Principe des moteurs de jeux
(8INF871) Plan de cours

Programmes dans lesquels se trouve le cours

Obligatoire *
1537 Maîtrise en informatique (Profil professionnel, concentration jeux vidéo)

Optionnel
3017 Maîtrise en informatique (Profil recherche)
3037 Maîtrise en informatique (Profil professionnel)

* L'étudiant doit suivre ce cours ou le cours 8INF955 *Principe de conception et de développement de jeux vidéo* dans le cadre de son parcours académique.

Objectif général

Introduire l'étudiant aux principes fondamentaux de la programmation des moteurs de jeux.

Objectifs spécifiques

- Rappels des notions d'infographie, de physique mécanique et d'intelligence artificielle propres à la programmation des moteurs de jeux.
- Moteurs de rendu.
- Systèmes d'animation.
- Détection de collision.
- Éditeurs de niveaux.
- Programmation en temps réel.
- Analyse d'un moteur de jeux.

Contenu

- Rôles, responsabilités et composition d'un moteur de jeu
- Retour sur la boucle de jeu
- Aperçu de haut niveau d'un moteur de jeu
- Structures de données et représentation du contenu
- Scènes, graphes et navigation
- Retour sur l'infographie
    - Vecteurs, quaternions et matrices
    - Coordonnées objet-monde-projection
    - Textures et *UV mapping*
    - *Pipelines* fixes et programmables, nuanceurs (*shaders*)
- Simulation physique mécanique
    - Aires englobantes
    - Croisement de vecteurs
    - Collisions de *sprites*
    - Lancement de rayons
    - Collisions en mouvements
    - Transferts de force
    - Joints
- Texte
    - Polices de caractères
    - Encodage de caractères
    - Glyphes et leurs propriétés
    - Crénage
    - Anti-crénelage
- Régionalisation
    - Langues et différences culturelles
    - Sémiotique
    - Texte et audio
- Effets sonores et musique
    - Composition
    - Rendu et effets
    - Compression
- Fonctionnalités réseau
- Animations classique, cinématique inverse, animations procédurales
- Liaisons aux langages de scripts
- Intégration de l'intelligence artificielle
- Développement rapide d'applications (*RAD*)
- Optimisation
- Modules d'extensions (*plugins*) natifs et programmation de bas niveau
- Intergiciels
- Outils et éditeurs de niveaux

Activités d'apprentissage

- Présentation magistrale;
- Échanges de connaissances avec les étudiants;
- Exercices en classe;
- Travaux sur ordinateur et sur papier;

Formules pédagogiques

- Démonstrations et études de cas fictifs
- Apprentissage par problèmes et résolution de problèmes
- Projets (dans le cadre des exercices)
- Cartes heuristiques (lors des révisions)

Qualité de la langue et du code

Nous sommes dans le domaine des technologies de l'information et des communications. Chaque document ou bout de code que l'on produit sera lu et relu et re-relu beaucoup plus souvent qu'il n'aura été écrit. Par conséquent, il est important de prendre soin de bien écrire et de faire attention aux erreurs courantes.

Vous pouvez choisir de produire vos textes ou votre code en français ou en anglais, mais assurez-vous d'être capable de vous exprimer clairement et de minimiser les incohérences et les fautes.

Jusqu'à 10% de la note obtenue pour un travail effectué sur support informatique pourrait être amputé du résultat si une trop grande concentration d'erreurs linguistiques ou d'incohérences injustifiées de vos propres conventions de programmation se retrouvait dans vos travaux.

Présentation des travaux

Les travaux doivent être remis dans un état facilitant leur correction.

Tout travail remis doit contenir les éléments suivants:

- Le nom complet de l'étudiant et de ses coéquipiers (et leurs codes permanents si plusieurs étudiants partagent le même nom...);
- Le titre de l'activité d'apprentissage auquel le document fait référence;
- Le titre ou sigle du cours auquel le document fait référence;
- La date de production ou de remise.

Les travaux remis sur papier comprenant plusieurs pages doivent inscrire, au moins sur la première page, le nombre total de pages, et chaque page doit être numérotée. Les feuilles doivent être jointes par une agrafe ou tout autre mécanisme fiable.

La remise des travaux sous format électronique devra se faire par des moyens technologiques démontrant les dates et heures de remise et les dates et heures de réception. À moins d'avis contraire, les fichiers remis doivent être consolidés en un seul fichier d'archive respectant un format standard (ex.: zip, tar, 7z).

Le Module ou le Département peuvent avoir des exigences supplémentaires concernant la remise des travaux.

Bibliographie et références

Livres

Martin, Robert C. Clean code: A handbook of agile software craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2009.
McConnell, Steve. Code complete: A practical handbook of software construction. Redmond: Microsoft P, 2004.
Fowler, Martin, and Kent Beck. Refactoring: Improving the design of existing code. Reading, MA: Addison-Wesley, 1999.
Gamma, Erich. Design patterns: Elements of reusable object-oriented software. Reading: Addison-Wesley, 1994.

Général

Gamasutra: http://www.gamasutra.com

Joel on Software :
http://www.joelonsoftware.com/

Technologies 2D

Atari 2600	Television Interface Adaptor
Nintendo NES	Picture Processing Unit
ClanLib	http://www.clanlib.org/download.html

Technologies 3D

id-tech-1 (Doom)	http://doomwiki.org/wiki/Doom_rendering_engine
Ogre3D	http://www.ogre3d.org/download/sdk
XNA	http://www.microsoft.com/en-us/download/details.aspx?id=23714

Physique 2D

Box2D	http://code.google.com/p/box2d/

Physique 3D

PhysX	https://developer.nvidia.com/physxapex-download-page
Havok	http://www.havok.com/solutions/students-and-educati

Moteurs 2D

Torque2D	http://mit.garagegames.com/Torque2D-2.0.zip
GameMaker	http://www.yoyogames.com/studio/download
RPG Maker	http://www.rpgmakerweb.com/download/free-programs/rpg-maker-vx-ace-lite

Moteurs 3D

Torque3D	https://github.com/GarageGames/Torque3D
Unreal UDK	http://www.unrealengine.com/udk/downloads/
Unity3D	http://www.unrealengine.com/udk/downloads/