Legal

The *Buff Builder IDE* is the work of Michael Greene (a.k.a. Volt Cruelerz) and is released under the MIT license.

History

For years, Volt complained about the lack of a better way to mod buff files. During the summer of 2014, his supervisor at his internship suggested that he take some time to study GUIs for use later in the summer. In response, Volt began to work on that which he had desired for quite some time as a means of practice. Over the next month, he put together the first IDE for modding Sins of a Solar Empire during his free time.

Summary

Its purpose is to enable editing of buffs without compelling the user to ever touch the ability or buff files themselves. While this may seem like a small difference, it is in fact quite useful as the entity files are required to follow a very particular format that humans are not terribly good at remembering.

Buff Builder is an application written in Java that enables the user to bypass all those inconveniences, focusing only on what the user wishes the buff chain to do rather than on what the game needs the format to be. While this is of minimal use for things such as simple balance mods that only tweak numbers and do not alter the flow of buff chains, for heavier modding where players are adding/removing InstantActions and other Pluggables, the value increases dramatically.

Definitions

Buff Chain: All buff chains originate with an Ability entity file and terminate in one or more Buff files. The sum total chain of all interacting Ability and Buff files forms the Buff Chain.

Pluggables: InstantActions, PeriodicActions, OverTimeActions, and the like. These are sub-blocks of code that plug into larger blocks. They cannot exist on their own, but they do a lot of the work in a Buff Chain.

Workspace: where you will actually be doing the work of editing entities

Toolbox: contains buttons that you can use to add new items to your Workspace

Menubar: contains several menus.

Panel: all entities and pluggables are represented by Panels in the workspace. Panels can be dragged around, minimized, and closed. Panels represent their properties in an editable list.

Invoking Line: line drawn from the invoker Panel to the invokee Panel. For example, an Ability that applies a Buff would have a line drawn from the Ability to the InstantAction and another from the InstantAction to the Buff. An Invoking Line will match the color of the invokee.

Nullbox: this is a small box in the upper-left-hand corner of the Workspace. Any Panel that is not invoked by another Panel will have its Invoking Line drawn from the Nullbox

Installation

To install the program, simply download deploy.zip and dump the contents alongside your GameInfo folder.

Usage

To run, simply double-click on the .jar file. If you do not have the Java Runtime Environment, you should install that first. The first thing you'll notice is the large central area. This is your Workspace. To the left, you'll see buttons in your Toolbox that allow you to add new Entities. At the top, you'll see your Menubar where you can access assorted features.

To begin, simply create an Ability via the button in the Toolbox. You will be given the option to name it. Please note that the name should not be prefixed with the term "Ability" as the software will automatically prepend that during exports.

You will now see a basic ability before you. Click on the white button to minimize or restore Panels. Click on the black button to delete Panels from the Workspace.

To edit a property, simply double click on it. Some properties cannot be changed while others have limited options. Boolean properties must either be *TRUE* or *FALSE*. Some Properties require a Buff that you have already created (if you have not yet created the Buff you specify when setting the value of the property, the software will prompt you to create it). Others have a finite list of options. In such a case, double clicking will bring up a list of all options with a number before each option. If option 3 were listed as "[3] Demo Option," you would enter the number 3 into the blank and submit that.

In many cases regarding properties with finite lists, one should attempt to fill out the properties from top to bottom as changing a higher property with a finite list can sometimes change the properties that are displayed below. As an example, only setting an ability to have an antimatter cost will display the fields in which you enter the amount of antimatter to charge.

The boxes running down the right side of a Panel are always in the same order: InstantAction, PeriodicAction, OverTimeAction, EntityModifier, EntityBoolModifier, FinishCondition and mirror the order in which they appear in entity files. Double clicking on one of these will create a new Pluggable of the corresponding type. Abilities and PeriodicActions only have the InstantAction box while Buffs have the full list.

Input/Output

Save/Load: data from Buff Builder is stored in Buff Chain Project (.bcp) files. These files store all components of a buff chain as well some additional data about the Workspace.

Import/Export

Because the game expects .entity files and not .bcp files, the software must export the data into a form that the game is comfortable with. During the export process, the user will be queried for the buff chain name. If the user were to respond with “Demo” while only exporting an Ability named “Do400Damage,” the resulting entity file would be named “AbilityDemoDo400Damage.entity.”

Because bcp files store more information than entity files, during imports, it is possible to lose the buff chain name. For this reason, it is possible to set the buff chain name during export to an empty string.