Informe 1

URL: https://github.com/jachana/Loncher

Introducción:

Para partir con el diseño decidimos dividir las labores. Vicente se encargaría de la interfaz gráfica, Julio y Jurgen de los aspectos técnicos y Ricardo del manejo de archivos para el registro de puntajes, ubicaciones de los juegos y demás información útil.

Descripción general:

La estructura de github está dada por una estructura de un proyecto en Visual Studio. Los jeugos a ser lanzados deben estar registrados en el xml (registroArcade.xml) y deben heredar de la clase Start e implementar el método Go.

Funcionamiento:

El archivo examplelauncher es el archivo principal del proyecto y lo que hace es crear un objeto de juego dummy, junto con el lanzador y le pide al lanzador que lance el juego. El juego dummy crea una clase juego que hereda de una clase start e imprime cosas en consola. LRArcadeLauncher.py es donde se lanzan propiamente los juegos, al poseer la clase GameCaller. Esta tiene solo un metodo importante que es GameCall que recibe dos parámetros, game ep (entrypoint) y un bool que indica si la llamada se hará en otro proceso o en el mismo que el arcade. Por defecto es falso.
De aquí tenemos dos casos.
Cuando es en el mismo proceso encierra en un try catch, tira una variable que almacena el código de error del juego e intenta que esta variable sea igual a lo que devuelve el método go del juego y retorna eso. Si hay una excepcion lo que hace es retornar -1.
En el otro caso se crea un nuevo proceso, el cual se le indica que a python que es principal y que no puede cerrar hasta que termine el proceso y se lanza. Después de esto se hace un join del proceso y retorna 0. Si lanza una excepción, no existe forma de conocer la excepción e informa de un error genérico.
StartBase es donde están todos los juegos de su clase de entrada, tiene el método go que todos tienen que implementar y no recibe parámetros. **Todos los juegos tienen que heredar de start e implementar el método Go.** Este es el requisito básico que deben cumplir los juegos para que funcionen.
Sobre el XML Manager, se busca que todos los juegos implementen esta clase y hagan modificaciones a un archivo xml en donde está guardada información que será usada por el launcher como puntajes, rutas a imágenes, rutas a videos, descripciones y demás. En el archivo xml están también guardadas otras informaciones básicas como la ruta de los scripts de los juegos y su nombre.

Problemas:

XMLManager:
El sistema de xml fue fácil en un comienzo pero luego fue marcado por una serie de errores que debido a una falta de un adecuado debug se hicieron difíciles de detectar. Sin darme cuenta estaba redactando un archivo xml con fallas que por lo tanto no podía ser leído. Tan pronto me di cuenta del error, para lo que pasé un buen número de horas cabeceándome de formas de debuggear el código, todo fue rápido.

GameCaller:
Originalmente se había planteado que en las llamadas multiprocesos, el sistema detectara automatizadamente cuando un juego se había quedado pegado (freeze), y terminase el proceso para volver al launcher. Debido a problemas que no pudieron ser resueltos con la comunicación inter-procesos, se optó por eliminar esta funcionalidad.

Pendientes:
Un menu que relacione todos los modulos del launcher, con interfaz grafica incluida.
Método que mediante reflection logre cargar el punto de entrada a un juego dada la ruta del script que lo contiene, sólo sabiendo que dicho punto de entrada hereda de la clase Start.
Se podría agregar más información al Xml, como el número de crushes de los juegos, y otras informaciones que surjan en el camino.