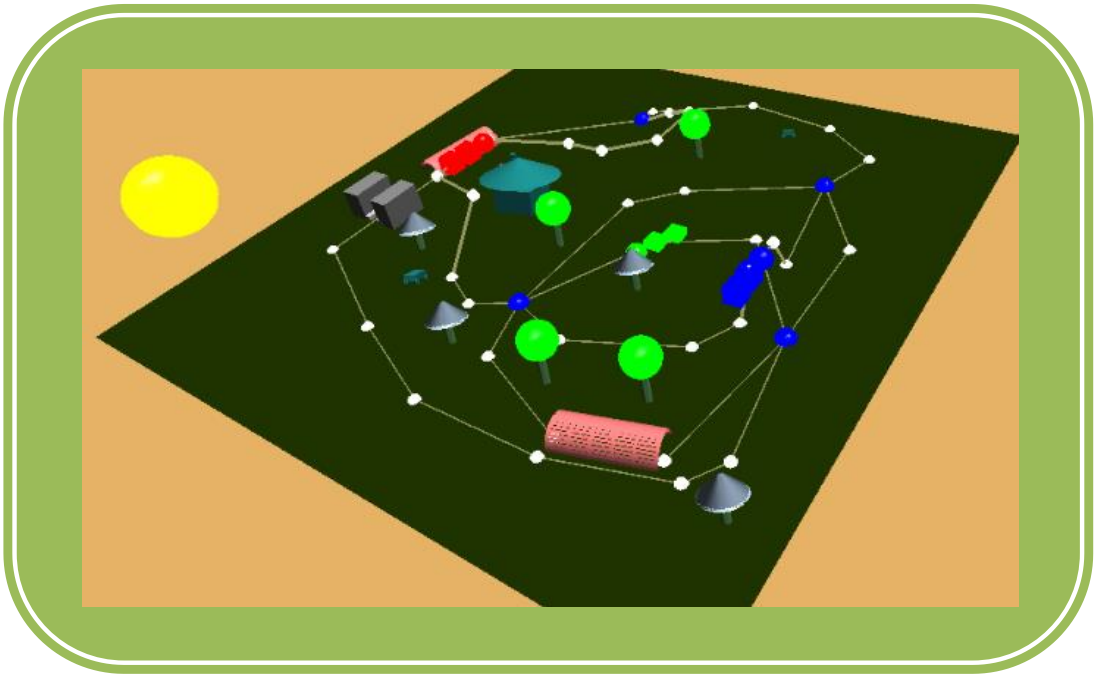
ENSG Géomatique

ÉCOLE NATIONALE
DES SCIENCES
GÉOGRAPHIQUES

Rapport Projet OpenGL

(2014-2015)



Réalisé par : ***Rhezali maryame*** **, Ait-omar Zakaria et Niamien David**

Plan

I. Installation des librairies

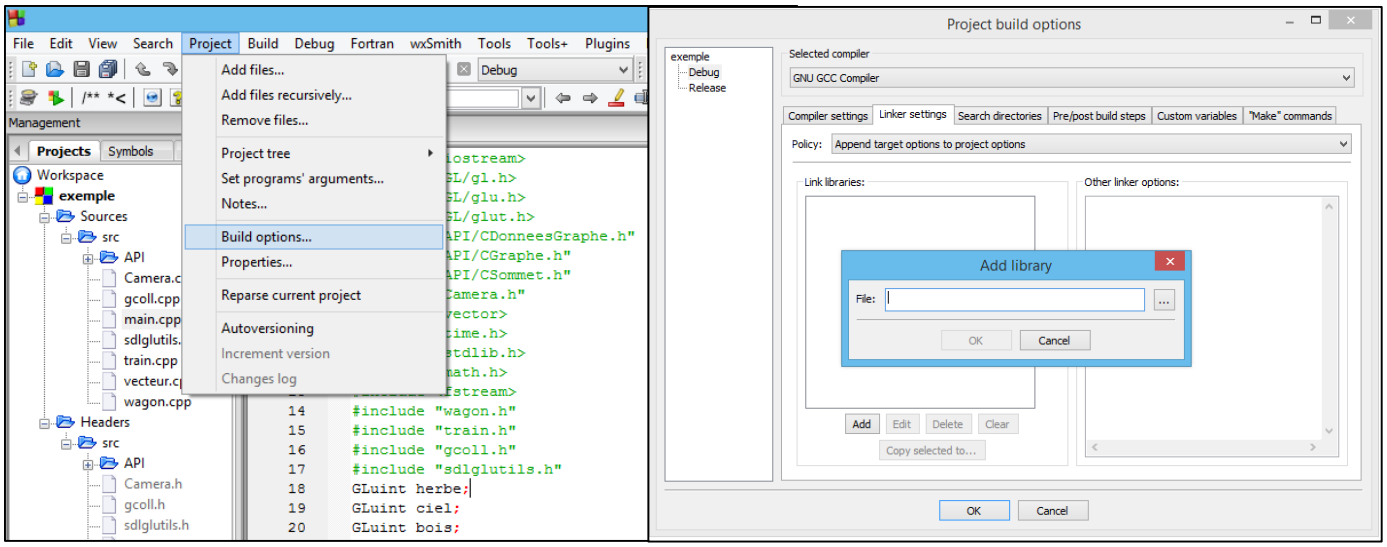
Dans ce projet nous avons opté pourCode ::Block comme environnement de développement sous
Windows,

1) Télécharger et installer Code::Blocks avec les compilateur MinGW : ici
2) Télécharger le fichier GLUT bin . Ouvrez le fichier GLUT bin , vous aurez besoin de copier:
   - glut32.dll à c: \ Windows \ System,
   - glut32.lib à c: \ program files \ mingw \ lib, et
   - glut.h à c: \ program files \ mingw \ include \ GL.

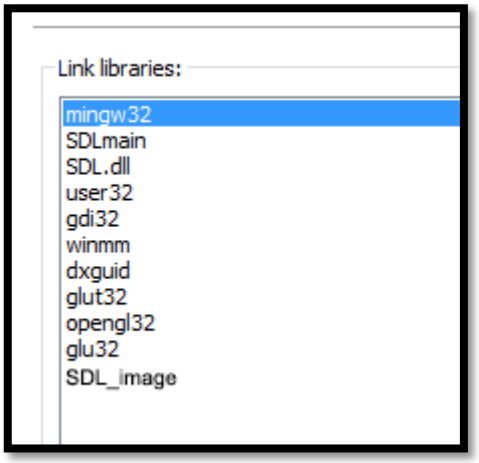
Ce sont des endroits défaut, vos chemins peuvent être différents. Mais fondamentalement, vous placez le .lib et .h à l'emplacement de votre compilateur (dans ce cas mingw). Le fichier .dll va dans le répertoire système de Windows.

Lorsque vous créez un nouveau projet, il faut ajouter des linkers au projet :

Cela se fait via le menu « Projet » => « Build Options » => onglet « Linker settings » :



Cliquer sur « Add » pour ajouter les librairies suivantes :



II. Dessin

1. *Graphe*

Graphe représente une combinaison des points et des points sommet et les arcs. Les points sont de type Points3f, CData2i. Pour dessiner les graphes nous avons utilisé la fonction «afficheDonnees()» .

La fonction « GL_LINE_STRIP » nous permet de récupérer les sommets initiaux et finaux et les points intermédiaires.

2. *Objets*

Pour décorer la scène on est amené à créer plusieurs objets :

Arbre à forme ronde / conique: la fonction qui dessine cet objet prend en paramètre la position basée sur les points annexes ainsi qu'un objet de type « GLUquadricObj »

Maison : la fonction qui dessine cet objet prend en paramètre la position basée sur les points annexes ainsi que les angles de rotation

Carrosse : la fonction qui dessine cet objet prend en paramètre la position basée sur les points annexes ainsi que les angles de rotation

Tunnel : la fonction qui dessine cet objet prend en paramètre la position basée sur les points annexes ainsi que les angles de rotation et un objet de type « GLUquadricObj »

Sol : la fonction qui dessine cet objet prend en paramètre la position basée sur les points annexes ainsi que les angles de rotation.

Soleil : la fonction qui dessine cet objet prend en paramètre la position basée sur les points annexes ainsi que les angles de rotation.

Gare : fonction qui dessine cet objet prend en paramètre la position basée sur les points annexes ainsi que les angles de rotation.

On appelle ces fonctions en leur fournissant les paramètres nécessaire pour ajouter l'objet à notre scène dans la fonction « affiche ».

3. *Trains/wagons*

Nous avons modélisé un train comme une suite de wagons qui se suivent, chaque wagon est représenté par deux cubes et une sphère.
Les couleurs et les formes sont gérés à l 'aide des fonctions « glcoulor3d » , « glutSolidCube » et « glutSolidShere ».

III. Gestion de l'application

1. *Gestions des collisions*

La gestion des collisions se résume en gestion de déplacement des trains dans la contrainte d'éviter que ces derniers se touchent, pour cela on a effectué un changement de vitesse lorsque la distance entre deux trains sur un même arc est inferieur à un seuil. Si deux trains sont sur le même sommet on arrête un train durant le déplacement de l'autre.

Afin de gérer les collisions, il faut savoir les cordonnées du déplacement des trains. Les déplacements sont gérés avec la fonction « Idle() » dont laquelle nous récupérons les coordonnées du train et du sommet d'arrivée, nous calculons après les déplacements en divisant chaque composante de ce vecteur par sa norme et nous multiplions par la vitesse du train et nous réinitialisons les coordonnées du train. Lors du passage par un point annexe nous incrémentons l'indice du point annexe suivant et lors du passage par un sommet nous réinitialisons les attributs du train .

2. Gestion de la vitesse

Le changement de vitesse est gére par la fonction « KeyListen() »dans laquelle on récupère les trains (T1,T2,T3) et après on change les vitesses des trains avec les touche du claviers.la méthode utiliser pour lier les touche au fonction de manipulation des vitesse est « switch » et la fonction qui fait l'interaction avec le clavier « glutKeyboardFunc() »
Les fonctionnalités des touches :

« v»:augmente la vitesse du train vert de 0,01
« x »:diminue la vitesse du train vert de 0,01
« b»:augmente la vitesse du train bleu de 0,01
« y »:diminue la vitesse du train bleu de 0,01
« r »:augmente la vitesse du train rouge de 0,01
« z »:diminue la vitesse du train rouge de 0,01
« q »:quitter la fenêtre

La fonction « glutPostRedisplay()» permet de réafficher les objets avec des nouvelles valeurs.

3. Gestion du camera

La camera est definit par deux paramètres :
-Œil de la caméra :

```
static int xcam ; static int ycam ; double distcam
```

-Points visé par la caméra :

```
static double xcentre ; static double ycentre ; static double zcentre ;
```

La fonction qui permet de visualiser l'objet est « «gluLookAt() » qui prend en paramètres l'œil de la camera, le point visé par la camera et le vecteur vertical.

Pour pouvoir déplacer la caméra et placer le point de vue n' importe où dans la scène, nous allons utiliser les flèche du clavier pour réorienter la caméra: déplacer le point visé par la camera.

« GLUT_KEY_UP »mouvement de la camera dans l'axe des x et des y d'un pas de 1
« GLUT_KEY_RIGHT » mouvement de la camera dans l'axe des x et des y respectivement d'un pas de 1 et de -1.
« GLUT_KEY_DOWN » mouvement de la camera dans l'axe des x et des y d'un pas de -1.
« GLUT_KEY_LEFT » mouvement de la camera dans l'axe des x et des y respectivement d'un pas de -1 et de 1.

Pour faire le zoom et le dé-zoom on incrémente par le click sur la touche « - » ou on dés-incrémente par le click sur la touche « + » le troisième paramètre de la position de la camera.

```
distcam = distcam+1;//zoom sur la la scène

distcam = distcam-1;//dé-zoom sur la scène
```

Conclusion : Ce projet nous a permit d'appliquer les différentes connaissances vues dans le cours d'OpenGl aussi que dans le cours c++. Ainsi il nous a permit de manipuler les multiples fonctionnalités d'OpenGl .

FIN