Projet
Architecture & C++ & OpenGL
Informatique
—IMAC 2e année—

Smashstein

Ave Caesar, Morituri te salutant
Devise IMAC2

Projet de fin de semestre pour les cours d'architecture logicielle, de programmation objet et d'infographie.

---

1 Smashstein

1.1 Description générale

Le projet Smashstein vous propose de construire un jeu temps réel orienté action/combat/plateforme en 2.5D. Au cours du projet vous créerez une *arène* 3D avec ses règles, ses contraintes. Vous placerez ensuite dans cette arène un *avatar* représenté par un modèle 3D qui sera piloté au clavier et à la souris par un joueur humain. Pour tenir compagnie à ce joueur humain vous ajouterez des adversaires appelés *bots* constitués d'un modèle 3D et d'une intelligence artificielle.

1.2 Gameplay

1.2.1 La seconde dimension et demi

Dans un jeu en 2.5D, les graphismes sont affichés en 3D, mais les protagonistes se déplacent tous sur le même plan. La caméra de jeu n'est pas libre, en fonction du gameplay, le scrolling peut être vertical, horizontal ou multidirectionnel.

1.2.2 Inspiration

Il existe un grand nombre de jeux récents en 2.5D mêlant combat et plateforme, vous pouvez notamment puiser l'inspiration dans les jeux tels que *Smash bros*, *Tee Worlds*, *Little Big Planet*, ou encore *Trine*.

1.2.3 Thème

Il est bien entendu conseillé de donner une thématique à votre jeu, cette thématique influencera le type de gameplay que vous choisirez, ainsi que la panoplie d'actions que pourront effectuer l'avatar et les bots.

Exemple 1 : Left 2.5 Dead L'avatar doit s'échapper de l'arène par la porte, il est attaqué par plusieurs bots qui collaborent pour le stopper.

Exemple 2 : Course Les avatars et les bots se livrent à une course en sautant de plate forme en plateforme tandis que l'arène se remplit d'eau, de lave ou autre liquide désagréable. Le gagnant est le dernier en lice.

Exemple 3 : Smashstein Les avatars et les bots cherchent à s'expulser mutuellement de l'arène.

1.3 L'arène

1.3.1 Liste désordonnée et non exhaustive des fonctionnalités à implanter

- Charger l'arène à partir d'un modèle 3D, d'une image 2D, d'un fichier XML ou les trois.
- Afficher l'arène en OpenGL en utilisant textures et modèles 3D.
- Charger et afficher plusieurs bots.
- Utiliser les caractéristiques des bots.
- Afficher les projectiles.
- Renseigner intelligemment le radar des bots.
- Implanter des conditions de victoire et de défaite.
- Gérer les collisions.
- Implanter une gestion de l'inertie et de la gravité.
- Utiliser les techniques OpenGL de texture, billboards, display lists et particules.
- Utiliser rendu multi-vue
  - Caméra poursuite centrée sur un bot ou sur l'avatar (choisi avec une IHM)
  - Caméra globale affichant l'ensemble des protagonistes.
  - Caméra *debug* libre.

1.3.2 Contraintes techniques obligatoires

- L'arène charge les IA des bots de façon dynamique pendant l'exécution au moyen d'un système de plugins.

1.4 Les joueurs

Dans cette sections vous trouverez les fonctionnalités communes aux bots et à l'avatar.

1.4.1 Liste désordonnée et non exhaustive des fonctionnalités à implanter

- Afficher le modèle en OpenGL
- Retranscrire graphiquement les actions des joueurs
- Les joueurs possèdent un panel d'actions à définir contenant au moins :
  - Déplacement vertical et horizontal
  - Envoi de projectile
  - Attaque de contact
  - Attente
- Animer le modèle.

1.5 L'avatar

L'avatar est divisé en deux parties une description et un ou plusieurs modèles 3D.

1.5.1 Liste désordonnée et non exhaustive des fonctionnalités à implanter

- Charger l'avatar à partir d'un fichier XML et d'un modèle 3D

1.5.2 Contraintes techniques obligatoires

- La description est au format XML
- Les modèles 3D sont au format MD2
- L'avatar est piloté au clavier, et/ou à la souris, par un joueur humain.

1.6 Les bots

Le bot est divisé en trois parties, une description, un ou plusieurs modèles 3D et un plugin représentant le cerveau. Le cerveau reçoit des données par son radar, prend une décision et finalement transmet un ordre. La description contient les caractéristiques de l'avatar ainsi que les liens vers chacun des modèles 3D représentant ses diverses parties.

1.6.1 Liste non exhaustive des fonctionnalités à implanter

- Le bot doit être capable de prendre des décisions parmi lesquelles :
  - Rejoindre un endroit
  - Attaquer un joueur au contact
  - Attaquer un joueur à distance
  - Fuir un joueur
  - Poursuivre un joueur
- Il existe plusieurs bots aux caractéristiques différentes
- Il existe plusieurs plugins d'IA représentant des caractères différents
- Le bot est utilisable dans l'arène d'un autre groupe

1.6.2 Contraintes techniques

- La description est au format XML
- Les modèles 3D sont au format MD2
- Le cerveau robot est compilé en un plugin binaire.

1.6.3 Contraintes techniques générales

Programmes Vous devez rendre un programme qui constituera le jeu en lui même ainsi que un à n plugins pour les intelligences artificielles. Il est fortement conseillé de s'inspirer du code fourni.

Bibliothèques obligatoires Le jeu possédera impérativement une IHM utilisant la bibliothèque Qt et un rendu en OpenGL. Le chargement des fichiers XML peut utiliser Qt.

Bibliothèques supplémentaires Il est tout à fait encouragé d'utiliser des bibliothèques supplémentaires à la condition expresse qu'elles soient validées par les enseignants. Sont dispensés de validation, les bibliothèques de traitement XML, l'ensemble des modules de Qt, et *boost*.

Compilation Vous utiliserez le système de compilation *qmake*.

Validation Tous les éléments produits par les équipes doivent impérativement pouvoir être lus, compilés, exécutés le plus facilement possible sur les machines de l'université. En cas d'impossibilité, il est obligatoire de documenter la marche à suivre pour compiler et exécuter. Tout manquement à cette règle entraînera l'invalidation du rendu.

2 Coopération et interopérabilité

Il est tout à fait permis et même encouragé de collaborer entre deux groupes. Il est possible de factoriser certaines parties du code comme le chargeur XML ou le chargeur de meshs MD2. Il serait de même intéressant que le robot produit par un groupe puisse être chargé dans l'arène d'un autre. La coopération au sens de partage de code est donc permise à la condition impérative qu'elle soit explicitement documentée dans le rapport sous peine d'invalider le rendu. L'interopérabilité des robots, notamment si elle est documentée et démontrée sera bien sûr récompensée.

3 Echéances

Le projet doit s'effectuer par équipes de 4 à 6. Il devra faire l'objet de deux rendus, le premier aura lieu le jeudi 14 janvier avant midi, chaque équipe devra alors fournir un rapport papier. Le second rendu prendra la forme d'une archive à télécharger contenant l'ensemble des données jugées nécessaires par l'équipe ainsi qu'une soutenance dont les horaires de passage vous seront communiquées ultérieurement, ce rendu aura lieu le XX janvier.

4 Le rapport

Chaque équipe devra produire un rapport d'une trentaine de pages maximum, qui contiendra la liste exhaustive des fonctionnalités implantées, un manuel détaillé de son programme, une liste des problèmes connus, les problèmes rencontrés, les piste d'amélioration, les choix techniques et leur justification. Vous inclurez aussi tous les documents nécessaires à l'évaluation de l'architecture logicielle.

5 La soutenance

La soutenance aura une durée de 30mn dont 5mn de démonstration et 10mn de questions. Au cours de cette soutenance vous évoquerez à tour de rôle les principales qualités de votre programme.

6 Évaluation

6.1 Architecture

Vous serez évalués sur la pertinence de la conception de votre programme, notamment sur l'organisation des classes et l'utilisation des concepts vus en cours. Pour cela vous devrez produire les diagrammes UML détaillés adéquats ainsi qu'une justification des designs patterns que vous aurez mis en oeuvre.

6.2 C++

La note de C++ tiendra compte de la qualité générale du code, du respect des conventions de codage, de la qualité de l'IHM ainsi que du respect des contraintes techniques du projet.

6.3 OpenGL

La partie OpenGL sera évaluée sur la qualité et la quantité des procédés OpenGL que vous mettrez en oeuvre dans votre projet : les textures, animations, particules, billboards ou encore brouillard. Le détail des fonctionnalités graphiques devra être accompagné de captures d'écran.

7 Références

Jouer avec la libdl - http://ilay.org/yann/articles/dlfcn/
Des idées sur la gravité - http://www.physics-lab.net/applets/projectile-motion
Le format MD2 décrypté - http://tfcduke.developpez.com/tutoriel/format/md2/