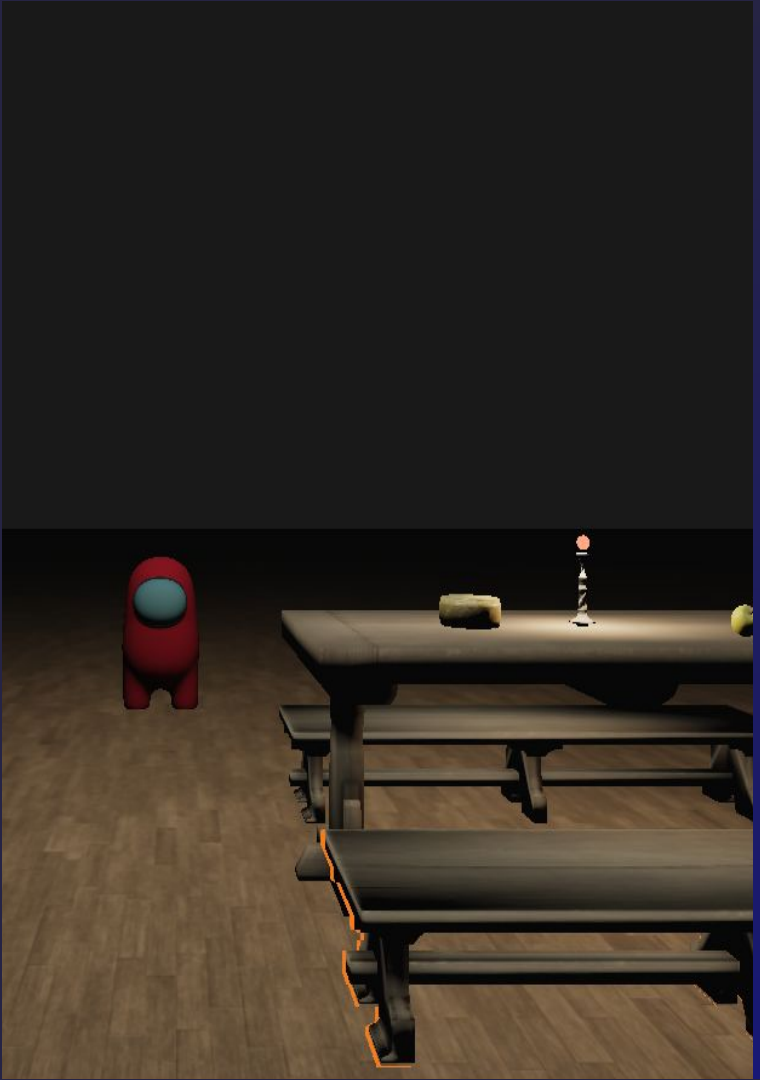


Projet Moteur de jeu

ELYS Engine

Benjamin VILLA - Pierre HENNECART - Samuel HELYE

SOMMAIRE

01

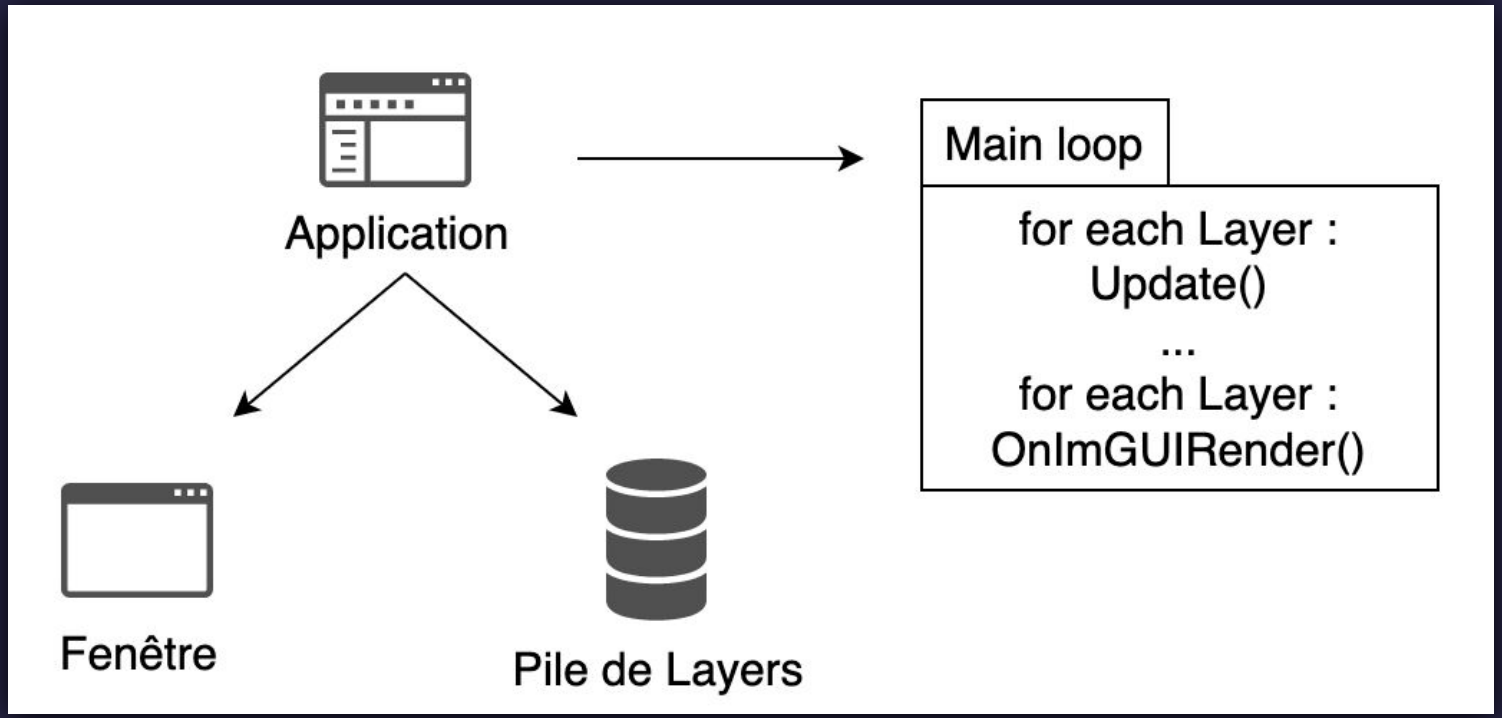
DÉMONSTRATION

Un aperçu du moteur

ARCHITECTURE

Application, Évènements, ECS
et Graphe de scène

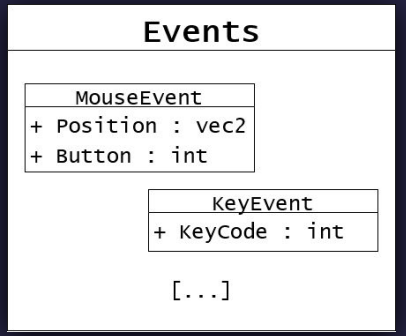
Structure de l'application

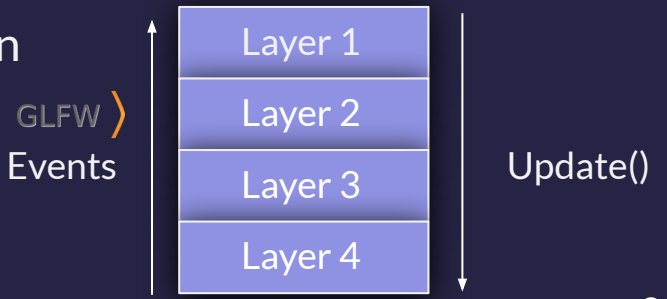


Gestion des évènements

Cycle de vie d'un évènement

- Récupéré depuis l'OS via GLFW
- Redirigé vers l'Application à travers des callbacks
- Propagé à tous les Layers dans l'ordre inverse
- Chaque Layer peut décider de stopper la propagation en détruisant l'évènement





Entity Component System

Entity Manager:

- Une entité est un ID (uint32_t)
- Garde une pool d'ID disponible et les distribue

Component Manager:

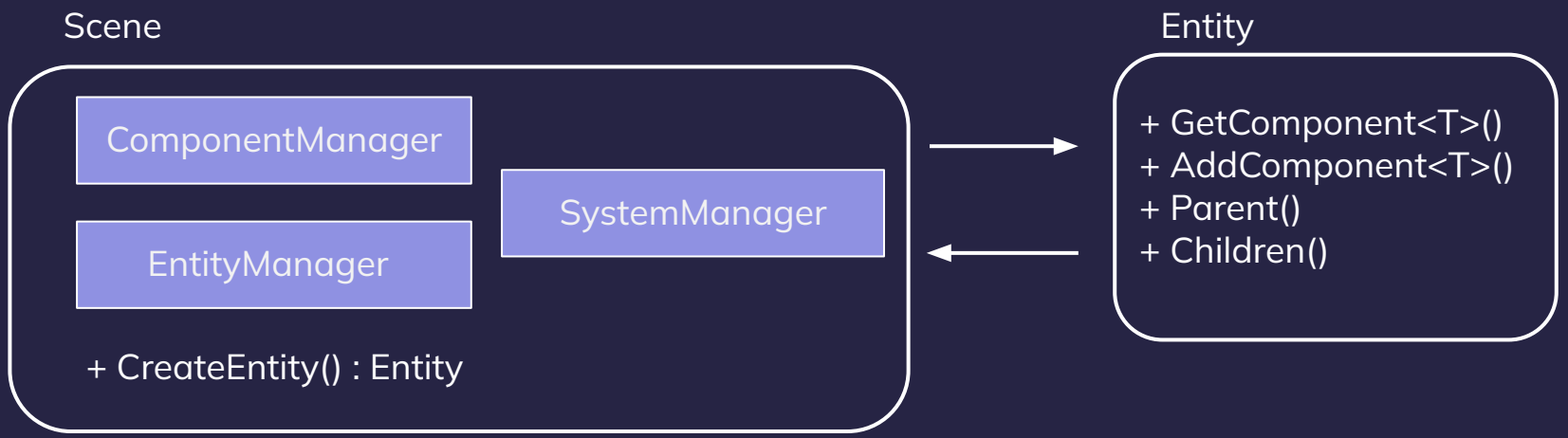
MAX_ENTITIES

0001	0	1	2					
0010	2							
0100	2	1						
1000	1	3	0					

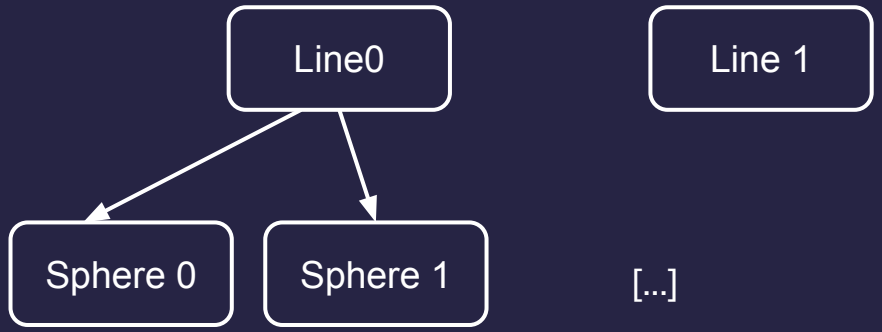
ComponentAmount

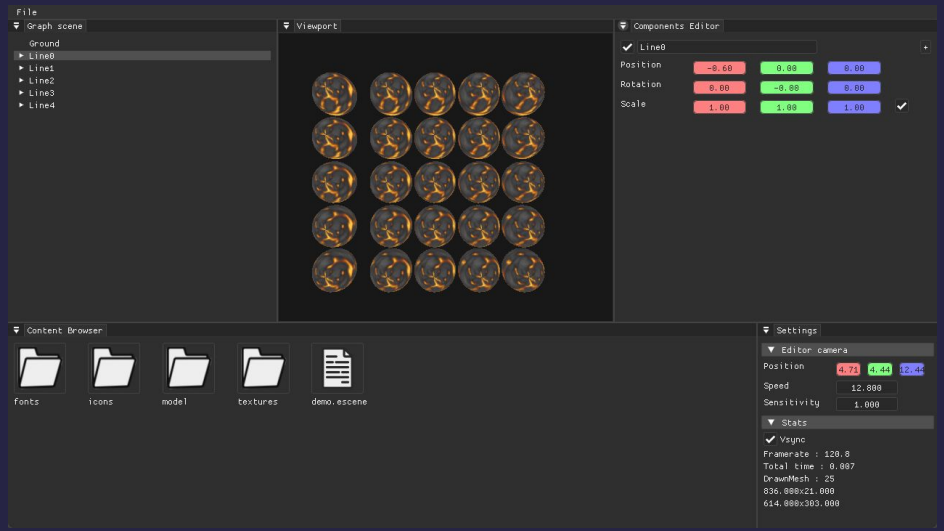
System manager:

Graphe de scène



Le composant important : Node





Le composant important : Node

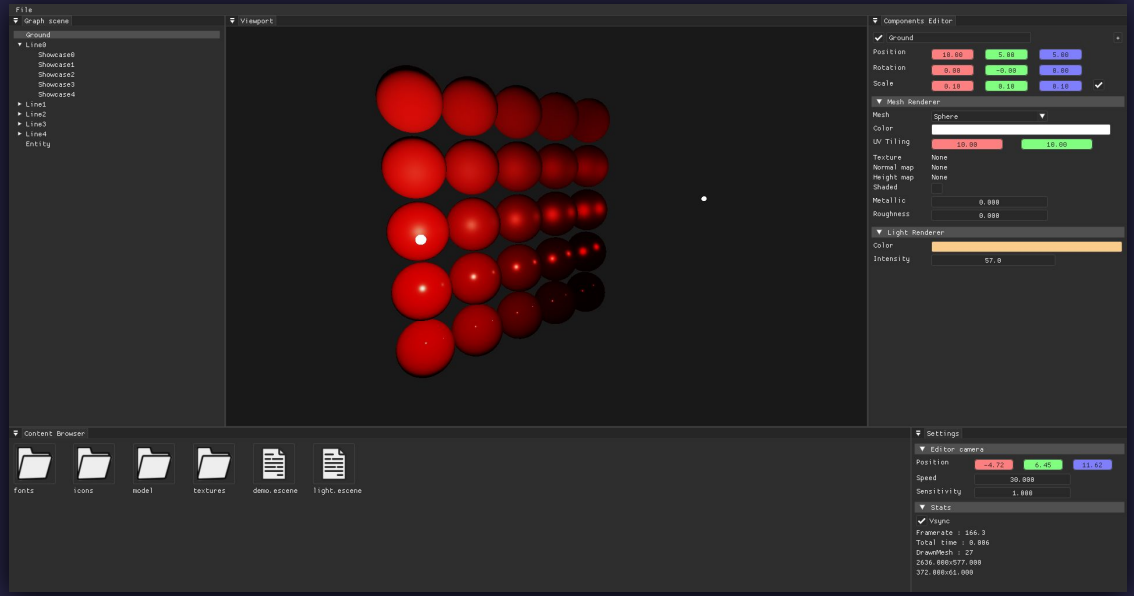
```
class Node {
  public:
    std::string name = "Entity";
  public:
    Node();

    void SetParent(Node* parent);
    void AddChild(Node* child);
    void RemoveChild(Node* node);
    [[nodiscard]] Node* Parent() const;
    [[nodiscard]] std::vector<Node*> Children() const;

    void OnDelete();

    [[nodiscard]] glm::vec3 InheritedPosition() const;
    [[nodiscard]] glm::vec3 InheritedScale() const;
    [[nodiscard]] glm::quat InheritedRotation() const;
    [[nodiscard]] glm::mat4 InheritedTransform() const;
    [[nodiscard]] bool InheritedEnabled() const;
    [[nodiscard]] glm::vec3 LocalPosition() const;
    [[nodiscard]] glm::vec3 LocalScale() const;
    [[nodiscard]] glm::quat LocalRotation() const;
    [[nodiscard]] glm::mat4 LocalTransform() const;
    [[nodiscard]] bool LocalEnabled() const;

  private:
    void InvalidateNode() const;
    void UpdateTransform() const;
  private:
    Node* mParent = nullptr;
    std::vector<Node*> mChildren;

    bool mLocalEnabled = true;
    glm::vec3 mLocalPosition{ x: 0.0f,  y: 0.0f,  z: 0.0f};
    glm::vec3 mLocalScale{ x: 1.0f,  y: 1.0f,  z: 1.0f};
    glm::quat mLocalRotation{ eulerAngle: glm::vec3( x: 0.0f,  y: 0.0f,  z: 0.0f)};
    mutable glm::mat4 mLocalTransform{ s: 1.0f};

    mutable glm::vec3 mGlobalPosition{ x: 0.0f,  y: 0.0f,  z: 0.0f};
    mutable glm::vec3 mGlobalScale{ x: 1.0f,  y: 1.0f,  z: 1.0f};
    mutable glm::quat mGlobalRotation{ eulerAngle: glm::vec3{ x: 0.0f,  y: 0.0f,  z: 0.0f}};
    mutable bool mGlobalUpdated = false;
    mutable glm::mat4 mGlobalTransform{ s: 1.0f};
};
```

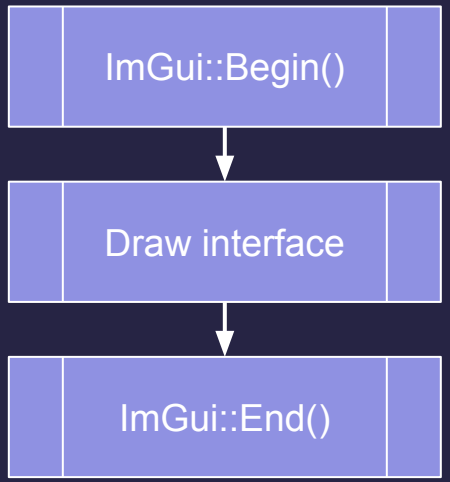
03

INTERFACE

Interface graphique de l'éditeur
via la bibliothèque ImGUI

Présentation de ImGUI (Immediate mode GUI)



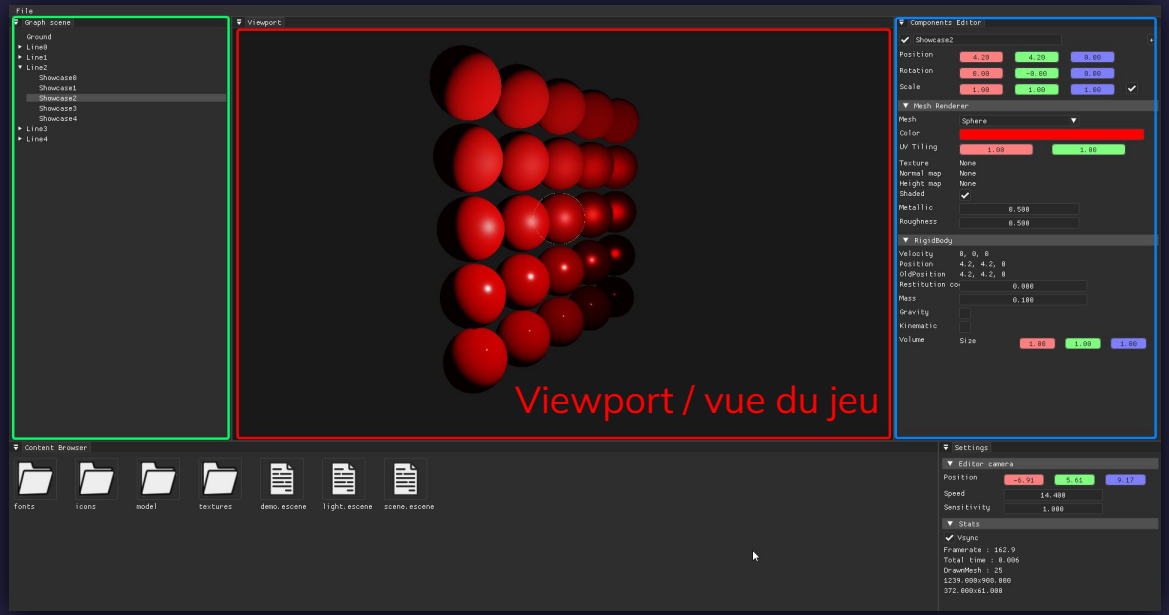


ELYS Engine

Les composantes de l'interface

Editeur des composants

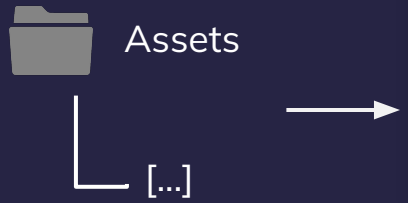
Graphe de scène

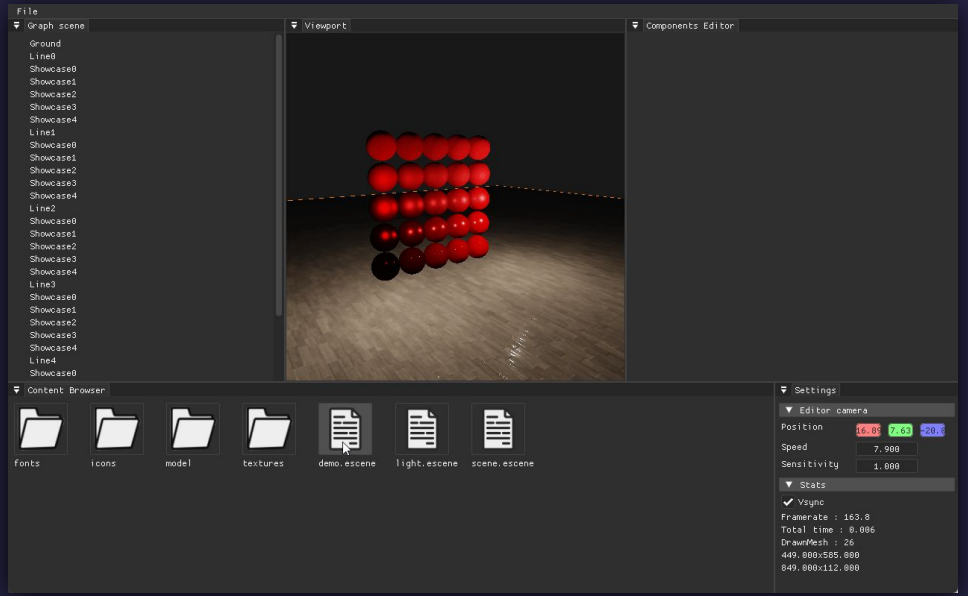


Possibilité de sélectionner depuis le graphe de scène ou en survol sur le viewport

Chargement des assets et navigateur

- Objet statique avec cache pour charger les assets
- Serialization/Déserialization de la scene





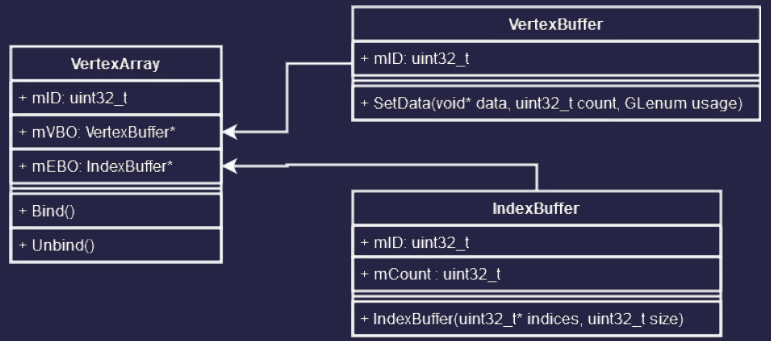
Rendu

Abstraction OpenGL, Systèmes de rendu,
Lumières et Recherches

Abstraction OpenGL

Pourquoi ?
- Réduire la verbosité
- Améliorer les contrôle des ressources (glGen/glDelete)

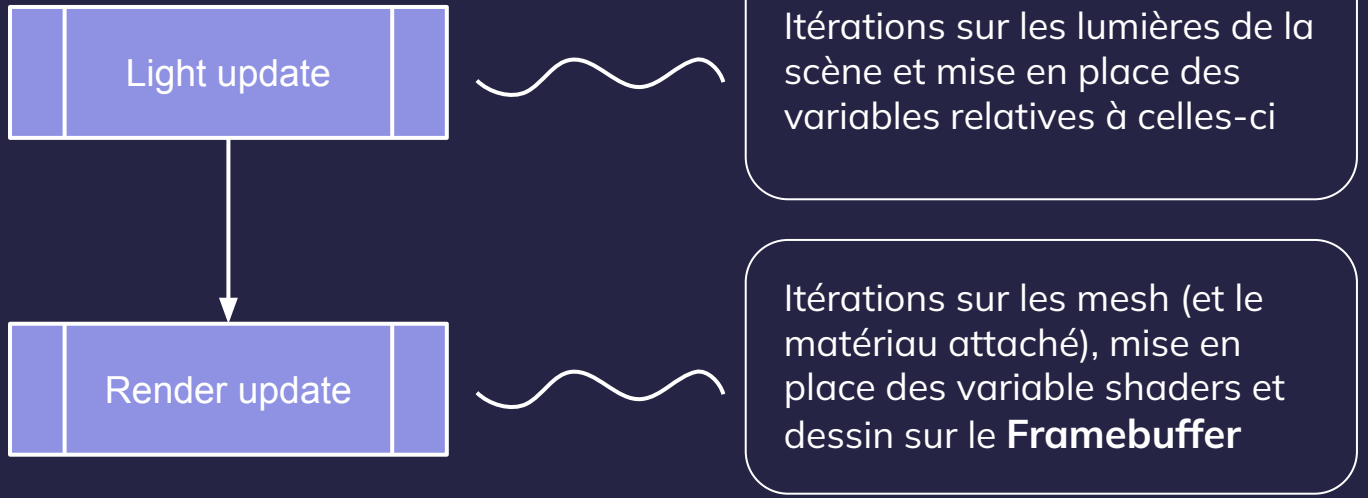
Quelles structures ?
- Texture
- Framebuffer (modulable avec les attachments)
- Buffers / VertexArray



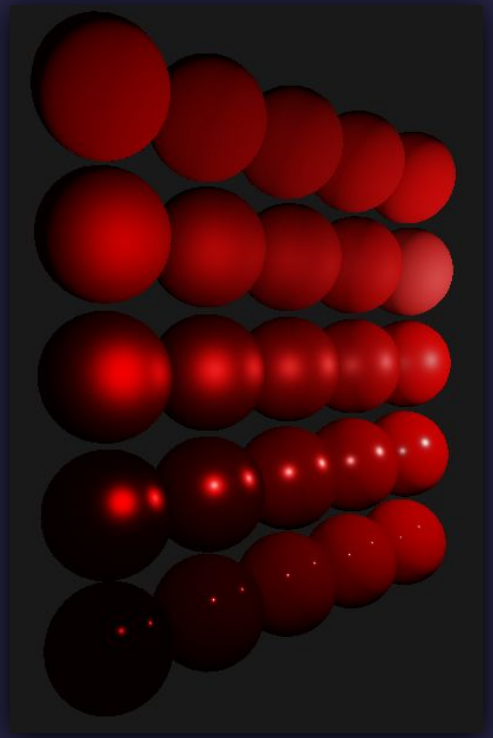
Système de rendu

Deux systèmes clefs :

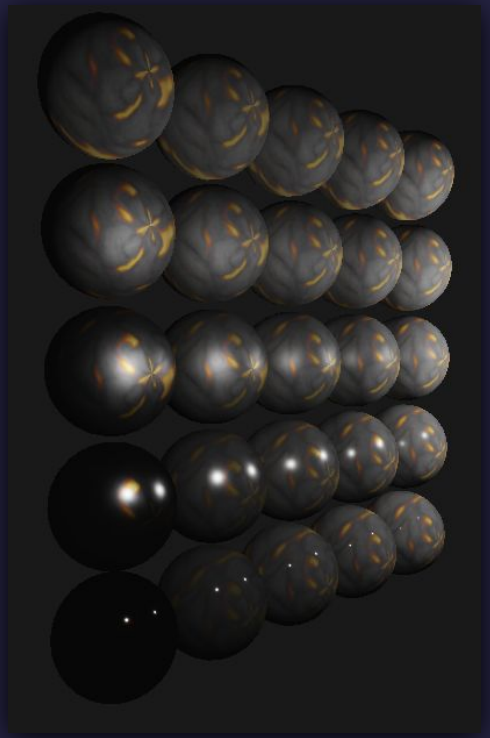
- **Light system (gère les entités avec le composant Light)**
- **Render system (gère les entités avec le composant MeshRenderer)**



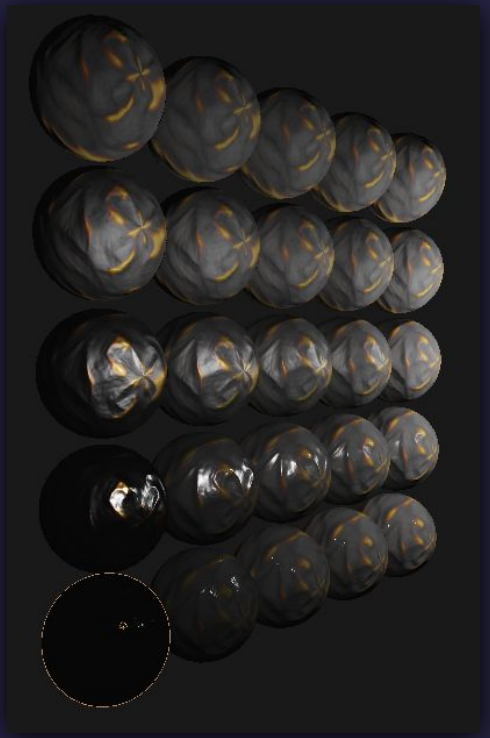
Lumières



PBR avec Metallic/Roughness



Application de texture simple



Application de normal maps

Recherches

Pré-calcul des espaces tangent

Qui implique parfois de dédoubler des sommets (sur un cube par exemple)

VS

Calcul on-fly dans le shader

Ne modifie pas la structure de stockage des sommets (avec une contrepartie sur du calcul)

05

PHYSIQUE

AABB/OBB, Collisions,
Impulsion linéaire et angulaire, RigidBody

Game Physics Cookbook, **Gabor Szauer**

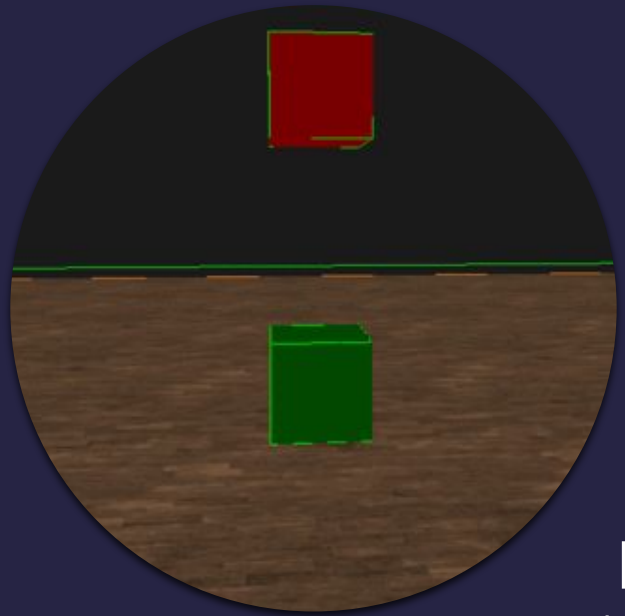
Forces

Linear Impulse

Impulsion modifiant la Vélocité Linéaire

Angular Impulse

Impulsion modifiant la Vélocité Angulaire



Gravité

Application de la force constante { 0, -9.81, 0 }

Friction

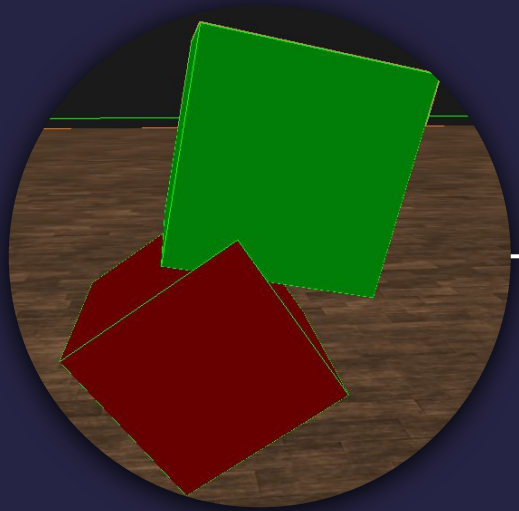
Impulsion tangentielle due au frottement

Volumes englobants

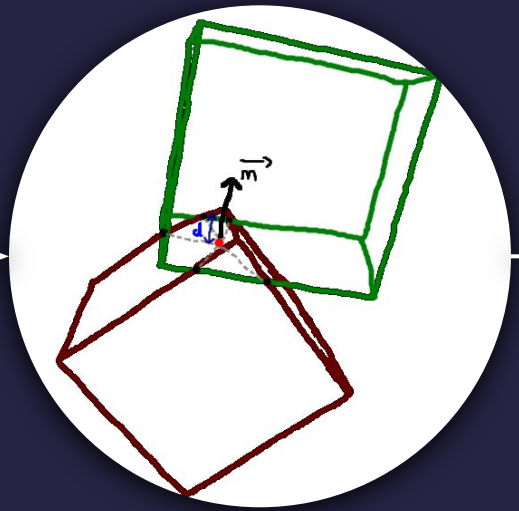
Axis-Aligned Bounding Box (AABB)

Oriented Bounding Box (OBB)

Détection de collision



Détection de collision
entre deux Bounding box



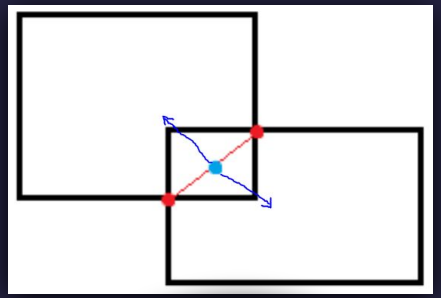
Calcul des points de contact,
de la normale et de la
profondeur de pénétration

```
struct CollisionManifold {
    bool colliding;
    glm::vec3 normal;
    float depth;
    std::vector<glm::vec3> contacts;
};
```

Contient les informations
de la collision

Réponse impulsionnelle

Vélocité **Linéaire**



Vélocité relative

$$V_r = V_B - V_A$$

Coefficient de restitution

$$e = \min(e_A, e_B)$$

Magnitude de l'impulsion

$$j = \frac{-(1+e)(V_r \cdot n)}{\frac{1}{mass_A} + \frac{1}{mass_B}}$$

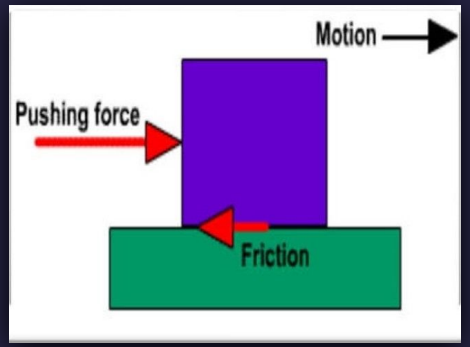
Impulsion linéaire

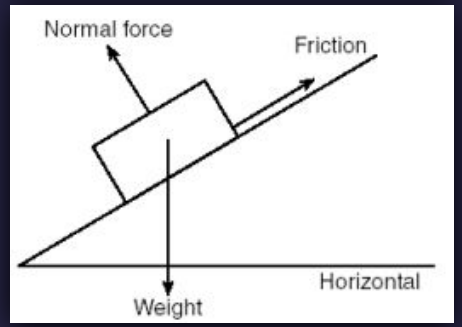
$$V'_A = V_A + \frac{j}{mass_A} n$$

$$V'_B = V_B - \frac{j}{mass_B} n$$

Réponse impulsionnelle

Friction





Vecteur tangent à la normale

$$t = V_R - (V_R \cdot n)\, n$$

$$jt = \frac{-(1+e)(V_r \cdot t)}{\frac{1}{mass_A} + \frac{1}{mass_B}}$$

Magnitude de la friction

$$friction = sqrt(friction_A, friction_B)$$

$$jt = \max(jt, -j * friction)$$

$$jt = \min(jt, j * friction)$$

Loi de Coulomb

Friction

$$V'_A = V_A + (t * jt)$$

$$V'_B = V_B - (t * jt)$$

Réponse impulsionnelle

Vélocité **Angulaire**

Tenseur d'inertie

$$\begin{bmatrix} \frac{1}{12}m(y^2+z^2) & 0 & 0 \\ 0 & \frac{1}{12}m(x^2+z^2) & 0 \\ 0 & 0 & \frac{1}{12}m(x^2+y^2) \end{bmatrix}$$

Magnitude de l'impulsion

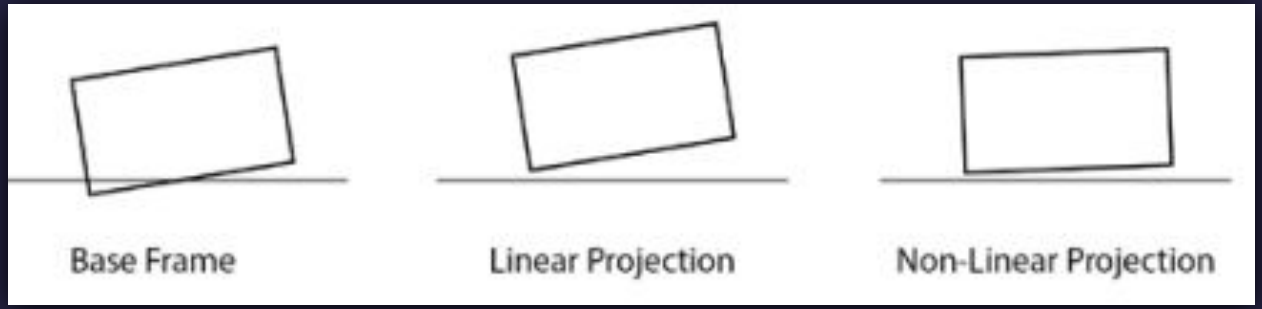
$$j = \frac{-(1+e)(V_r \cdot n)}{\frac{1}{mass_A} + \frac{1}{mass_B}} \rightarrow j = \frac{-(1+e)(V_r \cdot n)}{\frac{1}{mass_A} + \frac{1}{mass_B} + n \cdot \left[\left(\frac{r_A \times n}{I_A}\right) \times r_A\right] + n \cdot \left[\left(\frac{r_B \times n}{I_B}\right) \times r_B\right]}$$

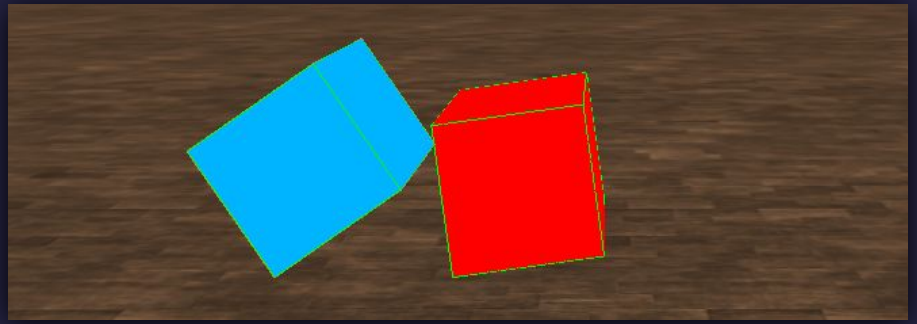
$$jt = \frac{-(1+e)(V_r \cdot t)}{\frac{1}{mass_A} + \frac{1}{mass_B}} \rightarrow jt = \frac{-(1+e)(V_r \cdot t)}{\frac{1}{mass_A} + \frac{1}{mass_B} + n \cdot \left[\left(\frac{r_A \times t}{I_A}\right) \times r_A\right] + n \cdot \left[\left(\frac{r_B \times t}{I_B}\right) \times r_B\right]}$$

Magnitude de la friction

Réponse impulsionnelle

Vélocité **Angulaire**





ELYS Engine

Amélio

+ Un jeu
+ Utilisation des colliders comme trigger
+ Structure d’optimisation pour les collision
+ Shadow map (accessible avec la code base disponible)
+ Composant et système pour le scripting
+ Déplacement personnage complet

MERCI !
DES QUESTIONS ?