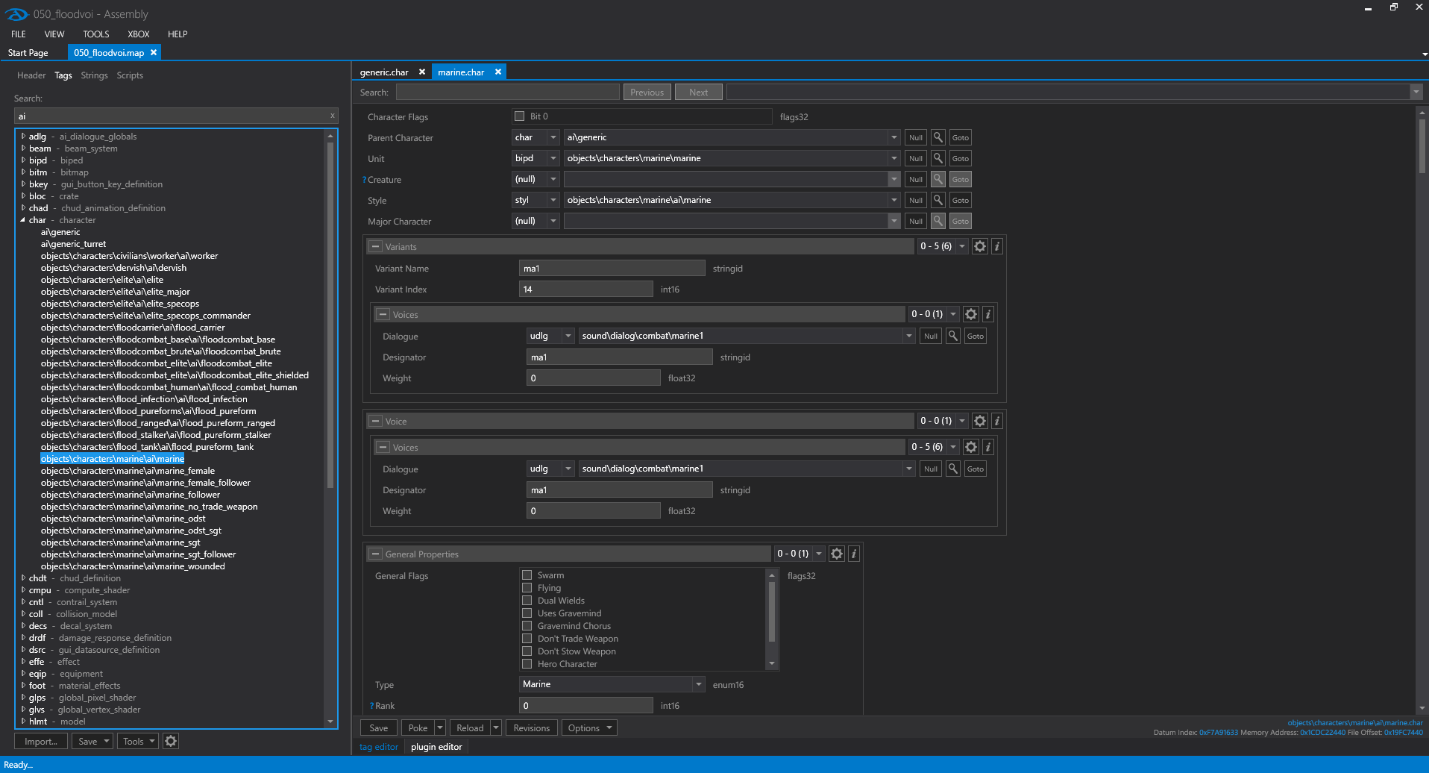
Game Engines Bonus Blog

Halo 3 is an iconic 1st shooter released on the Xbox 360 back in 2007. It made use of a variety of different enemy types and variants. I enjoyed playing the game on the Xbox One through the Master Chief Collection, and now I play it through PC. The mod support helps keep the game alive for me.

Since I focus on the campaign modding side of things, I want to focus on cool tricks Bungie used when developing these levels. Each level in the game contains a Generic AI character that every other character is a child of. They all inherit from the Generic AI and override certain parts of it to give each in-game character unique behaviour. I believe this to be an example of factory. The generic AI class is never used itself but lends itself to be the base of every other ai that is then used from it.



This is a close-up of all the character types, that inherit from ai\generic:

Text

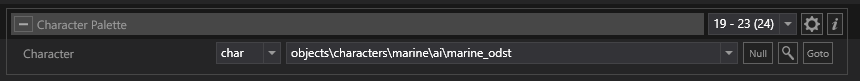
Description automatically generated

Then, the ai\generic is diversified through different data sections being overwritten, like variants, the character biped being used, ai style etc.

A screenshot of a computer

Description automatically generated with medium confidence

They are then labelled into an array to be used later.



After setting the characters up, you just spawn them into the game through the games scenario tag:

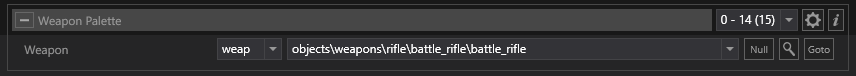
A screenshot of a computer

Description automatically generated with medium confidence

In this case we are spawning 2 squads in the Arbiter squad, 5 ODSTs and 1 Arbiter.

Another design pattern I picked up on was singletons. The original game had collectable skulls that modify the game. When you launch the game up, all the skulls are toggled off. If you collected a skull, it would remain active until the game is closed, in the case of the original Xbox and Halo 2, you would have to turn off the console to reset the skulls. This instance (the skull manager) is only created once and is active throughout the entire game’s runtime.

The last design pattern would be prototyping. The game creates each weapon upon loading the level, depending on each section, different weapons are being loaded in. Each battle rifle and assault rifle, SMG, pistol rocket launcher etc. are identical to each of their counterparts. They are all being cloned from the base prototype weapon. They are loaded in based off the scenario tag and weapon palette.



For example, this is an ODST captain being spawned in with a battle Rifle.

A screenshot of a computer

Description automatically generated with medium confidence

For part 2 I used a prototype design pattern, I create identical swords and bows that are outputted in the console.

UMLs are included as separate .png files.