1 Liste des projets

1.1 Aquarium artificiel

1.1.1 Descriptif général

Écrire un économiseur d'écran simulant un aquarium.

1.1.2 Fonctionnalités attendues

a. Pouvoir introduire plusieurs types de poissons / comportements ;
b. Pouvoir introduire / générer différents modèles de plantes sous marines ;
c. Pouvoir introduire des objets quelconques ;
d. Gérer les effets de lumière / de réfraction ;

1.2 Billard 3D

1.2.1 Descriptif général

Écrire un simulateur de jeu de billard (ou une de ses variantes). Plusieurs joueurs doivent pouvoir y jouer tour à tour.

1.2.2 Fonctionnalités attendues

a. Visualiser un billard texturé et éclairé sous différents points de vue ;
b. Implémenter un modèle physique permettant de simuler les effets de frappe ;
c. Gérer les ombres portées ainsi que les reflets ;
d. Jouer à plusieurs.

1.3 Coloriage de modèles

1.3.1 Descriptif général

Réaliser une application prenant la forme d'une palette graphique et permettant à l'utilisateur de colorier un modèle 3D chargé depuis un fichier (tel le coloriage de figurines de plâtre).

1.3.2 Fonctionnalités attendues

a. Charger un modèle 3D à partir d'un fichier ;
b. Modéliser (prévoir) plusieurs type d'outils (crayons, pinceaux et gommes) et permettre à l'utilisateur de les utiliser pour dessiner (colorier) sur le modèle 3D ;
c. Se déplacer autour du modèle ;
d. Éclairer le modèle : gérer plusieurs sources de lumières et permettre à l'utilisateur de les déplacer.

1.4 Combat aérien

1.4.1 Descriptif général

Placé dans la peau d'un aviateur, le joueur a pour mission de combattre tous les ennemis présents à l'écran (sur le radar). Le jeu place l'aéronef en vue arrière (voir cockpit aussi), il est de type Arcade. Vous pouvez vous inspirer des exemples suivants : Afterburner II et Comanche 2.

1.4.2 Fonctionnalités attendues

a. Piloter un aéronef au dessus d'un relief naturel (+ urbain si possible) texturé ;
b. Le rendu du relief doit utiliser du LOD (Level Of Detail) ;
c. Détecter les collisions ;
d. Combattre différents type d'ennemis (DCA, avions, hélicoptères, etc) gérés par le programme ;
e. Jouer en réseau.

1.5 Course acrobatique

1.5.1 Descriptif général

Ici, vous placez le joueur aux commandes d'un véhicule quelconque où il devra effectuer une course contre la montre. L'ensemble se déroule sur un circuit acrobatique (looping, rampes, franchissements et autres sauts périlleux) et le joueur devra faire en sorte de terminer le circuit sans trop de dégâts. Le jeu est de type Arcade et vous pouvez vous inspirer du jeux Stunt car.

1.5.2 Fonctionnalités attendues

a. Piloter un véhicule sur un circuit acrobatique ;
b. Pouvoir changer de point de vue : ras du sol, cockpit, arrière, haut et points fixes sur le circuit pour les ralentis ;
c. Réaliser un système physique, détecter les collisions et gérer les dommages sur le véhicule ;
d. Faire jouer le programme et gérer un tournoi ;
d. Éditer et/ou générer les circuits.

1.6 Course F1 / Karting / Moto

1.6.1 Descriptif général

Ici, vous placez le joueur aux commandes d'un véhicule (auto / kart et/ou moto) où il devra effectuer une course contre d'autres concurrents. Cette dernière se déroule sur un circuit et les coureurs devront effectuer un certain nombre de tours avant le finish.

Le jeu peut être de type Simulation ou Arcade et vous pouvez vous inspirer de F1 Grand Prix, Vroom / Super Mario kart / Moto racer, etc.

1.6.2 Fonctionnalités attendues

a. Piloter sur un circuit en mode entraînement ou course ;
b. Pouvoir changer de point de vue : ras du sol, cockpit, arrière, haut et points fixes sur le circuit pour les ralentis ;
c. Détecter les collisions ;
d. Le programme devra contrôler les autres concurrents ;
e. Éditer et/ou générer des circuits ;
f. Jouer à plusieurs en splitant l'écran ou via le réseau.

1.7 Course tout terrains

1.7.1 Descriptif général

Ici, vous placez le joueur aux commandes d'un véhicule quelconque où il devra effectuer une course dans laquelle il ralliera, sans autres contraintes, un point d'arrivée ; la course comporte des concurrents et se déroule dans un environnement naturel (reg, plaines ou petites collines) et/ou urbain. Le jeu est de type Arcade.

1.7.2 Fonctionnalités attendues

a. Conduire un véhicule quelconque / tout terrains d'un point de départ à un point d'arriver ;
b. Pouvoir changer de point de vue : ras du sol, cockpit, arrière, haut et points fixes sur le parcours emprunté pour les ralentis ;
c. Réaliser un système physique, détecter les collisions et gérer les dommages sur le véhicule ;
d. Générer et/ou éditer les terrains ;
f. Jouer à plusieurs via le réseau ;
d. Le programme contrôlera les concurrents.

1.8 Décoration d'intérieur

1.8.1 Descriptif général

Donner la possibilité à l'utilisateur de décorer son intérieur : modéliser son intérieur sur plan puis y placer différents meubles et objets de décoration.

1.8.2 Fonctionnalités attendues

a. Créer une interface permettant de modéliser une pièce : murs, portes et fenêtres ;

b. Personnaliser la pièce : texturer les murs et portes, choix des fenêtres ;
c. Importer des bibliothèques de modèles 3D de meubles et autres objets de décoration (dont les fenêtres) ;
c. Gérer les objets Lumières en tant que “vraies” sources lumineuses ;
d. Gérer l’ombre portée ;
e. Ajouter, déplacer et supprimer des meubles : implémenter une gestion des collisions ;
f. Visiter la pièce ;
g. Exporter la scène dans un format reconnu.

1.9 Exploration intergalactique

1.9.1 Descriptif général

Ici, le joueur incarne le commandant d’équipage d’un navire marchand explorant l’espace intergalactique. L’objectif principal du jeu est de développer des liens commerciaux avec les différents individus / groupes / peuples rencontrés mais le navire peut aussi bien être amené à combattre d’autres équipages afin de se défendre ou défendre sa cargaison. Vous pouvez vous inspirer des exemples suivants : Elite (Amiga) / Frontier : Elite 2.

1.9.2 Fonctionnalités attendues

a. Piloter un vaisseau dans l’espace ;
b. Gérer l’éclairage en fonction des étoiles les plus proches ;
c. Détecter les collisions avec d’autres vaisseaux et/ou astéroides ;
d. Gérer différents points de vue ;
e. Gérer des modes combats ;
f. Faire du commerce / gérer le stock ;
g. Jouer en réseau.

1.10 FPS ou Doom-like

1.10.1 Descriptif général

FPS pour *First Person Shooter* (généralement traduit par jeu de tir subjectif - relativement à la vue) est un type de jeu vidéo dans lequel le joueur est immergé dans un environnement 3D virtuel où il incarne un personnage devant éliminer ses adversaires ; à cet effet il a, à sa disposition, une ou plusieurs armes (blanches ou à feu). Le terme Doom-like est souvent utilisé pour qualifier ce type de jeux et fait référence aux premiers succès du genre : Wolfenstein 3D, Doom et Quake.

1.10.2 Fonctionnalités attendues

a. Pouvoir se déplacer (donc détection de collisions) dans un environnement 3D de type labyrinthe avec sas et zones à accès restreint ; l'utilisateur pourra déverrouiller l'accès via mot de passe / énigmes / clés. L'ensemble de l'environnement est texturé, bumpé et éclairé. L'environnement (= un niveau) pourrait comporter des étages (escaliers et/ou ascenseurs) ;
b. Gérer plusieurs classes d'ennemis ayant chacune ses aptitudes et stratégies de combat ;
c. Réaliser un éditeur et/ou un générateur de niveaux.

1.11 Jeux de réflexion : jeu d'Échecs / jeu de Go / Othello

1.11.1 Descriptif général

Créer un plateau virtuel pour l'un des jeux de réflexion suivants : échecs, jeu de Go, Othello, autres (à valider avec l'enseignant).

1.11.2 Fonctionnalités attendues

a. Créer le plateau virtuel, tout est en 3D texturée et l'ensemble est éclairé de plusieurs sources lumineuses ;
b. Déplacer (jouer) les pièces selon les règles du jeu ;
c. Se déplacer autour du plateau de jeu ;
d. Gérer les ombres portées ;
e. Interfaçage avec un programme (au choix, par ex. : gnuChess, gnuGo, etc.) jouant à ce jeu.

1.12 L-Systèmes

1.12.1 Descriptif général

Écrire un générateur de plantes basé sur les travaux d'Aristid Lindenmayer [PL90, PH92].

1.12.2 Fonctionnalités attendues

a. Écrire un générateur et un interprète géométrique de L-Systèmes ;
b. Le modèle géométrique doit être optimal : par exemple utilisant des metaballs pour déduire les surfaces des branches ;
c. Se déplacer autour du L-Système ;
d. Voir son évolution dans le temps.

1.13 Modeleur 3D

1.13.1 Descriptif général

Écrire un modeleur d'objets 3D.

1.13.2 Fonctionnalités attendues

a. Charger et sauvegarder un objet 3D sous un format reconnu ;
b. Modifier le maillage (ajout, suppression de points) ;
c. Déformer le maillage (attracteurs / répulseurs) ;
d. Gérer un mode édition par CSG (Constructive Solid Geometry) ;
e. Se déplacer, éclairer et texturer le modèle.

1.14 Pong 3D

1.14.1 Descriptif général

Permettre à un ou deux joueurs de jouer à un Pong en 3 dimensions. L'aire de jeu est délimitée par un tunnel et la raquette est déplaçable à la souris. Vous pouvez vous inspirer de la modeste mais efficace implémentation se trouvant à l'adresse : http://www.lelezard.com/jeux/ballon-courbe.html

1.14.2 Fonctionnalités attendues

a. Jouer a Pong en 3D ;
c. Gérer des effets de lumières (éblouissement, explosions, etc.) ;
c. Gérer les ombres portées ;
d. Jouer contre un programme ;
e. Jouer en réseau.

1.15 Randonnée

1.15.1 Descriptif général

L'idée est de proposer à l'utilisateur d'explorer un environnement naturel entièrement généré par ordinateur. Le joueur pourra, dans la mesure du possible et à pied, parcourir l'ensemble de ce monde virtuel.

1.15.2 Fonctionnalités attendues

a. Générer le terrain et le texturer ;
b. Créer un ou plusieurs lacs ;
c. Générer / placer les arbres et les rochers ;
d. Se déplacer à pied en posant des contraintes sur les déplacements possibles ;

e. Gérer les différentes conditions climatiques : neige, pluie, brouillard et beau temps ;
f. Modifier l'éclairage en fonction du temps.

1.16 Rendu de fonctions mathématiques

1.16.1 Descriptif général

Écrire une application permettant à l'utilisateur de visualiser la courbe ou la surface de l'équation mathématique donnée en entrée. Vous pouvez vous inspirer du célèbre programme *gnuplot*.

1.16.2 Fonctionnalités attendues

a. Afficher la courbe / surface correspondant à une équation donnée en entrée ;
b. Gérer les pas et intervalles pour la visualisation ;
c. Se déplacer autour de la courbe / surface créée ;
d. Éclairer les surfaces générées : calculer les normales pour un résultat facetté / lissé ;
e. Introduire la variable temps et permettre ainsi la création d'animations.

1.17 Stratégie temps-réel

1.17.1 Descriptif général

Il s'agit de gérer une population composée de différentes classes de personnages et de l'aider à occuper un terrain et consolider (protéger) son territoire. Selon le profile, chaque classe d'individus possède un panel d'actions réalisables ; votre population croit en fonction de l'occupation du territoire elle même liée au nombre de bâtiments construits. Par ailleurs, votre peuple serait amené à se confronter à d'autres populations ennemies ou amies gérées par la machine. Vous pouvez vous inspirer des exemples de jeux suivants : WarCraft (1 2 ou 3) / Starcraft, Age of empire, The settlers, etc.

1.17.2 Fonctionnalités attendues

a. Se déplacer au dessus du territoire ;
b. Sélectionner un ou plusieurs personnages et leur affecter une tâche : construire / démanteler un bâtiment, se déplacer / explorer, attaquer un ennemi ;
c. La machine pourra contrôler les autres populations ;
d. Jouer en réseau ;
e. Éditer des territoires (niveaux).

Références

[PH92] Przemyslaw Prusinkiewicz and James Hanan. *Lindenmayer Systems, Fractals, and Plants*. Springer Verlag, 1992.

[PL90] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *Algorithmic Beauty of Plants*. Springer Verlag, 1990.