Editeur de niveaux
Prog & Play

Benjamin BONTEMPS

29 mars 2016

Résumé

Ce document a pour objectifs de récapituler l'essentiel du travail réalisé dans le cadre du projet de conception d'un éditeur de niveaux pour le jeu sérieux Prog&Play et de permettre sa maintenabilité en cas de nécessité. Il suppose acquis les concepts de base liés à Prog&Play, à Spring (moteur sur lequel est basé le jeu) et au développement en Lua.

Table des matières

1 Architecture **2**

2 Description détaillée des modules **2**
2.1 Widget . . . 2
2.1.1 Chili UI . . . 2
2.1.2 Initialisation . . . 3
2.1.3 Unités et groupes d'unités . . . 3
2.1.4 Zones . . . 5
2.1.5 Forces . . . 6
2.1.6 Déclencheurs . . . 6
2.1.7 Paramètres de la carte . . . 6
2.1.8 Entrées / Sorties . . . 6
2.1.9 Fonctions de dessin . . . 6
2.1.10 Listeners . . . 6
2.2 Gadget . . . 6
2.3 Machines à états . . . 6
2.4 Fonctions utilitaires . . . 6
2.5 Fichiers de description . . . 6

1 Architecture

Cette section présente les choix effectués quant à l'architecture adoptée pour la réalisation de l'éditeur de niveux.

Ce dernier est composé de :
- Un widget central permettant d'afficher les éléments de l'interface utilisateur et de capter les interactions avec cette dernière ou avec le moteur pour effectuer des actions en conséquence. (editor_user_interface.lua)
- Un gadget permettant l'utilisation de code synchronisé [1]. (editor_gadget.lua)
- Un fichier contenant toutes les chaînes de caractères à afficher. (EditorStrings.lua)
- Un fichier définissant une classe [2] machine à états. (StateMachine.lua)
- Un fichier contenant des fonctions utilitaires. (Misc.lua)
- Des fichiers de description de tables Lua. (Actions.lua, Conditions.lua, TextColors.lua, Filters.lua)

2 Description détaillée des modules

2.1 Widget

Le widget est l'élément central de l'éditeur des niveaux autour duquel s'articulent les autres modules. Le choix d'un widget en tant qu'élément central se justifie par le fait que la majorité des interactions se font de façon asynchrone. L'unicité du widget vient du fait qu'il n'est possible de n'avoir des variables communes à plusieurs widgets qu'en passant par la table WG, ce qui aurait été gênant étant donné le très grand nombre de variables devant être utilisées à de nombreux endroits distincts.

2.1.1 Chili UI

Afin de simplifier le développement de l'interface graphique, j'ai choisi d'utiliser un framework d'interface graphique développé spécifiquement pour Spring dénommé Chili UI.

Ce framework a l'avantage de ne pas avoir à utiliser l'API OpenGL de Spring et les différents listeners pour afficher et permettre l'interaction avec des fenêtres, boutons etc. Tout y est déjà implémenté, ce qui a permis de réduire considérablement le temps requis pour obtenir les premiers résultats. Le problème majeur est que le framework est extrêmement mal documenté, et la meilleure façon de comprendre son fonctionnement pour s'en servir correctement voire même le modifier pour l'adapter à des besoins particuliers est de se référer au code source présent sur github [3].

Les modifications que j'y ai personnellement apporté concernent les boutons, qui possèdent un état supplémentaire *chosen* permettant d'avoir un feedback visuel lorsqu'on les sélectionne (les modifications de cet état sont gérées "à la main", mais le feedback visuel est géré automatiquement par le framework).

1. Par exemple, la création d'une unité sur le terrain.
2. Au sens de Lua.
3. https ://github.com/jk3064/chiliui/

Le widget comporte de nombreuses fonctions reprenant les fonctions *New* du framework en n'utilisant qu'un certain nombre de paramètres, ce qui permet une définition plus compacte de chaque élément de l'interface. Le point négatif est qu'il faut soit connaître par cœur l'ordre dans lequel les paramètres doivent être renseignés dans le prototype de la fonction soit se référer en permanence à la définition des fonctions. Ces fonctions sont définies dans la catégorie *Chili UI functions*.

Il faut faire extrêmement attention à une chose en particulier en ce qui concerne le framework : il ne faut surtout pas modifier les éléments d'interface qui ne sont pas affichés à l'écran (c'est-à-dire les éléments d'interface qui n'ont ni Screen0, ni un descendant de Screen0 en tant que parent). Ceci ne produit pas d'erreur lors de l'exécution, mais les éléments d'interface concernés ne s'actualiseront plus correctement, ce qui est très gênant d'un point de vue utilisateur (les opérations seront effectuées, mais l'utilisateur n'aura aucun feedback visuel).

2.1.2 Initialisation

L'initialisation de la quasi-intégralité de l'interface se fait par les fonctions d'initialisation appelées dans la méthode *widget :Initialize()*, et décrites dans la catégorie *Initialisation functions*. La grande majorité des fenêtres y est initialisée, en les affichant toutes d'un coup (pour les raisons évoquées précedemment), puis elles sont ensuite masquées pour n'afficher que la fenêtre correspondant à l'état actuel de la machine à états globale (voir 2.3).

Sont présentes également deux fonctions s'occupant de l'initialisation de fonctions liées aux changements d'état lors de la sélection d'un type et d'une équipe d'une unité pour la placer ensuite sur le terrain.

Les fonctions des catégories *Top bar functions* et *Forces window buttons functions* s'occupent de changer l'état courant de la machine à états globale et de choisir quelles fenêtres doivent être affichées. Il s'agit pour la plupart de fonctions appelées lors de la pression sur des boutons ou sur des touches du clavier.

2.1.3 Unités et groupes d'unités

Les fonctions présentes dans la section *Unit/Selection state functions* gèrent à la fois le placement des unités (choix du type et de l'équipe), la sélection, le positionnement, l'orientation, les attributs et l'appartenance à un groupe d'une ou plusieurs unités.

updateUnitWindow Fonction s'occupant d'afficher les boutons permettant de sélectionner le type de l'unité que l'on veut placer sur le terrain. Elle n'affiche que les boutons des unités appartenant à une faction dont le bouton est dans l'état *chosen*, ce qui permet de n'afficher que les unités de certaines factions.

applyChangesToSelectedUnits Callback lors de la validation de la modification de certains attributs des instances d'unité sélectionnées : applique et sauvegarde ces changements (le nombre de points de vie est sauvegardé dans un tableau, l'orientation est directement appliquée).

drawSelectionRect Affiche le rectangle de sélection d'unités. On dispose ici des coordonnées de la position initiale et de la position actuelle de la souris lors du cliquer-glisser pour sélectionner les unités. Or, dessiner un rectangle requiert d'avoir les coordonnées du point en haut à gauche ainsi que la longueur et la largeur, ce qui nécessite des calculs intermédiaires en fonction de la position actuelle par rapport à la position initiale de la souris, d'autant plus que l'ancrage de Chili se trouve en haut à gauche alors que l'ancrage en Spring (et en OpenGL) se trouve en bas à gauche.

previewUnit Affiche une unité en transparence lors de son placement sur le terrain. Cette fonction utilise des fonctions OpenGL pour modifier la matrice de transformation (et afficher l'unité sous le curseur), la couleur utilisée (ajouter de la transparence) et afficher l'unité en fonction de son ID. La modification du masque de profondeur est nécessaire pour éviter un bug d'affichage (comparable à une inversion des normales du modèle 3D).

showUnitAttributes Affiche la fenêtre permettant de modifier les attributs des unités sélectionnées. Les valeurs des attributs sont initialement affichées si elles sont partagées entre toutes les unités sélectionnées.

showUnitsInformation Affiche l'ID et la position des unités sélectionnées et de l'unité sur laquelle se trouve le curseur de l'utilisateur.

showUnitGroupsAttributionWindow Affiche la fenêtre permettant d'assigner les unités sélectionnées à un groupe d'unités déjà existant (en récupérant la liste et en créant les boutons correspondants) ou à un nouveau groupe en entrant un nom de groupe.

showUnitGroupsRemovalWindow De même que précédemment, mais pour retirer les unités sélectionnées d'un groupe auquel elles appartiennent. Si elles n'appartiennent à aucun groupe ou qu'elles n'ont aucun groupe en commun, un message apparaît, et il n'est donc pas possible de les retirer d'un groupe.

updateSelectTeamButtons Met à jour les boutons de sélection d'une équipe lors du placement d'une unité en fonction des équipes activées dans la fenêtre de configuration des équipes.

updateUnitList Met à jour la liste des unités sur la droite de l'écran lorsqu'une unité est ajoutée ou retirée du terrain. Pour chaque unité, ajouter un bouton permettant de centrer la caméra sur cette unité et une image permettant de savoir si l'unité est sélectionnée ou non.

updateGroupListUnitList De même que précédemment mais la liste qui est mise à jour est celle présente dans la fenêtre des groupes d'unités. Les deux listes se trouvent dans des méthodes distinctes pour éviter de mettre à jour la liste de la fenêtre des groupes d'unités alors qu'elle n'est pas affichée (et ainsi ne pas faire planter l'interface).

updateUnitHighlights Affiche un halo jaune autour des unités sélectionnées dans la liste des unités.

updateUnitGroupPanels Regarde si il faut update la liste des groupes (c'est-à-dire si le nombre de groupe a changé ou si le nombre d'unités dans un groupe a changé). Si oui, affiche et positionne un panneau par groupe (les panneaux sont positionnés cote-à-cote sur la première ligne pour les 4 premiers, puis sous la colonne de plus faible hauteur pour les suivants). La taille des panneaux est pré-calculée en fonction du nombre d'unités dans le groupe. Ensuite, on actualise les groupes en affichant des boutons pour chaque unité du groupe afin de pouvoir les retirer du groupe ou les visualiser sur le terrain. Cette fonction s'occupe également d'actualiser le nom des groupes en fonction de ce qui est écrit dans l'editbox.

addUnitToGroup Ajoute une unité à un groupe si elle n'y est pas déjà.

addSelectedUnitsToGroup Ajoute les unités sélectionnées à un groupe.

addChosenUnitsToSelectedGroups Ajoute les unités dont le bouton dans la fenêtre de gestion des groupes d'unités est dans l'état *chosen* aux groupes dont le bouton est dans l'état *chosen*. Il s'agit de la fonction appelée lorsque l'utilisateur appuie sur le bouton **[»]** de la fenêtre de gestion des groupes d'unités.

removeSelectedUnitsFromGroup Enlève les unités sélectionnées d'un groupe.

removeUnitFromGroup Enlève une unité d'un groupe.

addUnitGroup Ajoute un nouveau groupe.

addEmptyUnitGroup Ajoute un groupe vide avec un nom générique.

deleteUnitGroup Supprime un groupe.

Les états de la machine à états des unités permettent d'adapter le comportement des fonctions listeners d'évènements souris ou clavier. Ainsi, les fonctionnalités de sélection, rotation, déplacement... des unités se font par exemple dans *widget :MousePress* et *widget :MouseMove*. Ces fonctionnalités sont donc détaillées dans 2.1.10.

2.1.4 Zones

Les fonctions de cette section concernent la création, suppression, affichage, sélection et déplacement des zones logiques. Ces zones sont utilisées dans les déclencheurs.

computeZoneWorldCoords Cette fonction calcule les coordonnées dans l'espace de la zone en train d'être tracée en utilisant des raycasts et en triant les valeurs obtenues. Elle s'occupe également de forcer les zones à être carrées ou circulaires en appuyant sur alt.

drawZoneRect Dessine une zone rectangulaire aux coordonnées calculées par la fonction précédente.

drawZoneDisk De même, mais pour une zone elliptique.

displayZones Affiche les zones sélectionnées par l'utilisateur. Cette fonction s'occupe aussi de désélectionner une zone qui n'est plus affichée.

displaySelectedZoneAnchors Affiche les bords de la zone sélectionnée pour permettre à l'utilisateur de la redimensionner.

showZoneInformation Affiche les coordonnées des points importants de la zone sélectionnée ainsi que le nom de toutes les zones affichées.

updateZoneInformation Actualise le nom et l'état affiché des différentes zones.

clickedZone Retourne la zone se trouvant sous la souris. Cette fonction parcours la liste de toutes les zones dans l'ordre et retourne la dernière trouvée sous la souris, ce qui permet d'éviter une possible indétermination. Si la zone actuellement sélectionnée se trouve sous la souris, on garde sélectionnée cette zone : ceci permet une meilleure interaction puisque la zone peut continuer à être déplacée même lorsqu'elle est sous une autre zone.

2.1.5 Forces

2.1.6 Déclencheurs

2.1.7 Paramètres de la carte

2.1.8 Entrées / Sorties

2.1.9 Fonctions de dessin

2.1.10 Listeners

2.2 Gadget

2.3 Machines à états

2.4 Fonctions utilitaires

2.5 Fichiers de description