Creating your own game

This page explains how you can build upon the TopDown Engine to create your own game, and how to update safely.

Introduction

The TopDown Engine has been built with **extendability** in mind from the start. Of course, you can simply use the existing classes as they are, but you can also create new ones or build on what's there. This page covers a few of the ways you can add content and functionalities to create **your very own game**.

Visuals and sounds

A very simple way to go is to just **replace visual assets** with your own. Doing so is pretty easy, as all you have to do is locate the file and replace it with your creations. There's not much else to say about that. Unity good practices apply (you can divert from that of course, but if you want some reference you can use these). You should have a look at these guidelines that should ensure your assets work as intended on all platforms and that your project remains maintainable.

If you're having trouble figuring out **where a particular sprite is located** in the demo files, you can simply select the prefab in the scene view, then click on its sprite renderer's Sprite field. This will highlight the sprite in the hierarchy view. Then you can simply right click the sprite in the hierarchy view, and click on "reveal in Finder" (or show in Explorer on Windows). All you have to do now is replace this file with your own, and press play again.

Tweaking existing components

All the components in the asset have been built with **lots of use cases** in mind. From the same component you can usually, via their inspector, tweak values and select different behaviours that in the end will make your game really stand out. Before starting to code something new, maybe have a look at the existing similar classes if there are any, check out their API documentation and make sure you can't just get what you want via a few changes of settings.

Adding new classes

Maybe adding simply visuals won't be enough for you to create that game you want. Sometimes you'll find that you need **new code** to give life to that idea you have.

One way to go is to simply **create new classes**. Maybe you want an end credits scene, that would showcase everyone who's worked on that project of yours? There isn't such a component in the engine right now, and there isn't anything remotely close to it in the engine. That's typically the kind of situation where you'll want to create your own class. It's unrelated to anything, and you're free to do whatever you want. From your own classes you'll still be able to access the rest of the engine, like LevelManager or GameManager for example, get references to the current Character, etc.

Assembly Definitions

As is good practice, the engine uses assembly defintions. This separates code nicely, and improves compile times. But that also means that the engine scripts won't "see" any script outside of their assembly definitions. That's why it's encouraged to extend scripts instead of modifying them directly. If you really want to modify core classes instead of extending them, and need access to external APIs, you can either edit the assembly definitions to add access to them (dirty), or remove all assembly definitions (very dirty).

Folder Structure

Inside your project, you can put the TopDown Engine folder anywhere you want, it doesn't expect to be at a specific place. That said, it's good practice to not put your own content inside the TopDown Engine folder, as you may have to delete it entirely if you update it, for example. Instead, it's best to have a folder structure like this :

Assets/

- Game/ *where you'll put all your assets, scripts, etc.*
- ThirdParty/ *a folder where you'll put all third party plugins/frameworks/assets you plan on using*
-
    - TopDown Engine/ *the entire TDE folder*
-
    - Other Plugin/

If you decide to extend classes of the engine, they'll go into your Game folder (or likely a Game/Scripts/TDE/ subfolder). With such a structure, it becomes easy to update the engine, and you can organize your Game folder and subfolders any way you want, without risking to break anything in the TopDown Engine or your other plugins.

Inheritance

And sometimes you'll find that you'll want some of the existing classes to behave a bit differently. Maybe that walljump ability doesn't work *exactly* like you'd want it too. As much as I'd love for the engine to match all possible gameplay styles, that's just not possible, and sometimes choices have to be made. But the good thing is, that's been taken into account as well!

Most, if not all, of the methods in the Engine are marked as virtual, which means you can easily override them to suit your own needs. I strongly suggest using inheritance and polymorphism to reuse and extend as much code as possible.

In any case, I strongly recommend not modifying any of these, but rather **extend the engine's scripts into your own classes**. That way you won't have your work erased when updating to the next version.