placed.

Construction.cs
Constructions are objects that can be built. Once placed with the Buildable script, Colonist need to go to the construction and build it with resources. It has a build progress and will swap mesh depending off the progress (under construction or completed).

House.cs
A building that increases the population limit.

Factory.cs
A building that produces things (items, characters, tools). Some factory required colonists to be assigned to the building to work.

Storage.cs
A building were items can be dropped, to be placed into the global inventory.

Inventory.cs
Allow the object to hold items. If the global toggle is checked, will link to the global inventory (which is the one displayed in the UI and accessible from anywhere). For example, a storage should have the global set to on. But a character would have it to off, because their inventory is not shared with the global one.

Enemy.cs
Controls enemy behaviors like where they should move to and what they should attack.

Trader.cs
A NPC you can trade with. Add items as "starting items" in their inventory to add items to their list.

Save Data Scripts

SaveData.cs
This is the main data script that contains all the saved data. There is only one instance of this and it will be serialized when saving. It contains many useful functions to save, load, create a new game, or edit or access data.

Classes starting with "Save____" are usually objects contained inside SaveData.

Data Scripts

Data script are scriptable objects that define the structure of the data. A scriptable object data file can be created by right click in the Project files tab: Create->ColonySimulator->
These data file should be placed in the Resources folder so that TheData.cs load them automatically at the start of each scene. Most data files are linked to a prefab that will be spawned when this object is created.

AssetData.cs and GameData.cs

These contain generic settings for the whole game. There should be only 1 of each.

CSData.cs

Parent class for all Colonist Simulator data.

CraftData.cs

Anything that can be crafted, parent class. Inherited by ColonistData, ItemData, and ConstructionData.

SpawnData.cs

Object that can be spawned, and saved in the save file, but that is not a CraftData.

ItemData.cs

Data for Items

ColonistData.cs

Data for colonists. Multiple Colonists can share the same prefab.

ConstructionData.cs

Data for constructions.

TechData.cs

Data for technologies.

GroupData.cs

Groups do not do anything on their own, but they can be used to classify Selectables, or CraftData. And some actions will use groups to filter objects. (Example if you want to loop on all Items that are Food).

Action Scripts

Actions are a powerful tool that allow you to define your own character interactions with objects in the game. Actions are also scriptable objects, but they contain a code logic for different types of actions that can be performed. There are Actions and Works which are a bit different. There are many different types of actions but they all inherit from the same two scripts.

To create a custom Action, inherit a new script from one of these two basic classes, and override the main functions. More is explained in those specific scripts.

ActionBasic.cs

Actions are specific commands that will be executed by a character. For example, eating, gathering, building, sleeping. They usually target only 1 object and the action is performed by the character on that object (Selectable) when the character is within the use_range.

WorkBasic.cs

Works can only be performed by colonist, and can contain a series of automated actions (for example: move to tree, harvest tree, bring it back to base, then find the next tree to harvest, etc). They are more complex than actions because they can be executed on multiple targets. Works are the things that are used by the ColonistManager to auto assign work to Colonists.

Once you created a new script action, you must create a data file for this action by right clicking in the project window: Create->Data->Actions. OR you can also duplicate an existing action if you want a similar behavior with different properties.

You can take a look at all the existing actions included in this demo in the Resources/Actions folder. And also, the Scripts/Actions folder for their scripts.

Improving Colony Simulator

If you think that something important is missing from this document, or if you notice a bug. Feel free to contact me and I will take note of every request. The most popular ones will be added to the feature list for the next version. Mistakes will be corrected.

If you have any questions or suggestions send me an email:
contact@indiemarc.com

Or join Discord:
https://discord.gg/JpVwUgG

Thank you!

Credits

Programming and Unity integration
Indie Marc (Marc-Antoine Desbiens)
https://indiemarc.com

3D Art
Alejandra "Le_Pulgue" Pino (https://www.instagram.com/le_pulgue/)

Animation
Naufalrezki (https://www.artstation.com/naufalrezki30)

2D UI and Icons
Helen Lien (http://helenlien.com)