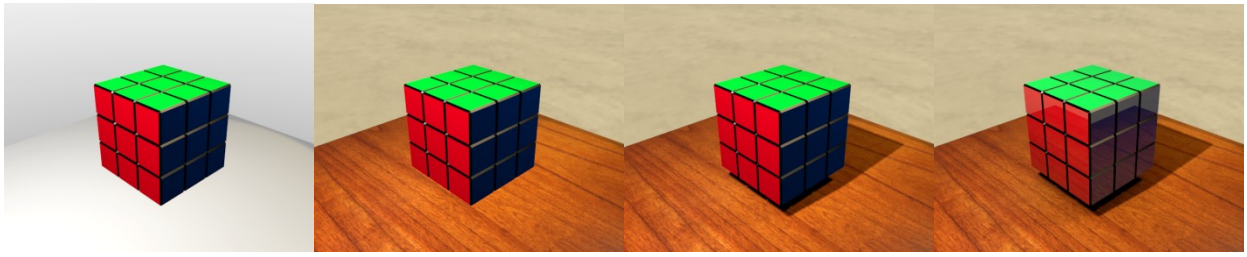
Informatique Graphique 3D et Réalité Virtuelle – Projet
Jeu 3D temps-réel



*Exemples d'é**tapes de la création d'un jeu : mécanique fonctionnelle, ajout des textures, des ombres et des réflexions.***

Principe
L'objectif de ce projet est d'implémenter un jeu vidéo 3D temps-réel.
L'implémentation sera faîte en C/C++ et OpenGL/GLUT. On pourra repartir de l'archive du TP « OpenGL » ou « Shaders» par exemple.
Cinq sujets au choix sont proposés :

- **Jeu d'échecs** : ce jeu doit permettre à 2 jours humains devant la même machine de s'affronter en jouant à tour de rôle leur coup. Le programme doit permettre de tourner autour d'un plateau de jeu virtuel placé sur une table virtuelle et de déplacer ses pions. Le déplacement des pions doit être animé et respecter les règles du jeu. A chaque coup, l'état courant du jeu est évalué (pièce détruite, victoire de l'un des deux joueurs, etc). Aucune IA n'est demandée. Des animations dynamiques sont envisageables.
- **Rubik's cube** : ce jeu doit permettre à un joueur unique de manipuler un Rubik's cube virtuel au-dessus d'une table virtuelle. La partie doit démarrer avec une permutation aléatoire du jeu, et permettre au joueur de :
  - faire tourner le cube pour l'observer sous tous les angles (la caméra est fixe)
  - manipuler les petits cubes en les faisant tourner par couche (animation)

  A chaque modification du cube, le programme doit vérifier si le joueur a gagné. Une gradation dans la complexité de l'état de base permettra d'implémenter plusieurs niveaux de difficulté. Pour rappel, la génération de l'état de base est simple : on part du cube résolu, et on applique un nombre donné d'actions joueur aléatoire.
- **Puissance 4** : ce jeu, plus simple, doit permettre d'empiler des jetons de couleurs différentes de manière à former des ligne verticales, horizontales ou obliques (cf. google pour plus d'infos sur ce jeu). On devra pouvoir tourner du jeu virtuel, et on profitera de la flexibilité du monde virtuel pour créer un jeu ou la géométrie de la structure de support varie (cylindre, surface arbitraire, etc.
- **Flipper** : reproduire l'expérience d'un vrai flipper dans un bar, avec ses sources de lumières, la fumée, ses partie animées, la physique de la bille de métal, etc. Beaucoup de réflexions attendus (cf. cube maps).
- **Aquarium** : créer un aquarium, coloré, vivant, dynamique, où l'on peut nourrir les poisons (système de particules). Attention, l'informatique graphique en milieu aquatique, c'est plus dur.

Dans tous les cas, le programme sera noté sur sa qualité graphique et sa performance graphique. La qualité de l'expérience de jeu sera aussi prise en compte, mais dans une moindre mesure.
Le jeu implémenté devra être fonctionnel. Au lancement du jeu, un écran doit permettre de démarrer une partie, de quitter ou de consulter les scores, et lorsque la partie est terminée, on doit revenir à l'écran d'accueil.

Elément techniques

Le jeu doit être implémenté en OpenGL, ne dépendre d'aucune bibliothèque extérieure, et idéalement offrir les caractéristiques suivantes :

- modèles de réflectance diffus et spéculaire (matériaux standard OpenGL 1.2 ou shaders)
- chargement de textures PPM et plaquage de texture (e.g., surface de la table virtuelle en bois)
- animations, en exploitant la fonction de callback idle des applications GLUT
- ombres portée par *shadow mapping* simple.

Bonus

On pourra implémenter l'un des éléments suivants :

- Ambient occlusion (attention, beaucoup de solutions possibles).
- Réflexions miroir (ex : cube map/environment mapping en OpenGL pour certains objets du jeu).
- Effets spéciaux (ex : effets pyrotechniques lors de la destruction de pièces, effets de fummée/poussière lorsqu'on actionne le Rubik's cube). Ou effets capteur (defocus).

Conseils

- Ces jeux existent dans la vie réelle, on les observera et on observera l'image que l'on en perçoit (voir ci-contre).
- Dans un premier temps, il est conseillé de réaliser un jeu fonctionnel à base de cubes, avant de passer à de véritable objets (ex : maillages pour les pions)
- Ne pas négliger la mise en scène du jeu (éclairage, manipulation contraintes de la caméra, sélection).
- Le sujet est volontairement court et une grande liberté est laissée aux élèves pour proposer des éléments originaux et techniquement intéressants.
- ***Il vaut mieux faire simple et bien, que trop ambitieux et… raté.***

Quelques références utiles

- Un remarquable tutorial et pleins de références sur le *shadow mapping* : http://codeflow.org/entries/2013/feb/15/soft-shadow-mapping/
- Comment faire des jeux « juicy » : https://www.youtube.com/watch?v=Fy0aCDmgnxg
- Comment « sélectionner » en OpenGL avec le mode GL_SELECT : http://www.lighthouse3d.com/opengl/picking/index.php?openglway2