Document de design

1) Sommaire

   Application 3D/2D générique permettant d'instancier et de visualiser des modèles 3D, des primitives géométriques et vectorielles dans une scène.

   L'application supporte l'importation de modèles, l'exportation d'une séquence d'images, le déplacement dans une scène 3D, la génération de primitives vectorielles et géométriques, et les transformations, de manière interactive et intuitive.

2) Interactivité
    a) Image
        i) Importation d'images
            - Interactivité: glisser-déposer des images directement dans la scène; l'image apparaît à l'endroit pointé et peut être
            - Entrées: fichiers PNG/JPG/JPEG/GIF déposés par l'utilisateur.
            - Sorties: nouvel objet image dans la scène, message d'état (succès/erreur).
        ii) Exportation d'images
            - Interactivité: panneau d'export pour choisir le dossier, le nom de base et le nombre d'images; lancement d'un export de séquence.
            - Entrées: paramètres d'export renseignés dans l'interface; action "Start Export".
            - Sorties: séquence de fichiers PNG correspondant au rendu.
        iii) Échantillonnage d'images
            - Interactivité: choix d'une ou plusieurs images sources, réglage de la taille et de la granularité, génération d'une nouvelle image; ajout automatique dans la scène.
            - Entrées: images sources sélectionnées et paramètres (taille, tuiles).
            - Sorties: fichier PNG généré et objet image ajouté à la scène; message d'état.
        iv) Palette de couleur
            - Interactivité: création/édition d'une palette (≥4 couleurs), sélection de la couleur active, ajout/suppression de couleurs.
            - Entrées: manipulations dans le panneau "Image".
            - Sorties: couleur active réutilisable pour d'autres outils; feedback visuel.
        v) Histogramme
            - Interactivité: sélection d'une image importée et affichage de son histogramme pour l'analyse visuelle.
            - Entrées: choix de l'image à analyser.
            - Sorties: histogramme affiché dans l'interface; message d'état.
    b) Dessin vectoriel
        i) Curseur dynamique
            - Interactivité: au moins 5 apparences du curseur selon l'outil/état courant.

- Entrées: changements d'outil/état.
- Sorties: retour visuel immédiat du curseur.

ii) Outils de dessin
- Interactivité: réglage de l'épaisseur, des couleurs de contour/remplissage et de l'arrière-plan (modes RGB et HSB).
- Entrées: valeurs choisies par l'utilisateur.
- Sorties: rendu vectoriel mis à jour.

iii) Primitives vectorielles
- Interactivité: création d'au moins 6 types de primitives (ex. point, ligne, rectangle, triangle, polygones, cercle/ellipse/arc).
- Entrées: choix de la primitive et de ses paramètres.
- Sorties: instances ajoutées à la scène.

iv) Formes vectorielles
- Interactivité: création d'au moins 3 formes composées (assemblages de primitives).
- Entrées: sélection/paramétrage des composants.
- Sorties: forme composée dans la scène.

v) Interface
- Interactivité: panneaux donnant un retour visuel (états, valeurs) et des contrôles pour agir sur la scène.
- Entrées: ajustements via l'UI.
- Sorties: mises à jour visuelles immédiates.

c) Transformation

i) Graphe de scène
- Interactivité: ajout, suppression, sélection et édition d'attributs des éléments.
- Entrées: actions via l'interface/panneaux.
- Sorties: scène actualisée et visible.

ii) Sélection multiple
- Interactivité: sélection de plusieurs éléments et modification d'attributs communs.
- Entrées: opérations de sélection multiples.
- Sorties: modifications appliquées à la sélection.

iii) Transformations interactives
- Interactivité: gizmo pour déplacer, faire pivoter et mettre à l'échelle; panneau numérique de transformation; option d'afficher/masquer les boîtes englobantes; sélection via panneau.

- Entrées: manipulations au gizmo/clavier/souris et panneaux.
- Sorties: transformations visibles sur les objets.

d) Géométrie
    i) Boîte de délimitation
        - Interactivité: option pour afficher les boîtes englobantes des modèles.
        - Entrées: activation/désactivation dans l'interface.
        - Sorties: arêtes de boîtes visibles.
    ii) Primitives géométriques
        - Interactivité: instanciation de sphères, cubes, cylindres depuis un panneau.
        - Entrées: paramètres des primitives (taille, segments, etc.).
        - Sorties: objets 3D ajoutés à la scène.
    iii) Modèles 3D
        - Interactivité: import de modèles 3D (.obj) dans une bibliothèque puis instanciation en scène.
        - Entrées: sélection de fichier .obj et du modèle à instancier.
        - Sorties: instances de modèles 3D visibles.
e) Caméra
    i) Caméra interactive
        - Interactivité: rotation à la souris (clic droit + mouvement) et déplacement au clavier (ZQSD), montée/descente (Espace/Ctrl); ajustements via l'interface.
        - Entrées: clavier/souris.
        - Sorties: point de vue mis à jour.
    ii) Caméras multiples
        - Interactivité: plusieurs caméras avec vues distinctes (perspective et orthographique).
        - Entrées: sélection/réglages par caméra.
        - Sorties: affichage multi-vues.
    iii) Point de vue multiple
        - Interactivité: visualisation simultanée de la scène depuis différents points de vue.
        - Entrées: configuration des vues.
        - Sorties: fenêtres de rendu multiples.

3) Technologies

Le projet est développé from scratch avec OpenGL pour le rendu. Des bibliothèques tierces sont incluses afin de faciliter certaines opérations.

l'environnement technologique du projet est composé des éléments suivants :

- C++ : Langage de programmation utilisé pour le projet. Standard 20 ou supérieur.
- CMake : Générateur de build. Utilisé pour la compilation du projet.
- IMGUI : Bibliothèque d'interface utilisateur, utilisée pour toutes les interfaces utilisateurs du projet.
- IMGUI File Dialog : Explorateur de fichiers pour IMGUI. Utilisé par le chargeur de modèle 3D.
- IMGuizmo : Gizmo pour IMGUI, utilisé pour déplacer des objets dans la scène.
- GLFW : Bibliothèque pour la gestion de fenêtre, d'input et pour créer un contexte OpenGL.
- GLM : Bibliothèque pour les opérations mathématiques. Utilisé principalement dans le projet pour les opérations sur les matrices et les vecteurs.
- GLAD : Chargeur de fonctions/helper pour OpenGL. Nécessaire pour appeler simplement les commandes OpenGL.
- STB : (stb_image) : Bibliothèque pour manipuler des images en C++.
- Optionnellement Ninja : En tant que build system pour compiler le projet plus rapidement.

4) Compilation

Le projet utilise C++ >= 20, Cmake et Make ou Ninja pour compiler, il est nécessaire d'avoir ces utilitaires installés sur le système qui doit compiler le projet.

Le projet est fourni avec toutes les dépendances dans la bonne version, situées dans le dossier 'external', il n'y a rien à ajouter.

Les fichiers de builds doivent être générés avec CMake. Nous préférons utiliser Ninja en tant que système de build, mais il est également possible d'utiliser simplement Make.

Depuis le dossier racine du projet, on génère les fichiers des build avec Cmake vers un dossier 'build' :

cmake -G Ninja -B build .

Note : Si l'on ne souhaite pas utiliser Make à la place de Ninja, il est possible d'utiliser la commande :

cmake -B build .

Puis pour compiler le projet :

cmake --build build

L'exécutable 'scenelab' se trouve dans le dossier 'build', il faut l'exécuter dans le dossier build:

cd build && ./scenelab

5) Architecture

L'architecture du logiciel s'organise autour de quelques axes clés :

- Les GameObject
    - C'est une classe de base permettant de représenter n'importe quel objet 2D ou 3D qui sera instancié dans la scène. Un GameObject offre des fonctions basiques, mais permet d'appliquer des translations, rotations et proportions et d'en extraire la matrice du modèle. Les GameObjects peuvent être hérités pour créer différents types d'objets ayant des attributs supplémentaires tels que les primitives vectorielles et géométriques.
- Les Renderers
    - Classe de base permettant l'implémentation de plusieurs méthodes de rendu différentes. Cela permet par exemple de créer une classe RasterizationRenderer dédié au rendu par rastérisation et une future classe dédiée au rendu par lancé de rayon. Cette classe fournir un ensemble de fonction nécessaire au rendu tel que la gestion (chargement et activation) de shaders, le chargement des textures, la configuration des matrices de vue et de projection, et de commander le rendu des objets.
- L'Application
    - L'application fournit les fonctionnalités interactives du programme. L'application initialise du renderer à utiliser.
    - L'application gère également la création, l'instanciation, et le stockage de tous éléments graphiques (GameObjects) qui seront rendus par le renderer. L'application est responsable des interactions (input) au clavier, à la souris, et des interfaces utilisateurs des différentes parties du projet.

6) Fonctionnalités

1. Image

1.1. Importation d’images

Description : L'application permet l'importation interactive d'images en cours d'exécution via un système de drag & drop. Les images importées sont automatiquement positionnées dans la scène 3D à l'emplacement du curseur de la souris et converties en objets 3D affichables.

Implémentation : Le système utilise une bibliothèque d'images pour charger les formats PNG, JPG, JPEG et GIF. Les coordonnées écran sont converties en coordonnées monde via un calcul de ray casting sur le plan Z=0. Chaque image est transformée en quad 3D avec mapping de texture et intégrée au système de rendu. Les images importées sont automatiquement ajoutées à la liste des sources disponibles pour l'échantillonnage.

1.2. Exportation d’images

Description : L'application permet l'exportation de séquences d'images correspondant au rendu de la scène. L'utilisateur peut configurer le nombre de frames, le répertoire de sortie et le nom de base des fichiers.

Implémentation : Le système d'export capture le framebuffer OpenGL et sauvegarde les images au format PNG. La capture gère le retournement vertical nécessaire (OpenGL utilise l'origine en bas-gauche) et l'écriture séquentielle. L'export s'exécute de manière asynchrone avec un système de notification pour informer l'utilisateur du statut. Les paramètres d'export sont configurables via l'interface utilisateur.

1.3. Échantillonnage d’images

Description : L'application génère de nouvelles images en échantillonnant aléatoirement des pixels provenant d'une ou plusieurs images sources. L'utilisateur peut sélectionner les images sources, configurer la taille de sortie et la taille des tuiles d'échantillonnage.

Implémentation : Le système charge les images sources sélectionnées, utilise un générateur de nombres aléatoires pour sélectionner des pixels aléatoires de chaque source, et génère une grille de tuiles colorées. Chaque tuile est remplie avec la couleur du pixel échantillonné. L'image résultante est automatiquement ajoutée

à la scène au centre de l'écran. Le système supporte l'échantillonnage depuis plusieurs sources simultanément.

1.4. Palette de couleur

Description : L'application fournit une palette de couleurs interactive permettant de créer, modifier et sélectionner des couleurs pour les éléments visuels. La palette supporte au minimum 4 couleurs avec possibilité d'ajout/suppression dynamique.

Implémentation : La palette stocke des couleurs RGBA dans un système de gestion dynamique. L'interface utilisateur permet l'édition des couleurs, la sélection via des boutons radio, et la gestion dynamique (ajout/suppression) avec une contrainte minimale de 4 couleurs. Un système de notification permet aux autres composants de récupérer la couleur sélectionnée et d'être informés des changements

1.5. Histogramme

Description : L'application calcule et affiche l'histogramme de luminance d'une image sélectionnée, permettant l'analyse de la distribution des intensités lumineuses.

Implémentation : Le système charge l'image, calcule la luminance de chaque pixel en utilisant la formule standard de conversion RGB vers luminance, et construit un histogramme normalisé sur 256 bins. L'histogramme est affiché graphiquement avec une interface de sélection d'image source. Le système gère automatiquement la mise à jour de l'histogramme lors du changement d'image sélectionnée et affiche le nom de l'image source.

2. Dessin vectoriel

2.1. Curseur dynamique

2.2. .

2.3. Primitives vectorielles

Le projet permet de générer les primitives vectorielles suivantes : point, ligne droite, triangle, carré, rectangle, polygone régulier à n segments, cercle et ellipse.

Les primitives possèdent des attributs leur permettant d'influer sur leur apparence tel que l'épaisseur de la ligne de contour, la couleur de la ligne de contour, le remplissage (actif ou inactif), la couleur du remplissage, et dans

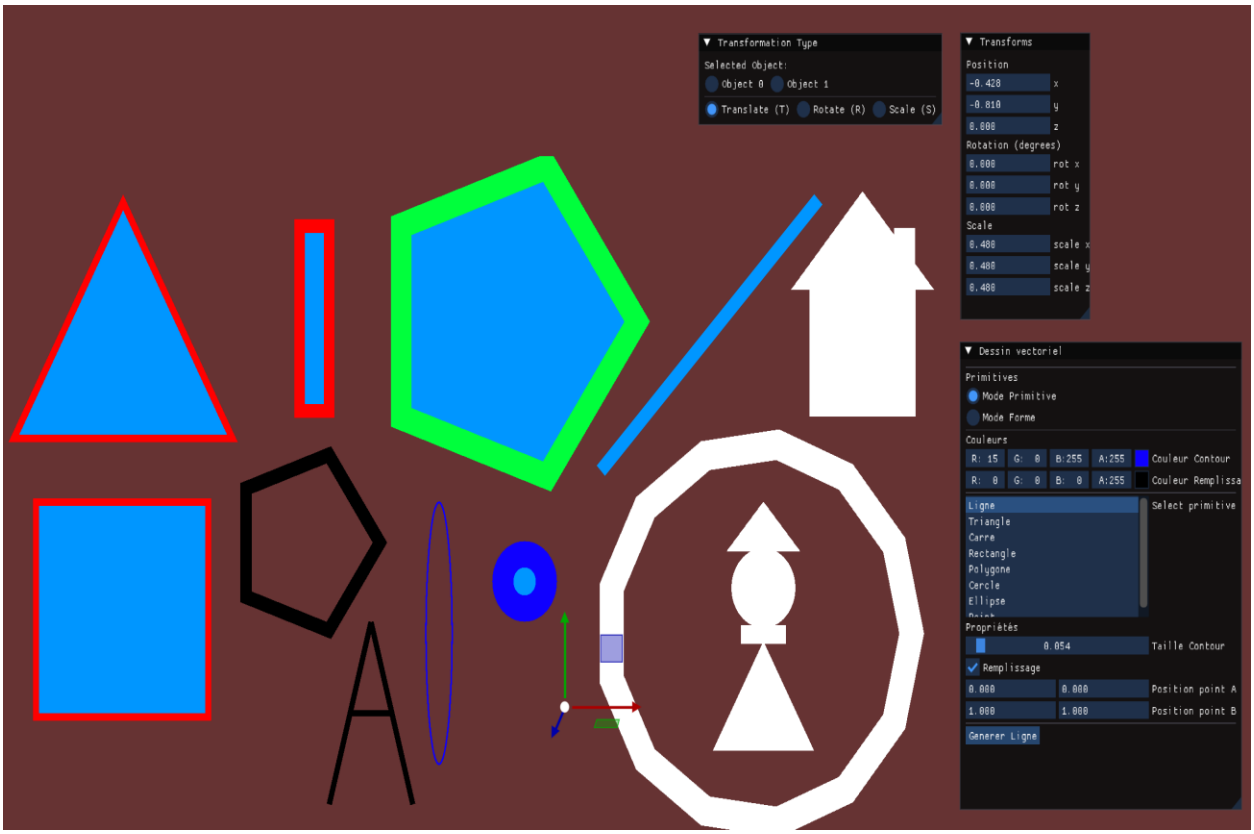
le cas des ellipse et rectangles : les proportions de génération sur les axes X et Y.

La majorité des primitives a été implémenté grâces aux concepts basiques d'un cercle trigonométrique.

Une classe RegularPolygon permet de générer toutes les autres à l'exception des lignes. RegularPolygon sert de base à la génération des triangles, carré, ellipse, cercle, point et polygones réguliers. Les classes dérivées influent sur les paramètres des axes du cercle trigonométrique et le nombre de segments à générer pour obtenir la primitive souhaitée.

La génération des vertices des primitives s'effectue en deux parties : le remplissage (si activé) et la bordure.

Afin de faire un rendu plaisant pour les primitives vectorielles, une fonction calcule des jointures (miter joints) à partir des positions des sommets suivant et précédent pour garantir une connexion propre et esthétique aux angles.



2.4. Formes vectorielles

Le projet implémente trois formes vectorielles : Une lettre A, réalisé à partir de trois lignes droites ; un pion/figure réalisé à partir de deux triangles, un

cercle et un rectangle ; et enfin une maison réalisée à partir d'un carré, d'un triangle et d'un rectangle.
Les formes vectorielles utilisent les primitifs précédemment implémentés et peuvent être colorés grâce à la couleur de remplissage.
Note : il n'y a pas de zone de remplissage, la couleur est appliquée à toute la forme (c'est à dire à toutes les primitives qui la compose).
Les formes sont implémentées sous forme de classe distinctes, ayant un vecteur de primitive prédéfinies. Elles ne sont pas modifiables.

2.5. Interface

Le projet possède un panneau dans l'interface graphique dédié au dessin vectoriel. Ce panneau permet de contrôler les propriétés de génération d'une primitive ou d'une forme vectorielle.
L'interface permet notamment de choisir le type de primitive ou de forme à générer, les couleurs de remplissage et de contour, l'épaisseur du contour, et des propriétés spécifiques aux primitives. Les contrôles se présentent sous la forme de listes de sélection, de color picker, de sliders et de boutons.
Les contrôles influent directement sur les paramètres de génération des primitives et des formes, et lorsque l'utilisateur appuie sur le bouton de génération, la primitive est instanciée dans la scène avec les bons paramètres. Cette interface a été implémenté grâce à la bibliothèque Imgui.

3. Transformation

3.1. Graphe de scène

Le projet possède un graph de scène qui permet à des objets d'avoir un parent et plusieurs enfants. L'origine d'un objet enfant est la position de son parent dans la scène. De ce fait toutes les transformations appliquées à un parent sont répercutées sur ses enfants.

3.2. Sélection multiple

Le projet permet d'effectuer une sélection sur de multiples objets de la scène. Les transformations sont appliquées à tous les objets sélectionnés et par leurs origines respectives. Cette sélection peut se faire directement dans le viewport par un raycast ou en cliquant sur la fenêtre du graph de scène.

3.3. Transformations interactives

Le projet permet d'appliquer des transformations aux objets de la scène par deux moyens différents : Une fenêtre des coordonnées / proportion / rotation des objets sélectionnés. Et un gizmo interactif positionné sur le dernier objet sélectionné et qui permet d'appliquer des translation, proportion et rotation, avec un visuel différent pour chacune de ces transformations grâce à la libraire ImGuizmo.

4. Géométrie

4.1. Boîte de délimitation (bounding box) | Il est possible d'afficher les boîtes de délimitation de soit tous les objets, soit d'un objet spécifique. L'algorithme utilisé est AABB. La boîte de délimitation s'affiche sous forme d'un wireframe verte juste assez grande pour envelopper l'objet.

4.2. Primitives géométriques | Il est possible de générer 3 primitives géométriques 3d différentes. Un cube, une sphère, et un cylindre. Pour le cube, on peut spécifier sa taille. Pour la sphère, on peut spécifier son rayon, son nombre de secteurs (délimitations verticales), et son nombre de stacks (délimitations horizontales). Pour le cylindre, on peut spécifier son rayon, sa hauteur, et son nombre de secteurs. Chaque primitive géométrique est générée avec une couleur aléatoire.

4.3. Modèles 3D | Il est possible de charger et de générer des fichiers sous format .obj depuis le disque local. Il faut tout d'abord cliquer sur "Load Model" et sélectionner le fichier souhaité dans l'explorateur. Ensuite, celui-ci est listé en-dessous du bouton mentionné. On peut supprimer l'entrée en appuyant sur la croix. On peut instancier l'objet autant de fois que souhaité en cliquant sur le bouton contenant son nom.

5. Caméra

5.1. Le projet possède une caméra interactive, grâce aux touches on peut déplacer la caméra dans la scène ainsi que la faire pivoter en utilisant le clique gauche de la souris. Ces actions vont venir changer la position et la rotation de la caméra, qui sont des éléments servent à construire la matrice de vu de celle-ci, ce n'est donc pas la caméra qui bouge mais tous les éléments qui vont être dessiné en fonction de la caméra

5.2. Le projet permet de crée et détruire de multiple caméra grâce à un gestionnaire de caméra, chaque caméra possède ses propres attributs de position et de rotation, ce qui fait que chaque caméra possède sa propre

matrice de vu indépendante des autres. Cela permet de changer entre différents points de vue.

5.3. Le projet permet aussi d'avoir de multiple caméra simultanément, chaque objet de la scène va être redessiner pour toutes les caméra qu'il y a à afficher selon la matrice de vu de chaque et ensuite chaque dessin est stocké dans un framebuffer qui est donné ImGui pour être affiché dans une sous-fenêtre et toutes ces sous-fenêtre sont géré grâce a ImGui, ce qui permet de les déplacer de changer le taille, ect.

7) Ressources

Partie dessin vectoriel : Utilisation de Google Gemini pour les calculs de jointures Miter.

Partie Image:

- Bibliothèques et dépendances externes
    - STB Image : Bibliothèque header-only pour le chargement et l'écriture d'images (PNG, JPG, JPEG, GIF). Utilisée pour l'importation d'images, l'export de séquences et l'échantillonnage.
    - ImGui : Framework d'interface utilisateur immédiate pour créer l'interface de gestion des images, de la palette de couleurs, de l'échantillonnage et de l'histogramme.
    - Formule de luminance : Utilisation de la formule standard ITU-R BT.709 pour le calcul de l'histogramme de luminance
    - OpenGL/GLAD : Pour le rendu des images importées et la capture du framebuffer lors de l'export.
    - GLM : Pour les calculs mathématiques de conversion de coordonnées écran vers coordonnées monde.
- Modèles d'intelligence artificielle
    - GPT-5 a été utilisé pour l'assistance au développement de la conversion de coordonnées écran vers coordonnées monde

Partie Géométrie:

- Utilisation de la bibliotèque ImGuiFileDialog pour l’explorateur de fichiers
- Les modèles utilisés sont les suivants :
    - Utah teapot
    - Stanford bunny
    - Suzanne (Blender monkey)

8) Présentation
    - Yohan Decamps
        - Développement de la partie transformation + base du projet
    - Virgile Legros
        - Développement de la partie Géométrie
    - Théo Fabiano
        - Développement de la partie image + merge et fix des bugs
    - Lucas Loustalot
        - Développement de la partie dessin vectoriel.