libParamTuner : réglage interactif de paramètres sans re-compilation de code
libParamTuner : interactive tuning of parameters without code recompilation

Marc Baloup, Veis Oudjail et Géry Casiez
Université de Lille, France
{marc.baloup,veis.oudjail}@etudiant.univ-lille1.fr,gery.casiez@univ-lille1.fr

ABSTRACT

We present libParamTuner, a cross-platform library designed to allow the interactive tuning of parameters in applications written in C++ or Java, without the need to re-compile code. libParamTuner provides a lightweight syntax to bind some variables of an application to the parameters defined in an XML file. Each modification of the XML file updates in real time the associated parameters in the application. A graphical interface allows editing the XML file, using interactive controls dynamically created for each parameter.

CCS CONCEPTS

• **Human-centered computing → User interface toolkits**;

KEYWORDS

library, parameters, interactive tuning.

RÉSUMÉ

Nous présentons libParamTuner, une librairie multi-plateformes conçue pour permettre le réglage interactif de paramètres dans des applications écrites en C++ ou Java, sans avoir besoin de recompiler le code source. libParamTuner propose une syntaxe légère pour associer des variables d'une application à des paramètres définis dans un fichier XML. Toute modification du fichier XML entraîne une mise à jour en temps réel des variables pour les paramètres correspondants. Une interface graphique permet d'éditer le fichier XML, en utilisant des contrôles interactifs créés dynamiquement pour chaque paramètre.

MOTS-CLEFS

librairie, paramètres, réglage interactif.

IHM '17 , August 28–September 1, 2017, Poitiers, France

1 INTRODUCTION

Le développement d'applications requiert souvent la définition d'un certain nombre de paramètres dont les valeurs par défaut sont difficiles à définir a priori. C'est particulièrement vrai pour les systèmes interactifs où, par exemple, la position d'un objet à l'écran ou sa vitesse de déplacement sont difficiles à définir précisément par avance. C'est également le cas pour le réglage de paramètres de filtres conçus pour des systèmes interactifs comme le 1€ filter [2]. Le réglage de chacun des paramètres de ce filtre demande en effet une appréciation par l'utilisateur de l'effet obtenu qui se fait plus facilement quand il est possible de régler interactivement chacun des paramètres.

Généralement le développeur utilise une stratégie essai/erreur : il donne une valeur plus ou moins arbitraire aux différents paramètres à régler, compile le code de l'application, exécute et observe le résultat obtenu. Il réitère l'opération jusqu'à obtenir un résultat satisfaisant, ce qui peut être long et fastidieux, sans garantie d'avoir obtenu le meilleur réglage possible. Le processus est d'autant plus difficile quand il est nécessaire de régler des paramètres qui sont corrélés.

Une alternative est de permettre la modification de ces paramètres dans l'interface, par exemple avec les touches du clavier. Cela demande d'écrire du code spécifique, ce qui prend du temps. Suivant le nombre de paramètres à régler, il peut être difficile de mémoriser les touches associées à chaque paramètre, et l'affichage des valeurs courantes des paramètres se fait le plus souvent dans la console. Un autre exemple est de développer une interface graphique dans l'application pour régler de manière interactive ces paramètres. Là encore le développement de l'interface graphique peut être couteux en temps. Ce coût est généralement jugé trop élevé par rapport aux bénéfices espérés.

Certains langages comme python en mode interactif ou des langages dérivés de Smalltalk comme Pharo [1] facilitent la modification de valeurs de paramètres à l'exécution. Cependant le contrôle interactif de ces paramètres via une interface graphique n'est pas prévu par défaut et les modifications ne sont pas enregistrées de manière persistante. Par ailleurs la majorité des applications développées aujourd'hui utilisent d'autres langages de programmation comme C++ ou Java.

D'autres approches comme Kaboom [3] ou Choc [1] s'inspirent d'idées popularisées par Bret Victor sur le *Leanable Programming* [2] [3]

1. https://www.fullstack.io/choc/
2. http://worrydream.com/LearnableProgramming/
3. https://vimeo.com/36579366

pour modifier de manière interactive des paramètres dans des IDE particuliers et pour des langages spécifiques. libParamTuner est inspirée de ces travaux mais se veut très facile à utiliser, sans avoir besoin de modifier l'environnement de développement de l'utilisateur. Son fonctionnement est détaillé ci-dessous.

2 LIBPARAMTUNER

libParamTuner comprend trois parties : 1) un fichier texte au format XML (définition des paramètres) 2) une librairie C++ qui observe quand le fichier texte est modifié et met à jour à la volée les valeurs dans l'application 3) une application de réglage interactif des paramètres qui charge le fichier, crée les widgets de contrôle des paramètres et sauvegarde les modifications dans le fichier texte.

L'utilisation d'un fichier XML permet une définition simple de chaque paramètre et surtout une sauvegarde persistante des réglages effectués. Les types booléen, entier, réel (float, double) et chaines de caractères sont supportés.

Un exemple minimaliste en C++ est le suivant :

```
#include "paramtuner.h"
int main() {
    int i; double d; string s;
    ParamTuner::load("settings.xml");
    ParamTuner::bind("varI", &i);
    ParamTuner::bind("varD", &d);
    ParamTuner::bind("varS", &s);
    loop(); // Boucle principale du programme.
}
```

Le développeur commence par déclarer les variables correspondantes aux paramètres. Il définit ensuite le chemin vers le fichier XML et associe chaque variable à un nœud du fichier XML, dont voici l'exemple associé au code ci-dessus :

```
<?xml version="1.0" encoding="UTF-8"?>
<ParamList>
    <varD type="double" value="2.3" min="0" max="100"/>
    <varI type="int"    value="12"  min="0" max="100"/>
    <varS type="string" value="lorem ipsum"/>
</ParamList>
```

Chaque nœud comprend le nom du paramètre, son type, sa valeur courante et, pour les types numériques, les valeurs minimales et maximales, utilisées par l'application de réglage interactif.

La surveillance des modifications du fichier XML se fait par l'utilisation de librairies natives spécifiques à chaque système d'exploitation. Nous utilisons la fonction `FindFirstChangeNotification` sous Windows, la librairie `inotify` sous Linux et les `FSEvents` sous macOS.

Chaque modification de fichier déclenche l'appel d'une fonction de callback qui modifie les valeurs stockées aux adresses mémoires des variables associées à chaque paramètre. Dans l'application, la seule contrainte est de forcer la mise à jour régulière de l'affichage afin de lire les nouvelles valeurs des paramètres.

3 RÉGLAGE INTERACTIF

Le réglage interactif des paramètres se fait par l'utilisation d'une application tierce écrite en Java. Celle-ci permet de charger un fichier XML et génère l'ensemble des widgets associés à chacun des paramètres : cases à cocher pour des booléens, sliders et champs de texte pour les entiers et réels et champs de texte pour les chaînes de caractères (Figure 1). La sauvegarde du fichier peut se faire manuellement ou automatiquement à chaque modification d'un paramètre. En pratique le délai de mise à jour des paramètres est très faible (inférieur à la seconde), même si nous n'avons pas pu le caractériser formellement.

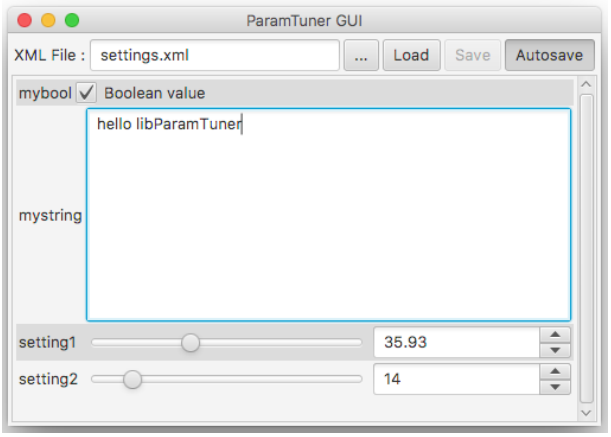

Figure 1: L'application de réglage interactif des paramètres. L'application permet de définir le fichier XML à charger et génère automatiquement les widgets permettant le réglage de chacun des paramètres. Le fichier est soit sauvegardé automatiquement à chaque modification ou manuellement.

4 CONCLUSION

libParamTuner permet de régler de manière interactive les paramètres d'une application sans avoir besoin de la recompiler. libParamTuner est librement disponible sur GitHub [4] sous licence GPL. La librairie peut être utilisée dans des programmes écrits en C++ ou Java mais nous espérons que la communauté aidera au développement de bindings dans d'autres langages.

RÉFÉRENCES

[1] Alexandre Bergel, Damien Cassou, Stéphane Ducasse, and Jannik Laval. 2013. *Deep into Pharo.* Square Bracket Associates. 420 pages. https://hal.inria.fr/hal-00858725

[2] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1€ Filter : A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems. In *Proceedings of CHI'12, the 30th Conference on Human Factors in Computing Systems (CHI '12).* ACM, New York, NY, USA, 2527–2530. DOI : http://dx.doi.org/10.1145/2207676.2208639

[3] Marchal Damien. 2014. Kaboom : une boîte à outils pour la programmation à la volée en C/C ++ . IHM'14, 26e conférence francophone sur l'Interaction Homme-Machine. (Oct. 2014). https://hal.archives-ouvertes.fr/hal-01089615 Poster.

4. https://github.com/casiez/libparamtuner