BeamToon

Guide du programmeur

Université de Namur

BAYET
Anthony

Idée originale:
JACQUET
Jean-Marie

année académique 2019-2020

Contents

1 Développeurs

Ce projet a été développé et conceptualisé conjointement par :

- Jean-Marie Jacquet
  Faculté d'informatique
  Université de Namur
  email: jean-marie.jacquet@unamur.be

- Anthony Bayet
  email: anthony.bayet@hotmail.com

2 Commanditaire

Le commanditaire du projet BeamToon est monsieur Jacquet Jean-Marie, professeur à la faculté d'informatique de l'université de Namur, qui a également participé au développement du projet.

3 Cahier des charges

3.1 Énoncé de base

BeamToon doit être un système permettant de créer des fichiers textuels décrivant des diapositives avec des animations, de la musique, etc. Et sur base de ces fichiers les affichaient.

3.2 Produit actuel

Actuellement, BeamToon est composé de 2 parties. Première, un langage, le BeamToon Markup Language (BTML), qui offre une syntaxe pour écrire les fichiers interprétables, ce langage est décrit précisément dans le guide de l'utilisateur (présent dans le répertoire Github). Deuxièmement, un programme appelé "interpréteur" qui sert à lire ces fichiers et afficher ce qu'ils décrivent.

L'ensemble des possibilités actuelles de BeamToon sont exhaustivement décrites dans le guide de l'utilisateur.

3.3 Pistes de développement

- Ajouter des éléments permettant d'afficher des vidéos, des gifs, etc.
- Porter l'interpréteur sur Android pour permettre d'utiliser BeamToon pour faire des prototypes d'application mobile.
- Permettre de lier des entrées claviers à des actions. Pour par exemple, changer de diapositive.
- Permettre de lier une action à d'autres événements, comme : déclencher une animation, jouer une musique, etc.
- Créer un affichage d'erreurs explicites lorsqu'un projet est en train de tourner.
- Créer un mode de chargement progressif, c'est-à-dire que l'interpréteur ne chargerait le contenu des diapositives que lorsque c'est nécessaire. Dans le but d'optimiser l'utilisation de la mémoire et ainsi pouvoir créer des projets BeamToon plus lourd.

4 Outils utilisés

L'interpréteur BeamToon a été développé en Processing (processing.org) car ce langage est désigné pour faire des logiciels graphiques et il est rapide d'apprentissage[1] et simple d'utilisation.

De plus, il possède un environnement de développement dédié, qui s'appelle le PDE. Cela a l'avantage d'offrir une grande facilité pour développer la partie Processing.

Mais la contre-partie est non négligeable, car cet environnement n'est pas autant développé que d'autres (comme Eclipse, IntelliJ, etc.) et donc certaines features cruciales au bon développement d'un projet de grande envergure n'y sont pas présentes, comme l'intégration d'outils de testing ou de documentation.

C'est parce que BeamToon est actuellement une version simple et concise de ce qu'il peut devenir que l'interpréteur a pu être développé en Processing jusqu'à présent.

[1] Grâce aux tutoriels sur le site de Processing.

Mais pour faire aller ce projet plus loin, il serait sage de le réécrire en Java avec un IDE plus poussé, car Processing n'est pas qu'un langage, mais aussi une librairie en Java, on peut donc facilement réécrire l'interpréteur BeamToon en Java et ainsi profiter des features d'un IDE plus développé.

4.1 Utiliser le PDE

Si vous souhaitez continuer en Processing, il suffit de télécharger le PDE à l'URL suivante: processing.org/download. Et ensuite, pour ouvrir le projet, après avoir cloné le projet depuis le Github, il suffit d'ouvrir le fichier "code/interpreter/interpreter.pde" avec le PDE.

5 Déploiement

Cette section illustre comment créer les dossiers "application.linux64" et "application.windows64" avec Processing. Ces dossiers servent à exécuter un projet Processing sans devoir faire aucune installation.

Pour les créer, il suffit d'ouvrir le fichier principal (interpreter.pde) du projet dans Processing. Et après, allez à File -> Export Application..., Ceci ouvrira une boite de dialogue servant à choisir le système d'exploitation que vous visez, et si vous voulez inclure une version de Java dans le dossier (ce que nous conseillons fortement). Après avoir appuyé sur "Export" le PDE générera vos dossiers.

Si vous voulez créer le dossier "application.windows64", il faut en plus ajouter dedans le fichier "code/iterpreter_launcher.bat". Qui est un script permettant d'exécuter plus facilement un projet BeamToon.

6 Code source

BeamToon est un projet OpenSource et son code est disponible sur la page Github nommée "BeamToon". Dans la suite de cette section, il y a une présentation de la structure de l'interpréteur BeamToon.

6.1 Principes généraux

L'interpréteur BeamToon fonctionne en 2 phases.

D'abord lire le fichier et convertir son contenu en une structure de données adaptée à être utilisée pour en afficher le contenu.

Ensuite, utiliser cette structure de données pour en exploiter le contenu et en plus réagir aux interactions de l'utilisateur comme décrit dans le fichier BTML de base.

6.2 Structure de données

La structure de données citée dans la section précédente est une HashMap associant un nom à un objet de la classe "Slide".

6.2.1 Slide

Un objet de la classe "Slide" permet de regrouper tous les éléments qui doivent être affichés à l'écran[2], ainsi que des informations telles que la musique à jouer. La classe Slide et celles des éléments affichables héritent toutes de la classe "Drawable".

6.2.2 Drawable

La classe abstraite "Drawable" sert à implémenter n'importe quel élément qui doit être affiché à l'écran. Pour ce faire, leur classe doit au moins implémenter une méthode, nommée "overridableDraw()", qui sert à dessiner concrètement l'élément en Processing.

De plus, cette classe a un objet du type ArrayList, nommé children, qui garde les enfants de l'élément. Ces enfants sont des drawables qui seront dessinés juste après leur parent.

Finalement, c'est aussi elle qui implémente les mécanismes pour qu'un élément réagisse à un clic de souris.

6.3 Conversion du fichier BTML

Pour convertir les fichiers BTML, il faut d'abord les lire, et pour ça nous utiliserons à chaque fois un parser XML built-in à Processing qui nous permet de convertir nos fichiers en un objet du type XML très pratique.

Dès lors, la conversion du fichier BTML vers la structure de données présentée dans la section précédente se fait en 3 phases : la résolution des alias principaux,

[2]Ces éléments sont en réalité des enfants de l'objet "Slide".

puis de ceux importés, et pour finir la conversion effective de l'objet du type XML vers la structure de données.

6.3.1 Résolution des alias principaux

Pour cette étape, on prend un à un chaque élément avec la balise "alias" présent dans le fichier principal.

Ensuite, on cherche récursivement dans tous les éléments du fichier principal s'il y en a un qui a la même balise que celle de la première partie de l'alias. Et quand on en trouve un, on le remplace par sa nouvelle valeur, qui est la deuxième partie de l'alias, en prenant soin de remplacer les valeurs des attributs de cette nouvelle valeur par celles précisées dans la balise trouvée ou bien par les valeurs par défaut qui se trouve dans la première partie de l'alias.

6.3.2 Résolution des alias importés

Cette phase va utiliser le même mécanisme de remplacement d'alias que la précédente. Sauf que les balises "alias" ne viendront pas du fichier principal, mais auront été récupérées dans les fichiers désignés par les balises "import" du fichier principal.

Dès lors, le programme va récupérer dans l'ordre d'apparition des balises "import" dans le fichier principal, les alias des fichiers qu'elles désignent. Pour ensuite les utiliser comme décrit précédemment dans la structure XML du fichier principal.

6.3.3 Conversion du XML

La phase finale consiste à convertir l'objet de type XML vers la structure de donnée souhaitée depuis le début.

Pour ce faire, nous allons d'abord récupérer les balises "slide" qui sont les racines de chaque slide, et ensuite nous allons récupérer tous ses enfants et les convertir en objet d'un type "Drawable" en fonction de sa balise.

Puis nous répétons cette action récursivement sur les enfants du slide et leurs enfants. Et finalement, une fois que nous avons obtenu un objet du type Slide, nous l'ajoutons à la structure de données avec son nom.

6.4 Fonctionnement d'exécution

Une fois que la structure de données a été générée sur base du fichier principal, l'interpréteur va afficher la première diapositive de la structure de données en appelant sa méthode "draw()", ce qui a pour effet de le dessiner et d'appeler les méthodes "draw()" de ses enfants directement après.

De plus, lorsque l'utilisateur clique, le programme donnera l'information de l'événement de la position de la souris au slide courant et ainsi le laissera déterminer si une action doit se produire ou non.

Pour finir, lorsqu'il y a un changement de diapositive, le programme change la valeur de la diapositive courante pour le nouveau demandé, l'initialise par sa méthode "start()" et ensuite, il appellera sa méthode "draw()".

6.5 Animation

Dans ce programme, les animations sont faites sur des Float uniquement. Nous avons choisi de créer une nouvelle classe de Float, AttributeFloat, qui est la classe de tous les attributs qui seraient du type Float naturellement et qui peuvent être animés.

Ainsi, pour mettre à jour la valeur de l'objet du type Attribute Float, il faut appeler une de ses méthodes "getValue()", qui permet de mettre à jour la valeur et de la récupérer.

Grâce à cette méthode, nous évitons des opérations inutiles, car les valeurs seront mises à jour uniquement quand c'est nécessaire.

7 Documentation

La documentation de l'interpréteur BeamToon a été générée par Doxygen et est disponible sur la page Github au chemin "doc/documentation". Pour l'ouvrir, il suffit d'ouvrir le fichier "index.html" du dossier dans un navigateur internet.