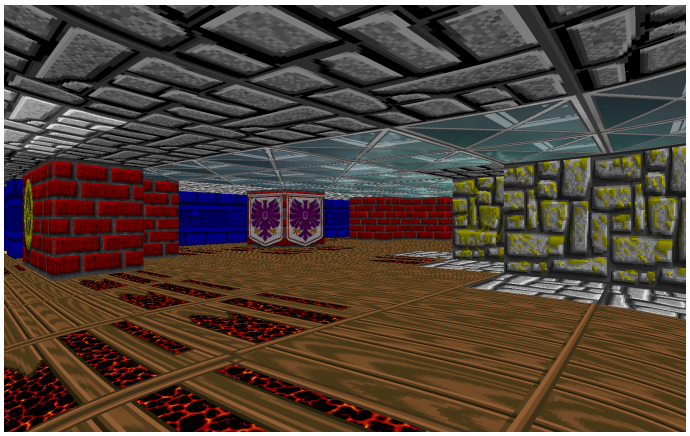
Projet - S2

Nom du projet : Héphaistos

Groupe : G.O



Yvan Fedaouche, Lenny Chiadmi-Delage, Clément Chang, Ryan Dahout.

Introduction

Vous allez entrer dans l'antre d'Héphaistos, contre toute attentes dans sa forge vous ne trouverez point de vieux dieux barbus, mais ses 4 jeunes apprentis forgerons.

Table des matières

1 Le projet.

1.1 Qui sommes nous et quel est le projet ?

Tout commença lorsque les grands anneaux furent forgés. . . Mauvaise histoire, désolé ! L'idée du projet nous vient de notre haine commune et totalement injustifiée contre Unity, quant au nom Héphaïstos, il provient du fait que nous voulons créer un outil (une sorte de mini-logiciel) permettant de forger ou plus précisément créer ses propres jeux en raycasting. Au cours des différentes soutenances nous allons créer notre propre moteur de jeu, celui-ci prendra la forme de librairies C# et d'un logiciel d'édition de cartes. Ces deux sous-parties seront détaillées plus bas dans le document. Le groupe quant à lui est composé de 4 personnes :

1.2 Présentation des membres

— Yvan Fedaouche :

Je me suis d'abord initié à la programmation avec scratch avant de créer tout au long de mes années de collège de nombreux petits jeux pour passer le temps. Je n'ai jamais réellement eu l'occasion de réaliser des projets de groupes en programmation, je suis donc assez enjouée à l'idée de réaliser ce projet avec des amis.

— Lenny Chiadmi-Delage :

J'ai sombré dans la programmation alors que je n'étais qu'un enfant. Même si j'ai continué à faire des petits projets de mon côté, comme des sites internet ou des programmes divers et variés. Je suis enthousiaste à l'idée de réaliser ce projet de groupe, car je vais pouvoir en apprendre plus sur le travail d'équipe.

— Ryan Dahout :

Pour ma part, je n'ai commencé la programmation seulement au lycée. Néanmoins, la programmation a rapidement évoqué un réel intérêt chez moi. J'ai commencé par des programmes simples et des petits sites internet. Malheureusement, je n'ai jamais eu l'occasion de faire des gros projets comme celui-ci. L'idée de ce projet m'a charmée et je suis empressé de commencer.

— Clément :

Un passionné de jeux vidéo ! Ma première console a été la "WII" accompagné de l'immanquable "Wii sport". Mon père m'obligeait à faire un peu de "Cérébrale Académie" avant de pouvoir jouer. Ralala... qu'est-ce que nos parents ne nous obligeaient pas à faire... Tous cela pour que mon frère gigotte la manette de gauche à droite pour jouer au tennis, tout en annonce de vive voix "TECHNIQUE SPECIALE" ou encore se faire battre à plate couture par ma grand-mère au bowling (aujourd'hui encore....). Par la suite j'ai obtenu la Nintendo DS ROUGE (Mes parents m'avaient associé au rouge et mon frère au bleu. Par chance, c'était des couleurs que nous apprécions, évitant toutes disputes !) A suivis la Xbox 360 puis, enfin, l'ordinateur (Rien de mieux qu'un bon et puissant ordinateur). Toutes ces anecdotes pour vous dire qu'un jour, j'ai décidé de passer derrière l'écran et de voir comment les machines, les jeux et les autres applications fonctionnent. C'est pourquoi j'ai rejoint l'EPITA, ne partant de rien j'ai pour but d'acquérir des compétences pour développer un projet, travailler en groupe et améliorer mon organisation.

2 Objet de l'étude.

Le but de ce projet est simple, tout faire pour ne pas utiliser Unity. En effet, Héphaistos est un adepte du Raycasting et c'est pourquoi il est primordial pour ses apprentis de mettre en lumière cette technique d'affichage qui n'est plus utilisée de nos jours alors qu'elle est simple d'utilisation et offre une nouvelle approche de la représentation d'un jeu. L'utilisation d'un moteur de jeu a beau au être devenu un standard dans le domaine du jeu vidéo, la création de notre propre moteur de jeu est une expérience enrichissante nous permettant d'approfondir nos compréhension du fonctionnement interne des jeux vidéos.

2.1 Qu'est-ce que le projet peut nous apporter ?

Ce projet peut nous permettre de nous confronter à des styles, des opinions et des comportements différents malgré nos profils similaires. Autant de manières de sortir de notre zone de confort et de nous apporter une expérience qui nous sera utile dans le monde du travail. Ce projet exige également une efficacité liée à une cohésion de groupe, un partage de connaissances et d'autres capacités que les tps ne peuvent nous apporter.

3 Détails du projet.

Un moteur de jeu est un ensemble de composants logiciels qui effectuent des calculs de géométrie et de physique utilisés dans les jeux vidéo. L'ensemble forme un simulateur en temps réel souple qui reproduit les caractéristiques des mondes imaginaires dans lesquels se déroulent les jeux.

Un moteur de jeu contient 5 composants principaux :

— Le programme principal du jeu,
— Un moteur de rendu,
— Un moteur audio,
— Un moteur physique,
— Et une intelligence artificielle.

Le but sera de créer un "mini" moteur de jeu en utilisant des algorithmes de raycasting pour la partie graphique de celui-ci. Le projet sera principalement divisé en deux sections : une partie logicielle et une librairie, ces deux parties du projet prendront en charge les 4 dernières composantes d'un moteur de jeu tandis que l'utilisateur n'aura qu'à réaliser le programme principal du jeu.

(Image utilisée à des fins d'illustration)

3.1 La partie librairie.

La partie librairie consistera en un ensemble d'outils facilitant la création d'un jeu. Cette section du projet sera divisée en de nombreuses sous parties qui pourront être distinctes ou inter-dépendantes.

3.1.1 IA :

Dans le domaine du jeu vidéo, l'intelligence artificielle est utilisée pour proposer une expérience de jeu de qualité, par exemple en conférant aux personnages non-joueurs un comportement s'approchant de celui de l'être humain, d'une interaction logique, ou en limitant les compétences du programme afin de donner au joueur un sentiment d'équité.

Définition : Un personnage non-joueur ou personnage non jouable (NPC) désigne tout personnage d'un jeu qui n'est pas contrôlé par un joueur.

Le but de l'IA sera de faciliter la création de NPC pour l'utilisateur, l'IA de la librairie permettra aux NPC de naviguer dans les cartes, interagir avec les joueurs de différentes manières (Exemple : NPC passif, agressif, marchand.), etc...

Pour faciliter l'implémentation d'algorithmes de pathfinding nous utiliserons un système de Mesh de navigation (Navmesh).

Définition : Un mesh de navigation est une structure de données similaire à un graph utilisée en intelligence artificielle et également en pathfinding. Cette structure de donnée permet de représenter les zones d'un environnement 3D et, couplée à certains algorithmes, permet de trouver des chemins dans ceux-ci. Cette structure de données est particulièrement adaptée à la description des larges espaces.

Exemples d'algorithmes utilisables pour le pathfinding : A*, Dijkstra.

3.1.2 Des cartes/niveaux fonctionnels :

Les cartes seront représentées par des matrices à 3 dimensions et découpées en plusieurs tronçons d'une taille prédéfinie et uniforme pour être stockées dans des fichiers spécialisés et générés dans cet objectif. Les tronçons censés être affichés au joueur seront également chargés dans une matrice à 2 dimensions pour en faciliter l'accès.

3.1.3 Un algorithme de raycasting (Moteur de rendu) :

Le raycasting est une technique de rendu 3D principalement utilisée dans de vieux jeux vidéo et qui à partir d'une simple matrice en 2D (ou 3D) permettait de réaliser en temps réel un environnement qui a l'apparence d'une 3D rudimentaire à l'image de wolfenstein, doom 64, etc...

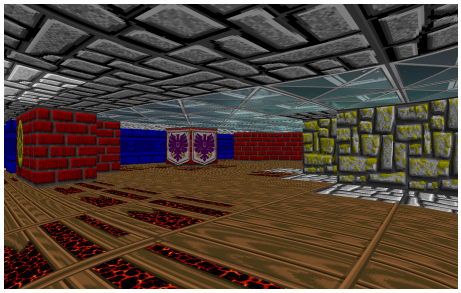
Attention à ne pas confondre le raycasting et raytracing qui sont deux techniques de rendu différentes malgré qu'elles soient confondues dans de nombreux ouvrages.

- **Raycasting :** La manière la plus simple de rendre le raycasting est de lancer un rayon par colonne de pixels, chaque rayon fait une recherche dans l'espace pour trouver l'intersection avec le mur le plus proche.

- **Raytracing :** Cette technique de rendu consiste, pour chaque pixel de l'image à générer, à lancer un rayon depuis le point de vue dans l'espace. Le premier point d'impact du rayon sur un objet définit l'objet concerné par le pixel correspondant. Des rayons sont ensuite lancés depuis le point d'impact en direction de chaque source de lumière pour déterminer sa luminosité. Cette luminosité, combinée avec les propriétés de la surface de l'objet, ainsi que d'autres informations éventuelles, déterminent la couleur finale du pixel.

Fonctionnement : On peut représenter la map en 2D (ou 3D), avec une matrice à deux dimensions dont Chaque case indique la présence d'un mur avec un bit. Quant au joueur il est représenté par un vecteur dont l'origine est la position du joueur et la direction est celle du regard. Avec ce rendu, on colorie une colonne de pixels à la fois sur l'écran, il faut pour cela connaître la hauteur du mur vue depuis l'écran. Pour la calculer la hauteur du mur depuis le point de vue du joueur, on peut utiliser un rayon passant par celui-ci et envoyé via un angle défini en fonction de la position de la colonne sur l'écran, du nombre de colonnes de l'écran, de la direction du regard et la distance de l'écran par rapport au joueur. Cependant il reste un dernier petit problème, l'effet fish-eye. En fait, les rayons du bord du regard parcourent une distance plus grande que les rayons situés au centre du regard. Heureusement cet effet est facilement corrigeable grâce aux propriétés d'un triangle rectangle, permettant d'obtenir un rendu propre.

Exemples de librairies utilisables : OpenGL, System.Drawing, SkiaSharp, etc...



(Image utilisée à des fins d'illustration)

3.1.4 Un moteur audio :

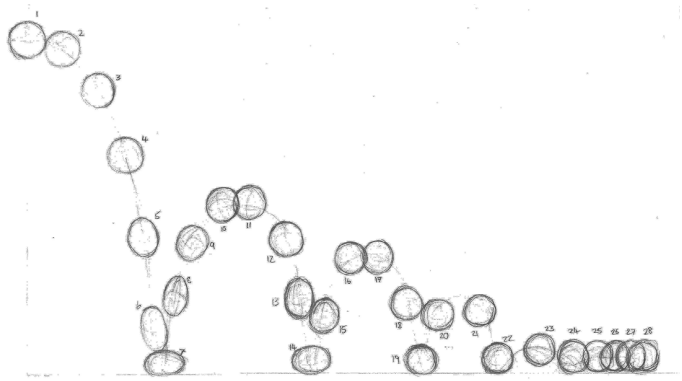
Le rôle d'un moteur de son est de rendre le plus fidèlement possible, par le biais du système audio utilisé, l'ambiance d'un jeu tel que le joueur l'entendrait s'il était lui-même présent dans l'univers du jeu.

Exemple de librairies utilisables : NAudio, CSCore, OpenTK, etc...

3.1.5 Un moteur physique :

Un moteur physique est, en informatique, une bibliothèque logicielle indépendante appliquée à la résolution de problèmes de la mécanique classique. Les résolutions typiques sont les collisions, la chute des corps, les forces, la cinétique, etc...

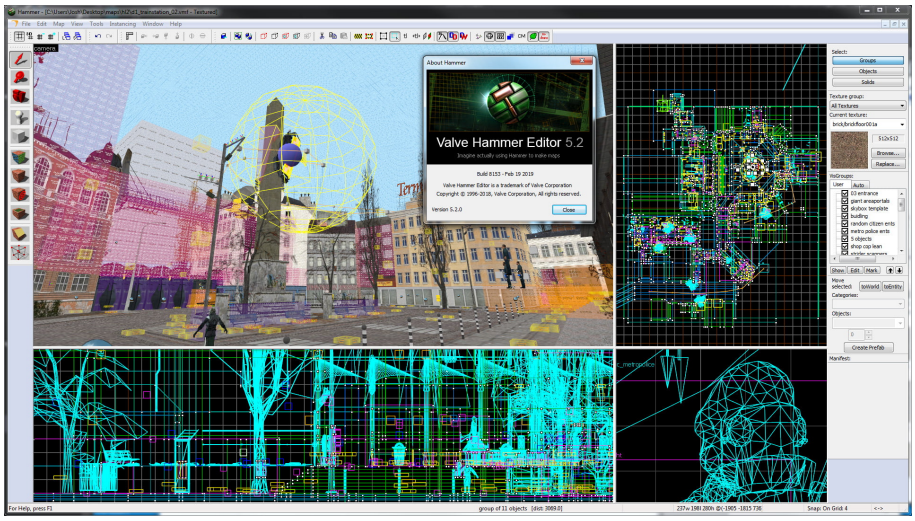
Au cours du projet nous tenterons d'implémenter un moteur physique (simplifié) afin de gérer les déplacements du joueur, objets et npc créés à l'aide des outils cités au-dessus.



(Image utilisée à des fins d'illustration)

3.2 La partie logiciel

La partie logicielle consistera en un éditeur de niveau le plus complet possible, permettant de créer ses propres environnements en raycasting. A l'image du logiciel d'édition de map du moteur source Hammer, l'édition de map sera faite via 3 vues de tranche (en 2D) et une vue permettant de prendre le point de vue d'un joueur en raycasting positionnés côte à côte.



(Image utilisée à des fins d'illustration.)

Pour réaliser cette partie du projet, nous aurons besoin de quelques prérequis tels que :

— **Des cartes/niveaux fonctionnels :** Voir le texte sur les cartes plus haut.
— **Des interfaces :** Une interface est une limite commune à deux systèmes, deux ensembles, deux appareils. Celles-ci seront composées de plusieurs éléments possédant leur propre classe tel que : Une classe bouton, Image, Entrée de texte, etc. . . .
Exemples de librairies utilisables : InputManager, WinputManager, etc. . .
— **Un algorithme de raycasting :** Voir le texte sur les cartes plus haut.

4 Les limites techniques du projet.

Cette section du cahier des charges sera centrée sur le moteur de rendu en raycasting car celui-ci représente la partie centrale du projet.

4.1 Raycasting :

Le raycasting était rapide et abordable lors de calculs en temps réel pour les pc des années 90, cette vitesse était due aux résolutions de l'époque qui étaient assez faibles. Malheureusement celui-ci est très lent sur des écrans à haute résolution comme ceux d'aujourd'hui. Notre moteur de rendu sera donc limité à de faibles résolutions tels que : 640x480, 800x600, etc...

Les premiers moteurs du genre possédaient de grosses limitations techniques, heureusement avec la sortie du premier doom qui permit par exemple de faire des variations de hauteur, des murs non perpendiculaires ou des effets de lumière ces limitations furent largement repoussées. Cependant de grosses limitations restent de mise :

- Le sol et le plafond doivent être plats,
- Les textures des murs doivent toutes avoir une résolution uniforme,
- La caméra ne peut être orientée verticalement,
- La hauteur des murs doit être un multiple de leur taille par défaut,
- La quantité d'images par seconde dépendra fortement du nombre d'objets affichés à l'écran, cependant cette valeur sera généralement comprise entre 30 et 60 fps.
- La distance d'affichage de la caméra sera également limitée pour des raisons d'optimisation,
- Pour finir, à l'image de doom, les entités seront représentées par une image constamment orientée vers le joueur.

(Image utilisée à des fins d'illustration)

4.2 Moteur physique :

Comme précisé plus haut, il s'agira d'un moteur physique simplifié, cela signifie que celui-ci ne prendra pas en charge de nombreux phénomènes tels que :

- Les Frictions ou la résistance de l'air,
- La gestion des fluides,
- La vitesse angulaire.

4.3 Les cartes/niveaux :

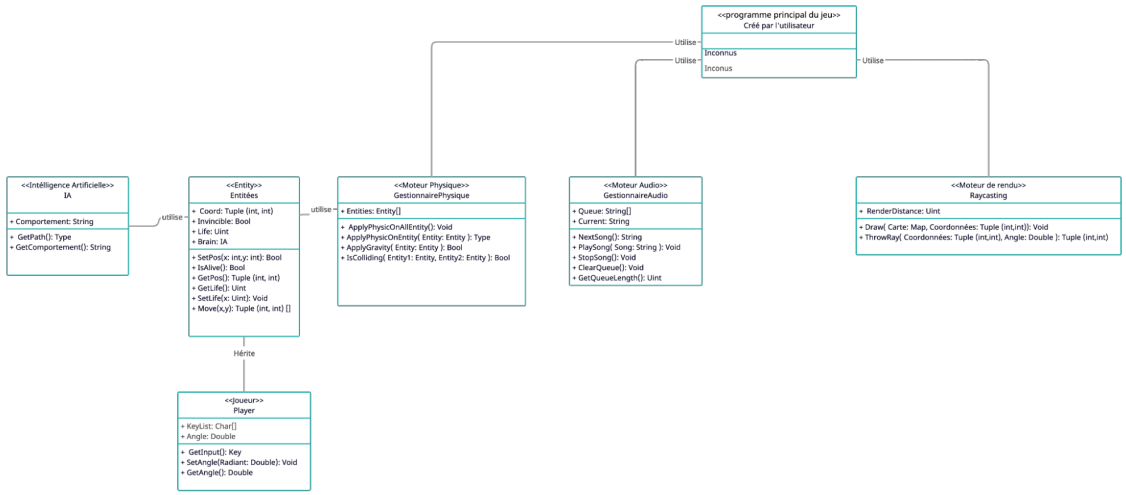
Pour des raisons intrinsèquement liées à la distance d'affichage et au principe même du raycasting, les cartes / niveaux devront être contenues dans un bloc creux entouré automatiquement lors de la création de nouvelles cartes, à l'instar des cartes du moteur Source.

(Image utilisée à des fins d'illustration, ça c'est une carte bien entourée)

Résumé

Un moteur de jeu contient 5 composants, les parties concernée par Héphaïstos seront :

— Un moteur de rendu,
— Un moteur audio,
— Un moteur physique,
— Et une intelligence artificielle.



(Image utilisée à des fins d'illustration, la structure de donnée pourra être changée pendant l'avancement du projet)

De plus le projet sera séparé en deux parties principales :

— La librairie C#,
— Le logiciel.

Ces différentes parties posséderont des composants en commun et seront indépendantes l'une de l'autre.

5 Répartition des taches.

Tâches/Prénoms	**Ryan**	**Clément**	**Lenny**	**Yvan**
Moteur de rendu				
Algorithme de Raycasting	X	X	Supléant	Résponsable
Moteur Physique				
Moteur Physique	X	X	Résponsable	Supléant
Moteur Audio				
Moteur Audio	Supléant	Résponsable	X	X
IA				
IA des NPC	Résponsable	Supléant	X	X
Autre				
Système de cartes / niveaux	X	X	Supléant	Résponsable
Interfaces	Supléant	Résponsable	X	X
Logiciel d'édition de cartes	X	Résponsable	X	Supléant

6 Avancement des taches.

Tâches/Soutenances	**1er**	**2éme**	**3éme**
Moteur de rendu			
Algorithme de Raycasting	25%	50%	100%
Moteur Physique			
Moteur Physique	20%	60%	100%
Moteur Audio			
Moteur Audio	50%	75%	100%
IA			
IA des NPC	25%	50%	100%
Autre			
Système de cartes / niveaux	50%	100%	100%
Interfaces	25%	50%	100%
Logiciel d'édition de cartes	0%	50%	100%

7 Monétisation

Notre projet n'utilisera que des librairie C# "standards", tandis que les assets utilisés seront libres de droits. Nous n'aurons donc aucuns frais en dehors du matériel informatique utilisé. De plus, si nous souhaitons publier notre projet celui-ci ne sera pas directement monétisable par la vente de licences, la monétisation pourra être faite via des paliers de prix à payer en fonction du nombre de ventes des jeux créés via notre moteur à l'image d'Unreal Engine ou d'autres moteurs de jeux.

8 Conclusion

En conclusion, le projet consiste en la création d'un moteur de jeu en raycasting et C#, accompagné d'une petite interface pour aider à la création de nouveaux jeux. Les différentes fonctionnalités ont été implémentées en fonction des capacités de chaque membre du groupe et nous comptons rendre un projet le plus propre possible malgré l'ambition de celui-ci.