Modélisation 3D en temps réel et physique dans Unity

Léo-Paul SEVIN

- Master IVI 2015-2016
- Stage et CDD chez Homido
- CDI chez Taktus 2017-2018
- Microentrepreneur depuis Janvier 2019

Projets en solo

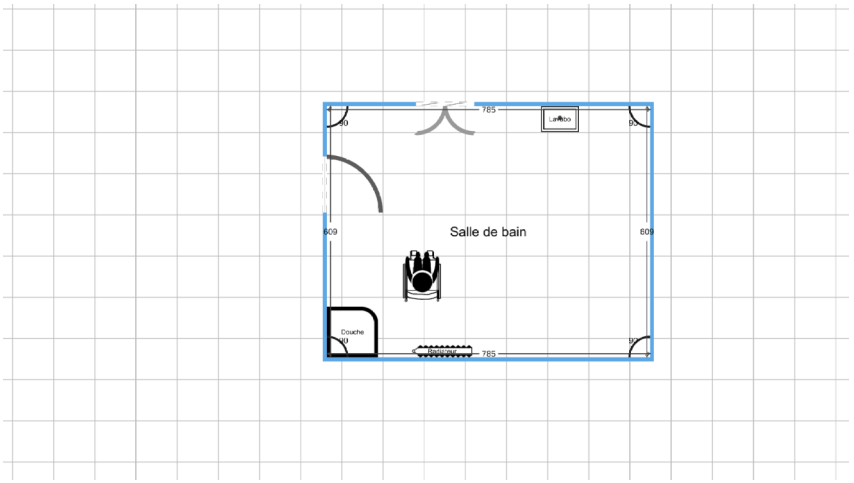
- Microentrepreneur depuis Janvier 2019

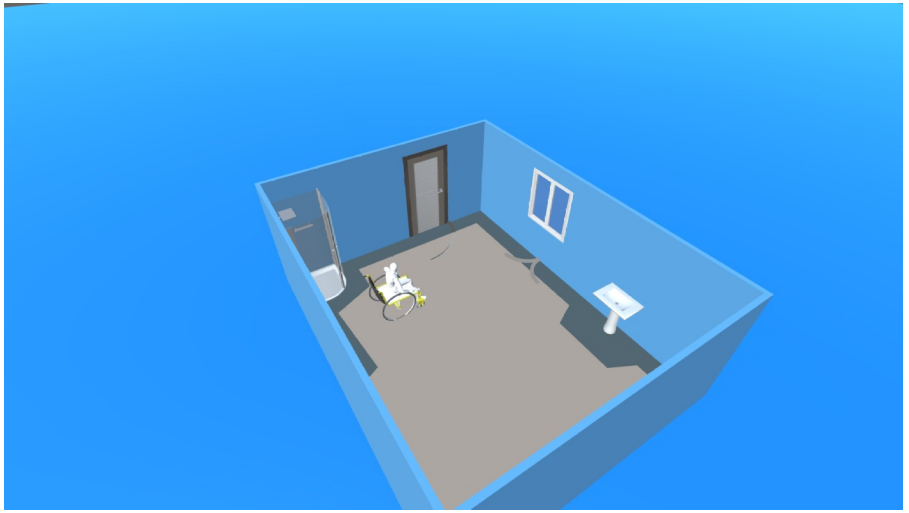
Projets en solo

- Microentrepreneur depuis Janvier 2019
  - Décathlon Trampo AR

Projets en solo

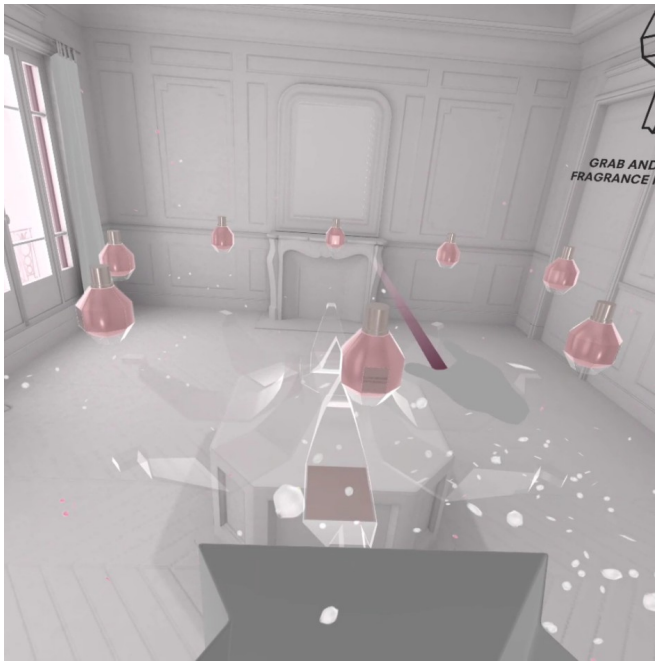
- Microentrepreneur depuis Janvier 2019
  - ErgoShop / Amen’Age





Projets en solo

- Microentrepreneur depuis Janvier 2019
  - Experience Oculus Quest pour L’Oréal



Projets en solo

- Microentrepreneur depuis Janvier 2019
  - Jeu Vidéo « Perpetual Dream » (achetez le merci)

Où sont les autres IVI ?

- Environ 50/50 IVI / Pas IVI

Modélisation temps réel

- Objectif : Voir le fonctionnement d’un Mesh sous Unity et d’un Rigidbody
- Étapes :
  - Création des vertices, faces, et normales
  - Ajout de materiau
  - Gestion des collisions et rigidbodies

Modélisation temps réel

- **Création des « cailloux »**
  - Créer un Mesh de cube
  - Demander à Unity de l’afficher
  - Modifier la méthode de la création du cube avec du random pour le déformer

Modélisation en temps réel

- Créer un cube :
  - A partir d'une longueur / largeur / profondeur
  - Créer 8 Vector3 a, b, c, d, e, f, g, h
  - Créer un Vector3[] contenant 6*4=24 vertices composés des 8 Vector3 créés
  - Créer un int[] contenant 6*2*3=36 indices pour indiquer les 12 triangles.
  - Créer un Vector3[] contenant 6*4 = 24 normales

Schéma du cube à faire

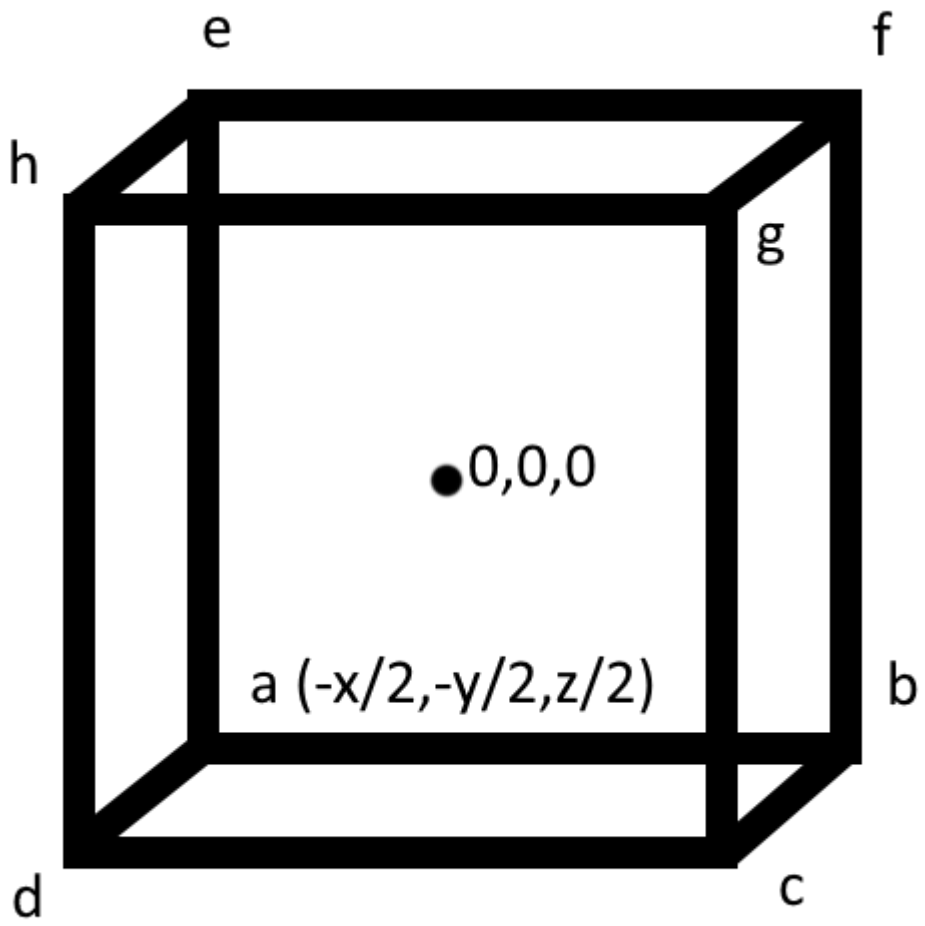


Schéma du cube à faire

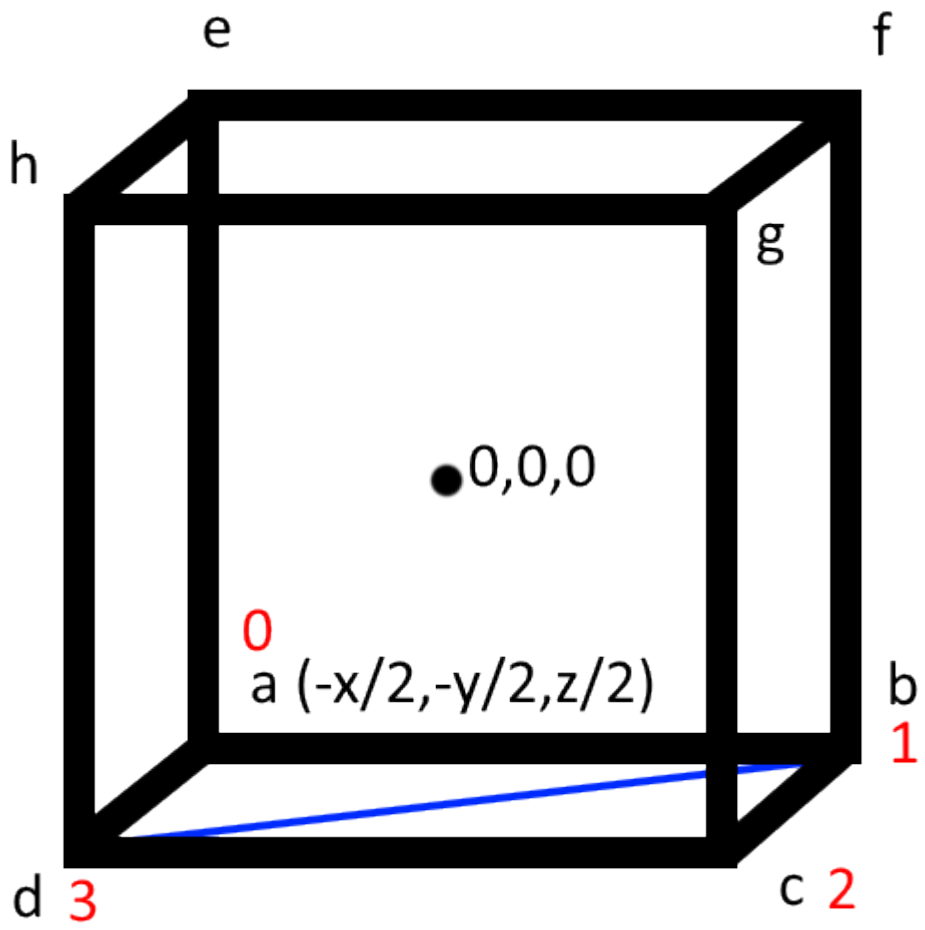


Schéma du cube à faire

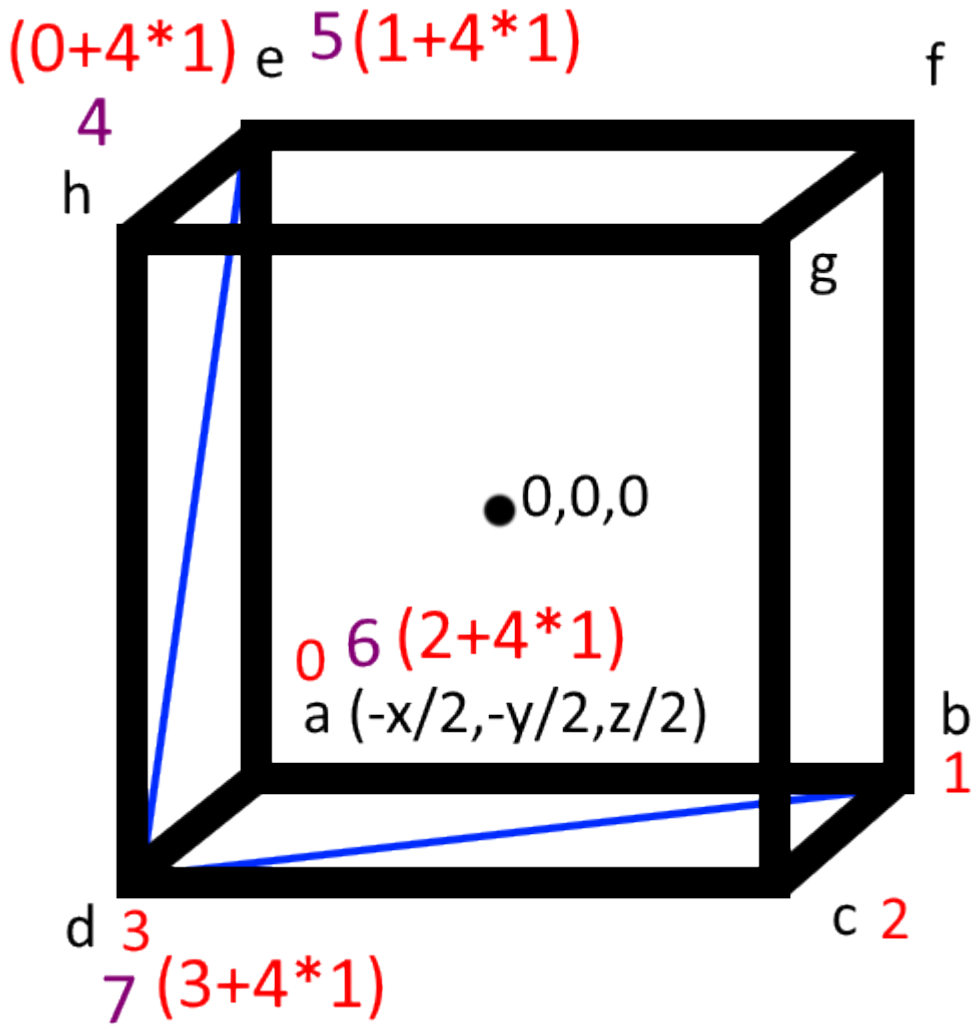
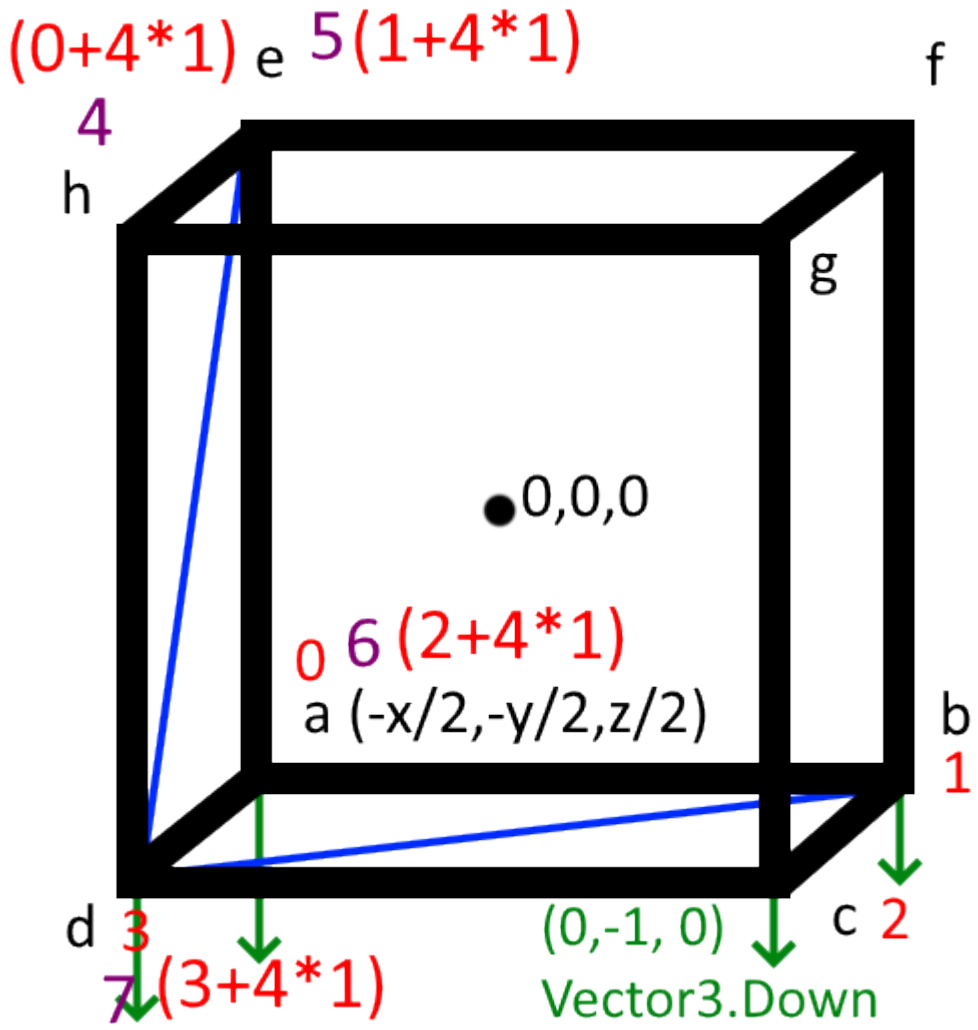


Schéma du cube à faire



Création du GameObject

- Components du *GameObject* :
  - *MeshFilter* : classe englobant le *Mesh*
  - *MeshRenderer* : classe qui affiche le *Mesh* passé par le *MeshFilter*
  - *MeshCollider* : Boite de collision autour du *Mesh*
  - *Rigidbody* : Component gérant la physique, avec le collider

PAUSE CAFÉ

La physique dans Unity

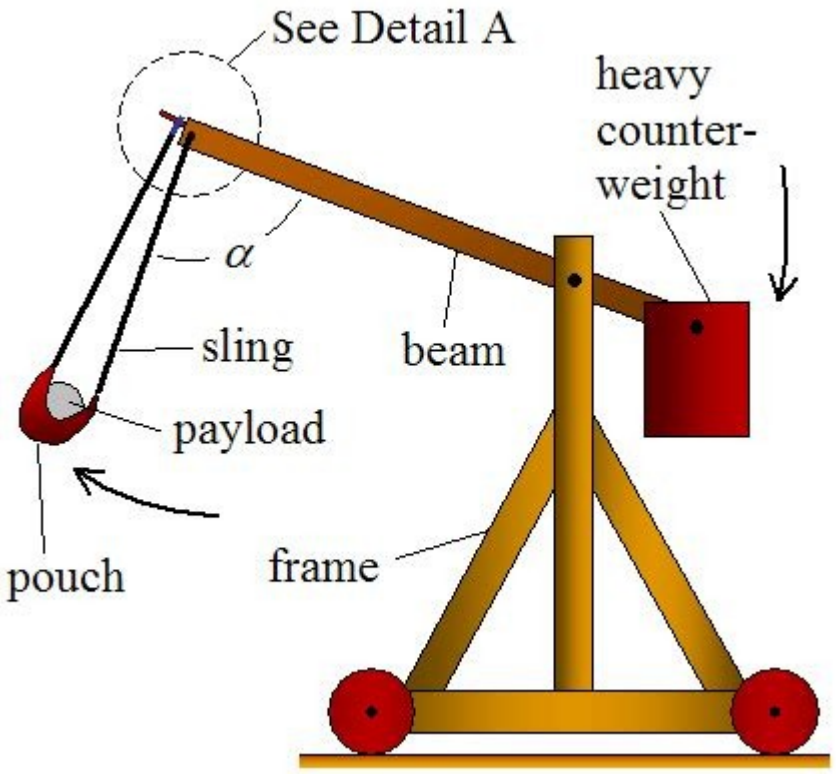
- *Rigidbody : Quelques propriétés*
- Velocity : Vector3 qui indique la direction du rb
- UseGravity : Force de gravité appliquée à chaque frame ou non (par défaut 9,81 m·s−2)
- Iskinematic : Aucune gestion des physiques (utile pour appliquer ses propres règles)

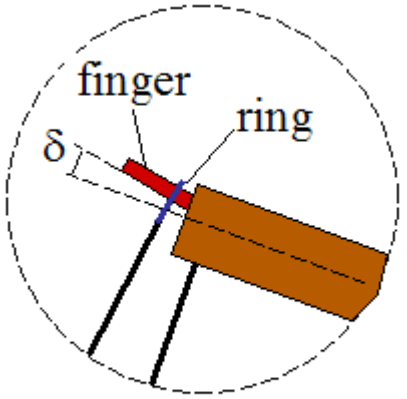
La physique dans Unity

- *Rigidbody : Quelques méthodes*
- *Toucher directement à la velocity : un peu random*
- *MovePosition(Vector3 position)*
  - *Déplace le rigidbody en ignorant sa vélocité actuelle (et la remet à jour)*
- *AddForce(Vector3 force)*

Faire un trébuchet

- Créer un système qui permet de projeter un caillou de 90kg sur une distance de 300m





Detail A

Faire un trébuchet

- Créer un système qui permet de projeter un caillou de 90kg sur une distance de 300m
- HingeJoint
  - Relie deux rigidbodies ensemble via un axe à définir (Vector3)
- SpringJoint
  - Relie deux rb ensemble comme un élastique/corde

Idées pour continuer

- Ajouter un mur à détruire
- Faire un caillou qui se déforme dans un Update
- Mettre le caillou généré dans le trébuchet et le lancer
- Méthode pour recharger le trébuchet