Développement d'une Application Interactive 3D du Jeu de Société Mancala avec OpenGL

Mini-Projet de Programmation Graphique 3D

Ayman Gharib Amine Izoughagen

15 janvier 2026

Table des matières

1 Introduction

La programmation graphique 3D constitue un domaine fondamental de l'informatique moderne, permettant la création d'expériences visuelles immersives et interactives. Dans le cadre de ce mini-projet universitaire, nous avons développé une application interactive tridimensionnelle du jeu de société Mancala en utilisant OpenGL, l'une des bibliothèques graphiques les plus répandues et performantes de l'industrie.

Ce projet s'inscrit dans un objectif pédagogique ambitieux visant à maîtriser les concepts fondamentaux de la programmation graphique en temps réel. Il permet d'acquérir des compétences essentielles telles que la manipulation des pipelines de rendu, la gestion des transformations géométriques, l'implémentation de modèles d'éclairage, ainsi que la conception d'interfaces utilisateur interactives en trois dimensions.

Le choix du jeu Mancala comme support d'implémentation s'avère particulièrement pertinent pour plusieurs raisons. Premièrement, ce jeu millénaire offre une structure géométrique simple et régulière, facilitant la modélisation 3D tout en permettant de se concentrer sur les aspects techniques du rendu graphique. Deuxièmement, la nature stratégique du Mancala nécessite une logique de jeu cohérente qui doit être parfaitement synchronisée avec la représentation visuelle. Enfin, l'interaction tour par tour permet d'explorer en profondeur les mécanismes de sélection d'objets, de détection de collisions et de feedback visuel.

L'application développée propose une expérience complète intégrant non seulement le rendu graphique et la logique de jeu, mais également des fonctionnalités avancées telles que la personnalisation thématique, différents modes de rendu, et un système d'éclairage dynamique. Ce rapport présente l'ensemble du processus de conception, d'implémentation et d'optimisation de cette application, en détaillant les choix techniques effectués et les solutions apportées aux défis rencontrés.

2 Présentation du jeu Mancala

Le Mancala désigne une famille de jeux de société africains millénaires, également connus sous diverses appellations selon les régions. Il s'agit d'un jeu de stratégie à deux joueurs qui se déroule sur un plateau comportant plusieurs cavités appelées puits, disposées en deux rangées parallèles. Chaque joueur contrôle une rangée de puits, généralement au nombre de six par côté, auxquels s'ajoutent deux cavités plus grandes situées aux extrémités du plateau, appelées magasins ou kalah.

La configuration initiale du jeu place un nombre égal de pierres, graines ou billes dans chaque puits, traditionnellement quatre par cavité. Les magasins, quant à eux, démarrent vides et servent à accumuler les points de chaque joueur au cours de la partie. L'objectif principal consiste à capturer le plus grand nombre de pierres possible et à les stocker dans son propre magasin.

Le déroulement d'un tour suit des règles précises qui définissent la mécanique du jeu. Lorsqu'un joueur choisit un puits de sa rangée contenant des pierres, il doit ramasser l'intégralité des pierres présentes dans ce puits et les distribuer une à une dans les cavités suivantes, en progressant dans le sens antihoraire autour du plateau. Le joueur dépose une pierre dans chaque puits consécutif, y compris dans son propre magasin lorsqu'il le croise, mais en ignorant le magasin adverse.

Plusieurs règles spéciales enrichissent la stratégie du jeu. Si la dernière pierre distribuée

atterrit dans le magasin du joueur actif, celui-ci obtient un tour supplémentaire. Si elle atterrit dans un puits vide de son propre côté, le joueur capture cette pierre ainsi que toutes les pierres présentes dans le puits directement opposé du côté adverse, et place l'ensemble dans son magasin. La partie se termine lorsqu'un joueur ne possède plus aucune pierre dans ses puits, moment où l'adversaire récupère toutes les pierres restantes de son côté.

Pour une implémentation informatique fidèle, plusieurs éléments essentiels doivent être pris en compte. La structure de données doit représenter avec précision l'état du plateau, incluant le contenu de chaque puits et magasin ainsi que l'identification du joueur actif. La logique de distribution des pierres nécessite une gestion rigoureuse de l'algorithme de parcours circulaire du plateau. Le système doit également vérifier les conditions de capture, détecter les situations de tour supplémentaire, et identifier correctement les conditions de fin de partie. Enfin, l'interface doit permettre une sélection intuitive des puits tout en empêchant les actions invalides, comme la sélection d'un puits vide ou appartenant à l'adversaire.

3 Analyse du problème

L'implémentation d'une application interactive 3D du jeu Mancala avec OpenGL soulève plusieurs défis techniques et fonctionnels qui nécessitent une analyse approfondie. Cette section examine les besoins essentiels du projet et les contraintes à prendre en compte pour garantir une expérience utilisateur satisfaisante.

D'un point de vue fonctionnel, l'application doit permettre aux joueurs d'interagir naturellement avec le plateau de jeu dans un environnement tridimensionnel. Cela implique la mise en place d'un système de sélection d'objets capable d'identifier avec précision le puits choisi par l'utilisateur au moyen de la souris. Le système doit fournir un feedback visuel immédiat pour indiquer les éléments interactifs et guider les actions du joueur. La logique de jeu doit s'exécuter de manière cohérente avec les règles du Mancala, en gérant correctement les tours des joueurs, la distribution des pierres, les captures et les conditions de victoire.

Le rendu 3D en temps réel constitue un aspect central du projet et impose des exigences techniques spécifiques. L'application doit maintenir un taux de rafraîchissement élevé, idéalement soixante images par seconde, pour garantir la fluidité des animations et la réactivité des interactions. OpenGL, bien que puissant, nécessite une gestion minutieuse du pipeline graphique, depuis la définition des vertex shaders jusqu'au rasterisation finale. Les transformations géométriques doivent être appliquées correctement pour positionner les objets dans la scène, contrôler la caméra et projeter l'espace tridimensionnel sur l'écran bidimensionnel.

La cohérence visuelle représente un enjeu majeur pour l'immersion du joueur. Le modèle d'éclairage choisi doit mettre en valeur la géométrie des objets tout en préservant leur lisibilité. Les matériaux appliqués aux différents éléments du jeu doivent créer une hiérarchie visuelle claire, permettant de distinguer rapidement les puits des magasins et les pierres du plateau. Le système de thèmes graphiques enrichit l'expérience en offrant des variations esthétiques, ce qui nécessite une architecture flexible capable de modifier dynamiquement les propriétés visuelles de la scène.

La gestion des états du jeu exige une synchronisation parfaite entre la représentation logique et la représentation graphique. Toute modification de l'état interne du jeu, qu'il s'agisse du déplacement d'une pierre ou de la mise à jour du score, doit se refléter immédiatement dans le rendu visuel. Cette synchronisation devient particulièrement critique

lors de l'animation de la distribution des pierres, où plusieurs objets doivent se déplacer de manière coordonnée tout en maintenant la cohérence de l'état du jeu.

Les contraintes de performance liées à OpenGL méritent une attention particulière. Bien que les scènes du Mancala demeurent relativement simples comparées aux applications graphiques modernes, l'optimisation reste essentielle. Le nombre d'appels de rendu doit être minimisé en regroupant les objets similaires et en utilisant efficacement les buffers d'OpenGL. La complexité géométrique des modèles 3D doit être adaptée aux capacités de rendu en temps réel, privilégiant des maillages optimisés sans sacrifier la qualité visuelle. La gestion de la mémoire graphique nécessite une attention constante pour éviter les fuites et garantir des performances stables.

L'interaction utilisateur soulève également des questions ergonomiques importantes. La caméra doit offrir des angles de vue adaptés permettant d'apprécier pleinement la scène tridimensionnelle tout en conservant une vue fonctionnelle du plateau de jeu. Les contrôles doivent être intuitifs, avec des raccourcis clavier pour les fonctionnalités avancées et une manipulation à la souris pour les actions principales. L'interface graphique intégrée doit afficher les informations essentielles comme les scores et l'indication du joueur actif, sans encombrer l'espace visuel ni détourner l'attention du jeu.

4 Conception générale de l'application

L'architecture de l'application repose sur une séparation claire des responsabilités, organisant le code en modules distincts qui interagissent de manière cohérente. Cette approche modulaire facilite non seulement le développement et la maintenance, mais permet également d'isoler les différentes préoccupations techniques inhérentes à un projet de programmation graphique interactive.

Le premier module fondamental concerne la logique de jeu, responsable de l'implémentation des règles du Mancala indépendamment de toute considération graphique. Ce module maintient l'état complet du jeu, incluant le contenu de chaque puits et magasin, l'identification du joueur actif, et l'historique des coups si nécessaire. Il expose des méthodes permettant d'exécuter les actions de jeu, comme la sélection d'un puits et la distribution des pierres selon les règles établies. Cette séparation garantit que la logique reste testable et réutilisable, indépendamment du système de rendu choisi.

Le module de rendu graphique constitue le cœur technique du projet, orchestrant toutes les opérations liées à OpenGL. Il gère l'initialisation du contexte graphique, la compilation des shaders, la création des buffers de géométrie, et l'exécution de la boucle de rendu. Ce module traduit l'état abstrait du jeu en représentation visuelle tridimensionnelle, en positionnant les objets dans la scène selon leurs coordonnées logiques et en appliquant les transformations appropriées. Il est également responsable de la gestion du modèle d'éclairage et des différents modes de rendu offerts par l'application.

La gestion des entrées utilisateur forme un troisième pilier architectural essentiel. Ce module capture les événements générés par le clavier et la souris, les interprète et les traduit en commandes compréhensibles par les autres composants de l'application. Pour les interactions au clavier, il gère les raccourcis permettant de basculer entre les modes de rendu, d'activer ou désactiver l'éclairage, de changer de thème graphique, et de contrôler la caméra. Pour les interactions à la souris, il implémente un système de picking permettant d'identifier l'objet 3D situé sous le curseur et de déclencher les actions appropriées lorsqu'un puits est sélectionné.

Le module d'interface graphique complète l'architecture en superposant des éléments informatifs bidimensionnels sur la scène tridimensionnelle. Utilisant généralement une bibliothèque comme ImGui ou un système de rendu 2D personnalisé, ce module affiche le score des joueurs, indique quel joueur doit jouer, présente des messages d'aide, et peut inclure des statistiques sur la partie en cours. Cette interface doit s'intégrer harmonieusement au rendu 3D sans perturber l'expérience visuelle.

La boucle principale de rendu orchestre l'exécution coordonnée de ces différents modules. À chaque itération, elle suit un cycle précis qui commence par le traitement des événements d'entrée, permettant de mettre à jour l'état de l'application en réponse aux actions de l'utilisateur. Elle calcule ensuite le temps écoulé depuis la dernière image, information cruciale pour garantir des animations fluides indépendantes de la fréquence de rafraîchissement. La boucle met à jour la logique de jeu si nécessaire, exécute les animations en cours, puis procède au rendu de la scène 3D en effaçant les buffers, en configurant les shaders, en dessinant tous les objets visibles, et en superposant l'interface graphique. Enfin, elle présente l'image complétée à l'écran en échangeant les buffers d'affichage.

Cette architecture favorise une grande flexibilité, permettant d'ajouter facilement de nouvelles fonctionnalités ou de modifier des aspects spécifiques sans affecter l'ensemble du système. Par exemple, l'ajout d'un nouveau mode de rendu ne nécessite que des modifications dans le module graphique, tandis que l'introduction d'une variante des règles du Mancala n'impacte que le module de logique de jeu. La séparation entre logique et présentation facilite également le débogage, chaque module pouvant être testé et validé indépendamment.

5 Modélisation 3D et assets

La création des modèles tridimensionnels constitue une étape fondamentale dans le développement de l'application, déterminant à la fois l'esthétique finale et les performances de rendu. La modélisation du jeu Mancala nécessite de représenter trois types d'éléments principaux : le plateau de jeu, les puits et magasins qui le composent, et les pierres qui constituent les jetons mobiles.

Le plateau de jeu forme la base structurelle de la scène et peut être modélisé selon plusieurs approches. Une méthode simple consiste à créer une surface rectangulaire légèrement surélevée, éventuellement avec des bords délimitant l'aire de jeu. Pour enrichir le réalisme visuel, des détails géométriques peuvent être ajoutés, tels qu'un cadre en relief ou des ornements décoratifs inspirés des plateaux traditionnels. La simplicité géométrique du plateau permet de maintenir un nombre réduit de polygones tout en offrant une base solide pour la disposition des autres éléments.

Les puits représentent les cavités dans lesquelles reposent les pierres et constituent les points d'interaction principaux avec le joueur. Géométriquement, ils peuvent être modélisés comme des cylindres creux ou des formes plus organiques selon l'esthétique souhaitée. La profondeur des puits doit être suffisante pour suggérer visuellement qu'ils peuvent contenir plusieurs pierres empilées, tout en permettant une visibilité claire de leur contenu. Les magasins, plus grands que les puits ordinaires, partagent une géométrie similaire mais avec des dimensions adaptées à leur rôle de zones de stockage principales.

Les pierres constituent les éléments les plus nombreux de la scène et méritent une attention particulière concernant leur complexité géométrique. Plusieurs options de modélisation s'offrent au développeur. Les sphères parfaites représentent le choix le plus simple et le plus performant, générées procéduralement avec un nombre contrôlé de sub-

divisions. Pour un rendu plus organique, des formes légèrement irrégulières peuvent être créées en appliquant une perturbation contrôlée aux vertex d'une sphère de base. Des modèles de galets réalistes peuvent également être utilisés, bien que leur complexité géométrique supérieure nécessite une gestion attentive du nombre de pierres simultanément affichées.

L'utilisation de Blender, logiciel de modélisation 3D libre et puissant, peut grandement faciliter la création et la finalisation des assets du projet. Blender permet de concevoir des modèles complexes avec une interface intuitive, d'optimiser les maillages en réduisant le nombre de polygones superflus, et d'appliquer des modificateurs pour créer des variations géométriques. Les modèles créés dans Blender peuvent être exportés dans des formats standards comme OBJ ou FBX, facilement importables dans une application OpenGL via des bibliothèques de chargement appropriées.

La simplification des maillages constitue un aspect crucial de la préparation des assets pour le rendu en temps réel. Les modèles créés dans un logiciel de modélisation comportent souvent un niveau de détail excessif pour une application interactive. Des techniques de décimation peuvent être appliquées pour réduire le nombre de triangles tout en préservant la silhouette générale et les caractéristiques visuelles importantes. Cette optimisation garantit que le système graphique peut afficher simultanément tous les éléments du jeu sans compromettre les performances.

L'organisation des assets suit également une logique réfléchie. Les objets partageant des propriétés géométriques similaires, comme l'ensemble des pierres, peuvent utiliser le même modèle de base instancié à différentes positions. Cette approche d'instanciation réduit considérablement la quantité de données à transférer vers la carte graphique et optimise les appels de rendu. Les puits d'une même rangée peuvent également partager une géométrie commune, différenciée uniquement par leur position et éventuellement leur matériau.

6 Rendu graphique et shaders

Le pipeline de rendu OpenGL constitue la chaîne de traitement qui transforme les données géométriques en pixels affichés à l'écran. Comprendre et maîtriser ce pipeline est essentiel pour créer une application graphique efficace et visuellement satisfaisante.

Le processus commence par la définition des vertex buffers, structures de données contenant les informations géométriques de chaque objet : positions des sommets, normales pour le calcul d'éclairage, et éventuellement coordonnées de texture. Ces données sont transférées vers la mémoire de la carte graphique pour un accès rapide lors du rendu. Pour chaque objet de la scène, l'application configure les attributs vertex appropriés et déclenche un appel de rendu qui lance le traitement par le pipeline graphique.

Les shaders représentent des programmes exécutés directement sur la carte graphique, offrant un contrôle précis sur le rendu. Le vertex shader constitue la première étape programmable du pipeline et s'exécute pour chaque sommet de la géométrie. Sa responsabilité principale consiste à transformer les coordonnées locales du modèle en coordonnées d'écran en appliquant successivement la matrice de modèle, qui positionne l'objet dans le monde, la matrice de vue, qui simule la position et l'orientation de la caméra, et la matrice de projection, qui applique la perspective et définit le volume de visualisation. Le vertex shader calcule également les vecteurs normaux transformés, nécessaires au calcul d'éclairage, et transmet ces informations aux étapes suivantes du pipeline.

Le fragment shader intervient après la rasterisation, processus qui convertit les primitives géométriques en fragments correspondant aux pixels de l'écran. Ce shader s'exécute pour chaque fragment et détermine sa couleur finale. Dans notre application Mancala, le fragment shader implémente le modèle d'éclairage choisi, combinant les propriétés matérielles de l'objet avec les caractéristiques des sources lumineuses de la scène. Il peut également appliquer des textures, gérer la transparence, ou créer des effets visuels spéciaux selon les besoins.

Les transformations géométriques méritent une attention particulière car elles déterminent comment les objets apparaissent dans la scène. La matrice de modèle combine généralement une translation pour positionner l'objet, une rotation pour l'orienter, et une mise à l'échelle pour ajuster sa taille. Ces transformations sont calculées pour chaque objet en fonction de son état dans la logique de jeu. Par exemple, les pierres sont positionnées au-dessus des puits correspondants, avec un léger décalage vertical lorsqu'elles sont empilées.

La matrice de vue implémente la caméra virtuelle qui observe la scène. Elle est construite à partir de la position de la caméra, du point vers lequel elle regarde, et d'un vecteur définissant l'orientation verticale. L'application permet au joueur de contrôler ces paramètres pour explorer différents angles de vue, facilitant ainsi la visualisation complète du plateau de jeu. Les mouvements de caméra comme le pan (déplacement latéral), le zoom (rapprochement ou éloignement), et la rotation orbitale autour d'un point central enrichissent l'interactivité.

La matrice de projection définit comment l'espace tridimensionnel est projeté sur l'écran bidimensionnel. Une projection perspective, la plus couramment utilisée, crée un effet de profondeur réaliste où les objets lointains apparaissent plus petits que les objets proches. Les paramètres de cette projection incluent l'angle de vue vertical (field of view), le rapport d'aspect de l'écran, et les plans de clipping proche et lointain qui délimitent la région visible de la scène.

L'application propose plusieurs modes d'affichage qui modifient la façon dont la géométrie est rendue. Le mode solide standard affiche les surfaces remplies avec leur éclairage et matériau. Le mode filaire (wireframe) ne dessine que les arêtes des triangles, révélant la structure géométrique sous-jacente, particulièrement utile pour le débogage ou l'appréciation de la complexité des modèles. Ces modes sont implémentés en modifiant l'état du pipeline OpenGL via des commandes comme glPolygonMode, permettant de basculer facilement entre les différentes représentations.

7 Éclairage et matériaux

Le modèle d'éclairage implémenté dans l'application joue un rôle crucial dans la perception de la tridimensionnalité et du réalisme de la scène. L'approche classique de l'éclairage en informatique graphique repose sur le modèle de Phong, qui décompose la lumière réfléchie par une surface en trois composantes distinctes : ambiante, diffuse et spéculaire.

La composante ambiante simule l'éclairage indirect omniprésent dans un environnement réel. Elle représente la lumière qui a été réfléchie de multiples fois par diverses surfaces et qui éclaire uniformément tous les objets, même ceux qui ne sont pas directement exposés aux sources lumineuses. Dans le contexte du Mancala, cette lumière ambiante garantit que toutes les parties du plateau restent visibles, évitant des zones complètement noires qui nuiraient à la jouabilité. Son intensité est généralement maintenue relativement faible pour ne pas aplatir la scène en éliminant les contrastes.

La composante diffuse constitue l'essentiel de l'éclairage perçu et dépend directement de l'angle entre la normale de la surface et la direction vers la source lumineuse. Lorsque la lumière frappe une surface perpendiculairement, l'éclairage diffus atteint son maximum, tandis qu'une lumière rasante produit un éclairage plus faible. Cette propriété crée naturellement des gradients de luminosité sur les surfaces courbes, révélant leur forme tridimensionnelle. Les puits cylindriques du Mancala bénéficient particulièrement de cet éclairage diffus qui met en évidence leur courbure.

La composante spéculaire génère les reflets brillants caractéristiques des surfaces lisses ou réfléchissantes. Elle dépend non seulement de l'angle d'incidence de la lumière mais également de la position de l'observateur, créant des points lumineux qui se déplacent lorsque la caméra change d'angle. L'intensité et la taille de ces reflets sont contrôlées par un coefficient de brillance (shininess) : une valeur élevée produit des reflets petits et intenses, typiques du métal ou du verre, tandis qu'une valeur faible crée des reflets larges et doux, caractéristiques des surfaces moins polies.

Les propriétés matérielles définies pour chaque type d'objet déterminent comment il réagit à l'éclairage. Le plateau de jeu peut recevoir un matériau bois avec des coefficients ambiants et diffus dans les tons marron, et une faible composante spéculaire pour simuler une surface légèrement satinée. Les puits peuvent utiliser un matériau similaire ou légèrement plus foncé pour créer un contraste visuel. Les pierres, éléments centraux du jeu, méritent des matériaux distinctifs : des propriétés similaires à celles de la pierre polie, avec des reflets spéculaires modérés, permettent de les différencier visuellement du plateau tout en maintenant une cohérence esthétique.

L'activation et la désactivation de l'éclairage via la touche L offre une comparaison intéressante entre un rendu éclairé et un rendu utilisant uniquement les couleurs de base des objets. Lorsque l'éclairage est désactivé, tous les objets apparaissent dans leur couleur uniforme sans variation de luminosité, créant un effet plat qui met en évidence l'importance cruciale de l'éclairage pour la perception de la profondeur. Cette fonctionnalité sert également d'outil pédagogique, illustrant concrètement l'impact du modèle d'éclairage sur le rendu final.

L'implémentation technique du modèle d'éclairage dans les shaders nécessite le calcul précis des vecteurs impliqués pour chaque fragment. Le vecteur normal, interpolé à partir des normales des sommets, indique l'orientation locale de la surface. Le vecteur lumière pointe de la surface vers la source lumineuse, tandis que le vecteur vue pointe vers la caméra. Ces vecteurs normalisés sont utilisés dans les formules de Phong pour calculer les contributions diffuse et spéculaire, qui sont ensuite combinées avec la composante ambiante pour obtenir la couleur finale du fragment.

Le système permet également de positionner stratégiquement les sources lumineuses dans la scène pour mettre en valeur certains aspects du jeu. Une lumière principale placée en hauteur et légèrement décalée par rapport au centre du plateau crée des ombres naturelles qui renforcent la perception du relief. Des lumières d'appoint peuvent être ajoutées pour éliminer les zones trop sombres ou créer des effets atmosphériques spécifiques en fonction du thème graphique actif.

8 Interaction utilisateur

La qualité de l'interaction utilisateur détermine en grande partie le succès d'une application graphique interactive. Dans le contexte du Mancala 3D, les mécanismes d'interaction doivent être suffisamment intuitifs pour permettre une prise en main immédiate

tout en offrant la profondeur nécessaire aux joueurs expérimentés.

Le contrôle de la caméra constitue l'interaction spatiale fondamentale permettant d'explorer la scène tridimensionnelle. L'application implémente plusieurs modes de manipulation de caméra adaptés à différents besoins de visualisation. Le mode pan permet un déplacement latéral de la caméra, généralement activé en maintenant un bouton de la souris enfoncé tout en effectuant un mouvement de glissement. Ce mode est particulièrement utile pour recentrer la vue sur une zone spécifique du plateau sans modifier l'angle d'observation. La rotation orbitale permet de faire tourner la caméra autour d'un point central, offrant différentes perspectives sur le jeu tout en maintenant le plateau dans le champ de vision. Le zoom, contrôlé par la molette de la souris ou des touches clavier, ajuste la distance entre la caméra et la scène, permettant d'obtenir une vue d'ensemble ou de se concentrer sur des détails spécifiques.

La sélection des puits à la souris représente l'interaction principale du gameplay et nécessite une implémentation soignée. Lorsque le joueur clique sur l'écran, l'application doit déterminer quel objet 3D se trouve sous le curseur. Cette opération, appelée picking, peut être réalisée selon plusieurs approches. La méthode du ray casting consiste à projeter un rayon depuis la position de la caméra à travers le pixel cliqué et à calculer les intersections de ce rayon avec les objets de la scène. Une alternative consiste à utiliser le color picking, technique où chaque objet est rendu dans un buffer hors écran avec une couleur unique identifiante, permettant de déterminer l'objet cliqué en lisant la couleur du pixel correspondant.

Une fois le puits identifié, l'application doit fournir un feedback visuel immédiat pour confirmer la sélection et guider les actions futures. Le survol d'un puits par la souris peut déclencher un effet de surbrillance, comme une légère augmentation de luminosité ou un contour coloré, indiquant que l'élément est interactif. Lors du clic, une animation ou un changement d'état visuel confirme l'action, avant que la distribution des pierres ne commence. Les puits non sélectionnables, comme ceux appartenant à l'adversaire ou les puits vides du joueur actif, peuvent être visuellement atténués ou ne pas réagir au survol, communiquant clairement leur état non interactif.

L'utilisation du clavier pour les fonctionnalités avancées enrichit l'expérience sans surcharger l'interface visuelle. La touche L bascule l'état de l'éclairage, permettant de comparer les rendus avec et sans calcul lumineux. La touche T fait défiler les différents thèmes graphiques disponibles, modifiant instantanément l'apparence de la scène. La touche M alterne entre les modes de rendu, passant du rendu solide standard au mode filaire révélant la structure géométrique. Des touches de navigation, comme les flèches directionnelles, peuvent contrôler la rotation de la caméra, tandis que les touches plus et moins gèrent le zoom. Cette organisation des contrôles suit les conventions établies dans de nombreuses applications 3D, facilitant l'apprentissage pour les utilisateurs familiers avec ce type d'interface. L'interface graphique intégrée superpose des informations essentielles sur la scène 3D sans perturber la visualisation du jeu. Un affichage permanent présente le score de chaque joueur, généralement positionné dans les coins supérieurs de l'écran en correspondance avec leur côté du plateau. Une indication claire signale quel joueur doit effectuer le prochain coup, évitant toute confusion lors des transitions de tour. Des messages contextuels peuvent apparaître temporairement pour communiquer des événements importants, comme l'obtention d'un tour supplémentaire, une capture de pierres, ou la fin de la partie avec l'annonce du vainqueur. Un système d'aide optionnel peut être intégré pour assister les nouveaux joueurs. Une touche dédiée, comme H, pourrait afficher ou masquer un panneau récapitulant les contrôles disponibles et les règles de base du jeu. Des infobulles

contextuelles pourraient apparaître lors du survol d'éléments spécifiques, expliquant leur fonction ou leur état actuel. Cette couche d'assistance garantit que l'application reste accessible même aux utilisateurs découvrant simultanément le jeu Mancala et l'interface 3D.

9 Logique du jeu Mancala

L'implémentation de la logique de jeu constitue le fondement fonctionnel de l'application, garantissant que toutes les interactions respectent fidèlement les règles du Mancala et maintiennent la cohérence de l'état du jeu à tout moment. La gestion des tours repose sur un système d'états qui identifie en permanence le joueur actif et contrôle les transitions entre les tours. L'état initial du jeu attribue le premier tour à un joueur, généralement choisi aléatoirement ou selon une convention. Après chaque coup, le système évalue si le joueur actif mérite un tour supplémentaire selon la règle spéciale : si la dernière pierre distribuée a atterri dans son propre magasin, le tour reste au même joueur. Dans le cas contraire, le tour passe à l'adversaire. Cette logique de transition doit être implémentée avec rigueur pour éviter toute incohérence qui compromettrait l'équité du jeu. L'algorithme de distribution des pierres représente le cœur mécanique du jeu et nécessite une implémentation précise. Lorsqu'un joueur sélectionne un puits valide contenant des pierres, l'algorithme commence par extraire toutes les pierres de ce puits, mémorisant leur nombre. Il identifie ensuite le puits suivant dans le sens antihoraire autour du plateau, en tenant compte de la structure circulaire de celui-ci. Pour chaque pierre à distribuer, l'algorithme dépose une pierre dans le puits courant, puis avance au puits suivant. Lors de ce parcours, il doit respecter plusieurs règles : inclure le magasin du joueur actif lorsqu'il est croisé, ignorer le magasin de l'adversaire, et traiter correctement les transitions entre les deux rangées du plateau. La dernière pierre distribuée déclenche potentiellement des actions spéciales qui enrichissent la stratégie du jeu. Si cette pierre atterrit dans le magasin du joueur actif, le système marque que le joueur conserve son tour. Si elle atterrit dans un puits vide appartenant au joueur actif, une capture se produit : la pierre qui vient d'être placée, ainsi que toutes les pierres présentes dans le puits directement opposé du côté adverse, sont transférées dans le magasin du joueur actif. Cette opération de capture nécessite d'identifier correctement le puits opposé, ce qui peut être réalisé par un calcul d'index ou une structure de données appropriée. Le calcul des scores découle naturellement de l'état des magasins. À tout moment, le score d'un joueur correspond au nombre de pierres accumulées dans son magasin. L'affichage de ces scores doit être mis à jour immédiatement après chaque action modifiant leur valeur, garantissant que les joueurs disposent d'une information constante sur l'état de la partie. Des statistiques supplémentaires peuvent être calculées et affichées, comme le nombre de coups joués, le nombre de captures effectuées, ou l'avantage numérique d'un joueur par rapport à l'autre. La détection de la fin de partie surveille en permanence une condition spécifique : lorsqu'un joueur ne possède plus aucune pierre dans les puits de sa rangée, la partie se termine immédiatement. À ce moment, l'adversaire récupère automatiquement toutes les pierres restantes de son côté, qui sont ajoutées à son magasin. Le système compare ensuite les scores finaux des deux magasins pour déterminer le vainqueur. Un message approprié est alors affiché, annonçant le résultat de la partie et offrant éventuellement la possibilité de recommencer une nouvelle partie. La synchronisation entre la logique de jeu et l'affichage graphique constitue un défi technique important. Idéalement, la logique de jeu maintient un état abstrait indépendant de toute considération graphique. Le système de rendu inter-

roge régulièrement cet état pour mettre à jour la position des objets 3D en conséquence. Lors d'une distribution de pierres, une approche naïve consisterait à mettre à jour instantanément l'état logique puis à refléter ce changement dans le rendu. Cependant, pour une expérience utilisateur fluide, il est préférable d'animer progressivement le déplacement des pierres. Cela nécessite un système d'animation qui interpole la position des pierres entre leur emplacement initial et leur destination finale, tout en maintenant temporairement un état intermédiaire entre la logique et le visuel.

10 Détection de collisions et contraintes d'interaction

La détection de collisions et la validation des interactions constituent des mécanismes essentiels pour garantir une expérience utilisateur cohérente et prévenir les actions invalides qui violeraient les règles du jeu ou créeraient des états incohérents. Les mécanismes empêchant les interactions invalides opèrent à plusieurs niveaux de l'application. Au niveau le plus fondamental, le système de sélection des puits doit vérifier que le puits ciblé appartient bien au joueur actif. Toute tentative de sélectionner un puits de l'adversaire doit être silencieusement ignorée ou accompagnée d'un feedback visuel indiquant l'impossibilité de l'action. De même, la sélection d'un puits vide du joueur actif ne doit pas déclencher d'action, car aucune distribution n'est possible sans pierres à déplacer. Ces vérifications sont généralement implémentées dans la couche de logique de jeu avant l'exécution de toute action. La détection de collisions lors de la manipulation des objets intervient principalement dans le contexte du picking, où le système doit déterminer quel objet 3D a été cliqué. Dans le cas d'une implémentation par ray casting, l'algorithme calcule l'intersection d'un rayon avec les volumes de délimitation (bounding volumes) des objets sélectionnables. Ces volumes, souvent des sphères ou des boîtes englobantes, simplifient les calculs d'intersection tout en fournissant une précision suffisante pour l'interaction. Lorsque plusieurs objets se trouvent le long du rayon, le système sélectionne généralement celui le plus proche de la caméra, correspondant à l'objet visible sous le curseur. Pour les pierres empilées dans un puits, une gestion particulière peut être nécessaire. Bien que les pierres individuelles ne soient généralement pas sélectionnables (c'est le puits entier qui est l'unité d'interaction), le système de rendu doit positionner correctement les pierres empilées pour éviter qu'elles ne se chevauchent visuellement de manière inesthétique. Un espacement vertical calculé en fonction du nombre de pierres garantit que chaque pierre reste au moins partiellement visible, permettant une appréciation visuelle approximative du nombre de pierres dans chaque puits. Les contraintes temporelles constituent un autre aspect important de la gestion des interactions. Pendant l'animation de distribution des pierres, le système doit empêcher toute nouvelle sélection de puits jusqu'à ce que l'animation soit terminée et l'état du jeu stabilisé. Permettre des interactions pendant une animation risquerait de créer des conflits d'état et des comportements imprévisibles. Un système de verrouillage d'interface (input lock) désactive temporairement la réactivité aux clics pendant les phases d'animation, garantissant que chaque action se déroule complètement avant qu'une nouvelle ne puisse être initiée. Le feedback visuel des états d'interaction joue un rôle crucial dans la communication de ces contraintes à l'utilisateur. Lorsqu'un puits ne peut pas être sélectionné, qu'il s'agisse d'un puits adverse, vide, ou pendant un verrouillage temporaire, l'absence de réaction au survol communique implicitement cette restriction. À l'inverse, les puits sélectionnables réagissent visuellement au survol, indiquant clairement leur état interactif. Cette approche de design préventif, où l'interface elle-même guide vers des actions valides, réduit considérablement les tentatives

d'interactions invalides et améliore la fluidité de l'expérience. La validation des actions s'étend également à la prévention des états de jeu impossibles. Le système doit garantir que le nombre total de pierres dans le jeu reste constant, à l'exception des transferts entre puits et magasins. Toute opération modifiant ces quantités doit être atomique et vérifiable, permettant de détecter rapidement d'éventuelles incohérences résultant de bugs d'implémentation. Des assertions ou des vérifications périodiques de l'intégrité de l'état peuvent être intégrées pendant le développement pour identifier précocement les erreurs logiques.

11 Personnalisation et thèmes

Le système de thèmes graphiques enrichit considérablement l'expérience utilisateur en offrant des variations esthétiques de l'application tout en conservant la même structure fonctionnelle et les mêmes règles de jeu. Cette personnalisation visuelle permet aux joueurs de choisir une ambiance correspondant à leurs préférences et contribue à renouveler l'intérêt visuel lors de parties répétées. L'implémentation d'un système de thèmes repose sur la séparation entre la géométrie des objets et leurs propriétés visuelles. Chaque thème définit un ensemble de palettes de couleurs, de propriétés matérielles et éventuellement de textures qui sont appliquées aux différents éléments de la scène. Par exemple, un thème classique pourrait utiliser des tons de bois naturel pour le plateau avec des pierres dans des teintes de pierre naturelle. Un thème moderne pourrait privilégier des couleurs vives et contrastées avec des matériaux réfléchissants. Un thème nocturne pourrait opter pour des tons sombres avec des accents lumineux, créant une atmosphère feutrée. La structure de données représentant un thème regroupe typiquement les propriétés suivantes : couleur de base du plateau, couleur et propriétés matérielles des puits, couleur de fond de la scène, couleurs distinctes pour les pierres des deux joueurs permettant de les différencier visuellement, intensité et couleur de la lumière ambiante adaptées à l'atmosphère du thème, et éventuellement des paramètres d'éclairage spécifiques comme la position ou la couleur des sources lumineuses principales. Le changement de thème via la touche T déclenche une mise à jour immédiate de toutes ces propriétés dans le système de rendu. Cette transition peut être instantanée ou, pour un effet plus fluide, animée progressivement en interpolant les couleurs et propriétés matérielles de l'ancien thème vers le nouveau sur quelques images. L'interpolation de couleurs crée un effet de fondu visuellement agréable qui évite les changements brusques pouvant être perturbants. L'impact sur l'expérience utilisateur va au-delà de la simple variation esthétique. Des thèmes bien conçus peuvent améliorer la lisibilité du jeu en augmentant le contraste entre les éléments importants et le fond, facilitant ainsi la distinction des puits et l'évaluation de leur contenu. Certains joueurs peuvent préférer des thèmes épurés minimisant les distractions visuelles pour se concentrer sur la stratégie, tandis que d'autres apprécient des thèmes riches et détaillés créant une expérience plus immersive. La personnalisation peut être étendue au-delà des thèmes prédéfinis en permettant aux utilisateurs de créer leurs propres combinaisons de couleurs. Une interface de configuration pourrait offrir des curseurs pour ajuster les teintes et la saturation des différents éléments, ainsi que des préréglages inspirés de différentes cultures ou styles artistiques. Cette flexibilité transforme l'application d'un simple jeu en un outil d'expression personnelle, renforçant l'engagement des utilisateurs. D'un point de vue technique, la gestion des thèmes illustre parfaitement les principes de conception modulaire. Les paramètres visuels sont découplés de la logique de jeu et de la structure géométrique, permettant de modifier l'apparence sans affecter le comportement. Les

shaders reçoivent les propriétés matérielles comme paramètres uniformes, facilitant leur modification dynamique. Cette architecture rend l'ajout de nouveaux thèmes aussi simple que la définition d'un nouvel ensemble de valeurs, sans nécessiter de modification du code de rendu lui-même.

12 Performances et optimisation

La fluidité d'une application graphique interactive dépend fondamentalement de sa capacité à maintenir un taux de rafraîchissement élevé et constant. Pour une expérience utilisateur optimale, l'objectif standard consiste à atteindre soixante images par seconde, ce qui correspond à un budget temporel d'environ seize millisecondes par image. L'analyse des performances commence par l'identification des goulots d'étranglement potentiels dans la chaîne de rendu. Le transfert de données entre la mémoire système et la mémoire graphique représente souvent une source de ralentissements, particulièrement si des données géométriques volumineuses sont envoyées à chaque image. Pour minimiser cet impact, l'application charge les géométries dans des buffers GPU lors de l'initialisation et conserve ces données en mémoire graphique. Les objets statiques comme le plateau et les puits n'ont besoin d'être envoyés qu'une seule fois, tandis que seules les matrices de transformation sont mises à jour à chaque image. Le nombre d'appels de rendu (draw calls) influence significativement les performances, car chaque appel implique une communication entre le processeur et la carte graphique. L'application du Mancala pourrait naïvement effectuer un appel de rendu séparé pour chaque pierre, ce qui deviendrait coûteux avec quarante-huit pierres simultanément affichées. L'instanciation géométrique permet de dessiner plusieurs copies d'un même objet avec un seul appel de rendu, en fournissant un tableau de matrices de transformation. Cette technique réduit drastiquement le nombre d'appels tout en permettant de positionner individuellement chaque pierre. La complexité géométrique des modèles 3D doit être adaptée aux capacités de rendu en temps réel. Bien que les scènes du Mancala demeurent relativement simples, l'optimisation des maillages contribue à maintenir des performances élevées. Une sphère représentant une pierre n'a pas besoin de milliers de triangles pour paraître lisse ; quelques centaines suffisent généralement, exploitant l'interpolation de normales pour créer une illusion de surface parfaitement courbe. Les techniques de level of detail pourraient même être appliquées pour réduire la complexité des objets distants, bien que leur nécessité soit limitée dans le contexte d'un plateau de jeu de taille modeste. La gestion du rendu en fonction de la complexité de la scène peut également être optimisée. Le frustum culling élimine du pipeline de rendu les objets situés en dehors du champ de vision de la caméra, évitant de traiter inutilement leur géométrie. Dans le Mancala, cette optimisation a un impact limité car la plupart des éléments restent généralement visibles, mais elle devient pertinente si l'utilisateur zoome fortement sur une zone spécifique du plateau. Les shaders eux-mêmes méritent une attention particulière en termes de performance. Des calculs complexes dans le fragment shader peuvent ralentir considérablement le rendu, car ce shader s'exécute pour chaque pixel de chaque objet affiché. Déplacer certains calculs vers le vertex shader, qui s'exécute beaucoup moins fréquemment, peut améliorer les performances si l'interpolation des résultats entre vertices reste acceptable. Par exemple, certains calculs d'éclairage peuvent être approximés au niveau des vertices plutôt que des fragments, réduisant la charge de calcul au prix d'une légère perte de précision visuelle. La mesure régulière des performances pendant le développement permet d'identifier rapidement les régressions et de valider l'efficacité des optimisations. Des outils de profilage peuvent révéler quelles

parties du code consomment le plus de temps, guidant les efforts d'optimisation vers les zones les plus critiques. L'affichage en temps réel du nombre d'images par seconde aide également à évaluer l'impact de modifications sur la fluidité de l'application. Enfin, la gestion de la mémoire graphique nécessite une vigilance constante. Les textures, buffers et autres ressources GPU doivent être correctement libérés lorsqu'ils ne sont plus nécessaires pour éviter les fuites mémoire qui dégraderaient progressivement les performances. Un système de gestion de ressources structuré garantit que chaque allocation est associée à une désallocation correspondante, maintenant ainsi la stabilité de l'application même lors d'utilisations prolongées.

13 Démonstration de l'application

Cette section présente le fonctionnement pratique de l'application à travers une série d'illustrations démontrant les différentes fonctionnalités et interactions disponibles.

13.1 Vue générale du jeu

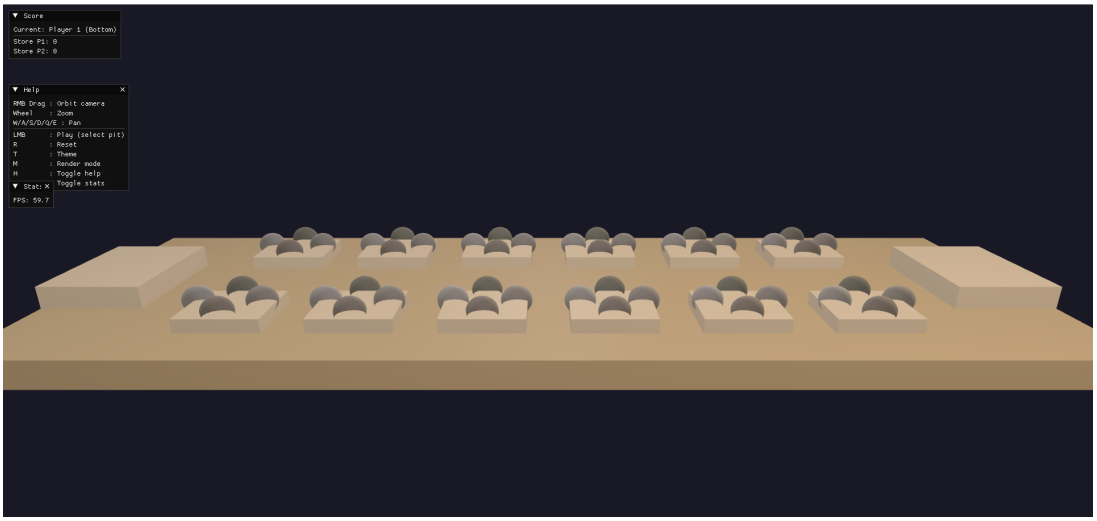

FIGURE 1 – Vue générale de l'application Mancala 3D

Le lancement de l'application présente immédiatement le plateau de jeu dans une vue tridimensionnelle claire et engageante. L'interface principale affiche le plateau avec ses deux rangées de six puits disposées symétriquement, flanquées des deux magasins plus grands aux extrémités. Les pierres sont visibles dans chaque puits, leur nombre correspondant à la configuration initiale du jeu. L'éclairage met en valeur le relief des éléments, créant des ombres subtiles qui renforcent la perception de la profondeur. En haut de l'écran, l'interface graphique affiche les scores actuels de chaque joueur ainsi qu'une indication du joueur dont c'est le tour. Cette vue d'ensemble permet aux joueurs de saisir immédiatement l'état du jeu et de planifier leur stratégie.

13.2 Déplacement de la caméra (PAN)

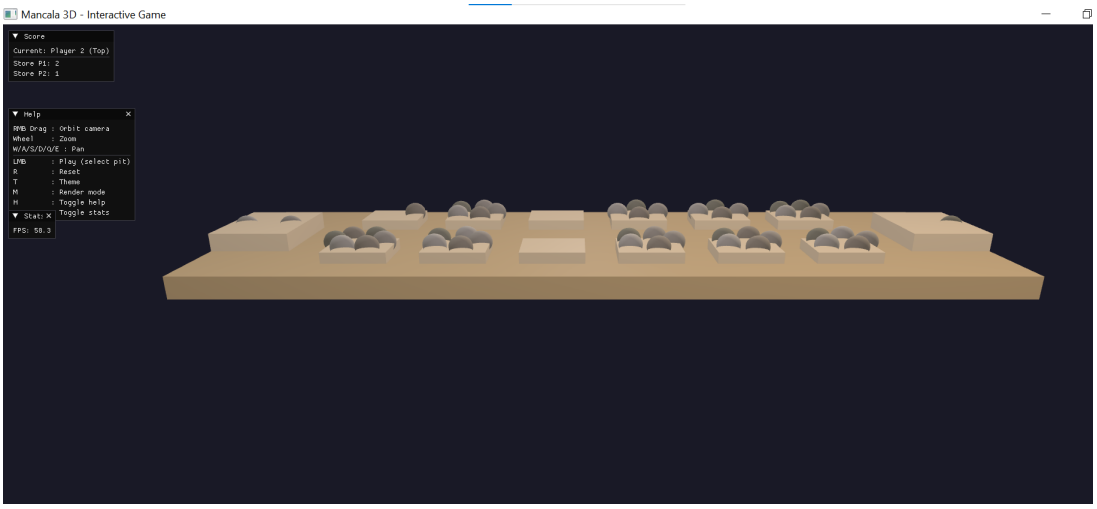

FIGURE 2 – Déplacement de la caméra (PAN) dans la scène 3D

La fonctionnalité de déplacement latéral de la caméra permet d'explorer différentes portions de la scène sans modifier l'angle de vue. En maintenant un bouton de la souris enfoncé et en effectuant un mouvement de glissement, ou en utilisant les touches directionnelles du clavier, l'utilisateur peut recentrer la vue sur une zone spécifique du plateau. Cette illustration montre comment la caméra a été déplacée pour offrir une perspective légèrement différente, peut-être pour mieux visualiser un groupe particulier de puits ou pour adapter la vue à une préférence personnelle. Le mouvement fluide de la caméra maintient l'orientation et l'angle constants, offrant une sensation de contrôle précis sur le point de vue.

13.3 Sélection et exécution d'un coup

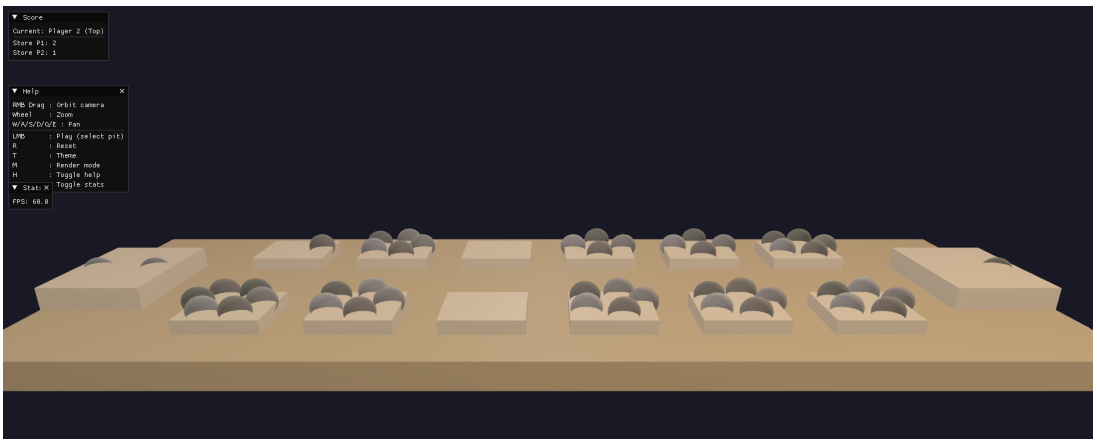

FIGURE 3 – Sélection d'un puits et exécution d'un coup

Cette capture illustre le moment où un joueur sélectionne un puits pour exécuter son coup. Le survol du puits avec la souris a probablement déclenché un effet de surbrillance,

indiquant qu'il s'agit d'un élément interactif valide. Après le clic, la distribution des pierres commence, les pierres se déplaçant animatement d'un puits au suivant selon les règles du jeu. L'image peut montrer soit l'instant de la sélection, soit un état intermédiaire de l'animation où certaines pierres sont en cours de déplacement. L'interface affiche la mise à jour du nombre de pierres dans les puits affectés, et le score peut avoir changé si des pierres ont atteint un magasin. Cette interaction constitue le cœur du gameplay, démontrant la fluidité de la connexion entre l'action du joueur et la réponse visuelle de l'application.

13.4 Activation et désactivation de l'éclairage

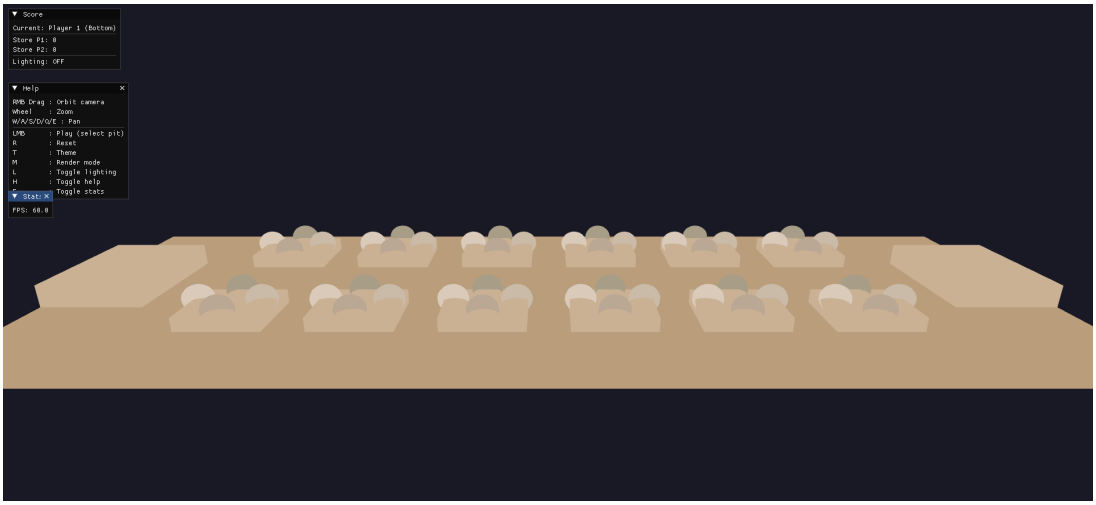

FIGURE 4 – Activation et désactivation de l'éclairage

La touche L permet de basculer entre un rendu avec calcul d'éclairage complet et un rendu utilisant uniquement les couleurs de base des objets. Cette illustration montre probablement le mode sans éclairage, où tous les objets apparaissent dans des couleurs uniformes sans variation de luminosité. L'absence de gradients d'ombre et de reflets crée un aspect plat qui contraste fortement avec le rendu éclairé standard. Cette comparaison met en évidence l'importance cruciale du modèle d'éclairage pour la perception de la tridimensionnalité. Les contours des objets restent visibles et la scène demeure fonctionnelle, mais le réalisme et la profondeur visuelle sont considérablement réduits. Cette fonctionnalité sert à la fois d'outil pédagogique et de curiosité technique, permettant d'apprécier la sophistication du système de rendu.

13.5 Changement de thème graphique

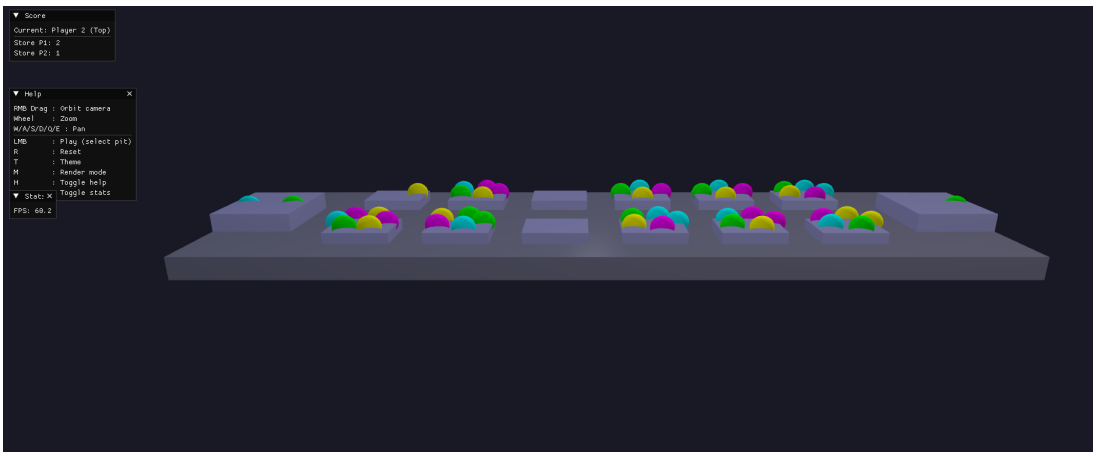

FIGURE 5 – Changement de thème graphique

L'activation de la touche T fait défiler les différents thèmes graphiques disponibles, modifiant instantanément l'apparence esthétique de toute la scène. Cette image présente probablement un thème distinct du thème par défaut, avec une palette de couleurs alternative appliquée au plateau, aux puits et aux pierres. Le nouveau thème pourrait utiliser des tons plus chauds ou plus froids, des couleurs plus saturées ou plus sourdes, créant une atmosphère visuelle différente tout en conservant la même structure géométrique et la même jouabilité. Les propriétés matérielles peuvent également avoir changé, affectant la réflectivité et l'apparence des surfaces. Cette personnalisation permet aux joueurs de choisir une ambiance visuelle correspondant à leurs goûts personnels, renouvelant l'intérêt esthétique du jeu même après de nombreuses parties.

13.6 Changement du mode de rendu

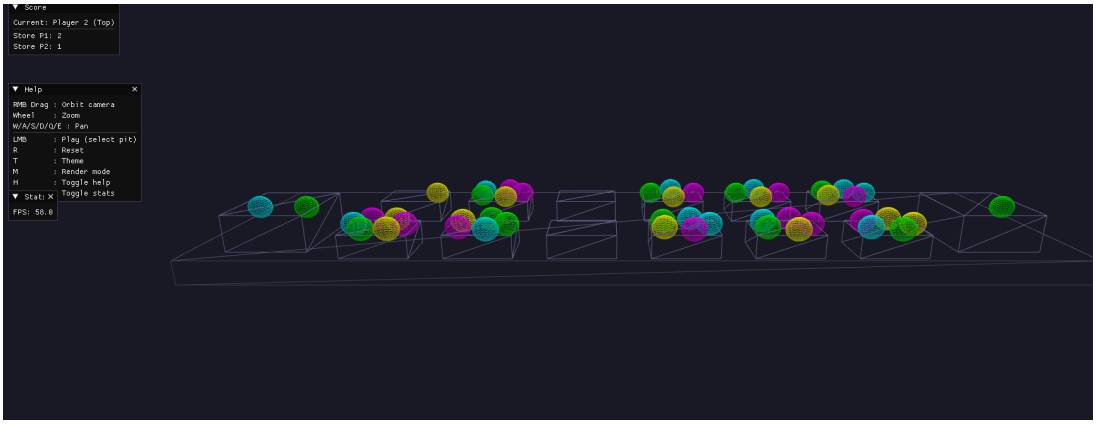

FIGURE 6 – Changement du mode de rendu (solide / filaire)

La touche M permet de basculer entre différents modes de rendu, notamment entre le mode solide standard et le mode filaire. Cette illustration présente vraisemblablement le mode filaire, où seules les arêtes des triangles composant les modèles 3D sont dessinées. Ce

rendu révèle la structure géométrique sous-jacente des objets, permettant d'apprécier la complexité des maillages et la tessellation des surfaces courbes. Le plateau, les puits et les pierres apparaissent comme des cages de fils métalliques, créant un effet visuel technique et artistique. Bien que moins pratique pour le jeu effectif, ce mode offre une perspective fascinante sur la construction technique de la scène 3D et peut servir d'outil de débogage pour vérifier la qualité des modèles géométriques.

14 Conclusion et perspectives

Ce projet de développement d'une application interactive 3D du jeu Mancala avec OpenGL a permis d'atteindre les objectifs pédagogiques fixés tout en créant un produit fonctionnel et visuellement engageant. L'implémentation complète couvre l'ensemble de la chaîne de création d'une application graphique interactive, depuis la modélisation des assets 3D jusqu'à la gestion des interactions utilisateur en passant par l'implémentation de la logique de jeu. Les compétences acquises au cours de ce projet sont nombreuses et fondamentales pour la programmation graphique. La maîtrise du pipeline de rendu OpenGL permet désormais de comprendre comment les données géométriques sont transformées en images à l'écran. L'implémentation de shaders personnalisés a développé une compréhension approfondie de la programmation GPU et du traitement parallèle des données graphiques. La gestion des transformations géométriques et de la caméra virtuelle constitue une base solide pour tout travail ultérieur en visualisation 3D. L'intégration d'un modèle d'éclairage complet illustre les principes physiques sous-tendant le rendu réaliste. Enfin, la conception d'interactions utilisateur intuitives dans un environnement tridimensionnel développe des compétences essentielles en design d'interface et en ergonomie. La séparation architecturale entre logique de jeu et rendu graphique s'est révélée particulièrement bénéfique, facilitant le développement parallèle de ces aspects et permettant de tester chacun indépendamment. Cette approche modulaire constitue une excellente pratique applicable à des projets de plus grande envergure. L'utilisation potentielle de Blender pour la création d'assets a également démontré l'importance de maîtriser les outils de modélisation 3D en complément des compétences en programmation graphique. Les résultats obtenus sont satisfaisants tant du point de vue technique que visuel. L'application fonctionne de manière fluide, maintenant un taux de rafraîchissement élevé même avec tous les éléments affichés simultanément. Les interactions sont réactives et intuitives, permettant une prise en main rapide. Le système de thèmes et les différents modes de rendu enrichissent l'expérience utilisateur en offrant de la variété et de la personnalisation. Plusieurs perspectives d'amélioration et d'extension s'ouvrent naturellement suite à ce travail. L'ajout d'animations plus sophistiquées pourrait enrichir considérablement l'expérience visuelle. Actuellement, les pierres se déplacent probablement de manière linéaire d'un puits à l'autre ; des trajectoires paraboliques simulant un mouvement de lancer créeraient un effet plus dynamique et réaliste. Les pierres pourraient également rebondir légèrement en atterrissant dans leur destination, ajoutant une sensation de poids et de physique. Des effets de particules lors de captures importantes ou de victoire apporteraient une dimension festive aux moments clés du jeu. L'implémentation d'une intelligence artificielle permettrait aux joueurs de s'entraîner seuls contre un adversaire virtuel. Une IA simple basée sur des heuristiques évaluerait les coups possibles et choisirait celui maximisant le score ou minimisant les opportunités de l'adversaire. Une IA plus sophistiquée utiliserait des algorithmes de recherche comme minimax avec élagage alpha-bêta pour explorer l'arbre des possibilités sur plusieurs coups. Différents niveaux

de difficulté offriraient un défi adapté aux joueurs de tous niveaux. L'enrichissement des effets visuels pourrait inclure des ombres projetées en temps réel, augmentant considérablement le réalisme de la scène. Des textures détaillées appliquées aux surfaces du plateau et des puits remplaceraient les couleurs unies actuelles, simulant l'apparence de matériaux réels comme le bois sculpté ou la pierre polie. Des effets de post-processing comme la profondeur de champ ou le bloom ajouteraient une dimension cinématographique au rendu. L'intégration de musique d'ambiance et d'effets sonores pour les actions de jeu créerait une expérience multisensorielle plus immersive. Le support de variantes des règles du Mancala permettrait d'explorer différentes versions régionales du jeu, chacune avec ses spécificités tactiques. Un mode multijoueur en réseau étendraitl'application au-delà du jeu local, permettant de défier des adversaires distants. Un système de sauvegarde et de chargement de parties permettrait de reprendre des sessions interrompues. L'ajout de statistiques détaillées et d'un historique des parties enrichirait l'expérience pour les joueurs réguliers. En conclusion, ce projet a non seulement permis de développer une application fonctionnelle et agréable à utiliser, mais a également constitué une expérience d'apprentissage riche en enseignements techniques et méthodologiques. Les fondations posées dans ce travail serviront de base solide pour de futurs projets en programmation graphique et en développement d'applications interactives. Les perspectives d'amélioration identifiées offrent des pistes stimulantes pour poursuivre l'exploration de ce domaine fascinant qu'est la création d'expériences visuelles tridimensionnelles.