Editeur de niveaux
Prog & Play

Benjamin BONTEMPS

30 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

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.

getClickedZone Cette fonction parcours la liste de toutes les zones de façon circulaire [4] et renvoie la première zone se trouvant sous la souris. Ceci permet de cycler les zones et pouvoir ainsi toutes les sélectionner même si elles sont superposées.

getZoneSide Renvoie le côté de la zone sélectionnée sur lequel l'utilisateur a cliqué pour lui permettre de redimensionner la zone ou de la déplacer. Le côté est codé sous forme d'une chaîne de caractères. Les variables *left*, *right*, *top* et *bottom* correspondent à la distance entre l'endroit cliqué et l'un des bords du rectangle, tandis que *outer* et *inner* correspondent respectivement à l'appartenance de l'endroit cliqué à l'ellipse extérieure et intérieure (en fonction de leurs valeurs par rapport à 1).

applyChangesToSelectedZone Applique les déformations à la zone sélectionnée. Pour un rectangle, cela revient à changer les coordonnées des points correspondant à ce qui a été renvoyé par la méthode précédente. Pour une ellipse, cela revient à modifier les longueurs des petit et grand axes. La méthode est appelée récursivement dans le cas où la souris sortirai du terrain : ainsi, la zone peut s'étendre jusqu'au bord de la carte [5].

updateZonePanel Si le nombre de zones a changé, mettre à jour la liste des zones avec le nom et une checkbox pour choisir ou non de l'afficher.

De même que précédemment, les états de la machine à états régissent les interactions souris, qui sont donc gérées dans les listeners du widget.

2.1.5 Forces

Les fonctions de cette section s'occupent d'afficher les paramètres liés aux équipes et de gérer les alliances entre les différentes équipes actives.

updateAllyTeamPanels Cette fonction s'occupe de l'affichage des alliances entre les équipes. La première partie concerne l'affichage des alliés de chaque équipe ainsi que d'un bouton permettant de supprimer un allié. La seconde partie s'attache à n'afficher que les panneaux des équipes activées.

4. La dernière zone pointe vers la première.

5. Si on ne prends pas ces précautions, la zone s'arrête à l'endroit où était la souris à la frame qui précède la frame où la souris se trouve en dehors du terrain, ce qui rendait impossible le fait d'étendre la zone jusqu'au bord à moins d'aller très lentement.

addTeamToSelectedAllyTeam Défini une équipe en tant qu'alliée d'une autre.

removeTeamFromAllyTeam Supprime l'allié d'une équipe.

removeTeamFromTables Supprime une équipe de toutes les alliances. Ceci assure qu'une équipe inactive n'apparaisse pas dans une alliance.

updateTeamsWindows Lorsque le nombre d'équipes actives est modifié, on met des switchs sur true pour actualiser certaines fenêtres lors de la prochaine ouverture de ces dernières. Ces actualisations concernent les boutons sélection de l'équipe lors du placement d'une unité, les paramètres des équipes et les alliances entre les équipes.

updateTeamConfigPanels Permet de n'afficher les modifications liées aux équipes (couleur en jeu, contrôle) que lorsque l'équipe en question est activée.

2.1.6 Déclencheurs

Les fonctions de cette partie concernent la gestion du système de déclencheurs (création, édition, suppression des évènements, conditions et actions ainsi que la gestion des variables).

createNewEvent Création d'un nouvel évènement vide. Les IDs sont attribués aux évènements en commençant par 1 et en étant incrémentés à chaque nouvel évènement. Cette fonction instancie également les tableaux qui contiendront les boutons pour modifier et supprimer les conditions et les actions de cet évènement.

editEvent Cette fonction ouvre la fenêtre de gestion d'un évènement, la génère en fonction des paramètres de l'évènement et l'assigne à l'évènement courant.

removeEvent Supprime l'évènement. Par mesure de sécurité, les fenêtres sont fermées.

createNewCondition, editCondition, removeCondition, createNewAction, editAction, removeAction Il s'agit des mêmes fonctions que celles décrites précédemment mais pour les conditions et les actions.

removeSecondWindows, removeThirdWindows Cache respectivement les fenêtres en "deuxième" et "troisème" position, en allant de gauche à droite.

updateEventList Actualise la liste des évènements (fenêtre la plus à gauche) lorsque le nombre d'évènements change. Cette fonction s'occupe également d'actualiser le nom de l'évènement courant lorsque la valeur de l'editbox change.

updateEventFrame Actualise l'évènement courant : lorsque le nombre de conditions ou d'actions change, on actualise les boutons qui permettent de les modifier ou de les supprimer. Cette fonction s'occupe également d'actualiser le nom de la condition ou de l'action courante lorsque la valeur de l'editbox change.

currentEventFrame Génère la fenêtre d'édition de l'évènement courant en fonction de son nom, de ses conditions et de ses actions.

currentConditionFrame, currentActionFrame Démarre le processus de génération de la fenêtre d'édition de la condition ou de l'action courante. Un point notable : si l'action ou la condition a déjà été définie et possède donc un type, on sélectionne dans la combobox l'action correspondante pour faciliter l'édition. Sinon, on sélectionne le premier élément. La fonction de callback liée à la sélection d'un item de la combobox s'occupe de générer les paramètres restants (cf fonctions suivantes). La variable *dontUpdateComboBox* est un verrou permettant de sélectionner les items de certaines combobox sans appeler une partie de la fonction de callback de celles-ci.

selectFilter Fonction de callback liée à la sélection d'un item dans la combobox liée au filtre sur les types de conditions ou d'actions. Cette fonction génère la liste des conditions ou des actions correspondantes au filtre sélectionné.

selectConditionType, selectActionType Fonction de callback liée à la sélection d'un item dans la combobox correspondant au choix du type de condition (resp. action) pour la condition (resp. action) courante. Elle récupère le type de la condition (resp. action) sélectionnée et affiche le texte et les différents paramètres pour ce type.

drawConditionFrame, drawActionFrame Cette fonction récupère le template du type de la condition (resp. action) sélectionnée, affiche son texte et trace chacun de ses attributs en fonction du type de l'attribut. Ces tracés sont appelés *features* par la suite. L'argument *reset* permet de réinitialiser les valeurs des paramètres de la condition (resp. action) courante, dans le cas où le type serait changé.

drawFeature Cette fonction trace une feature correspondant au type de l'attribut actuellement étudié. Par exemple, si il s'agit d'un attribut de type "groupe", il faut afficher une combobox permettant de choisir le groupe parmi tous les groups disponibles.

configureEvent Cette fonction ouvre et génère la fenêtre de configuration de l'évènement courant. Cette fenêtre est composée d'une editbox permettant de modifier le déclencheur de l'évènement en utilisant des opérateurs logiques (not, and, or), d'un panneau permettant de changer l'ordre d'exécution des actions et d'un gestionnaire d'importation de conditions ou d'actions provenant d'autres évènements.

updateImportComboBoxes Fonction de callback de la combobox permettant de choisir de quel évènement on veut importer les conditions ou actions. Cette fonction génère donc les items des combobox permettant de choisir quel condition ou action on veut importer.

importCondition, importAction Créer une nouvelle condition (resp. action) possédant les mêmes paramètres que la condition (resp. action) importée sauf l'id qui continue à être incrémentée classiquement.

preventSpaces Empêche l'utilisateur de mettre des espaces dans les noms des évènements, conditions et actions.

showVariablesFrame Affiche la fenêtre permettant d'ajouter, supprimer et modifier des variables pouvant être utilisées dans les évènements.

addVariable Ajoute une nouvelle variable.

removeVariable Supprime une variable.

updateVariables Actualise le nom et les paramètres des différentes variables.

drawVariableFeature Affiche les éléments d'interface permettant de modifier les variables. Si une variable possède déjà des arguments (type, valeur initiale) les sélectionne lors de la génération des éléments d'interface.

showPickText Affiche un texte lorsque l'on choisi une unité ou une position.

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