**LSI - 2013**

Projet OpenGL : JEUX D’ECHEC en 3D



# Réaliser par DIOUBATE SEKOU ET

SOMMAIRE

1. OBJECTIF
2. OUTILS UTILISES
3. JEUX D’ECHECS
   1. But du jeu et principes généraux
   2. Les règles officielles
   3. La mise en échec et ses conséquences
   4. Le pat
   5. L'échec perpétuel
4. LES FONCTIONALITES AJOUTER
   1. Effet de la lumière
   2. Effet d’ombre
   3. Reflet
   4. Application de la texture
   5. Déplacement des pions le **picking**
   6. Les cas possible d’un pion sélectionné lors du déplacement
   7. Prise de pion adversaire
5. REALISATION
   1. Chargement du model « échiquier »
   2. Modification que nous avons apportée à la bibliothèque **glm**
   3. Structure de données
   4. Description des fonctions utilisées
6. COMMANDE DU JEUX
7. **OBJECTIF:**

La synthèse d’image en 3D est un ensemble de techniques, qui permet la représentation d'objets en perspective sur un moniteur d'ordinateur. Elle est actuellement très utilisée en art numérique dans l'industrie du film, dans beaucoup de jeux vidéo.

L’objectif de ce projet est de réaliser un jeu d’échecs en 3D composé d'un échiquier de 64 cases (8 lignes et de 8 colonnes), et d'un lot de 32 pièces (16 claires et 16sombre), en appliquant les connaissances élémentaires et techniques apprises en cours.



1. **OUTILS UTILISES**

OpenGL

OpenGL est une interface de programmation d’application comprenant plusieurs fonctions prédéfinies, qui permettent de décrire des objets (cameras, lampes, modèles 3D) et les opérations que l’on peut effectuer pour les manipuler.

Librairies

* OpenGL Utility Library (GLU) : surcouche a OpenGL, fournit des fonctions plus évoluées pour la gestion des caméras ou la création de NURBS.
* OpenGL Extension Library(GLX) : pour le système graphique X11, fournit les moyens de créer un contexte OpenGL et l’associer avec un drawable X11.
* GL Utility Toolkit (GLUT) : fournit une interface simple portable et indépendante de l’OS pour la GUI.
* GLM : permettant d’importer les objets prédéfinis dans des fichiers .obj.
* TGA : permettant l’importation d’image de type .tga pour la texture.

1. **JEUX D’ECHECS**

Le jeu d’échecs oppose deux joueurs de part et d’autre d’un plateau ou tablier appelé échiquier composé de soixante-quatre cases claires et sombres nommées les cases blanches et les cases noires. Les joueurs jouent à tour de rôle en déplaçant l'une de leurs seize pièces (ou deux pièces en cas de roque), claires pour le camp des blancs, sombres pour le camp des noirs. Chaque joueur possède au départ un roi, une dame, deux tours, deux fous, deux cavaliers et huit pions. Le but du jeu est d'infliger à son adversaire un échec et mat, une situation dans laquelle le roi d'un joueur est en prise sans qu'il soit possible d'y remédier.

But du jeu et principes généraux

* Le but du jeu est la capture du roi adverse. C'est l'échec et mat.
* Les blancs commencent toujours la partie.
* Chaque joueur joue un coup à tour de rôle. (jouer est obligatoire, on ne peut pas passer son tour).
* Une partie d'échecs peut se terminer par un match nul

Les règles officielles

Les règles du jeu d'échecs sont très simples et identiques dans le monde entier ! Les règles officielles sont établies par La FIDE (Fédération Internationale Des Echecs).

Pour savoir jouer il faudra maîtriser les points suivants

* la position initiale de l'échiquier et des pièces
* le déplacement des pièces (roi, dame, fou, cavalier, tour, pion)
  + le roque (cas particulier du déplacement du roi)
  + la prise en passant (cas particulier du déplacement du pion)
  + la promotion
* la mise en échec et ses conséquences
* les cas de parties nulles

Une partie d'échecs se jouera souvent avec une pendule afin de limiter la durée de la partie.

Il faudra donc également connaître le règlement du jeu à la pendule.

Il existe également un règlement pour le jeu par correspondance.

La mise en échec et ses conséquences

Lorsque le roi est en prise, il est en échec. Le joueur devra obligatoirement parer cet échec. S'il n'existe aucun coup pour soustraire le roi à l'échec, le roi n’est alors échec et mat et la partie est perdue.

Il existe trois façons de parer un échec:

1. Déplacer le roi.
2. Prendre la pièce qui met le roi en échec.
3. Couvrir l'échec en intercalant une autre pièce.

Il est interdit de mettre son roi dans une position d'échec. Si c'est le seul coup jouable le roi est dans une position de pat et la partie est nulle.

L'expression "Echec et mat" signifie que le roi adverse reçoit un échec imparable. Cela constitue le dernier coup d'une partie d'échecs.

Une partie est déclarée nulle dans les cas suivants

* sur un commun accord des joueurs.
* si le roi est pat.
* si une position se répète 3 fois à l'identique (consécutive ou non).
* sur un échec perpétuel.
* si 50 coups sont joués sans prise et sans mouvement de pion.
* si le matériel est insuffisant pour mater.

Le pat

Si le seul coup jouable est de mettre son roi en échec (ce qui est interdit) le roi est dit pat et la partie est nulle. Une position de pat peut arriver en fin de partie quand il ne reste que les rois et quelques pions.  
Volontairement provoquer une situation de pat peut parfois sauver une position désespérée.

L'échec perpétuel

Si le roi ne peut se soustraire aux échecs répétitifs donnés par le camp adverse la partie est déclarée nulle.

1. **LES FONCTIONALITES AJOUTER**
2. Effet de la lumière



*GLfloat light\_ambient[] ={ 0.0, 0.0, 0.0, 1.0 };*

*GLfloat light\_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };*

*GLfloat light\_specular[] = { 1.0, 1.0, 1.0, 1.0 };*

*GLfloat light\_position[] = {0.0, 5.0, 0.0, .7};*

*glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);*

*glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);*

*glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);*

*glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);*

*glEnable(GL\_LIGHTING);*

*glEnable(GL\_LIGHT0);*

1. Effet d’ombre

Dans cet application, l’effet d’ombre peut être activer ou désactiver par la touche « o »



**Le Principe :**

- On dessine d’abord l’objet,

*glEnable(GL\_LIGHTING); glEnable(GL\_TEXTURE\_2D);*

*glmDraw2( pmodel, GLM\_SMOOTH | GLM\_2\_SIDED | GLM\_MATERIAL | GLM\_TEXTURE);*

-On Créer une matrice qui projettera l'ombre sur le plan souhaitée. Les paramètres sont ceux de l'équation d'un plan : ax + by + cz + d = 0

*static GLfloat floorPlane[4];*

*static GLfloat floorShadow[4][4];*

*GLfloat light\_pos[] = { .3, .5, .2, 0.0 };*

*shadowMatrix(floorShadow, floorPlane, light\_pos);*

-Trouvez l’équation du plan : *ax + by + cz + d = 0*.

*findPlane(floorPlane, vfloor[1], vfloor[2], vfloor[3]);*

-Redessiner l’objet:

*glMultMatrixf((GLfloat \*) floorShadow);*

*glmDraw22( pmodel, GLM\_NONE);*

1. Effet de brouillard

De même, l’effet de brouillard peut être activer ou désactiver par la touche « b »



**Le Principe de l’effet brouillard:**

*-*On active le mode GL\_FOG

*glEnable(GL\_FOG);*

-On range le mode *GL\_EXP* dans la variable GL\_FOG\_MODE

*glFogi(GL\_FOG\_MODE, GL\_EXP);*

-On fait la même chose pour la couleur et la variable GL\_FOG\_COLOR

*GLfloat fogcolor[4] = {0.5, 0.5, 0.5, 1} ;*

*glFogfv(GL\_FOG\_COLOR, fogcolor);*

-On définit la densité du brouillard (plus ou moins épais)

*glFogf(GL\_FOG\_DENSITY, 0.05);*

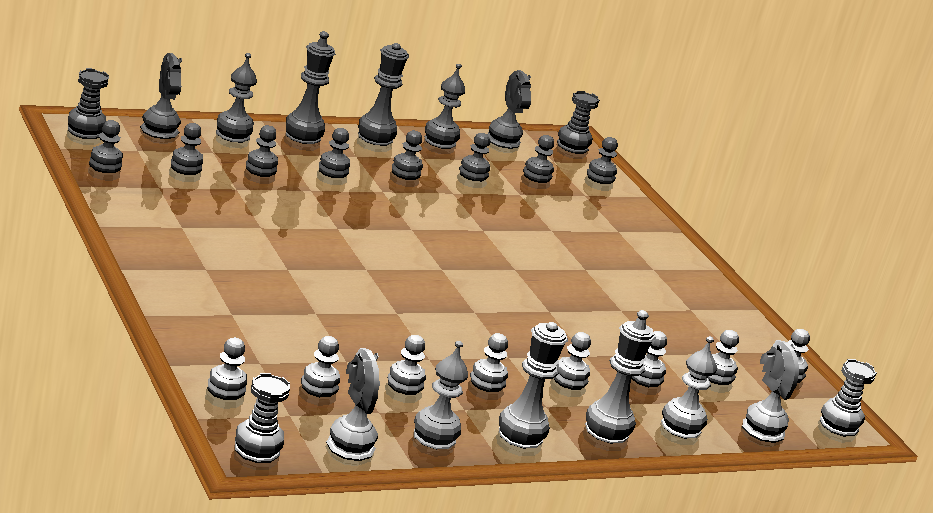
-En fin le Debut et fin du brouillard

*glFogf(GL\_FOG\_START, 1.0);*

*glFogf(GL\_FOG\_END, 5.0);*

1. **Le reflet:**

Pour activer ou désactiver le reflet, on utilise la touche « f » du clavier



**Le Principe du reflet:**

Le but est de redessiner la scène, avec un Y inversé. Mais le reflet va s'afficher partout. Nous allons donc utiliser le Stencil Buffer. Le principe est le suivant : Nous effaçons le stencil buffer en le rempliant de " GL\_FALSE». Ensuite nous dessinons notre eau dans ce tampon en remplaçant les " GL\_FALSE" par des " GL\_TRUE". Nous afficherons ensuite le reflet uniquement aux endroits ou le tampon de stencil est à " GL\_TRUE " !

- Sauvegarder la matrix : glPushMatrix  
- utilisation glScalef(1, -1, 1) pour inverser le dessin par rapport au plan Y=0  
- dessin de l'objet  
- Recharger la sauvegarde de la matrix : glPopMatrix  
- positions originales des sources lumineuses  
- dessin le support de l'objet qui servira de miroir.

1. Application de la texture

Nous importer l’objet sans texture. En s’inspirant du TP « lab26b » (celui de la pyramide), nous avons applique les texture avec la bibliothèque « tga », d’abord sur le échiquier et les case que nous avons dessiné nous-même (signalons qu’ils n’ont pas été importées, seule les pions ont été importées), ensuite sur le fond, background.

1. Déplacement des pions avec le picking

Chaque case est une structure qui contient son identifiant et l’identifiant du pion qu’il contient (-1 s’il ne contient pas de pion), ainsi que ses coordonnée.

Chaque pion aussi est une structure qui contient son identifiant, ses coordonnée qui sont ceux de la case où il se trouve.

Pour le déplacement des objets, il suffit de cliquer dans la case qui le contient puis dans la case cible, et l’objet effectue une translation vers la case cible.

La sélection s’effectue par le picking

Le principe ou l’algorithme du Picking :

1. *Initialiser un tampon de sélection, qui contiendra des données sur les objets sélectionnés*
2. *Sauvegarder quelque part les informations sur la fenêtre courante*
3. *Passez en mode GL\_SELECT*
4. *Initialiser le nom de pile*
5. *Restreindre la zone de dessin autour de la position de la souris*
6. *Redessiner les objets en les nommant*
7. *Récupérer le nombre de hits et obtenir les noms des objets sélectionnés*
8. Les cas possible d’un pion sélectionné lors du déplacement

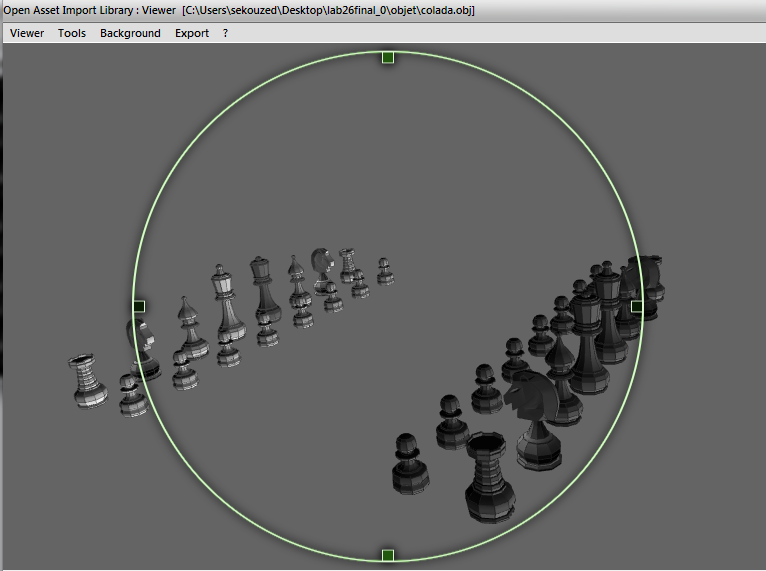
Chaque pion possède un tableau contenant les identifiants des cases possibles. Ce tableau est rempli lorsque le pion est sélectionné pour le déplacement. Ainsi les cases possibles seront représentées en couleur vive comme suggestion au joueur.



1. **REALISATION**

Chargement du model « échiquier »

A partir du logiciel « SkechUp » de google, on crée les groupes en les nommant, puis on l’export sous format « .obj ». La bibliothèque glm nous permettra d’importer ce model dans le projet.



*Le model .obj exporter a partir de SkechUp*

Modification que nous avons apportée à la bibliothèque glm :

- Modification apporter à la structure groupe

Nous avons ajouté une variable de plus à cette structure, à savoir GLfloat position[3], un tableau qui va contenir les coordonnées x, y, z du groupe dans le repère de la scène.

*typedef struct \_GLMgroup {*

*char\* name; /\* name of this group \*/*

*GLuint numtriangles; /\* number of triangles in this group \*/*

*GLuint\* triangles; /\* array of triangle indices \*/*

*GLuint material; /\* index to material for group \*/*

*GLfloat position[3]; /\*position of group\*/*

*struct \_GLMgroup\* next; /\* pointer to next group in model \*/*

*} GLMgroup;*

-Modification apporter à la fonction : glmDraw(GLMmodel\* model, GLuint mode)

-la fonction glmDraw2(GLMmodel\* model, GLuint mode)

Cette fonction dessine les éléments de l’objet groupe par groupe, alors nous avons ajoute la fonction translate pour translater chaque groupe par ces cordonnées stocker dans le tableau décrit ce dessus *(GLfloat position[3]) .*

GLvoid glmDraw2(GLMmodel\* model, GLuint mode)

{

*……code…..*

group = model->groups;

while (group) {

*……code…..*

glPushMatrix();

glTranslatef(group->position[0],0.0, group->position[1]);

*……code…..*

glBegin(GL\_TRIANGLES);

*……code…..*

glEnd();

glPopMatrix();

group = group->next;

}

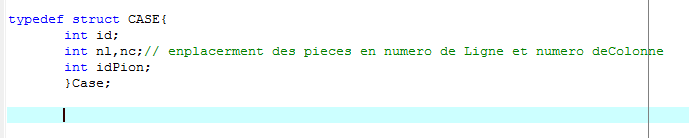
}

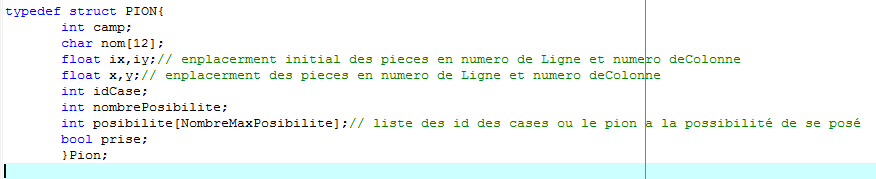
-la fonction glmDraw22(GLMmodel\* model, GLuint mode)

Cette fonction dessine les éléments sans aucun effet. Tous les effets de texture ou autre ont été supprimés. Elle dessine les objets uniquement en couleur sombre (noir). On la utilisé pour dessiner l’ombre de l’objet

Structure de données

On a deux structure Case et Pion. Puis on définit deux tableaux de cases et de de pions ou l’on va stocker les éléments, à savoir les 32 pions et 64 cases : lesCases[65] et lesPions[32] ;





Description des fonctions utilisées

* **init(void) :** permet d’initialiser la scène ainsi que l’ objet. Aussi, elle initialise la position de chaque groupe de l’objet.
* **DrawPlateau():** dessine le plateau de l’échiquier
* **DrawGrille();** dessine les case de l’échiquier
* **drawCasePosible();** dessine les cases possibles d’un pion
* **DrawSky();** dessine le background de la scène
* **bool chargerTextures(int it, char\* nom);** pour charger la texture
* **setPicking(int x, int y);** applique le picking
* **getPickingResult(GLint hits, GLuint \*names);** le résultat du picking et le déplacement
* **casePossible(int idPion**); pour trouver les cases possibles des pions
* **posibiliteDuTour(int idPion);** trouver les cases possibles **Du Tour**
* **posibiliteDuPion1(int idPion);** les cases possibles **Du Pion blanc**
* **posibiliteDuPion2(int idPion);** les cases possibles **Du Pion noir**
* **posibiliteDuCavalier(int idPion);** les cases possibles **Du Cavalier**
* **posibiliteDuFou(int idPion);** pour trouver les cases possibles **Du Fou**
* **posibiliteDuRoi(int idPion);** pour trouver les cases possibles **Du Roi**
* **posibiliteDeLaDame(int idPion);** les cases possibles **de la dame**
* **Reshape(int w, int h) :** gère le mode de vu de la scène( le viewport et le perspective)
* **Mouse(int button, int state, int x, int y) :** évènement souri
* **fleches(int key, int x, int y);** la rotation de la scène
* **Keyboard(unsigned char key, int x, int y);** évènement clavier
* **Display(void);** affichage de la scène
* **HelpDisplay(GLint ww, GLint wh);** affichage d’aide
* **HelpRenderBitmapString(float x, float y, void \*font, char \*string);** écriture sur la scène

1. **LES COMMANDES DU JEUX**

* **touche "H" ou "h" => Masquer ou Afficher l'Aide**
* **touche "O" ou "o" => Masquer ou Afficher l'ombre**
* **touche "B" ou "b" => Activer ou deactiver le brouillard**
* **touche "F" ou "f" => Activer ou deactiver le reflet**
* **touche "R" ou "r" => La rotation automatique**
* **touche "Fleches" => La rotation manuelle**
* **touche "Espace" => Affichage en plain ecran**
* **touche "Esc" => Quitter le jeux**
* **Pour le déplacement des objets, il suffit de cliquer dans la case qui le contient puis dans la case cible, et l’objet effectue une translation vers la case cible.**