Suggestions de projets option INFO-IA

1 Introduction

Sourcepawn est un langage de programmation, compilé, à typage statique. Il est principalement utilisé pour la création de plugins pour les jeux utilisant le moteur de jeux Source.

Depuis 2020, je développe des outils pour faciliter le développement de code Sourcepawn: sourcepawn-vscode, sourcepawn-lsp, SPFormat. Les suggestions de projets suivants sont des idées d'outils qui viendraient aider le développement de plugins Sourcepawn.

Sourcepawn est un langage simple, très proche du C, sans les allocations de mémoire manuelles, comme en C++. La simplicité du langage permet de rendre le développement d'outils plus atteignable que pour d'autres langages plus complexes comme Swift ou Rust.

J'ai déjà développé plusieurs librairies pour instrumenter du code SourcePawn (Lexer, Preprocesseur, Parser et Analyseur semantique). Ces librairies peuvent être utilisées dans les projets pour faciliter le développement.

2 Projets

2.1 Linter

Les linters font une analyse statique du code pour trouver des erreurs dans le code (erreurs de syntaxe, nombre d'arguments insuffisants, signature de fonction incorrect, ...). Le compilateur de Sourcepawn, *spcomp*, permet déjà de produire certains de ces diagnostics, mais il est très limité. En-effet, il se limite à des erreurs simples, et ne peut pas détecter plus de quelques erreurs en même temps.

Créer un linter pour Sourcepawn nécessite de faire du préprocessing sur le code source (comme en C/C++) et ensuite de parser le code. On obtient ainsi un arbre de syntaxe abstrait (AST), et on peut ensuite analyser ce graph et vérifier, par exemple, que toutes les variables déclarées sont au moins utilisées une fois, que les appels de fonctions utilisent la bonne signature, etc (Plus d'exemples sur la documentation du linter Clippy).

Pour aller plus loin dans ce projet, on peut également prévoir un mode incrémentale, qui recalcule partiellement les diagnostics quand un fichier change. Cela est utile pour les éditeurs de code notamment.

2.2 Formatter / Beautifier

Un formatter de code comme Black pour Python ou Clang Format pour C/C++ permet de formatter du code source selon des règles prédéfinies.

Avant formattage

```
int main() {int foo=1;return 1;}
```

Après formattage

```
int main() {
        int foo=1;
        return 1;
}
```

La difficulté en Sourcepawn est qu'on ne peut pas se servir de l'AST pour formatter le code, car, comme en C, les déclarations du préprocesseur peuvent casser la syntaxe, par exemple, le code suivant est correct, mais un parser ne parviendra pas à le transformer en AST:

Déclaration préprocesseur cassant la syntaxe

```
#define FOO
#if defined FOO
void OnPluginStart() {
#else
void OnPluginStart(int bar) {
#endif
}
```

Il faudra donc utiliser le flux de symbols du lexer directement, comme Clang-Format (cette présentation détaille l'implémentation par un développeur de Clang-Format).

Une des difficultés de ce projet est de mettre en place un algorithme qui trouve le formattage idéale du code lorsque plusieurs représentations sont possibles (où faire un retour à la ligne dans un long appel de fonction par exemple).

2.3 Détection de fuites de mémoire

Sourcepawn est un langage compilé en bytecode, qui est ensuite interprété par une machine virtuelle (comme Java). Le langage implémente les Handles, qui représentent des pointeurs sous forme d'entiers. Cela permet notamment de garder un compteur des *Handles* d'un plugin, pour le désactiver automatiquement si une fuite de mémoire est détectée. Il y a une fuite de mémoire lorsque l'on oublie de fermer une *Handle* (comme en C++ lorsque on alloue de la mémoire sur la *heap*).

Cette détection se fait au moment de l'exécution mais il est préférable de détecter ces erreurs à la compilation, afin de faciliter le débuggage.

Une idée de projet serait de détecter statiquement les fuites de mémoire en Sourcepawn. Il n'est normalement pas possible d'utiliser l'analyse statique pour trouver les fuites de mémoire en C++ mais la simplicité de Sourcepawn devrait le permettre.

Ce projet peut également être fait en même temps que l'idée de projet n1.