Université des sciences et de la technologie Houari Boumediene

Master 2 IV

IA & SD

Module : Synthèse d'images

Rapport sur le jeu de puzzle River Crossing

ABDALLAH Abderraouf
HANAGRIA Issam
AMOURA Youcef

Date : 28 janvier 2024

1 Introduction

Le jeu du passage de la rivière est un casse-tête classique impliquant un fermier devant transporter un loup, un mouton et un chou d'une rive à l'autre d'une rivière. Les règles du jeu stipulent que le fermier ne peut laisser le loup seul avec le mouton ni le mouton seul avec le chou, sous peine que le loup mange le mouton ou que le mouton mange le chou.

L'objectif de ce projet est de mettre en œuvre ce jeu en utilisant les technologies de rendu graphique, notamment OpenGL, GLFW, Glad, et GLM. De plus, le rapport présente l'utilisation de bibliothèques supplémentaires pour la gestion du texte, la création d'une interface utilisateur et d'autres fonctionnalités du jeu.

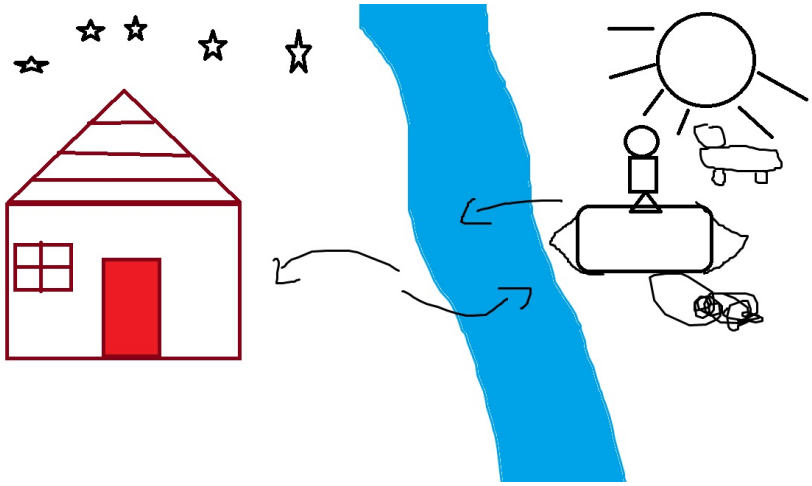

FIGURE 1 – Capture d'écran illustrant la conception initiale du jeu.

2 Project Setup

Avant de commencer le développement du jeu du passage de la rivière, une configuration initiale du projet a été nécessaire. Les étapes suivantes décrivent la mise en place du projet, y compris l'installation des bibliothèques essentielles utilisées pour le rendu graphique et d'autres fonctionnalités :

1. **Installation d'OpenGL, GLFW, Glad et GLM :** Les bibliothèques OpenGL, GLFW (une bibliothèque de gestion de fenêtres), Glad (un gestionnaire de chargement OpenGL) et GLM (bibliothèque mathématique pour OpenGL) ont été installées pour fournir la base du rendu graphique du jeu.

2. **Gestion des images avec STB_Image :** La bibliothèque STB_Image a été intégrée pour faciliter le chargement des textures et des images nécessaires au rendu du jeu.

```
void InitTexture(void)
{
    for (int i = 0; i < nObjects; ++i)
    {
        int width, height, channels;
        unsigned char *image = stbi_load(
           filenamesTextures[i], &width, &height, &
           channels, STBI_rgb);

        // Generate and bind texture
        ...
        // Free the image data
        stbi_image_free(image);
    }
}
```

Listing 1 – Extrait de code de chargement des textures.

3. **Intégration de gltext pour la gestion des polices :** La bibliothèque gltext a été utilisée pour la manipulation des polices de texte, permettant d'afficher des informations à l'utilisateur.

```
void drawtext_func()
{
    gltBeginDraw();

    gltColor(1.0f, 1.0f, 1.0f, 1.0f);
    GLTtext *text = gltCreateText();
    gltSetText(text, text_buffer);
    gltDrawText2DAligned(text, 540.0f, 360.0f, 5.0f,
       GLT_CENTER, GLT_CENTER);

    gltEndDraw();
}
```

Listing 2 – Extrait de code permettant d'afficher des informations à l'utilisateur

4. **Utilisation de SFML pour la gestion du son :** La bibliothèque SFML a été ajoutée pour gérer les aspects sonores du jeu, offrant des fonctionnalités telles que la lecture d'effets sonores ou de musique de fond.

```
1
2  // global variables
3      sf::SoundBuffer wolfEatBuffer;
4      sf::Sound wolfEatSound;
5      GLboolean isPlayingWolf = false;
6
7  // load the sounds
8      wolfEatBuffer.loadFromFile("sounds/wolf_eat_sound
           .wav");
9      wolfEatSound.setBuffer(wolfEatBuffer);
10
11 // play the sound
12     wolfEatSound.play();
13     isPlayingWolf = true;
```

Listing 3 – Extrait de code de la gestion du son.

5. **Intégration d'Assimp pour le chargement des modèles 3D :** La bibliothèque Assimp a été utilisée pour le chargement des modèles 3D, facilitant l'intégration d'objets complexes dans le jeu.

```
1  for (int i = 0; i < nObjects; ++i)
2  {
3      loadModel(filenamesModels[i], meshVertices[i],
           meshIndices[i], meshMaterials[i]);
4  }
5
6  // Function to load model with Assimp
7  void loadModel(&path, &meshVertices, &meshIndices, &
       meshMaterials)
8  {
9      Assimp::Importer importer;
10     const aiScene *scene = importer.ReadFile(path,
11     .
12     .
13     .
14 // Store the vertices, indices, and material for the
       current mesh
15     meshVertices.push_back(vertices);
16     meshIndices.push_back(indices);
17     meshMaterials.push_back(material);
18     }
```

Listing 4 – Extrait de code de chargement des textures.

Cette configuration initiale a fourni la base nécessaire pour le développement du jeu, permettant d'exploiter les fonctionnalités avancées des bibliothèques graphiques et sonores tout au long du processus de création du jeu du passage de la rivière.

3 Rendu Graphique

La mise en place du rendu graphique du jeu a été réalisée en utilisant OpenGL, GLFW, Glad et GLM. Les étapes suivantes détaillent la configuration et l'implémentation du rendu graphique :

1. **Initialisation de GLFW et OpenGL :** Le projet a commencé par l'initialisation de GLFW et le chargement de la version OpenGL 3.3.

2. **Chargement des modèles 3D :** Les modèles 3D requis pour le jeu ont été obtenus en explorant des sites Web proposant des modèles 3D, tels que `https://sketchfab.com`. Après avoir sélectionné des modèles appropriés, des modifications ont été apportées à l'aide du logiciel Blender pour répondre aux besoins spécifiques du jeu. Les modèles ont ensuite été exportés au format .obj, conservant leurs propriétés telles que la position, la texture, les matériaux, les normales et la couleur.

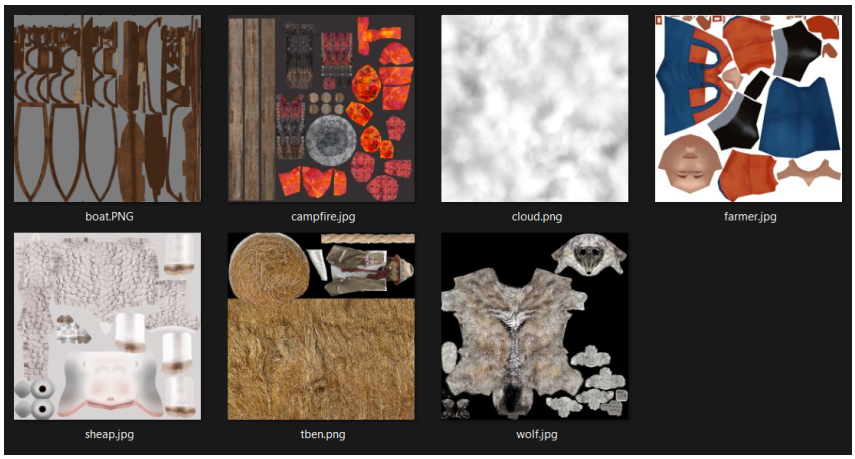

FIGURE 2 – Capture d'écran sur les textures des objets utilisées.

Un processus itératif a été employé pour charger chaque modèle dans le jeu, garantissant une intégration cohérente avec les propriétés associées.

3. **Création des tampons de données :** Les tampons OpenGL ont été créés et liés pour stocker les données des modèles 3D.

4. **Chargement des shaders :** Les shaders ont été chargés à l'aide de la fonction `LoadShaders`, créant ainsi le programme de shaders utilisé pour le rendu. Le programme de shaders intègre une gestion avancée de l'éclairage, incluant les composantes lumière ambiante, diffuse et spéculaire.

   Pour chaque fragment rendu, les calculs d'éclairage sont effectués en utilisant les propriétés matérielles telles que l'ambiante, la diffuse et la spéculaire. Ces propriétés sont ensuite combinées avec les valeurs de lumière ambiante, diffuse et spéculaire, créant un rendu réaliste et dynamique.

   Le code shader utilisé intègre également la possibilité d'utiliser une texture pour le rendu. La couleur finale du fragment est déterminée en tenant compte des propriétés du matériau, de la position de la lumière, de la direction de la lumière, de la vue de la caméra, et des coordonnées de texture.

```
out vec4 FragColor;
void main() {
    // Calculs de l'ambiante, de la diffuse et de la
       speculaire
    // ...

    vec3 result = ambient + diffuse + specular;
    FragColor = vec4(result, texColor.a);
}
```

Listing 5 – Extrait de code de fragmentShader.

5. **Initialisation des modèles et des textures :** Les modèles ont été initialisés, et les textures nécessaires ont été chargées pour être utilisées dans le rendu.

6. **Configuration des callbacks GLFW :** Les fonctions de rappel GLFW ont été configurées pour gérer les entrées utilisateur, telles que les touches du clavier et la position du curseur.

7. **Définition de la couleur d'effacement et activation du test de profondeur :** La couleur d'effacement a été définie, et le test de profondeur a été activé pour assurer un rendu approprié.

8. **Boucle de rendu principale :** Une boucle principale a été mise en place pour effectuer le rendu de la scène à chaque itération, tout en prenant en compte les entrées utilisateur.

```
while (!glfwWindowShouldClose(window))
{
    cameraMovement();
    renderingScene(MatrixID, TextureID);

    // Swap buffers and poll events
    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

Listing 6 – Extrait de code de la boucle principale de rendu.

Pour illustrer l'implémentation, des extraits de code et des captures d'écran du rendu graphique peuvent être consultés ci-dessous.

FIGURE 3 – Capture d'écran du rendu graphique du jeu.

4 Rendu de Texte

Le rendu de texte dans le jeu a été réalisé en utilisant la bibliothèque gltext. Les étapes suivantes décrivent l'implémentation du rendu de texte :

1. **Intégration de gltext :** La bibliothèque gltext a été intégrée dans le projet pour faciliter le rendu de texte à l'écran.
2. **Initialisation de la police de caractères :** Une police de caractères appropriée a été choisie et initialisée pour être utilisée avec gltext.
3. **Positionnement du texte :** Les coordonnées et la taille du texte à afficher ont été définies en fonction des besoins du jeu.
4. **Rendu du texte :** Le texte a été rendu à l'écran à l'aide des fonctionnalités de la bibliothèque gltext.

Pour illustrer l'implémentation, des extraits de code et des captures d'écran du rendu de texte peuvent être consultés ci-dessous.

FIGURE 4 – Capture d'écran du rendu de texte dans le jeu.

```
// Utilisation de gltext pour le rendu de texte
drawText = true;
snprintf(text_buffer, sizeof(text_buffer), "W:wolf t:tben
    k:kebch");
snprintf(text_buffer2, sizeof(text_buffer3), "Press Enter
    to start");
```

Listing 7 – Exemple de code pour le rendu de texte avec gltext.

5 Logique du Jeu

La logique du jeu de la traversée de la rivière a été implémentée en respectant les règles du puzzle. Voici comment la logique du jeu a été intégrée dans le code :

1. **Définition des objets :** Les objets du jeu, tels que le fermier, le loup, le mouton et le chou, ont été représentés en tant qu'entités dans le code.
2. **Déplacement des objets :** Les règles du puzzle ont été respectées lors de la mise en œuvre des déplacements des objets. Le fermier peut transporter un seul objet à la fois, et il ne peut pas laisser le loup seul avec le mouton ni le mouton seul avec le chou.
3. **Gestion de l'état du jeu :** L'état du jeu a été suivi pour déterminer si le puzzle a été résolu avec succès ou s'il y a eu une violation des règles.
4. **Affichage de messages :** Des messages informatifs ont été affichés à l'utilisateur pour l'informer des actions possibles et de l'état actuel du jeu.

Pour mieux comprendre l'implémentation, des extraits de code pertinents et des captures d'écran du jeu en cours peuvent être trouvés ci-dessous.

(a) Capture d'écran en cas de défaite.

(b) Capture d'écran en cas de victoire.

FIGURE 5 – Captures d'écran illustrant la logique du jeu en action.

6 Conclusion

Ce projet a implémenté avec succès le jeu du casse-tête de la traversée de la rivière en utilisant plusieurs technologies telles qu'OpenGL, GLFW, Glad, GLM et d'autres bibliothèques. Voici un récapitulatif des points clés du projet :

- **Objectif du jeu :** Le jeu implémente le casse-tête classique de la traversée de la rivière, mettant en scène un fermier, un loup, un mouton et un chou. L'objectif est de transporter chaque élément d'une rive à l'autre sans laisser le loup seul avec le mouton ou le mouton seul avec le chou.
- **Technologies utilisées :** Le projet a utilisé OpenGL pour le rendu graphique, GLFW pour la gestion de la fenêtre, Glad pour charger les fonctions OpenGL, GLM pour les mathématiques, et d'autres bibliothèques telles que gltext et assimp pour le rendu de texte et le chargement d'objets 3D respectivement.
- **Configuration initiale du projet :** La configuration initiale a impliqué l'installation et la configuration de plusieurs bibliothèques, notamment OpenGL, GLFW, Glad, GLM, gltext et assimp. Les étapes détaillées sont présentées dans la section *Project Setup*.
- **Rendu graphique :** Le rendu graphique a été mis en place avec OpenGL. Des modèles 3D ont été chargés à l'aide de la bibliothèque assimp, et des textures ont été appliquées pour améliorer la qualité visuelle. Des captures d'écran illustrant la logique du jeu sont présentées dans la section *Graphics Rendering*.
- **Rendu de texte :** L'implémentation du rendu de texte a été réalisée avec la bibliothèque gltext. Des détails sur la manière dont le texte a été intégré à la fenêtre du jeu sont fournis dans la section *Text Rendering*.
- **Logique du jeu :** La section *Game Logic* détaille comment les règles du casse-tête ont été incorporées dans le code. Les contrôles de la caméra, la gestion des entrées clavier et la logique de déplacement des éléments du jeu y sont expliqués.
- **Captures d'écran :** Les captures d'écran illustrant la logique du jeu en action sont présentées dans la section *Game Logic*. Ces captures montrent des scénarios de victoire et de défaite.
- **Défis rencontrés :** Pendant l'implémentation, certains défis ont pu être rencontrés, tels que la gestion des mouvements des éléments du jeu de manière logique tout en respectant les règles du casse-tête.

En conclusion, ce projet a fourni une expérience pratique dans le développement de jeux en utilisant des bibliothèques graphiques et des technologies associées. Il a démontré l'application des concepts OpenGL dans un contexte de jeu de puzzle classique.