Rapport de projet

Synthèse d'images

Master 2 en informatique visuelle

Année 2020/2021

Rédigé par : BENAISSA Rania

Matricule : 2015 0000 8321

I. Introduction

Ce présent document recense les étapes suivies pour concevoir une application de simulation d'un jeu de bowling. Cette application est conçue avec OpenGl 3 et regroupe toutes les notions vues en TP (fragments, gestion des évènements etc…).

II. Importation d'objets

Tous les objets de la scène sont des objets de la classe « Solid » qui à été implémentée de sorte à pouvoir stocker les informations clés sur un objet de format « obj » et d'être en mesure de l'afficher (gestion de la matrice des transformations géométriques, des VAO et des VBO etc..).

Tout objet de la scène est un modèle 3D au format « .obj » et qui est accompagné d'un fichier « Material » au format « .mtl ».

Pour pouvoir l'importer, j'ai créé un Parser (classe « OBJParser ») qui lit et récupère les informations contenues dans ces fichiers, à savoir :

- Les vertex
- Le vecteur normal associé à chaque vertex
- Les vecteurs de textures (si l'objet a une texture)
- Les vecteurs de couleur ambiante
- Les vecteurs de couleur diffuse
- Les vecteurs de couleur spéculaire et leurs paramètres de réflexion spéculaire « n ».

III. Eclairement

Grâce aux informations récupérées, on peut attribuer une couleur à chaque fragment selon sa position et l'éclairement de la scène. Tout cela est calculé au niveau du fragment shader, comme le montre la figure ci-dessous :

```
void main()
{

	//DANS CE PROGRAMME JE SUPPOSE QUE LA COULEUR DE LA LUMIERE EST BLANCHE
	vec3 norm = normalize(vnormal);

	// LUMIERE AMBIANTE
	float Iambiante = 0.4f;
	vec3 ambiante = lightA * Iambiante;

	//LUMIERE DIFFUSE:
	vec3 lightDir = normalize(lightPos - vpos);
	float diff = max(dot(norm, lightDir), 0.0);
	float Idiffuse = 0.5f;

	vec3 diffuse = diff * lightD * Idiffuse;

	// LUMIERE SPECULAIRE
	float specIntens = 1.0f;
	vec3 viewDir = normalize(camPos - vpos);
	vec3 reflectDir = reflect(-lightDir, norm);
	float spec = pow(max(dot(viewDir, reflectDir), 0.0), int(lightN));
	vec3 specular = spec * specIntens * lightS;

	// si c'est un objet avec une texture
	if (isTextured == 1.0f)
	{

		color = vec4(texture(myTexture, vtexture).xyz * (ambiante + diffuse) + specular, 1.0);
	}
	else
	{

		color = vec4(ambiante + diffuse + specular, 1.0);
	}
}
```

IV. Interactions et collisions

a. Interactions

- On peut pivoter la caméra vers la gauche ou vers la droite selon le point de visualisation de la caméra en appuyant sur les flèches (gauche et droite) du clavier.
- On peut aussi mouvoir la caméra vers le haut ou vers le bas selon le vecteur ViewUp en appuyant sur les flèches haut et bas du clavier.
- Il est possible faire des zoom arrière et avant en pressant respectivement la touche « z» et la touche « a » du clavier.
- On peut également déplacer la balle tout au long de l'axe des X en appuyant sur les touches « q » et « s » du clavier.
- Pour lancer la balle, il suffit d'effectuer un clic gauche de la souris et un curseur apparait pour permettre de donner une direction à la balle.
- Pour sélectionner la direction de la balle, cliquer à nouveau avec la souris (clic gauche toujours).
- Si toutes le quilles sont mises à terre, le jeu recommence à nouveau.

b. Collisions

- Après avoir orienté la balle grâce à la flèche verte, elle se lance vers les quilles.
- Pour pouvoir détecter les collisions de la balle avec les quilles, j'ai calculé mathématiquement certains attributs tels que : la position de la balle sur les axes X et Y, le vecteur de direction de la balle, le vecteur normal des quilles et les angles entres eux ainsi j'ai pu approximer de la façon la plus réaliste possible : la collision entre la balle et les quilles.
- Si la balle touche une quille, celle-ci est translatée vers le haut (axe des Y) et est pivotée de -90° sur l'axe des X avec une direction réaliste, calculée selon l'axe des X et l'axe des Y.

V. Conclusion

Grâce à ce projet, j'ai appris à :

- ✓ Importer et mouvoir des objets dans OpenGL3.
- ✓ Gérer l'éclairement d'une scène selon le modèle d'éclairement étudié en cours.
- ✓ Gérer les interactions avec la scène et les collisions entre les objets.