Architecture Overview

The aim of the current architecture is to allow as much of the code as possible to be written in a fully functional language, in this case C++. The removal of the large majority of Papyrus script is done to enable much more control over the system. Only things that directly influence the game are left in papyrus script. We would recommend the continuation of this design choice.

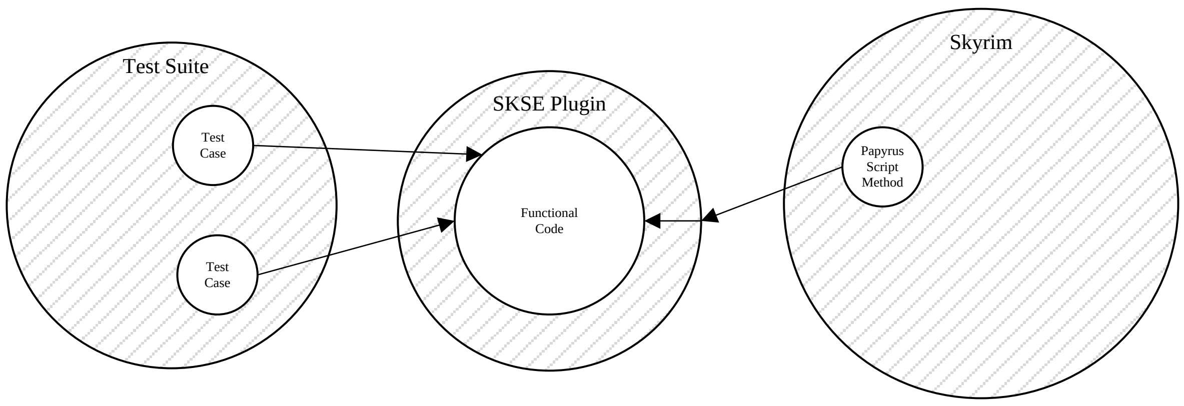
Any math operations need to be done through C++ as Papyrus Script is not capable of doing things as simplistic as division reliably.

With the removal of FISS and the use of JSON for data transfer, we believe we have a more stable and portable system. Each JSON file has its own handler in the plugin. These are intended to restrict access to that which is intended.

Unit Testing

Each method called by the plugin is a wrapper for a native C++ method which takes in standard C++ types. The native C++ methods are exported by their project so that others may access them. It is this wrapping of native methods that allows for unit testing as the unit tests can simply call the native methods with test case data. As we can assume they will behave as though they are being called by the plugin, we can leverage them to perform white box unit testing on the code. Unit tests are provided in the Test project which can be run from within VisualStudio. This greatly saves on time and uncertainty when compared to acceptance testing from within Skyrim.

A basic illustration is provided below:



Data Flow

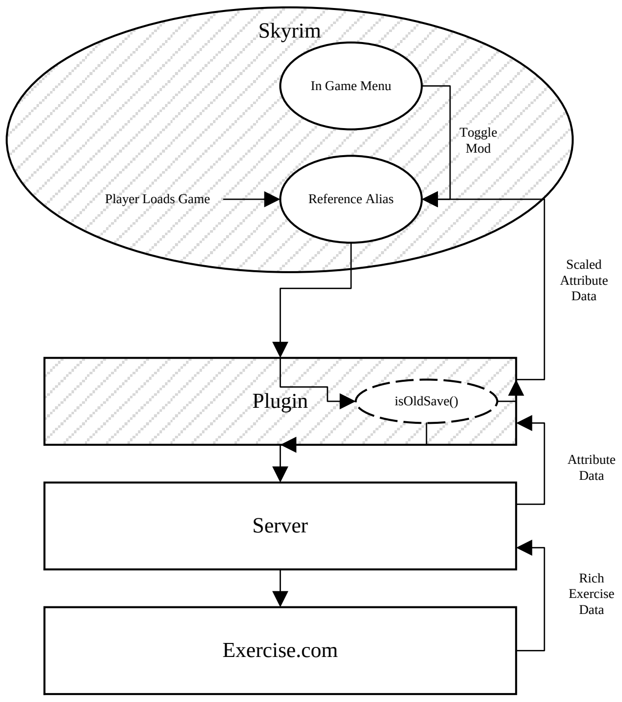
A typical data flow will start in either the PlayerReferenceAlias or the ExergamingConfigMenu. Control flows from the ExergamingConfigMenu to the PlayerReferenceAlias. From the PlayerReferenceAlias, control flows to the plugin from which it may flow to the web service executable if the current loaded save is the most recent(see Loading an Old Save for the alternative). The web service executable is responsible for requesting data from the server given the username and dates supplied from the plugin by the PlayerReferenceAlias. This data is stored in the Raw_Data.json file located in the Skyrim folder. The raw data contains workouts in the following form:

```
{
    "data": {
        "workouts": [{
            "syncDate": 1467670756,
            "workoutDate": 1464566400,
            "health": 1682,
            "stamina": 0,
            "magicka": 122
        },{
            "syncDate": 1467670756,
            "workoutDate": 1466812800,
            "health": 0,
            "stamina": 0,
            "magicka": 1002
        }]
    }
}
```

This raw data is then processed by the plugin into weeks based on the workoutDate field. Once this is done, the plugin decides whether levels should be rewarded. The plugin uses the config file through the configure() method to decide how many levels a workout is worth. This “weight” in stored in the workouts information in the Weeks.json file. This is called in a loop which returns control back to the script with a stringified list of the levels to award.

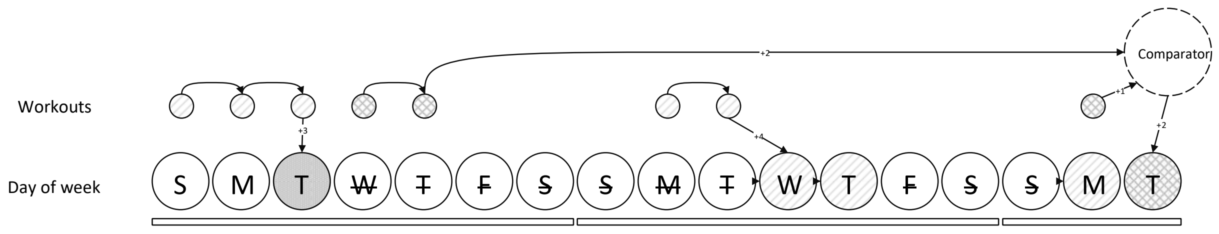
These levels are separated as there may be use in showing them individually to the user, those currently they are summed and presented in one message. This potential usage is the reason the control is passed between the Papyrus script and the plugin more than once in this section. At the end of this data flow, the number of levels are added to the user’s character in the papyrus script. The attributes are also added to with the distributions as determined by the plugin.

A basic overview of the data flow is given in the illustration below:



Loading an Old Save

As many people think of exercise in weeks we have decided to store workouts in an abstraction of them. When the user loads an old save, they receive points for workouts from a so called “best week”. The best week is determined by the number of workout points that are of benefit to the player. This week must be between the current day and the day the save was created. The points that are used (the points that are of benefit to the player) are those form workouts from the day of the week the save was made, to the end of that week. The figure below illustrates how this week is chosen. The example illustrated is based on weeks which run Sunday to Monday (represented by S-M). The player has loaded the dark shaded Tuesday and the current day is the cross-hatched Tuesday. Points for workouts completed from Tuesday (the day the old save was created) to the end of the week (in this case Sunday) are summed and compared to find the best week. The contributing workouts are shown as cross-hatched in the figure.


Workouts
Day of week
Comparator
+3
+2
+4
+1
+2
S M T ~~W~~ ~~T~~ ~~F~~ ~~S~~ ~~S~~ ~~M~~ ~~T~~ W T ~~F~~ ~~S~~ ~~S~~ M T

Weeks are stored in the Weeks.json file in the Skyrim directory as follows:

```
{
    "weeks": [
        {
            "startDate": 1467936000,
            "workouts": [
                {
                    "health": 1614,
                    "magicka": 67,
                    "stamina": 51,
                    "syncDate": 1468976159,
                    "weight": 1,
                    "workoutDate": 1468454400
                },
                {
                    "health": 825,
                    "magicka": 139,
                    "stamina": 0,
                    "syncDate": 1468976750,
                    "weight": 0.717024803161621,
                    "workoutDate": 1468195200
                }
            ]
    ]
}
```

Note:

This document is intended as a reference to aid in the understanding of the Skyrim Exergaming code base. It is by no means a complete description of the exact implementation. If you have any questions regarding more specific detail than what is provided here, please do not hesitate to contact me via email at mitchell.longair1@gmail.com.