SuperB1copter

LE JEU DE SIMULATION DÉJANTÉ !

Documentation technique

Architecture globale

L'architecture est constituée de différentes briques qui s'assemblent par le biais de liens entre les structures.
On compte notamment GameResources qui gère les ressources du jeu (musiques, textures, ...), les charge et les libère ; Configuration qui gère les informations qui devront être pérennisées dans un fichier de configuration ; Audio qui gère tout ce qui concerne l'audio, en encapsulant la (trop) riche bibliothèque FMOD dans une interface adaptée aux besoins du jeu; GameShowObjects (voir Moteur graphique) ; GameControl qui permet de contrôler toutes les informations relatives au jeu telles que le nombre de vies restant au joueur ainsi que tout le gameplay qui y est implémenté. Cette architecture a l'avantage d'être très flexible et tout à fait extensible.

Moteur graphique

La séparation est faite entre les ressources elles-mêmes (gérées dans GameResources) et les objets graphiques montrés à l'écran (gérés dans GameShowObjects). Cette séparation permet en particulier de rendre la gestion des deux totalement indépendante.
Le GameShowObjects est chargé de s'assurer du bon affichage des sprites et animations (en choisissant la frame appropriée en fonction du temps). Il offre également des fonctions helpers qui permettent de manipuler les textes d'interface du jeu.

Scrolling

Tous les éléments sont positionnés dans le référentiel de l'univers du jeu. Tous ? Non. Un hélicoptère qui fait exception en devant être positionné toujours au centre de l'écran fait exception en ayant sa coordonnée x fixée par rapport à l'écran et non l'univers. Un rapide calcul (souvent sous le pseudonyme de helico_real_position) permet de retrouver sa position dans l'univers. Tous les calculs de gameplay se font par rapport au référentiel de l'univers ce qui évite ainsi tout parasite subtil tel que l'affichage d'un élément en dehors de l'écran. Tous les calculs d'affichages se font par rapport au référentiel écran. Le calcul se base sur gso->backgroundPosition qui contient les coordonnées permettant de faire le lien entre l'origine du référentiel univers (à l'emplacement de la base de l'hélicoptère) et l'origine de l'écran (en haut à gauche).

Mouvement des otages/ennemis

Les otages et les ennemis se déplacent dans certaines conditions particulières fixées par le gameplay (celui-ci se trouve pour la majeure partie dans la fonction processEvents(GameControl* ctrl).
Leur mouvement est déterminé par leur position par rapport à l'hélicoptère.

Gestion des balles/missiles

Les balles et missiles sont gérés de façon tout à fait identique, puisqu'il s'agit au final d'objets mortels se déplaçant en continu. A chaque sprite est associé une position et un mouvement. Cela permet de gérer leur déplacer, ainsi que leurs collisions en masse indifféremment de leur type.
A chaque frame la position de chaque sprite se voit incrémenté des valeurs contenues dans le mouvement qui lui associé.
A chaque frame les calculs de collisions sont effectués : à l'aide de calculs d'intersection de rectangles, chaque balle peut avoir l'occasion d'apporter son mortel message à la cible qui a le malheur de croiser son chemin. Les balles sont aveugles et ne différencient pas l'ami de l'ennemi, que voulez-vous c'est la guerre.