Survial Engine

Survival kit of this asset v2.00

Thank you for downloading Survival Engine!

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

Youtube: https://www.youtube.com/channel/UC0LYig0AgPT9T5IN5DCUiqQ
Discord: https://discord.gg/JpVwUqG

Important Concepts:
- To get started, take a look at the demo scenes in the Scenes to understand how the engine works.

- The Resource folder contains all the data files: items, constructions and plants, and are using scriptable object files, so you can add new objects or edit existing ones directly from the Unity editor.

- Prefabs in the Prefabs folder can be dragged n dropped into the scene for testing.

- 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.

- The Upgrades folder contains extra packages based on your project type, for example there is a package for the New Input System and another for WebGL builds.

Scripts

Here are the most important scripts in this package.

For more details on the scripts public parameters, open that specific script, there should be comments next to each variable.

PlayerCharacter.cs

Probably the most important script, controls the main character and allow it to interact with everything else. (Moving, Crafting, Attacking, Taking items, …).

PlayerControls.cs and PlayerControlsMouse.cs

Manager script that handle all player controls. Part of the Managers prefabs that should be included in every scene. PlayerControls is for keyboard/controller controls, while PlayerControlsMouse if for mouse and touch (mobile). Each script has events and functions that allow you to know what the player is pressing or clicking. For example, the onClickObject event will be fired every time the player clicks on an object, passing the clicked object as a parameter.

TheGame.cs, TheRender.cs, TheAudio.cs

Managers scripts that are part of the Managers prefab (include it in every scene). TheGame handles the generic events of the game, like loading the UI, detecting if the main player is dead, or spawning objects after a save file is loaded.
TheRender is an optimization script, and will disable any object (Selectable) that are too far from the player and camera. It is not necessary but strongly recommended for large maps.
TheAudio manages all sound effects and music, its two main functions are PlaySFX() and PlayMusic().

Selectable.cs

Most objects that are placed on the map are Selectables. These are everything the player can click on and interact with. You can attach actions or other scripts to it like Destructible, Item, Construction, Plant, Animal. (These all require the Selectable script).

Destructible.cs

Destructibles are anything that can be destroyed by the player. They have HP and can be attacked. Resources that can be gathered are destructibles, as well as animals that can be attacked. All destructibles also need to have the Selectable script.

Buildable.cs

Buildable is a new script introduced in 1.06, that allow player to create objects and place them on the map manually. This was previously done by Construction.cs, but since plants can now also be placed in the same way, a separate script is now doing this. Both plants and constructions require this script.

Item.cs, Construction.cs, Plant.cs and Character.cs

These are directly linked to their data section. Except that these ones are MonoBehavior and added to the prefab that is put on the map. While the ones ending with 'Data' are scriptable objects and just contains the data (but not attached to a prefab). Items are objects that you can add to your inventory or equip. Constructions are objects that you can build and place on the map. Plants are objects that can spawn and grow on the map, some of them also grow fruits/vegetables.

Character.cs

Generic character script that can be used for NPC, pets, etc. Give it orders by calling the functions like MoveTo(). Compatible with the navmesh.

CraftData.cs, ConstructionData.cs, PlantData.cs, ItemData.cs, AttributeData.cs

Scriptable objects that contain all the stats and data of items, constructions and plants. CraftData is the parent object and represent things that can appear in the crafting panel. ItemData, PlantData, and ConstructionData inherit from CraftData. See the data section for a better understanding of the properties of these objects.

UniqueID.cs

Unique ID of the object, used by the save system. Each instance of an object that have properties saved in the save file should have a Unique ID. Without a unique ID, nothing can be saved about this object. Add only to things that need to be saved.

PlayerData.cs

Basically, the save file. All variables inside this script will be serialized to a save file when the game is saved. It also contains multiple function that allow you to easily access or update the data contained inside the save file. Make sure to call Save() to save the game. (There's no autosave by default).

Data (Scriptable Objects)

Scriptable objects are the files that contain all the data about items, buildings, plants, and characters. Each object is represented by a file with all it's properties. Take a look inside the Resources folder to see all the data files. The files are located in that folder so that the game can load them automatically. To create a new data asset, right click in the Project Files window and select Create->Data. You can also duplicate an existing one with Ctrl-D.

GameData

Contains generic data about the game. There should be only 1 of this. In the demo, that file is located inside. It is referenced by the Managers prefab.

GroupData

Groups don't do anything by themselves, but they are useful for grouping objects and reference them by groups (instead of one by one). For example, you could have an action that can only be performed on a specific 'group' of objects. Groups can have a title and icon to represent them in the game.

CraftData

Parent script of ItemData, ConstructionData and PlantData. Does not do anything by itself, but contain the common data that are in all other data types.

ItemData

An item is an object you can hold in your inventory. Items can be crafted as they inherit from CraftData.

ConstructionData

A construction is an object that can be built and placed on the map by the player.

PlantData

A plant is an object that can be placed on the map (Usually with a seed). It will grow into different stages. The final stage can often produce items (fruits or veggies).

CharacterData

Any object that can move, its position will be saved in the save file.

AttributeData

Attributes represent the life meters of the player. In order to survive, the player must keep its attribute in check. The health attribute will cause the player to die when it reaches 0.

Actions

Actions are a powerful tool that allow you to define your own player interactions with objects in the game. Some Selectable or Items will have a list of possible actions you can do when you click on them. You can code a script to create your own action by inheriting one of the three base action scripts: SAction, MAction or AAction.

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. For example, you could duplicate the action 'Fill Water' and make a 'Fill Juice' action if you had a juice source in the game. In that example, they would both use the same action script but would have 2 different action files.

After you create your own action data file, reference it on a Selectable or ItemData. Actions put on a selectable will appear when the player click on that object in the scene. Actions put on an ItemData will appear when the player right-click on that item in their inventory.

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.

Creating your own actions scripts

If you are up to the challenge to create your own action scripts, there are 2 important functions you will need to override: DoAction and CanDoAction
DoAction is the code that is ran when the action is performed.
CanDoAction returns a bool that tells if the action can be performed or not, for example it may check if you have the required items to perform it. If the action has no conditions you may just return true.

Actions all need to inherit from one of the three base action scripts:

SAction: Are the basic 'Standard' actions. When the player clicks on an object with such actions, a UI list will popup and the player can select which action to perform. The action will be performed when the player selects one of them. If applied to an Item, the UI will popup when the player right clicks on that item in their inventory instead.

AAction: Are 'Automatic' actions. They are performed automatically when the player clicks on the Selectable in the scene and if the action conditions are respected. You can only have 1 action of this type on any given object.

MAction: Are 'Merge' or 'Mix' actions. They are performed by combining two items (example: axe and coconut to open it), or by combining one item with a selectable (example: jug and water source to fill it, or raw meat and fire to cook it).

Improving the Engine

If you think that something important is missing 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.

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

Or discuss on the forum:
https://forum.unity.com/threads/released-survival-engine-complete-project-template-for-survival-games.972804/

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

Thank you!

Credits

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

3D Art
Antonio Maričić (https://www.instagram.com/mocacinodesign/)
Marwan Hany (https://www.fiverr.com/marwan_hany)
Pogwitch (https://www.fiverr.com/pogwitch)

Animation
Naufalrezki (https://www.fiverr.com/naufalrezki)
Hamzasketches (https://www.fiverr.com/hamzasketches)

2D UI
Helen Lien (http://helenlien.com)
Ayes Studios (https://www.fiverr.com/ayesstudios)

SFX
Andey Fellowes (https://soundcloud.com/andeyfellowescomposer)