Sorbonne Université (Master ANDROIDE)

Projet ISG
LightMare

Auteur :
Adrien Arts
Binh Thanh Luong

Encadrants :
Tristan Briant
Amel Yessad
Mathieu Muratet

17 janvier 2019

Table des matières

1 Introduction **2**

2 6 facettes de la conception du jeu **2**
2.1 Facette 1 : Objectifs pédagogiques 2
2.2 Facette 2 : Simulation du domaine 2
2.3 Facette 3 : Décorum 2
2.4 Facette 4 : Interactions avec le modèle 3
2.5 Facette 5 : Problèmes et Progression 4
2.6 Facette 6 : Conditions d'utilisation 5

3 Architecture Entité-Composant-Système **5**
3.1 Transformation 5
3.2 Comparaison 6

4 Éditeur de niveaux **7**

5 Traçage **7**

1 Introduction

Dans le cadre de l'UE ISG, nous souhaitons développer un jeu sérieux pour le domaine de l'éducation. Nous utilisons dans ce projet une librairie pour manipuler les éléments du jeu.

2 6 facettes de la conception du jeu

2.1 Facette 1 : Objectifs pédagogiques

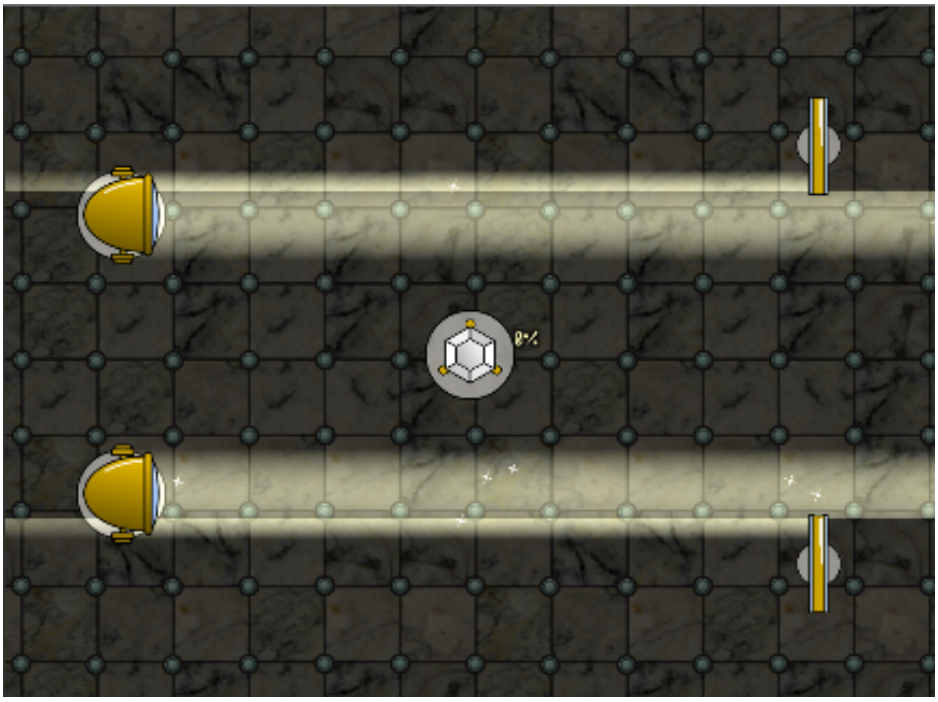
Notre jeu porte sur l'optique physique. Le but est d'étudier les propriétés différentes des éléments optiques ainsi que leurs comportements lors de contact avec des rayons lumineux. Le jeu sera utilisé en parallèle d'un cours de physique de L1. Il ne sers cependant pas à enseigner des formules mais des intuitions. C'est pourquoi aucune formule n'est visible dans l'interface.

2.2 Facette 2 : Simulation du domaine

La zone d'espace couverte par un faisceau lumineux étant théoriquement infini, nous avons discrétisé la lumière sous forme d'un ensemble fini de rayons. Nous utiliserons ici 100 rayons pour représenter ce faisceau. Ces rayons permettent, en faisant varier leur angle de propagation, de représenter a la fois la dispersion de la lumière et sa déviation par un objet.

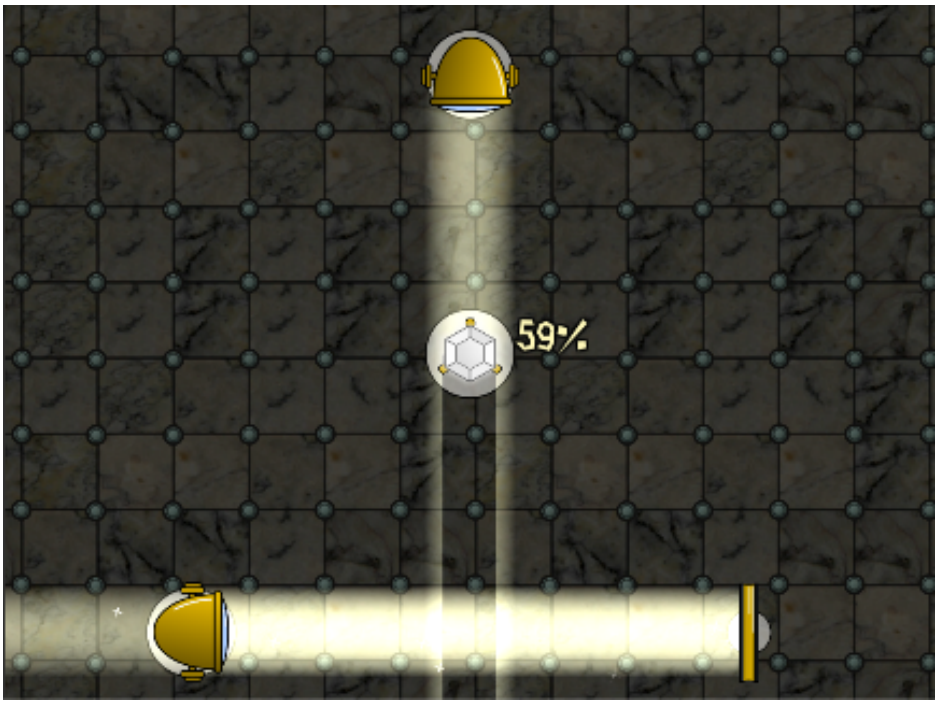
2.3 Facette 3 : Décorum

Le simulateur de comportement de la lumière est recouvert pars une interface de puzzle game classique. Nous disposons d'une source lumineuse que l'on doit dévier jusqu'à une cible en appliquant les connaissances acquise sur le comportement de cette lumière face a différents éléments. On retrouve ici quelques éléments de gamification tels que la cible affichant un score (représentant le pourcentage de la lumière de départ qu'elle reçoit) ou les étincelles parcourant les rayon de lumière afin d'attirer l'oeil.



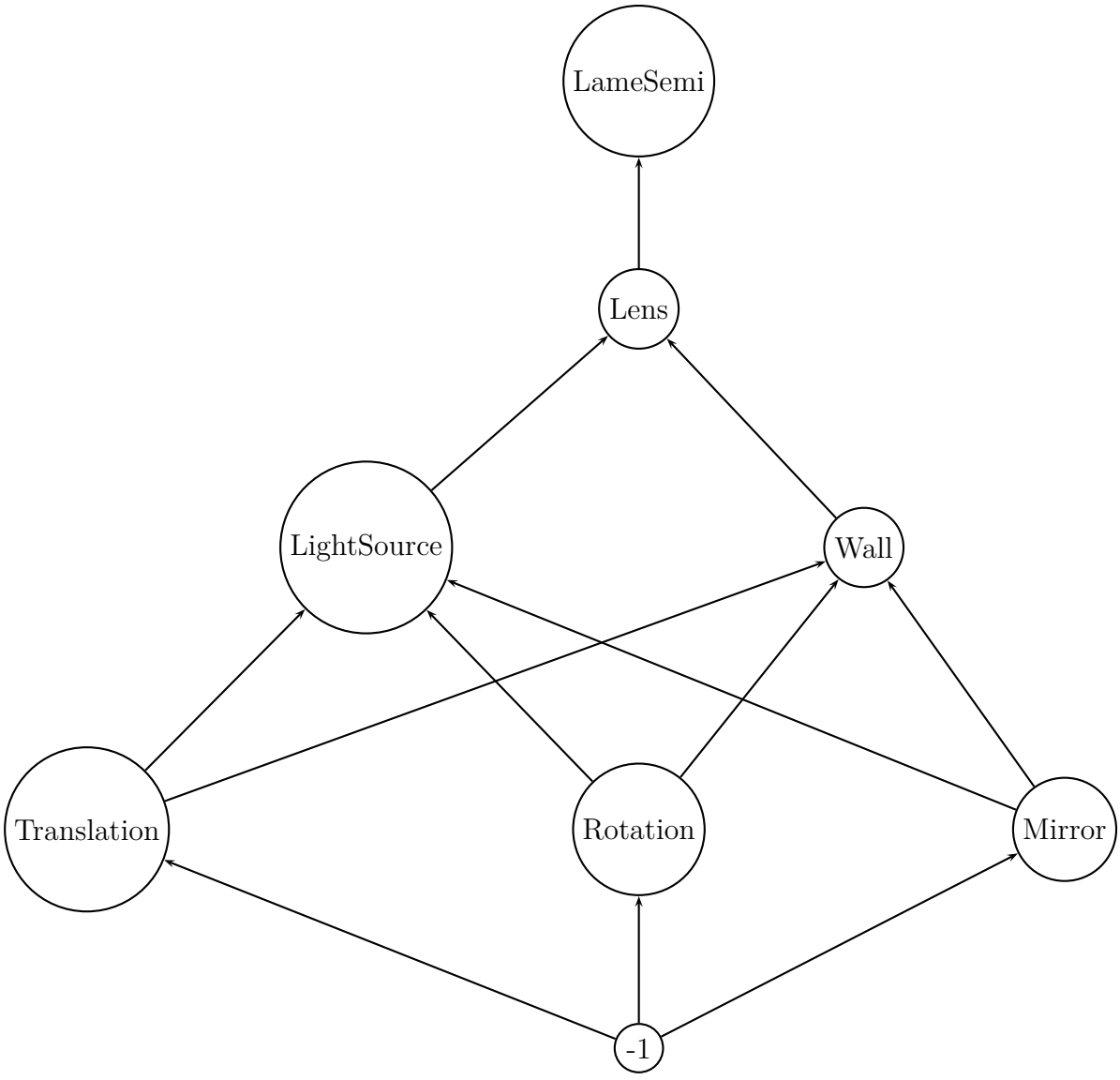
2.4 Facette 4 : Interactions avec le modèle

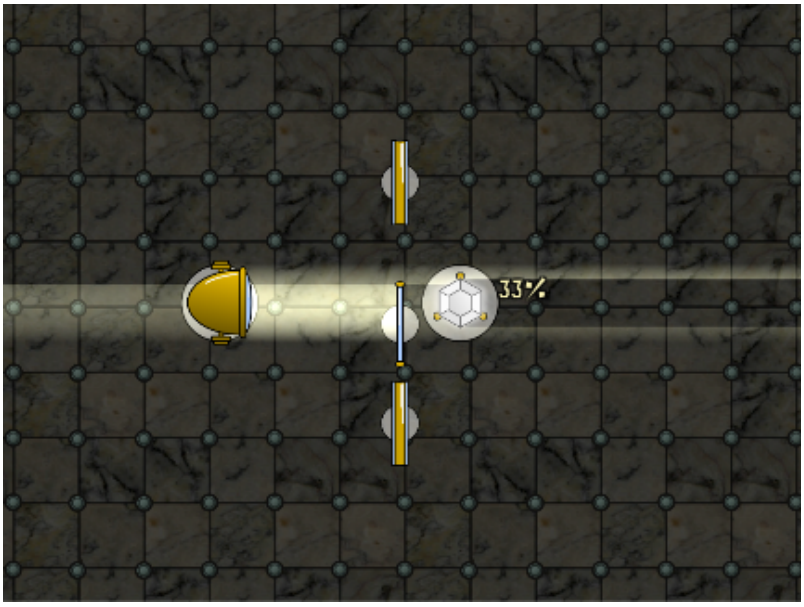
La règle du jeu est très simple : le joueur dispose des éléments optiques sur un plan et il doit faire en sorte que les rayons lumineuses aillent vers une cible fixée sur la scène. Il ne peut que déplacer ou tourner ces éléments pour atteindre le but.



2.5 Facette 5 : Problèmes et Progression

Nous supposons que le miroir est l'élément le plus familier, nous l'introduisons dès les premiers niveaux en même temps avec les actions possibles du gameplay (Drag And Drop). Au fur et à mesure, nous insérons d'autres éléments tels que le mur, les lentilles. Le graphe ce-dessous montre la complexité du jeu.





Un niveau avec 2 miroirs et une lame semi-réfléchissante

2.6 Facette 6 : Conditions d'utilisation

Notre jeu a été pensé pour être le complément d'un cours de L1, en tant qu'application mobile externe. C'est pourquoi l'interface et les commandes sont simplifiées. L'idée derrière cette application est d'occuper les longs trajets avec une activité en rapport avec le cours. Les étudiants de L1 utiliseront ainsi notre jeu en total liberté par rapport au cours. Ce ne serait en aucun cas un devoir maison mais une envie de la part de l'élève de développer des intuitions quand a son cours.

3 Architecture Entité-Composant-Système

Étant donné le moteur de jeu avec toutes les formules optiques en programmation orientée objet, nous souhaitons le transformer en architecture Entité-Composant-Système (ECS). Nous avons utilisé la librairie FYFY développée par l'équipe Modèles et Outils en ingénierie des Connaissances pour l'Apprentissage Humain (MOCAH) de Sorbonne Université. Nous avons tout d'abord transformé le modèle fourni en ECS puis nous comparerons les avantages et inconvénients des 2 structures.

3.1 Transformation

Le jeu a été implémenté en Unity 2D avec plusieurs scènes dont la première scène vide qui crée un moteur de jeu avec les propriétés principales qui existe à travers tout le long de l'exécution du jeu. La deuxième scène correspond au menu principal avec un menu roulant et des boutons pour sélectionner les niveaux. Enfin nous avons regroupé tous les niveaux dans une seule scène générique avec la gestion de génération des niveaux par fichiers.

Le tableau ci-dessous montre tous les systèmes associés aux scènes correspondantes :

Base	Menu	Générique
GameEngine LevelSelector	GameEngine LevelSelector	GameEngine LevelSelector Loading LightSource LightRay OpticalComponent LaunchStar StarFollowRay DragAndDrop FixedRotation

— GameEngine : Créer des ressources lumineuses et dessiner des rayons.
— LevelSelector : Gérer le bouton appuyé en utilisant PointerOver et changer de scènes.
— Loading : Charger le fichier correspondant au niveau sélectionné au menu principal et instancier des composants optiques.
— LightSource : Détecter si la situation des ressources a changé.
— LightRay : Dessiner les rayons.
— OpticalComponent : Détecter si la situation des composants a changer et calculer le score.
— LaunchStar & StarFollowRay : Créer les étoiles et faire l'animation.
— DragAndDrop & Fixed Rotation : Gérer la situation des composants optiques et faire l'animation.

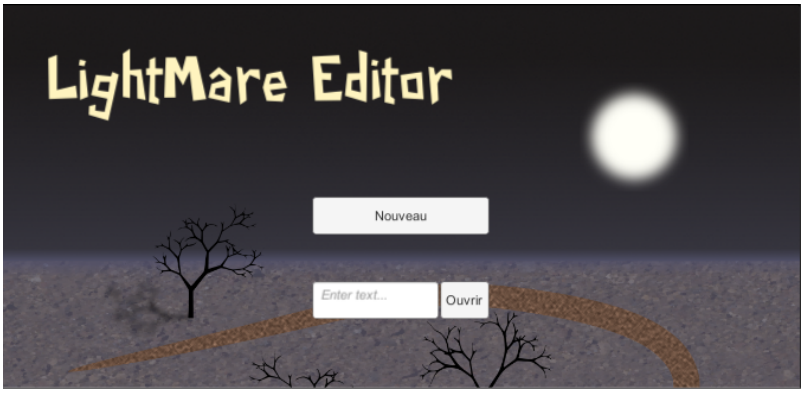
3.2 Comparaison

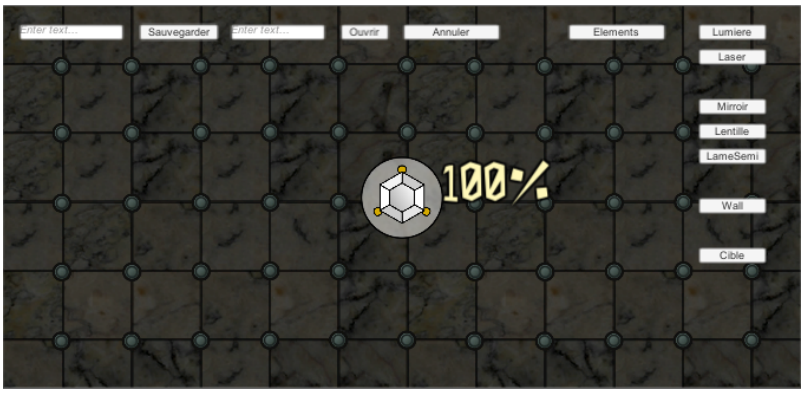
En utilisant, la librairie FYFY, nous pouvons conclure que l'architecture ECS est très efficace dans le cas où les objets ont beaucoup de propriété différentes. FYFY nous permet de créer les filtres pour générer les objets qui ne sont pas de même type. Cependant, dans notre cas, l'ECS rend le projet très complexe. Voici quelques remarques sur l'utilisant du FYFY :

— Migration d'un objet : En créant un objet, l'abonnement de l'objet à GameObjectManager est obligatoire.
— Multiplication du code : Des calculs d'un objet pourrait être utilisés dans plusieurs systèmes mais des méthodes ne sont pas autorisées.
— Complexification du code : Certaines méthodes héritent de la classe MonoBehavior qui ne sont pas utilisables dans les systèmes. Par exemple, la gestion de Drag And Drop (glisser déposer) est très simple si on utilise OnMouseDown(), OnMouseUp(), OnMouseClick(), OnMouDrag(). Cependant, avec ECS, on doit vérifier tous les objets pour savoir quel objet avec lequel on a interagit et parfois la détection n'est pas stable car les systèmes sont dynamique.
— Héritage : Pour les sous classes, il faut tester le type de l'objet pour affecter de bons calculs.
— Parallélisme : Les systèmes peuvent dépendre les uns des autres car ils sont créés en même temps. Les objets traités dans un constructeur d'un système ne pourraient être présents dans la mémoire car ils sont instanciés par un autre système. Il faut donc des traitements spéciaux dans la boucle principale.

4 Éditeur de niveaux

En s'inspirant de l'interface du jeu, nous avons créé une deuxième application qui permet à l'expert pédagogique de créer de différents. Cependant ce gadget n'est pas totalement en ECS car le but est créer des fichier texte.





5 Traçage

Nous avons mis en place une méthode de traçage simple pour le jeu, les événements essentiels sont sauvegardés dans un fichier pour le but de resimuler les actions effectuées par les joueurs. À partir de ces fichier, les experts pédagogique et ludique peuvent détecter auxquels points sont bloqués les joueurs. La méthode qui logge les événements n'a pas été implémenté dans les systèmes de FYFY pour une raison de simplicité. Les experts peuvent l'utiliser s'ils veulent étendre le jeu dans un autre aspect.