Rapport de Soutenance 2
The Maze VR

S-TEAM

3 juin 2019

Martin Camincher
Geoffroy du Mesnil du Buisson
Lucien Leseigle
Hugo Belot-Deloro

Table des matières

1 Présentation du projet

1.1 Le jeu

The Maze VR est un projet de jeu d'horreur en VR. Celui-ci repose sur le principe de l'exploration de labyrinthe.

1.2 Modes de jeu

1.2.1 Mode histoire

Le joueur apparaîtra sur une grande carte ouverte, depuis laquelle il sera possible d'accéder à plusieurs labyrinthes de plus petite taille. Il devra donc explorer ces labyrinthes afin d'obtenir divers objets lui permettant de progresser sur la carte principale. Plusieurs monstres gêneront sa progression, certains pouvant l'attaquer dans les labyrinthes, d'autres sur la carte extérieure.

1.2.2 Mode multijoueur

Dans ce mode de jeu, deux joueurs s'affronteront. L'un explorera le labyrinthe et tentera de s'en échapper, tandis que l'autre aura un plan du labyrinthe et pourra interagir avec son adversaire en faisant apparaître des monstres ou en activant des pièges, voire même en modifiant la structure du labyrinthe en cours de partie pour le gêner.

1.2.3 Éditeur de Labyrinthe

Il sera possible de créer, exporter et importer des labyrinthes entièrement personnalisés grâce à un éditeur de labyrinthe. Ils seront ensuite jouables en solo ou en multijoueur.

2 Planning de réalisation

2.1 Avancement du projet

Tâche	Gameplay	Interface	Multi	IA	Graphismes	Audio	Web
Prévisionnel	75%	80%	90%	60%	80%	60%	40%
Effectif	75%	80%	90%	60%	80%	70%	50%

2.2 Taches communes

- Réflexions sur le gameplay
- Entraide en cas de problèmes

2.3 Taches individuelles

Hugo :
-Ajout de différentes mécaniques de jeu au joueur (inventaire, équipement, points de vie)
-Affichage tête haute
-Gestion de l'éclairage
-Gestion des scènes

Martin :
-Création de différentes structures (porte, lampes)
-Modification du détecteur de structure s'adaptant à l'axe vertical
-Création d'un éditeur de labyrinthe

Geoffroy :
-Construction de six nouveaux labyrinthes
-Assemblage de la pyramide avec tous les labyrinthes
-Deboguage de la carte
-Création du site web
-Gestion du projet

Lucien :
- composition et intégration de la musique dans le jeu
- gestion des effets audio
- implémentation de l'IA des ennemis

3 Avancement individuel

3.1 Hugo

3.1.1 Objets et inventaire

Le principal élément auquel je me suis intéréssé depuis la dernière soutenance est l'inventaire du joueur et les objets ramassables qui y

sont associés. J'ai d'abord créé le ScriptableObject Item représentant ces objets. Il contient simplement un Sprite qui permet d'afficher l'objet dans un inventaire. J'ai également créé un prefab représentant un objet pouvant être ramassé, qui contient donc un Item et un script ajoutant cet objet à l'inventaire du joueur et détruisant le GameObject lorsque le joueur le touche.
J'ai ensuite créé l'inventaire, qui fonctionnait alors en utilisant une liste d'Item et une liste de UI.Image pour afficher les Sprites des Item (leur représentation graphique dans l'inventaire).
J'ai par la suite voulu ajouter une structure similaire à l'inventaire pour stocker des outils comme un compas ou une lampe, par opposition à l'inventaire visant plus à stocker des clés et des objets génériques par exemple. Pour cela j'ai dû changer une grande partie du système d'inventaire pour le rendre plus flexible.
Premièrement, j'ai créé une classe ItemSlot qui contient un Item et une Image où afficher son Sprite. Je l'ai associée au prefab ItemSlot que j'avais utilisé dans l'inventaire pour afficher les Item. L'inventaire est maintenant représenté par une liste de ItemSlot. De plus, j'ai rajouté à Item une énumération représentant le type de l'objet (non-équipable, compas, lampe, etc) et j'ai créé un Item représentant l'Item vide pour remplacer la valeur nulle lors de la manipulation d'Item vides.
J'ai donc pu ensuite créer le système d'équipement, mais celui-ci n'est pas encore testé. Son fonctionnement diffère de celui de l'inventaire en ce que un Item peut être placé dans n'importe quel ItemSlot de l'inventaire, mais ne peut être placé que dans l'ItemSlot correspondant à son type dans l'équipement. Pour cela, j'utilise un dictionnaire afin de mapper le type d'un objet à ajouter à l'inventaire à l'ItemSlot correspondant.
Résultat final :
Objet dans le monde : (1)
Objet dans l'inventaire : (2)

3.1.2 Eclairage

Je me suis brièvement intéressé à l'éclairage afin de rajouter un "soleil" à la map de Geoffroy, et une sorte de lampe frontale au joueur, qui sera éventuellement activée lorsque celui-ci équipera l'objet associé. J'ai aussi ajouté des commandes ('a' et 'w') pour lever et bais-

ser la tête (caméra) du joueur, dans une certaine limite. Cela déplace également la lampe.

3.1.3 Vie

J'ai ensuite ajouté un système basique de points de vie au joueur. Les points de vie sont représentés par un int, et si le joueur passe suffisamment de temps sans subir de dégâts, ses points de vie remonteront jusqu'à leur maximum. La possibilité de mourir n'est pas encore implémentée. Pour ce qui est de l'affichage des points de vie, je voulais éviter quelque chose de trop numérique qui brise l'immersion, et j'ai donc simplement rajouté un overlay (3) rouge sur l'écran du joueur, qui disparaît au fur et à mesure que ses points de vie remontent.

3.1.4 Loader de scènes

La zone de jeu étant assez grande, j'ai voulu créer les bases d'un système pouvant nous éviter de garder les parties actuellement inutilisées de la carte. Par exemple, un labyrinthe souterrain ne devrait pas être chargé tant que le joueur ne s'en approche pas, et une fois le joueur à l'intérieur, le reste de la zone de jeu n'a pas besoin d'être activé. J'ai donc créé un prefab invisible qui détecte la présence du joueur et charge et décharge des scènes en conséquence. Celui-ci n'est pas encore testé, et il faudra sans doute un certain nombre de modifications pour le faire fonctionner correctement mais à terme, il pourrait aussi nous permettre de remplacer les zones lointaines par des versions moins détaillées afin de limiter la charge graphique du jeu.

3.1.5 Objectifs

Je voulais implémenter un menu pause au jeu, ce que je n'ai pas encore fait, et c'est donc sans doute l'une des premières choses dont je vais m'occuper. J'en profiterais pour ajouter un système plus avancé pour gérer les entrées du joueur, afin de pouvoir plus facilement changer les keybindings. Il me faudra ensuite tester le système d'équipement et implémenter le compas. Geoffroy m'a également demandé d'ajouter des échelles au jeu. Une fois que j'aurais le temps, je voudrais implémenter véritablement le système de chargement/déchargement

de scènes mentionné plus haut.

3.2 Martin : Structures et éditeur

3.2.1 Structures : Portes et Lampes

L'un de mes objectifs après la dernière soutenance était la création de différentes structures. En effet, nous pouvions déjà créer des labyrinthes contenant des murs et un sol, mais cela était assez limité car ces structures constituent le labyrinthe entier, et n'apportaient aucune diversité dans celui-ci. J'ai donc créé des structures permettant plus d'originalité dans le labyrinthe.

La première structure que j'ai créé est une porte. Celle-ci est composée de différents éléments : Les bords de la porte, la porte elle-même, qui s'ouvre et se ferme, et un trigger invisible. Lorsque le joueur est proche de la porte et regarde dans sa direction, le détecteur de structure qui avait été créé avant la première soutenance (et qui a modifié pour mieux suivre la vue du joueur, maintenant que celui-ci peut bouger la caméra selon l'axe vertical) repère le trigger de la porte. Lorsque la porte est ainsi observée, il est possible de l'ouvrir ou de la fermer grâce à la touche 'E'. Afin que la porte ne s'ouvre et se ferme à chaque frame tant que la touche 'E' est maintenue, ce qui rend l'ouverture de la porte plutôt aléatoire, une coroutine à été implémentée. Ainsi, à chaque fois que la porte est activée, celle-ci passe dans un état de "sommeil" pendant 0.25 secondes, avant de se réveiller et ainsi pouvoir être à nouveau activée.

La deuxième structure que j'ai créé est plus simple, il s'agit d'une simple lampe. Celle-ci est constituée d'un modèle de capsule, provisoire, permettant de repérer l'emplacement de la lampe, et d'une area light émettant une lumière orangée censée rappeler la lumière d'une flamme. Avec l'ajout d'un plafond dans le labyrinthe et la diminution de la luminosité ambiante, ces lampes, avec la lampe frontale du joueur, sont les seuls sources de lumière dans les labyrinthes.

3.2.2 L'éditeur de labyrinthe

Un des modes de jeux que nous avons prévus est l'éditeur de labyrinthe, permettant au joueur de créer ses propres labyrinthe.

Dans un premier temps, le joueur dois choisir les dimensions du labyrinthe qu'il souhaite créer. Il entre donc ces dimensions dans deux zones de textes pour la hauteur et la largeur, et valide avec un bouton. un Try-Catch empêche tout problème qui serait causé par l'insertion de caractères invalides dans les zones de texte. Une fois que les dimensions sont insérées et validées, le joueur prend le contrôle de la caméra affichant le labyrinthe en construction en vue du dessus, sans perspective afin de mieux repérer le plan du labyrinthe. Le labyrinthe n'est alors qu'un sol selon les dimensions choisies par le joueur. C'est alors à lui de placer les murs, portes, et autres éléments constituants son labyrinthe. La caméra peut être déplacée dans toutes les directions, et peut zoomer et dézoomer, afin de pouvoir voir le labyrinthe dans son intégralité, ou bien de pouvoir l'observer de plus près. A l'heure actuelle, la suite du fonctionnement de l'éditeur n'est pas encore opérationnelle. Cet outil étant très complexe, sa création est longue et présente de nombreuses difficultés, mais voici comment celui-ci fonctionnera :
Afin de pouvoir placer les structures dans le labyrinthe, le joueur utilisera le pavé numérique. les touches 1, 2, 3, 4, 6, 7, 8 et 9 permettront de déplacer les structures, la touche 5 confirmera son emplacement et les touches + et - permettront de changer de structure à placer. L'une des plus grandes difficultés que j'ai rencontrées lors du développement de ces fonctionnalités est la détection des touches sur lesquelles le joueur appuie. Je voulais à l'origine attendre que le joueur appuie sur une touche, puis récupérer cette touche et agir en fonction, mais les différentes solutions que j'essayais ne fonctionnaient pas et faisaient freeze le jeu. Après 5 tentatives différentes ayant toutes conduites à un échec, j'ai changé ma méthode et ait utilisé la fonction Update pour vérifier à chaque frame si l'une des touches ayant un fonctionnement dans l'éditeur est pressée. Il ne reste à présent plus qu'a finir de programmer le déplacement des différentes structures.

3.2.3 Objectifs

Mes objectifs sont à présent de terminer l'éditeur de labyrinthe et si possible de l'améliorer. En effet, j'utilise dans celui-ci une version modifiée du générateur de labyrinthe afin de créer la structure de base, utilisant une matrice d'int à la place d'une image. Ce format permet-

tra une plus grande facilité dans le stockage et le transfert de données pour les labyrinthes, pouvant être compressé en un simple int. Mais j'envisage de modifier ceci afin d'utiliser à la place une matrice de listes d'int, ce qui permettrait de générer plusieurs structures différentes aux mêmes coordonnées.
Il me faudra également créer un moyen pour le joueur de supprimer un structure déjà placée dans l'éditeur.
La réalisation de cet éditeur est très importante, car celui-ci, dans le mode multijoueur, permettra de créer plus facilement les contrôles du deuxième joueur. En effet, celui-ci utilisera les mêmes mouvements de caméra, et placera les ennemis et pièges de la même manière.

3.3 Geoffroy

3.3.1 Construction de nouveau labyrinthes

Maintenant que j'ai pu me familiariser un maximum avec Blender j'ai donc pu avancer sur la carte du mode solo. Je me suis servi de la même technique que j'avais utilisé pour faire mon premier labyrinthe à savoir de faire comme le générateur de labyrinthe mais à la main en prenant une image de labyrinthe sur internet et découpant un cube pour avoir le labyrinthe. J'ai commencé par faire le premier sous sol de la pyramide qui est un labyrinthe de complexité moyennement forte, j'ai pu le faire essayer à l'équipe et le temps moyen pour le résoudre est entre cinq et dix minutes. J'ai fait ensuite le deuxième et le troisième niveau de la pyramide en créant deux labyrinthes distinct d'une difficulté moyenne. Le joueur à un nombre limité de chemin possible, il ne peut donc que réussir ces labyrinthes. J'ai par la suite fait le dernier étage du labyrinthe soit le deuxième sous sol. Parmi les deux labyrinthes souterrains et les cinq en surface, celui-ci est de loin le plus dur : il est plus grand, plus complexe et surtout beaucoup plus étroit au niveau des passage. Cela donnera au joueur une sensation d'oppression constante. Ce dernier labyrinthe contient une infinité de possibilité mais reste néanmoins faisable. Néanmoins nous comptons donner une boussole au joueur qu'il devra bien entendu trouver afin qu'il puisse à peu près se repérer dans le dernier labyrinthe, car j'admets avoir moi même eu du mal à le finir en aillant le plan à coté de moi, alors que je l'ai créé moi-même.

3.3.2 création de Pont

J'ai créé les deux ponts qui serviront pour transiter de zone en zone.

3.3.3 Réunion des objets

Pour des raisons pratique, j'ai conçu chacun des labyrinthes de telle sorte que chacun soit un objet indépendant, afin de pouvoir les manipuler plus librement, de pouvoir les modifier à ma guise et que je puisse changer leurs taille et leur orientation sans que cela n'affecte la carte dans sa généralité. Une fois que chacun des labyrinthe était fini individuellement, il fallu les fusionner à la carte principale. Pour cela je suis passé par un boolean de Blender qui m'a permis de fusionner deux objets entre eux. Mais j'ai fait face à un problème, car quand on fusionne deux objet on ne peut plus les séparer. Il a fallu que je positionne moi-même à la main chacun des labyrinthes les uns après les autres de telle sorte que les labyrinthes se chevauchent parfaitement et que le joueur puisse passer d'un niveau à l'autre de manière fluide, agréable et implicite. Pour ce faire j'ai principalement utilisé le mode wireframe qui permet de voir uniquement les arrêtes et les points de tous les objets. Cela pourrait être comparé au fait de rendre les murs invisibles, c'était donc plus simple pour l'alignement des labyrinthes.

Une fois que tous les labyrinthes étaient superposés, j'ai du les relier entre eux arrête par arrête afin que le joueur ne puisse pas tomber et qu'il n'y ait pas de trous. Cette partie était assez compliquée mais plutôt rapide, en effet Blender n'est pas fait pour la manipulation de grands objets à coté de petits objets donc pour cela, j'ai principalement utilisé le mode exploration de Blender qui permet de simuler une vision à la première personne avec des contrôles pré-enregistrés. Cela m'a permis de, à la manière d'un petit ouvrier, rentrer dans le labyrinthe et relier les points en étant précis.

Cette partie fut plus longue que prévu car les labyrinthes n'étaient pas tous de la même taille, donc le couloir ne faisait parfois pas la même largeur d'un étage à l'autre. J'ai donc dû combler moi même les trous.

3.3.4 Deboguage de la carte

Après avoir passé toutes les étapes précédemment cités, nous avons exporté la carte Blender sur Unity et nous avons pu pu constater bon nombres de problèmes. En effet, nous avons pu avoir l'occasion de faire face au "backface culling problemes". Pour résumer ce problème je dirais que les faces ont un sens sur unity qui n'est que partiellement visible. En effet, dans un sens la face est opaque et solide alors que de l'autre coté elle est transparente et "vide" comme une sorte de vitre sans tain par laquelle on pourrait passer à travers d'un coté. Notre carte entière se transformait donc en un immense champ de trous et pour augmenter la difficulté ce problème ne pouvais pas être résolu directement sur Unity. Il a fallu réimporter la carte sur Blender et retourner chacune des faces les unes après les autres, soit une petite matinée de travail.

Une fois ce problème corrigé, il a fallu faire une deuxième face pour chacun des plafond car sinon si le joueur était dans un labyrinthe sous un autre labyrinthe, il aurait vu tout le niveau du dessus comme si il avait un wallhack. Pour cela il a fallu que je refasse un plafond a chaque étage, alors on pourrait penser qu'il ne suffirait que de faire 7 grands rectangles qui partent de chaque coin des labyrinthes. Mais malheureusement nous avions des passages entre chacun des étages afin que le joueur puisse monter au niveau supérieur ou inversement descendre d'un niveau. Par conséquent, je devais faire quatre toits différents pour chacun des 7 labyrinthe, tout cela en fonction des communications de chaque labyrinthes. Or les niveaux étant déjà reliés entre eux, il était extrêmement difficile de parvenir aux points communiquants entre les niveaux. J'en ai du coup profité pour ajouter un effet d'optique de telle sorte que nous puissions voir l'intérieur des labyrinthes en étant à l'extérieur de la pyramide et au dessus, et que quand nous soyons à l'intérieur nous verrions bien un plafond.

Une fois tout ces problèmes réglés, nous avons eu un nouveau problème. En effet, j'avais mal prévu les pentes, elles étaient trop raides pour le joueur à tel point qu'il ne pouvais pas les monter sans glitcher à l'intérieur des murs. J'ai donc adouci l'ensemble des pentes auxquelles le joueur pouvait avoir accès.

3.3.5 Création du site web

Pour le site Web, je suis passé par le site Wix. Le but de cette démarche était surtout pour nous donner une idée précise de ce que nous voulions faire avant de nous lancer à l'aveugle dans du code HTML sans être tous parfaitement d'accord sur l'apparence de notre site web. Pour l'instant, j'ai acheté notre nom de domaine et nous avons réussi à trouver un serveur pour nous héberger. Le site sur Wix n'est donc pas complètement définitif. En effet, comme nous l'avions prévu dans le cahier des charges, le site web de notre jeu n'est pas une de nos priorités. Si nous n'avons pas assez de temps, nous utiliserons le Wix tel qu'il est maintenant en le modifiant sûrement un peu. Dans le cas contraire nous le ferons en HTML.

3.3.6 Gestion de Projet

En tant que Responsable, je me suis pleinement impliqué dans ce projet afin de contribuer à la réussite de celui-ci. Nous avons créé un groupe de discussion pour faciliter les échanges entre nous. J'ai eu l'idée de monter une page Facebook ainsi qu'un compte Instagram dans l'optique de mettre en avant notre projet, mais c'est surtout afin d'optimiser notre force de travail que j'interviens. En effet j'ai mis à disposition mon appartement tous les jours pendant toute la semaine avant la soutenance afin que nous puissions travailler tous ensemble sur ce projet.

3.4 Lucien

3.4.1 composition de la musique

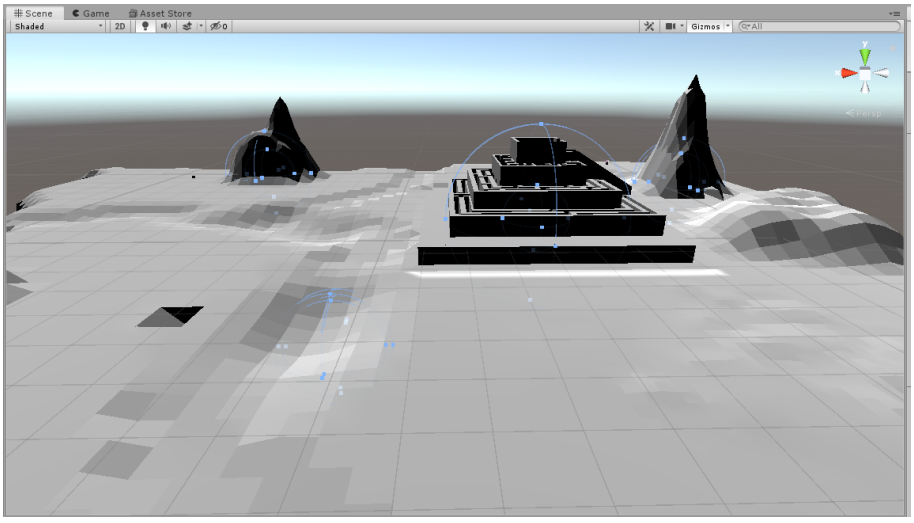
La bande son du jeu dure à présent 12mins environ. Elle est composée de manière à pouvoir être mise en boucle sans coupure ; la fin de la musique et le début sont similaires et la transition n'est pas perceptible. Elle contient des passages effrayants et quelques parties plus épiques. Le jeu pourrait être publié avec la musique telle qu'elle est maintenant, cependant je compte faire 3 autres musiques pour changer l'ambiance musicale en fonction de la zone dans laquelle se trouve le joueur.

3.4.2 Intégration de la musique

La musique du menu tourne en boucle sans transition perceptible lors du retour au début. La musique du jeu commence par une introduction jouée une seule fois, puis les 11mins de bande son tournent en boucle. Elle est envoyé dans un mixer qui permet de compresser la musique pour que le volume sonore reste similaire malgré l'ajout d'effets, tout en conservant plus de dynamique sonore qu'avec une normalisation. Le script de gestion des pistes audio en jeu ("Audio Manager.cs") est attaché au joueur et a comme paramètres les objets vides contenant les diverses sources audio, rattachées aux fichiers mp3 (320kbs soit la plus haute qualité possible, le format wav étant trop lourd pour justifier l'apport en qualité audio).

3.4.3 Reverb zones

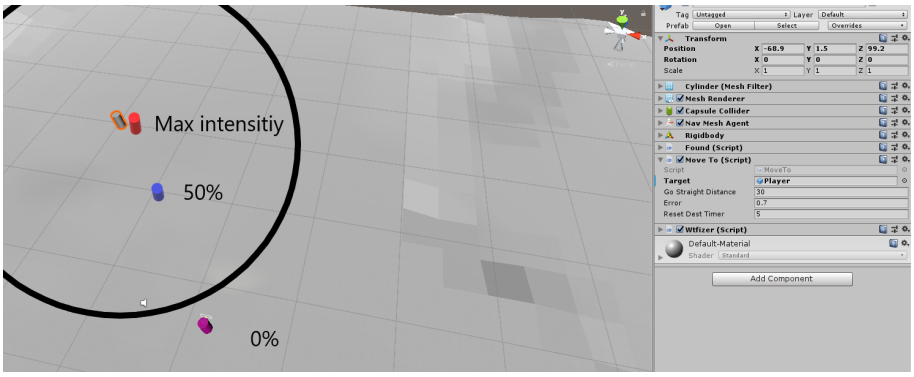
Il y a quelques couloirs dans le jeu, j'ai donc ajouté des Reverb Zones dans ceux-ci. Cela donne l'impression que la musique est jouée depuis l'intérieur du tunnel grâce aux réverbération adaptées a ce type d'environnement. plus le joueur s'approche de la "Full Zone", et plus l'effet est intense. Ces zones de réverbération permettent un réalisme et donc une immersion accrue. Ci-dessous les différents zones dans lesquelles l'effet est activé :



Il faudra rajouter des zones et réduire leur taille / changer leur réglages pour rendre cet effet encore plus réaliste.

3.4.4 Effets audio liés au monstre

J'ai également ajouté une fonctionnalité qui distord le son quand le monstre s'approche du joueur. Cela crée une sensation de pression qui augmente dans la musique quand le monstre est sur le point de tuer le joueur. Ce script accepte 3 paramètres :
- Threshold : la distance à partir de laquelle les effets s'activent
- Maxchorus : l'intensité maximale de l'effet chorus
- Maxdisto : l'intensité maximale de l'effet distortion
Le son est "dry" quand la distance entre le joueur et le monstre est supérieure au threeshold. Quand le montre et le joueur sont proche, l'intensité des effets est proportionnelle a cette même distance, jusqu'à atteindre les valeurs maximales quand les deux objets se touchent.



3.4.5 Déplacement du monstre

J'ai d'abord réalisé une autre implémentation du réseau de neurones, cette fois un réseau FeedForward classique avec un algorithme génétique. Cependant l'apprentissage (ou le réseau lui même) dysfonctionnement et les performance atteintes après l'entraînement du réseau laissent à désirer et j'ai abandonné cette solution. J'ai donc géré les déplacements de l'IA avec la fonction NavMesh intégrée dans Unity. Les réglages sont les suivants :

- Go Straight Distance : distance à partir de laquelle l'IA suit le joueur sans faire d'erreurs.
- Error : intensité des erreurs commises par l'IA
- Reset Dest Timer : durée entre les changements de destination de l'IA.
Le monstre choisit une destination basée sur la position du joueur, avec une erreur plus conséquente plus il est loin du joueur. Cela a pour effet que les déplacements du monstre semblent assez aléatoire en début de partie, quand l'IA est loin du joueur. L'IA se rapproche du joueur plus ou moins rapidement en fonction du taux d'erreur et du délai entre chaque changement de trajectoire. Quand la distance qui la sépare du monstre est inférieure au threeshold, l'IA suit le joueur sans se tromper jusqu'à l'attraper ou bien que le joueur la sème (ce qui est, soit dit en passant, impossible si la vitesse de l'IA est supérieure ou égale à celle du joueur).
Les différents paramètres doivent encore être réglés en fonction de la difficulté souhaitée. Après avoir finit d'implémenter des fonctionnalités plus importantes, je referais si le temps le permet l'IA avec les outils ML-agents de Unity, par exemple en remplacement l'algorithme génétique par du (Deep ?) Q-Learning qui est beaucoup plus adapté.

3.4.6 Algorithme génétique

Si l'algorithme génétique ne sera pas utilisé pour entraîner l'IA, je compte l'utiliser pour optimiser les différents paramètres du jeu. Le projet est le suivant :
- A la sortie du jeu, les paramètres seront réglés par nos soins de manière à ce que le gameplay nous convienne.
- Chaque paramètre du jeu pourra être muté de manière aléatoire en début de partie, avec une fréquence et une intensité très faible (par exemple une diminution de 1% de la vitesse de l'IA...)
- En fin de partie, le joueur note la qualité de sa partie (par exemple avec des étoiles)
- Les réglages des parties les mieux notées sont conservées.
- Quand un joueur lance une partie, les paramètres sont donc une légère variante d'une des parties sélectionnées par l'algorithme.
- Si il y a assez de joueurs, les paramètres évolueront constamment en fonction de ce que préfère la communauté. Les problèmes d'équi-

librages seront gérés par cet algorithme sans aucune intervention requise de notre part.
Cet algorithme requiert un serveur pour stocker les paramètres des différents parties conservées dans la population. Cependant il sera quand même possible de joueur offline, en utilisant les derniers meilleurs paramètres par exemple.

3.4.7 Bruitages

Je n'ai pas encore commencé la réalisation des bruitages puisque nous n'avons pas encore décidé du design des différents élément du jeu, et qu'il me semble nécessaire de faire cette tache avant pour que les bruitages soient cohérents avec l'univers du jeu.

3.4.8 Multijoueur

Le multijoueur en est au même stade qu'à la soutenance précédente, c'est à dire une confection client serveur fonctionnelle. Nous n'avons pas encore implémenté les événements en solo, il n'est donc pas possible de les implémenter dans le multijoueur. Il manque également le système de lobby (découverte d'hôtes), qui fonctionnera à priori en réseau local.

4 Conclusion

En conclusion, nous avons rajouté à notre jeu des mécaniques plus complexes avec l'inventaire, la vie, les structures, l'éditeur et les IA, et des graphismes avec le modèle de la zone de jeu et la musique. Nos prochains objectifs sont de peaufiner le jeu en ajoutant de la texture et des modèles, en créant la progression dans le jeu et en finissant d'implémenter les monstres et l'éditeur de carte.
Lien du site

5 Annexes

5.1 Hugo

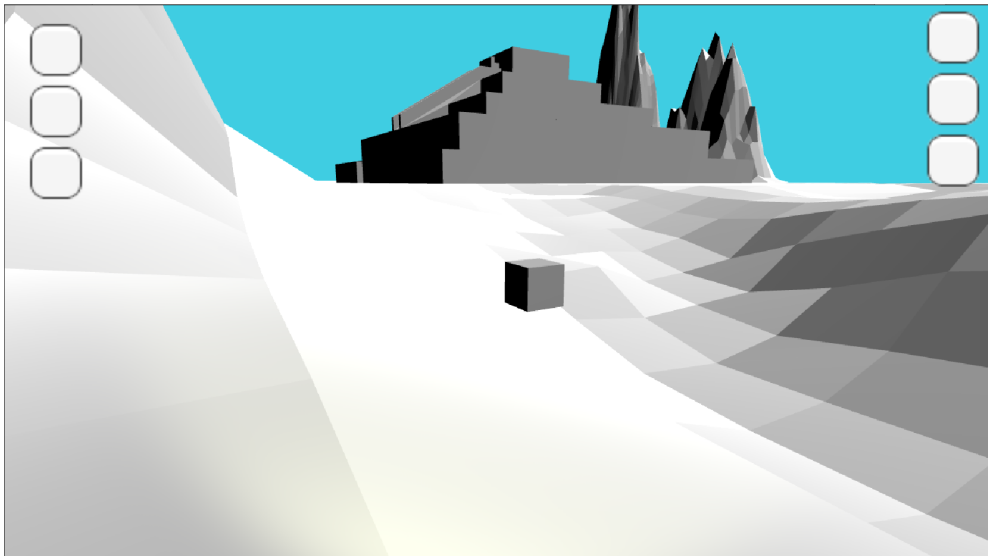


FIGURE 1 – L'objet dans le monde (cube gris)

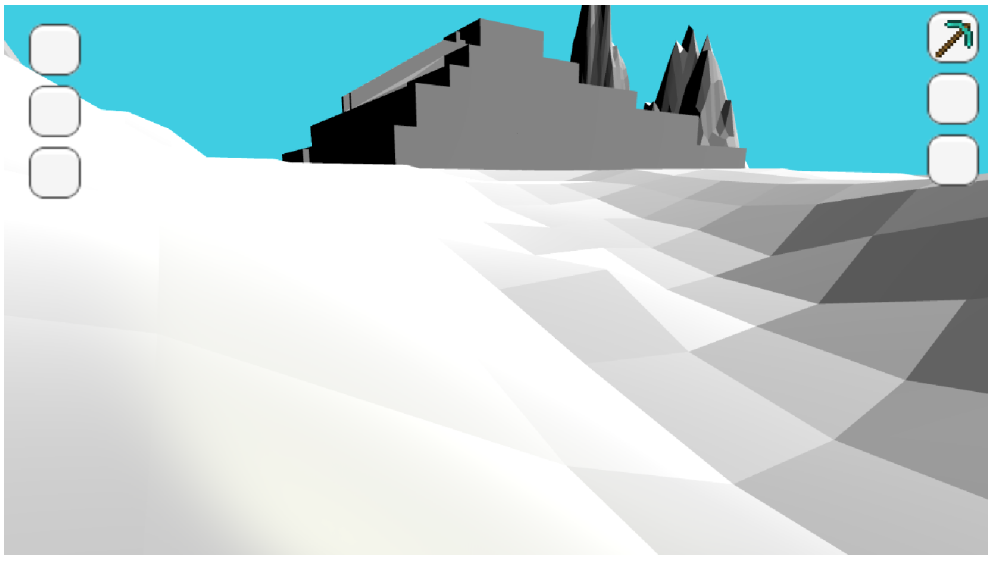


FIGURE 2 – L'objet à été ajouté à l'inventaire

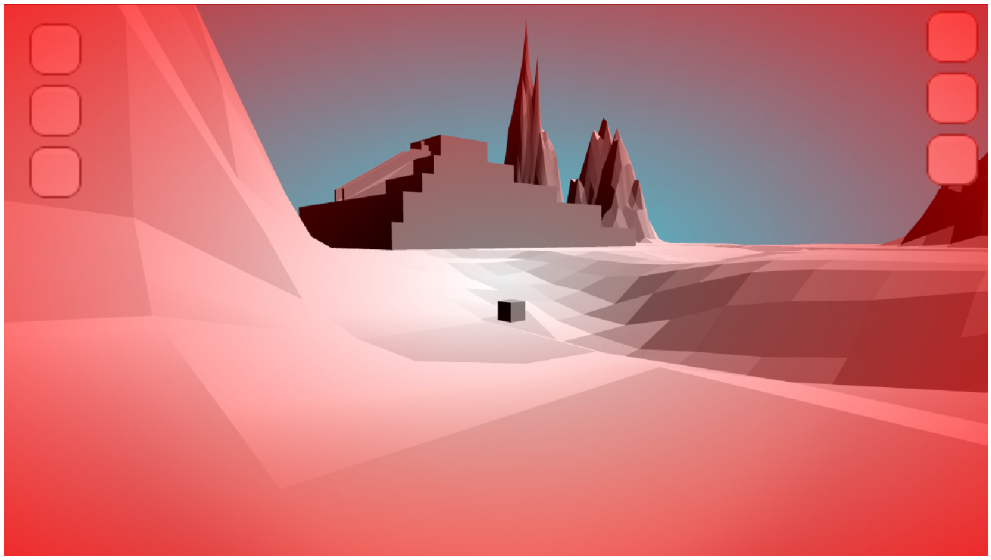


FIGURE 3 – Le joueur a subi des dégâts

5.2 Geoffroy

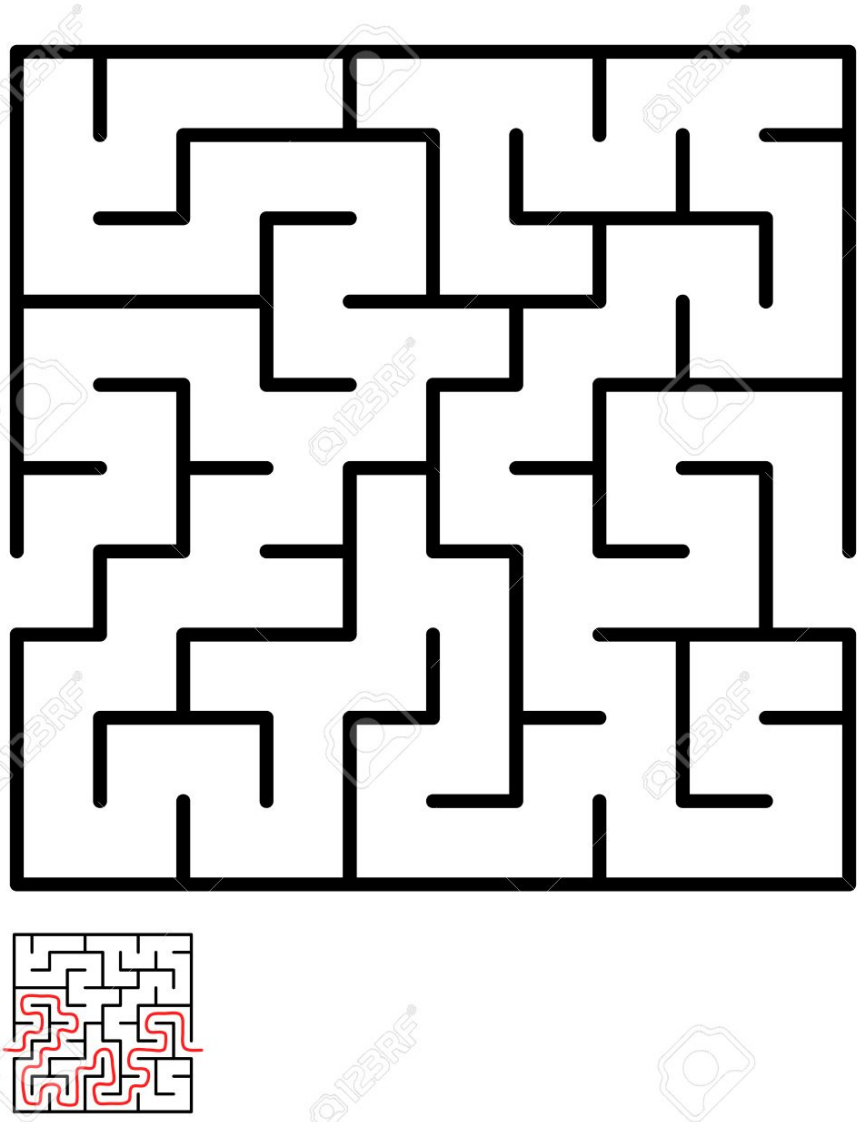


FIGURE 4 – premier étage

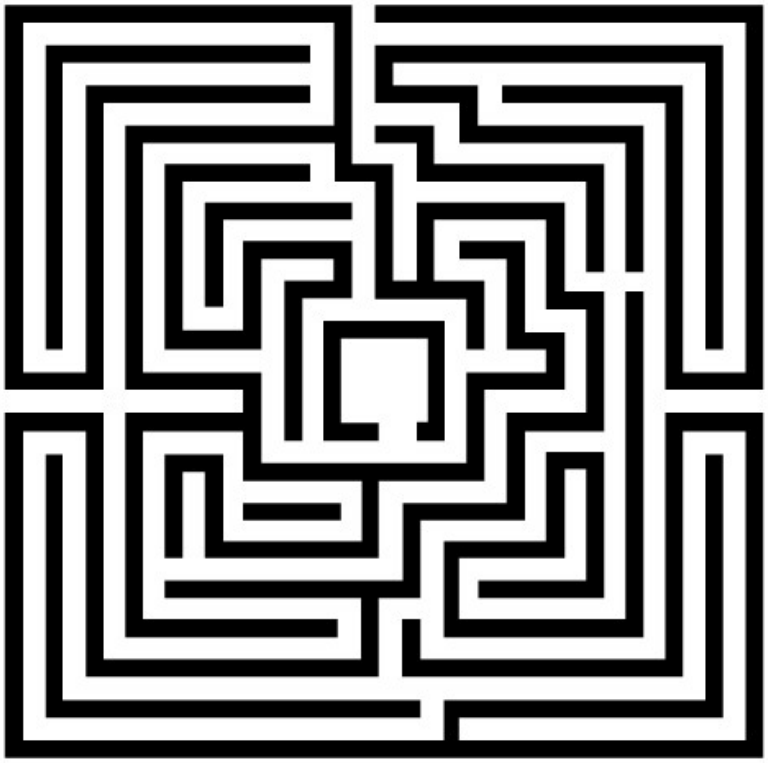


FIGURE 5 – deuxième étage

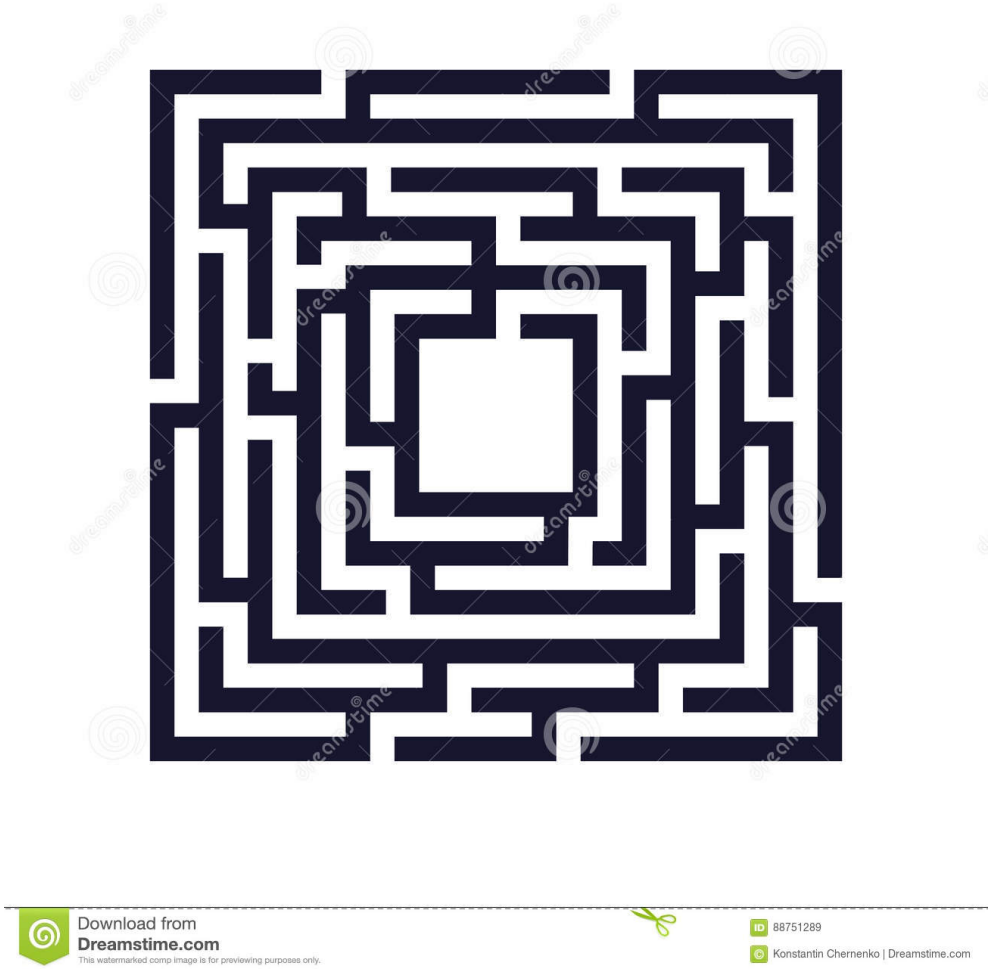


FIGURE 6 – troisième étage pyramide

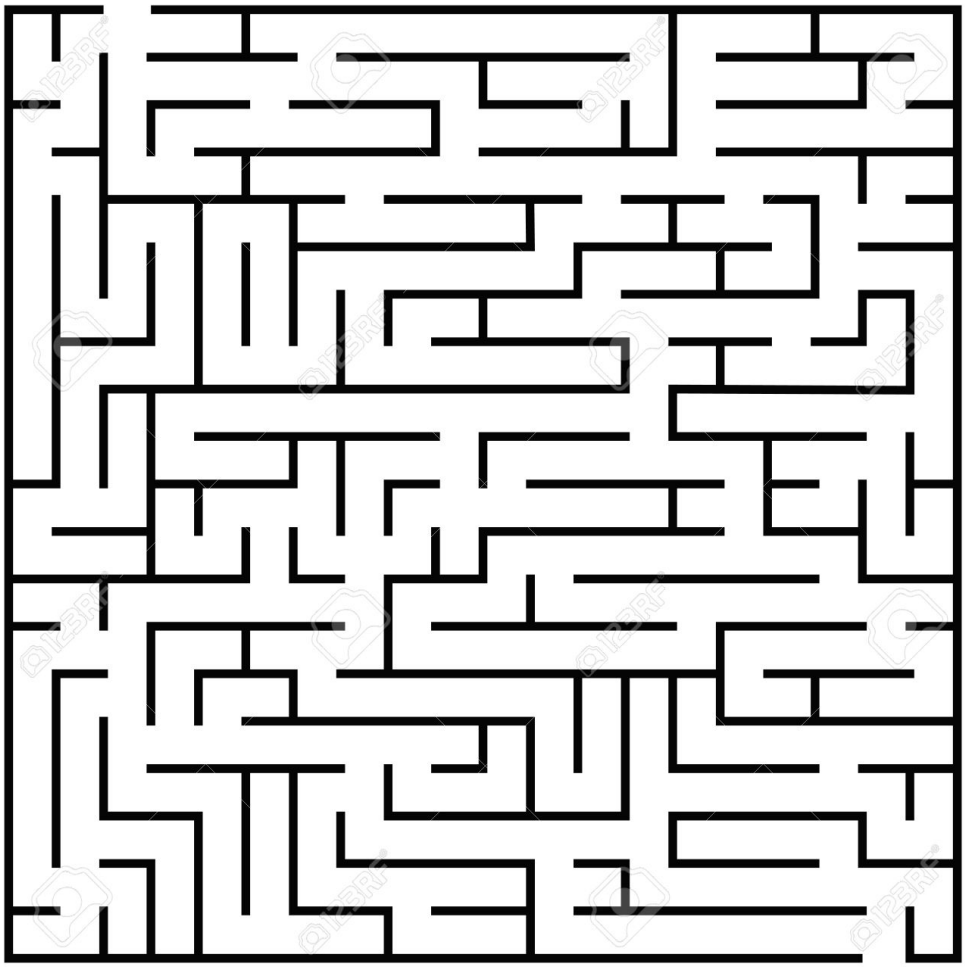


FIGURE 7 – premier sous sol de la pyramide

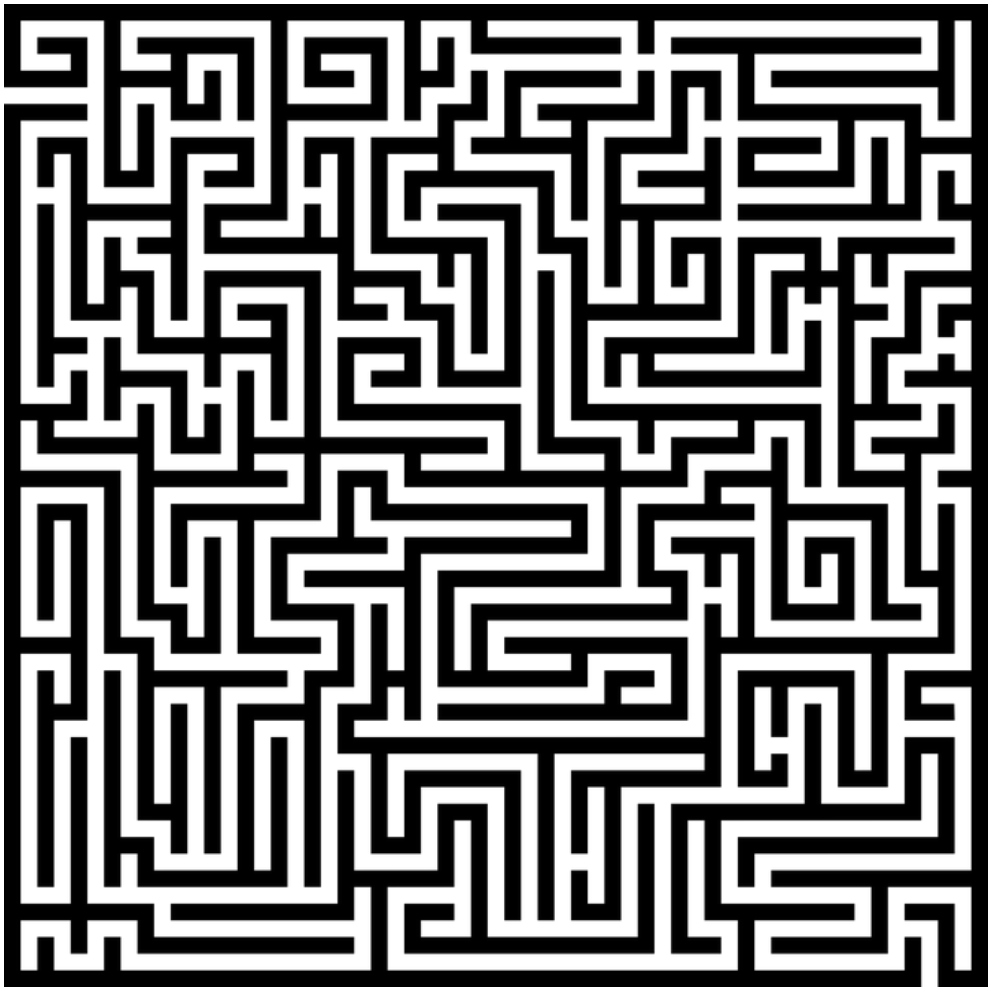


FIGURE 8 – deuxième sous sol de la pyramide

5.3 Martin

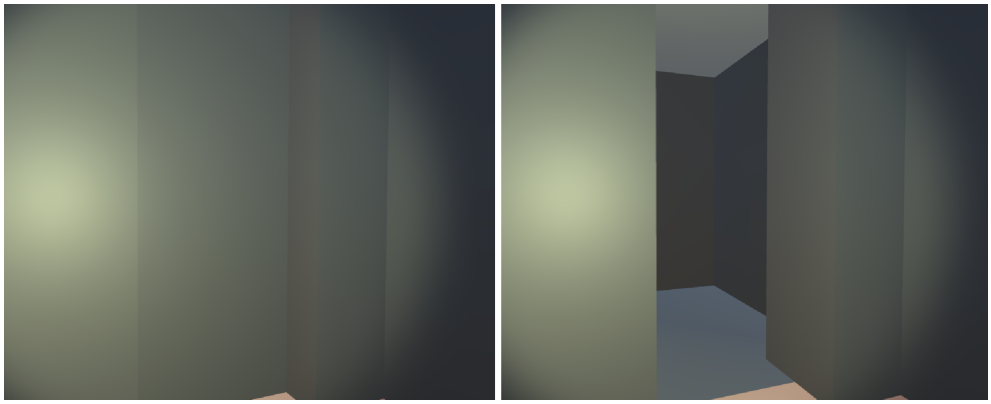


FIGURE 9 – Une porte ouverte et fermée

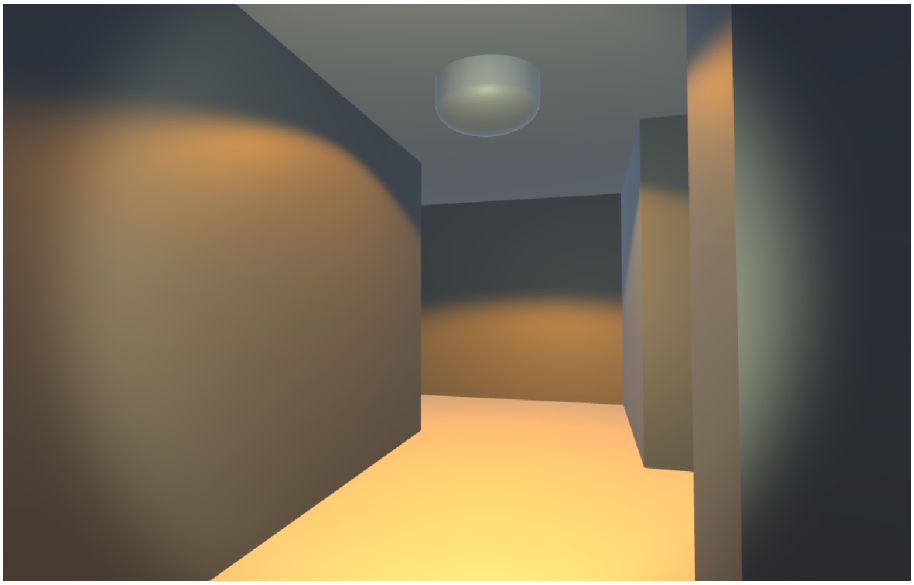


FIGURE 10 – Une lumière

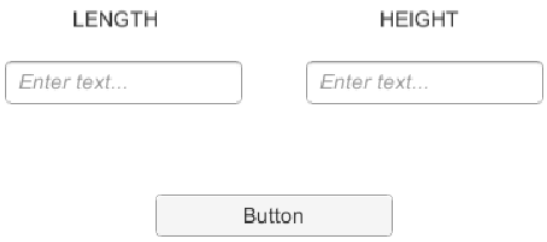


FIGURE 11 – Le menu de sélection des dimensions dans l'éditeur