RAPPORT FINAL du PROJET de SEMESTRE en INFOGRAPHIE

Tugce Celik

Mon projet de terme pour l'infographie consistait à concevoir un personnage et à lui transmettre une texture. Mais j'ai eu des problèmes pour utiliser certaines bibliothèques sur macos, et j'ai dû changer de projet car je ne trouvais pas la texture, l'image, l'objet appropriés.

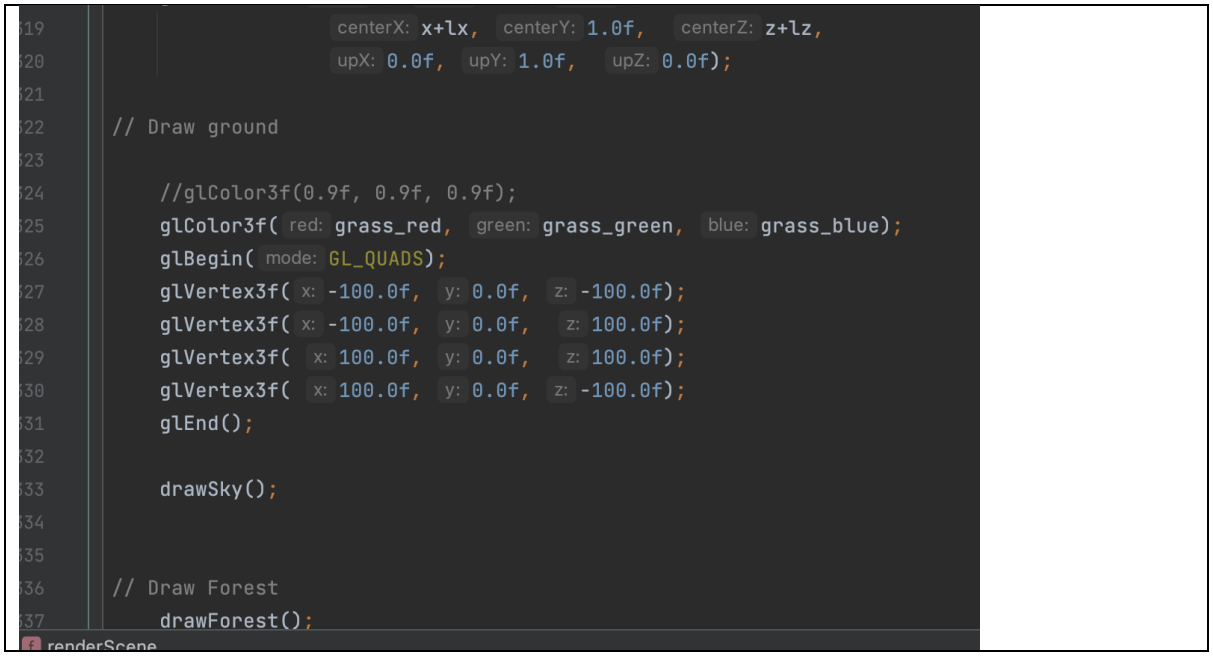
Le projet mentionné dans ce rapport consiste à créer une forêt simple en utilisant opengl et c++ et à modifier cette forêt en fonction des saisons.

La principale caractéristique que je voulais créer dans le projet était de marcher dans la forêt créée avec la caméra. J'ai pu le faire, mais j'aurais pu améliorer l'angle de vue en ajoutant le mouvement de la caméra dans l'axe y.

J'ai suivi ces étapes pour réaliser ce projet:

1- Déterminer les limites et les caractéristiques de la forêt et du ciel

La forêt créée a été construite sur le plan XZ avec la taille de 200x200. Des quads ont été tirés du sommet à cet effet.



```
                    centerX: x+lx,  centerY: 1.0f,   centerZ: z+lz,
                    upX: 0.0f,  upY: 1.0f,   upZ: 0.0f);

// Draw ground

    //glColor3f(0.9f, 0.9f, 0.9f);
    glColor3f( red: grass_red,  green: grass_green,  blue: grass_blue);
    glBegin( mode: GL_QUADS);
    glVertex3f( x: -100.0f,  y: 0.0f,  z: -100.0f);
    glVertex3f( x: -100.0f,  y: 0.0f,   z: 100.0f);
    glVertex3f(  x: 100.0f,  y: 0.0f,   z: 100.0f);
    glVertex3f(  x: 100.0f,  y: 0.0f,  z: -100.0f);
    glEnd();

    drawSky();


// Draw Forest
    drawForest();
renderScene
```

Afin d'ajuster et de changer la couleur du ciel, la fonction drawSky a été écrite et la fonction glutSolidCube() a été utilisée. La taille du cube a été choisie à 90 car ce n'est qu'à ces dimensions qu'un ciel plus précis est obtenu.

2- Créer des fonctions pour dessiner des éléments de la forêt

Il y a 5 types d'arbres dans la forêt que j'ai conçue.

Pour dessiner ces arbres, les méthodes glutSolidCone et glutSolidSphere ont été utilisées.

L'objectif principal est d'obtenir une arborescence en ajustant la taille et la position des cônes et des sphères.

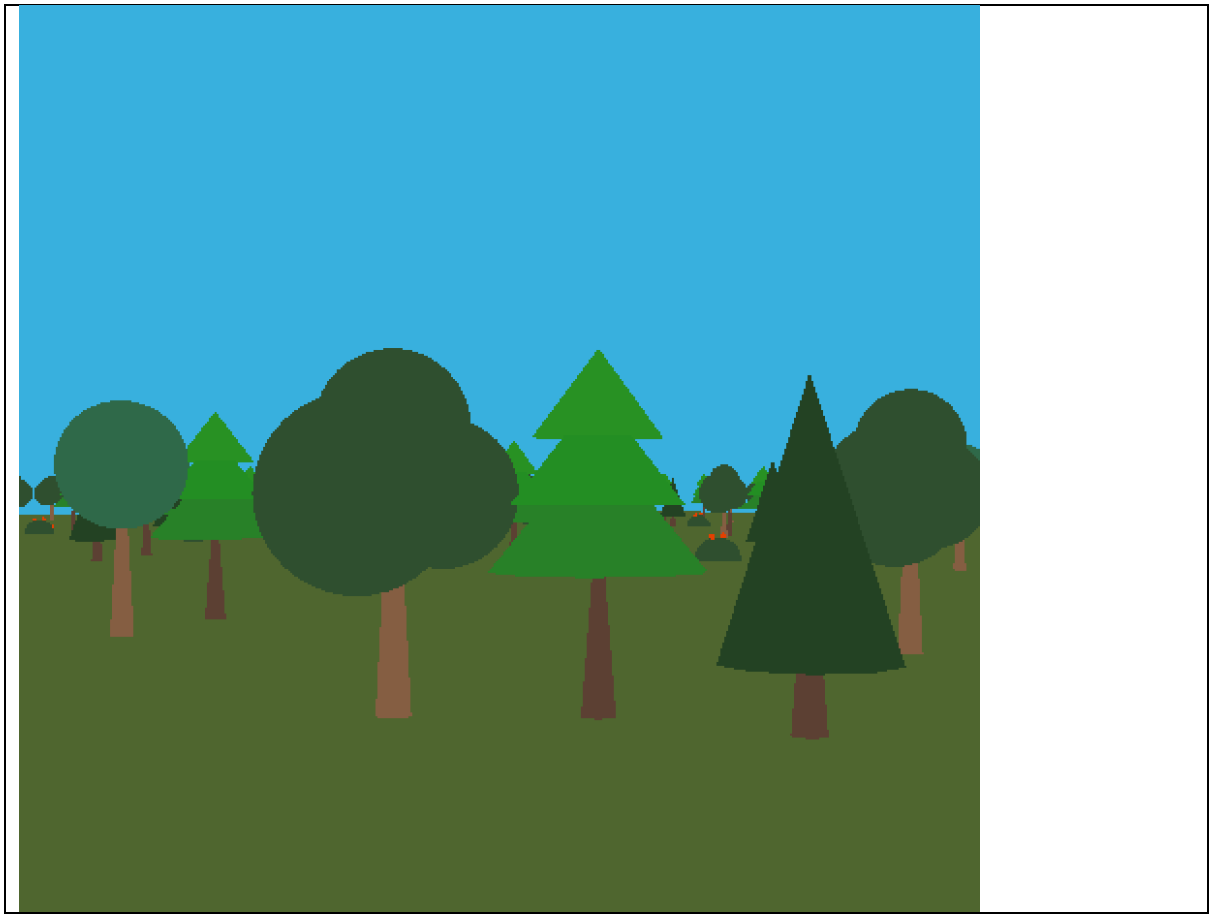
```
void Tree(){...}

void BubleTree(){...}

void pineTree(){...}

void pineTree2(){...}

void berryBush(){...}
```



Vous pouvez voir les types d'arbres ci-dessus.

La fonction drawForet() a été créée pour placer les arbres dans le champ. Avec cette fonction, l'emplacement du stylet est défini à l'aide de la boucle for imbriquée. De cette façon, les dessins ne se chevauchent pas.

Pour cela, glTranslatef(); est utilisé.

3- Faire les réglages de couleur pour les transitions saisonnières automatiquement

Des couleurs spéciales ont été attribuées à chaque saison pour assurer les transitions saisonnières.

Les couleurs des saisons d'automne, d'été et d'hiver ont été définies respectivement pour les touches 'a', 's' et 'w'.

glutKeyboardFunc(); Nous appelons la fonction processNormalKeys() que nous y avons écrite.

4- Définir la caméra à déplacer dans la zone définie et contrôler l'angle et les mouvements de la caméra avec les touches du clavier

Nous avons utilisé la fonction gluLookAt() pour définir la caméra. Nous avons assigné les paramètres de ces variables de fonction afin que nous puissions appliquer les commandes de changement de position dont nous avons besoin pour nous déplacer.

Tout d'abord, nous avons utilisé les touches fléchées pour déplacer la caméra.

Pendant que les touches gauche et droite font pivoter la caméra autour de l'axe Y, c'est-à-dire dans le plan XZ,

Les touches haut et bas déplacent la caméra vers l'avant et vers l'arrière dans la direction actuelle.

Nous avions besoin de l'angle lors de la rotation sur l'axe XZ. Nous avons calculé cela comme suit :

lx = sin(ang)

lz = -cos(ang)

Tout comme nous voulons convertir des coordonnées polaires en coordonnées euclidiennes. Puisque la valeur initiale est -1, lz est négatif. Nous tenons à souligner que lors de la mise à jour de lx et lz, la caméra ne bouge pas, la position de la caméra reste la même, seule la perspective change.

Et voici les nouvelles valeurs de x et z pour faire avancer la caméra

x = x + lx * fraction

z = z + lz * fraction

Une fraction est une application possible de la vitesse. Nous savons que (lx,lz) est un vecteur unitaire, donc si la fraction est maintenue constante, la vitesse sera également maintenue constante.

L'autre problème que j'ai rencontré après avoir fait cela était que je ne voulais pas que ma caméra bouge tant que le bouton était enfoncé. et quand j'ai arrêté d'appuyer sur la touche, mon appareil photo aurait dû s'arrêter aussi. Pour résoudre ce problème, nous désactiverons les rappels lorsqu'une répétition de clé se produit avec glutIgnoreKeyRepeat.

Lorsqu'une touche était enfoncée, une variable était définie sur une valeur différente de zéro.

Lorsque la touche a été relâchée, la variable a été remise à zéro.

D'autre part, comme il n'y a pas de rappel actif entre la touche enfoncée et la touche relâchée, nous devons vérifier ces variables dans la fonction de rendu et mettre à jour la position et l'orientation de la caméra en conséquence. Dans la section d'initialisation, nous avons deux nouvelles variables : deltaAngle et deltaMove . Ces variables contrôlent respectivement la rotation et le mouvement de la caméra. Lorsqu'il est différent de zéro, une action de la caméra se produit, lorsqu'il est nul, la caméra est immobile. Ces deux variables prennent la valeur zéro initiale, ce qui signifie que la caméra est initialement au repos. Au début de notre code, nous ajouterons deux variables pour suivre l'état de la clé, une pour l'orientation, deltaAngle, et l'autre pour la position de déplacement delta.

Grâce à ces processus, nous avons résolu notre problème.

5- Affectation de la touche ESC pour fermer le programme.

Fermer facilement le programme est tout aussi important que de l'exécuter. J'ai choisi la touche ESC pour fermer la fenêtre, et j'ai fait les définitions dans la fonction processNormalKeys() pour traiter la commande reçue du clavier.