INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
ROUEN NORMANDIE

Rapport de PAO

Développement d'un démonstrateur 3D sous Unity

Marine RICHARD
Paul SMACQUE
Paul TESSE

A l'intention de
Maxime GUERIAU

Table des matières

1 Introduction **3**

2 Spécifications **4**
2.1 Liste des fonctionnalités prévues : . . . 4
2.2 Liste des objets à implémenter : . . . 4

3 Journal de bord **5**
3.1 Première semaine - 17/01 . . . 5
3.2 Deuxième semaine - 24/01 . . . 5
3.3 Troisième semaine - 31/01 . . . 5
3.4 Vacances - 07/02 . . . 5
3.5 Quatrième semaine - 21/02 . . . 6
3.6 Cinquième semaine - 28/02 . . . 6
3.7 Sixième semaine - 07/03 . . . 6
3.8 Septième semaine - 14/03 . . . 6
3.9 Huitième semaine - 21/03 . . . 7
3.10 Neuvième semaine - 28/03 . . . 7
3.11 Dixième semaine - 04/04 . . . 7
3.12 Vacances - 11/04 . . . 7
3.13 Onzième semaine - 25/04 . . . 7
3.14 Onzième semaine - 09/05 . . . 8

4 Gestion 3D **9**
4.1 Le manoir . . . 9
4.1.1 Les assets utilisés . . . 10
4.2 Collider et Trigger . . . 10
4.2.1 Collider . . . 10
4.2.2 Trigger . . . 10
4.3 Les cachettes . . . 12

5 Implémentation **13**
5.1 Liste des fonctionnalités implémentées . . . 13
5.2 Liste des objets implémentés . . . 13
5.3 Gestion des objets . . . 14
5.4 Le Q-Learning . . . 15
5.5 Liste de ce que nous n'avons pas pu implémenter . . . 20

6 Sources **21**

1 Introduction

Ce document a pour but de tenir compte de l'évolution de notre PAO tout en proposant une synthèse de ce que nous avons appris.

Étant tous les trois de grands fans des jeux de rôles et des jeux vidéos, nous voulions trouver un projet qui nous permettrait de découvrir l'univers du développement de jeux vidéos. Nous avons tout de suite pensé à travailler sous Unity, logiciel très utilisé par les studios de développement professionnels.
Nous avons donc proposé ce PAO à M.Gueriau qui a accepté de nous accompagner et de nous guider tout au long de ce projet.

Ce projet consiste à développer un démonstrateur 3D sous Unity. Enfin, dans l'objectif d'ajouter du challenge et un aspect recherche au PAO, nous avons donc décidé d'intégrer une IA basée sur de l'apprentissage par renforcement à notre jeu.

Les objectifs de ce PAO vont donc être d'apprendre à utiliser l'outil Unity, développer un démonstrateur qui prendra la forme d'un premier prototype de jeu et d'explorer un aspect de recherche scientifique en essayant d'intégrer de l'apprentissage par renforcement à notre jeu.

Dans ce rapport, vous trouverez la liste des spécifications de notre jeu, le tableau de bord de nos avancées et un rapport sur comment nous avons avancés et les problèmes que nous avons rencontrés.

2 Spécifications

Notre jeu est un jeu à la troisième personne où nous contrôlons un personnage dans un environnement 3D. Le personnage en question pourra interagir avec des objets divers et variés. Il sera poursuivi par une entité cherchant à freiner ses découvertes.

Dans l'idéal nous aimerions ajouter un scénario à notre jeu avec une ligne directrice à suivre par l'intermédiaire d'événements, de documents à trouver et d'énigmes.

2.1 Liste des fonctionnalités prévues :

- Jeu en 3D : On modélisera dans un premier temps une pièce seule dans laquelle le personnage pourra évoluer. Dans un second temps, nous ajouterons des pièces pour se rapprocher d'un manoir et avoir un environnement suffisamment grand. Le but sera d'éviter la modélisation 3D au maximum en utilisant des assets gratuits pour passer un maximum de temps sur le développement.
- Jeu à la première personne : On modélisera un personnage qui pourra se déplacer dans l'environnement 3D. Le personnage aura également la possibilité de sprinter et de s'accroupir.
- Interaction avec l'environnement : Le personnage pourra ramasser des objets (indices/notes) et éventuellement les observer (faire tourner pour observer), se cacher dans des placards, ouvrir des portes et des tiroirs. Ceci est une première liste qui pourra être adaptée en fonction de nos idées et de notre avancée.
- Système d'inventaire : Le personnage pourra stocker les objets trouvés dans son inventaire (tels que des clés ou des notes) et ainsi les utiliser à tout moment.
- Gestion d'événements : Suite à des actions du personnage, on scriptera des événements scénarisés, tels que des portes qui se ferment/s'ouvrent ou des lumières qui s'éteignent.
- IA basée sur de l'apprentissage par renforcement : L'idée serait d'utiliser de l'apprentissage par renforcement pour entraîner un ennemi et augmenter progressivement la difficulté du jeu à mesure que l'enquête du joueur progresse.
- Audio : Est-ce que la gestion de sons et de musiques est complexe ? Mettre des sons et de la musique d'ambiance. On pourra éventuellement s'enregistrer en lisant à haute voix les notes du jeu pour renforcer l'aspect narratif/ambiance en jouant ces enregistrements lorsque le personnage examine les notes.

2.2 Liste des objets à implémenter :

- **Lampe torche :** gérer différentes sources de lumière
- **Clef :** bloquer certaines interactions avec des portes, des coffres, etc et associer 2 objets (clef/objet)
- **Piles (pour la lampe) :** gérer le niveau de batterie de la lampe (une sorte de timer) et le ré-augmenter avec les piles
- **Meubles à cachette :** armoire, coffres, lits, etc ; l'idée est de pouvoir se dissimuler dans certains endroits ; gérer l'interaction et la visibilité du monstre

3 Journal de bord

3.1 Première semaine - 17/01

- **18/01 :** Réunion entre nous pour discuter de ce que nous voulions faire dans ce projet.
- **21/01 :** Réunion avec M.Gueriau pour présenter ce que nous souhaiterions faire. Nous avons donc décidé d'un objectif et fixé le cadre du projet (fin du PAO, modalités d'évaluation, etc).
- **23/01 :** Réunion pour rédiger une première version des spécifications.

3.2 Deuxième semaine - 24/01

- **26/01 :** Finalisation des spécifications et prise en main de Unity. Nous avons tous installé le logiciel et commencé à suivre un tutoriel sur la création d'environnement 3D.
- **28/01 :** Réunion avec M. Gueriau pour présenter nos avancées et discussion autour des spécifications.
- **Prochains objectifs :** Déplacement d'un personnage et gestion d'un trigger.

3.3 Troisième semaine - 31/01

- **02/02 :** Nous avons suivi un tutoriel pour apprendre à déplacer un personnage à la 3ème personne et avons ajouté un trigger. Celui-ci consistait en une plaque au sol qui projetait le personnage dans les airs quand celui-ci marchait dessus.
- **04/02 :** Réunion avec M. Gueriau pour présenter ce que nous avions fait.
- **Prochains objectifs :** Modélisation de notre manoir, intégrer notre personnage qui se déplace (notre prefab) et interactions avec l'environnement.

3.4 Vacances - 07/02

A partir de là nous avons décidé de nous répartir le travail au vu des bases acquises par chacun afin d'augmenter notre productivité.

- Développement d'un premier système d'UI : le personnage peut lire des notes sur un mur, il avance vers un mur, appuie sur un bouton et un cadre avec du texte apparaît à l'écran.
- Premier système d'inventaire : conception fait maison avec une super classe *Objet* qui hérite de *GameObject*. Tous les objets (clef, lampe, piles, etc) hériteront de la classe *Objet*. Le joueur a, en guise d'inventaire, une liste d'*Objet*.
- Implémentation de la clef et de la porte : le joueur peut ramasser une clef qui lui permet d'ouvrir une porte ; pour le moment la porte passe à travers le sol quand elle est déverrouillée.
- Recherche d'assets et modélisation d'une première salle de notre manoir.
- **11/02 :** Réunion avec M.Gueriau pour faire un bilan de notre avancée.
- **Prochains objectifs :** système d'inventaire et utilisation d'objets (lampe, piles et clé) et continuer à fouiller dans l'asset store pour trouver ceux qui nous conviendraient le mieux.

3.5 Quatrième semaine - 21/02

- Objectif : modéliser notre manoir mettre notre personnage qui se déplace (notre prefab) et interactions avec l'environnement
- Décor : assets principaux trouvés ; début de la création d'une pièce (un bureau) pour pouvoir tester nos futures fonctions d'interactions.
- Inventaire : début de la création d'un inventaire pour le personnage ainsi que de l'implémentation de fonctions telles que ramasser et utiliser.
- UI : début de la création d'une UI ayant pour objectif d'afficher le statut de certains objets lorsque l'on interagit avec eux ; afficher une boite avec du texte lisible sans interaction de la part du joueur.
- Cachettes : début de la réflexion sur comment les mettre en place.

3.6 Cinquième semaine - 28/02

- Objectif : interactions avec l'environnement (ramasser, ouvrir, se cacher, etc.)
- Décor : bureau terminé, une clef, une porte, un coffre et un endroit où se cacher y sont présents afin de pouvoir faire nos différents tests.
- Inventaire : inventaire fonctionnel ; système d'interaction entre un objet de l'inventaire et un objet de l'environnement fonctionnel (d'un point de vu statut des objets).
- UI : ajout de l'interaction déclenchant l'affichage de l'interface ; ajout du script permettant d'afficher le statut de l'objet.

3.7 Sixième semaine - 07/03

- Objectif : modélisation de l'ennemi (choix d'un prefab à implanter) et développement de l'IA par renforcement
- Cachette : mise en place de transforms servant à fixer la caméra de la vue du joueur dans la cachette ainsi que d'un trigger au niveau du meuble ou se cacher.
- Interaction : réflexion sur comment faire en sorte qu'une porte ayant le statut ouvert puisse pivoter lorsque le joueur la "pousse".

3.8 Septième semaine - 14/03

- Objectif : avoir un système de cachette fonctionnel.
- Cachette : mise en place du script permettant de dés-afficher le joueur et de changer d'emplacement la caméra pour que le joueur ait le point de vu de là où il est caché ; le script se déclenche lorsque le joueur interagit et qu'il est dans la zone de trigger.

3.9 Huitième semaine - 21/03

- Correction des différents bugs : problèmes d'UI qui passe à travers les murs, problèmes avec le système de cachette liés à des problèmes de référentiel, fonctionnement des Hinge Joint pour la porte.
- **Nous avons une V1 de notre scène qui est fonctionnelle**
- Recherches sur le Q-Learning
- **Prochains objectifs :** tester et prendre en main l'algorithme du Q-Learning pour en comprendre le fonctionnement ; agrandir notre manoir ; commencer à implémenter notre ennemi sur Unity

3.10 Neuvième semaine - 28/03

- Décor : structure globale du manoir finie ; recherche d'assets pour le meubler
- Recherche approfondie sur le Q-Learning : tests et manipulation de l'algorithme pour en comprendre le fonctionnement sous Python
- Sous Unity, création d'un ennemi (sous forme d'un golem) et début d'implémentation d'une IA : l'ennemi suit le joueur en toute circonstance ; recherches et utilisation de la NavMesh
- **01/04 :** Réunion avec M.Gueriau pour montrer nos avancées ; discussion autour du Q-Learning et de comment l'intégrer dans notre jeu (notamment comment lier le script Python et le jeu sous Unity)
- **Prochains objectifs :** Travailler Paul T. et Paul S. ensemble pour mettre en place le Q-Learning entre Unity et le script Python ; Préparer une table des États et des Actions ; meubler le décor

3.11 Dixième semaine - 04/04

- Objectif : meubler le décor ; mettre en place le Q-Learning entre Unity et le script Python ; Préparer une table des États et des Actions.
- Décor : Définition des espaces ; mise en place des assets pour un décor cohérent ; placement des assets pour créer des cachettes pour le joueur.
- Golem : Paul s'amuse avec le golem, ça fait rire M. Gueriau ; le golem suit bien le joueur mais est particulièrement bête.

3.12 Vacances - 11/04

- Objectif : meubler le décor et tenter d'intégrer du Python sous Unity

3.13 Onzième semaine - 25/04

- Objectif : Rédaction du rapport et mise en place de l'IA en dur.
- Décor : Ajout d'un toit pour avoir une luminosité assez sombre et vérifier que les éclairages fonctionnent et donnent l'ambiance souhaitée.

3.14 Onzième semaine - 09/05

— Objectif : Finalisation du rapport - codage de l'IA - livraison (sauvetage de la scène)

— Soutenance : Livraison + Démo

4 Gestion 3D

4.1 Le manoir

Très rapidement après avoir commencé à regarder des tutoriels et commencé à faire des tests nous nous sommes rendus compte que sans environnement 3D un minimum défini et travaillé, il serait compliqué de tester nos scripts et nos idées. Nous avons donc décidé que l'un de nous se chargerait de créer cet environnement dans lequel notre personnage évoluerait par la suite.

Tout d'abord, nous nous sommes concentrés sur la création d'une seule pièce, un bureau. Après avoir créé le terrain et les quatre murs il a fallu trouver des assets afin de pouvoir mettre des objets autres que des cubes et de rajouter des textures pour pouvoir mieux nous projeter.

Lorsque nous nous sommes lancés dans la recherche des assets nous avons vite vu que l'asset store de Unity, d'où tous nos assets sont issus, est très fourni. De nombreux assets contenant tout ce dont nous avions besoin était proposés mais à des prix loin d'être abordables pour un projet sans budget tel que le notre. Notre premier filtre a donc été la gratuité des assets (en plus du filtre 3D mais cela va sans dire).

Nous avons finalement trouvé notre bonheur en mélangeant plusieurs assets, le problème principal à ce niveau là est que l'on ne peut pas savoir avant de les tester si leurs résolutions/textures/styles étaient cohérents ensembles. Il a donc fallu de nombreux téléchargements d'assets avant de trouver ceux fonctionnant bien ensembles.

Un des soucis que nous avons aussi rencontré est le fait que certaines textures, par exemple celle pour les murs, ont une résolution trop faible. Cela fait qu'il est impossible d'étirer le mur sans déformer complètement les briques, c'est pour ça que l'on a du coller plusieurs murs les uns à la suite des autres pour pouvoir créer la structure globale du manoir.

De plus, étant donné que nous utilisions uniquement des assets gratuits il a fallu parfois faire preuve d'inventivité afin de créer certaines chose que nous ne trouvions pas dans l'asset store, comme une porte d'entrée ancienne par exemple.

Enfin, certains assets, étaient assez compliqués à prendre en main pour des novices, notamment celui pour les flammes, ce qui rendait certaines créations très chronophage.

4.1.1 Les assets utilisés

- Free Medieval Room, utilisé principalement pour le toit et les meubles
- Free Medieval Structure, la porte principale est une superposition de différents prefabs dont certains appartiennent à cet asset.
- Old Sea Port Environment, les textures des murs ainsi que certains prefabs ayant servis pour la porte principal proviennent de cet asset.
- Procedural fire, utilisé pour toutes les flammes dans les cheminées ainsi que pour les flammes magiques.
- Furniture Free Pack, utilisé pour la plupart du mobilier.
- Books, utilisé pour tous les parchemins et les livres.
- Mega Fantasy Props Pack, utilisé pour la plupart du mobilier et pour tous les éclairages.
- Old Medieval Chest, l'asset pour les coffres et les clefs.
- Rusty Flashlight, l'asset pour la lampe torche.
- Survival Game Tools, asset contenant du matériel typique de survival horror, utilisé pour les batteries.

4.2 Collider et Trigger

Les colliders et triggers sont des propriétés propres aux objets Unity. On peut se les représenter par des "zones", par exemple sphériques ou rectangulaires, représentées par des contours verts, propres aux objets.

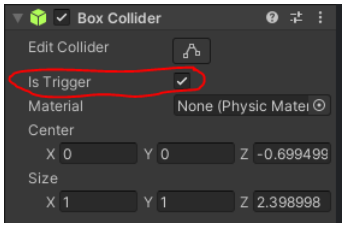
4.2.1 Collider

Le collider a pour but d'éviter qu'en jeux les objets passent les uns au travers des autres. C'est une zone que l'on définit en général extrêmement proche de l'objet de manière à épouser un maximum ses contours. Un collider précis est, certes, très réaliste mais très difficile à réaliser pour des objets complexes, comme les buissons, et peut engendrer des bugs. C'est pour cela que l'on privilégie les box ou capsule collider.

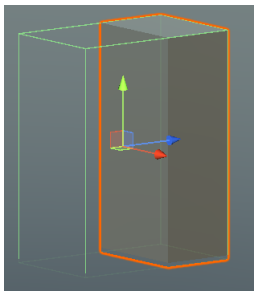
4.2.2 Trigger

Le trigger est utilisé, principalement, pour le déclenchement de scripts et l'acquisition d'informations. C'est une zone, en général plus large que l'objet, qui va en quelque sorte servir de capteur. Unity permet, notamment grâce aux fonctions OnTriggerEnter et OnTriggerExit de définir des actions à réaliser. Ce sont dans notre cas les triggers qui permettent de déclencher la possibilité de ramasser une clef lorsque le personnage se situe à moins d'un mètre de celle-ci par exemple.

Il est important de préciser que pour créer un trigger, on crée un box collider et en éditant ce collider, on le transforme en trigger en cochant une case "Is Trigger" dans l'inspecteur.



(a) Exemple de paramétrage d'un box trigger



(b) Exemple de box collider utilisé en trigger

```
public bool playerInRange;

0 references
private void OnTriggerEnter(Collider col)
{
    if(col.name == "Player")
    {
        playerInRange = true;
    }
}

0 references
private void OnTriggerExit(Collider col)
{
    if(col.name == "Player")
    {
        playerInRange = false;
        dialogBox.SetActive(false);
    }
}
```

FIGURE 2 – Exemple de code gérant un trigger

Ce script 2 lié à une porte en l'occurrence, permet de détecter le joueur. Cette information peut ensuite être utilisée plus loin dans le script afin de permettre l'ouverture de la porte si le joueur est in range.

4.3 Les cachettes

Dès le début, étant donné que nous voulions partir dans un jeu de type survival horror, nous avons évoqué le fait que notre personnage puisse se cacher afin d'éviter/d'échapper à la créature le poursuivant. Une fois notre première pièce créée nous avons pu mettre ça en place et le tester.

Initialement, nous ne voyions pas vraiment comment procéder. C'est M. Gueriau qui nous a mis sur le chemin en nous parlant de changement de caméra. Paul T. a donc mis en place un script qui, lorsque le joueur entre dans une zone de trigger, ajoute une action possible : se cacher.

Si le joueur appui sur le bouton permettant de déclencher cette action la suite du script se déclenche et la caméra change. La caméra du joueur quitte le transform lié au joueur (point d'accroche), pour se replacer sur un transform mis en place au préalable dans la cachette (angle de la caméra déjà paramétré), devient le point de vue du joueur et au même moment son personnage se "désactive" afin qu'on ne le voit plus et qu'on ne puisse pas le bouger. Le joueur se retrouve donc "caché" dans la cachette sans vraiment y être. En appuyant sur le bouton d'action, l'opération inverse se produit et le joueur reprend alors possession de son personnage avec la caméra placée en vue première personne.

Pour ce qui est de la création d'un transform, la seule façon de procéder est de créer un cube et de supprimer tous ses éléments sauf le transform (repère avec les 3 flèches sur la figure 4).

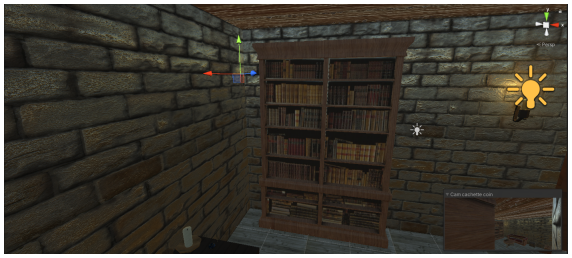


Figure 3 – La cachette dans le coin avec le transform

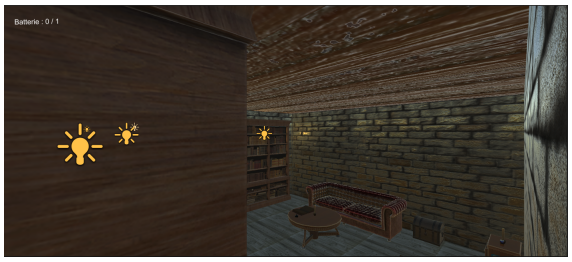


Figure 4 – Point de vue du personnage lorsqu'il est caché

5 Implémentation

5.1 Liste des fonctionnalités implémentées

- Jeu en 3D : Un manoir en 3D meublé avec des assets gratuits.
- Jeu à la première personne : Personnage pouvant se déplacer et interagir avec l'environnement.
- Interaction avec l'environnement : Le personnage possède un système d'inventaire et peut interagir avec son environnement.
- Système d'inventaire : Ramasser et utiliser les objets à des endroits pré-définis.
- IA basée sur de l'apprentissage par renforcement : Script Python servant de base/exemple pour le future Q-Learning de l'IA. Prémices d'IA déjà implémentée=s "en dur". Pour le moment elle n'apprend pas. Un dictionnaire est créé en dure pour gérer la prise de décisions en fonction de l'état actuel.

5.2 Liste des objets implémentés

- **Lampe torche :** gérer différentes sources de lumière
- **Clef :** bloquer certaines interactions avec des portes, des coffres, etc et associer 2 objets (clef/objet)
- **Batteries (pour la lampe) :** gérer le niveau de batterie de la lampe (une sorte de timer) et le ré-augmenter avec les piles
- **Meubles à cachette :** armoire, coffres, lits, etc ; l'idée est de pouvoir se dissimuler dans certains endroits ; gérer l'interaction et la visibilité du monstre

5.3 Gestion des objets

Notre diagramme de classe :

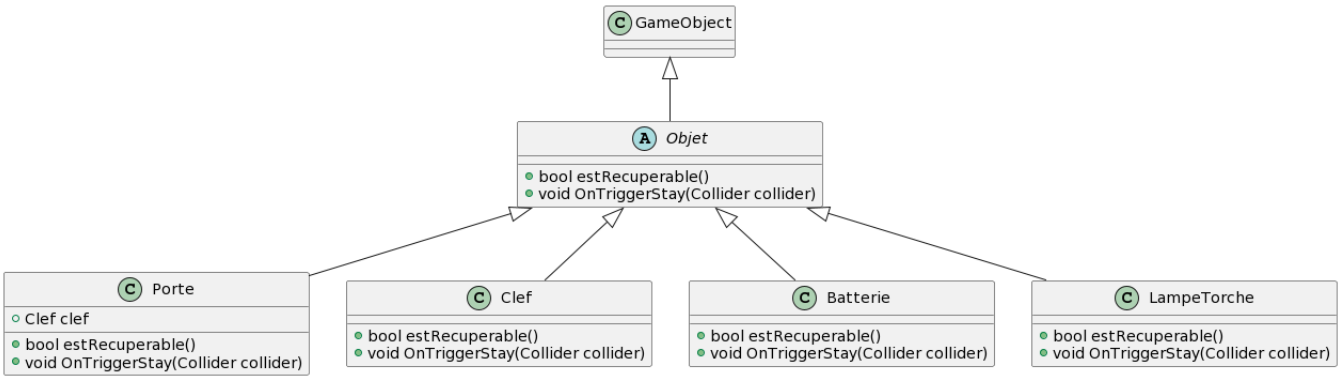


FIGURE 5 – Diagramme de classe de la classe GameObject

La classe *Objet* est abstraite donc ses méthodes le sont aussi et doivent être redéfinie dans chaque classe fille. La fonction *estRecuperable* retourne True ou False en fonction de si l'objet peut être ramassé ou non. Par exemple une Porte ne peut pas être ramassée alors qu'une clef oui.
Remarquons également que la classe *Porte* possède un attribut *Clef* pour savoir quelle clef permet d'ouvrir cette porte. Ce qui est pratique c'est que le champs apparaît dans l'interface Unity, il nous suffit simplement d'y glisser l'objet Clef correspondant.
L'objectif du projet étant avant tout de coder et de mettre la main à la pâte, cette conception a été faite sur le tas en codant. Le diagramme a été fait une fois l'implémentation finie, afin de conserver une vision claire. Cependant, cette conception n'est pas parfaite. Par exemple, la fonction *estRecuperable* aurait du se trouver dans une Interface (*ObjetRecuperable* par exemple).

5.4 Le Q-Learning

Le Q-Learning est une technique d'apprentissage par renforcement. Ce type d'apprentissage est très utilisé dans le jeu vidéo pour créer des IAs. En effet, l'apprentissage par renforcement permet d'entraîner un agent à prendre des décisions, sans avoir besoin de créer au préalable un modèle mathématique lui permettant de prendre ses décisions. La modélisation de l'environnement de l'agent est également extrêmement simplifiée, ce qui offre un net avantage. Cela provient du fait que l'agent va explorer lui même l'environnement pour apprendre et n'a donc pas besoin de le connaître au préalable.

Le principe même du Q-Learning repose sur la Q-table. Cette matrice fait office de matrice des récompenses. L'agent prend ses décisions de sorte à toujours maximiser la récompense qu'il peut obtenir suite à sa prochaine action. Observons cette Q-table plus en détail.

$$
\begin{array}{cc}
 & \textbf{Action} \\
\textbf{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
R = \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}
\end{array}
$$

FIGURE 6 – Exemple d'une Q-table

Chaque ligne correspond à un état et chaque colonne à une action. Pour le moment, nous ferons abstraction de ce que sont concrètement les états et les actions. Ce qu'il est important de visualiser lorsque l'on voit cette matrice c'est comment l'agent l'utilise.

Prenons pour exemple le cas où l'agent serait actuellement dans l'état 1. Celui-ci a le choix entre 6 actions différentes mais privilégiera l'action 5 étant donné que la récompense est de 100 ([1,5]) et correspond au maximum de ce qu'il peut obtenir. Suite à cet action l'état changera ou non. Voici en résumé comment l'agent prend ses décisions à partir de cette Q-table.

A présent, nous allons nous intéresser à l'algorithme de génération de cette Q-table :

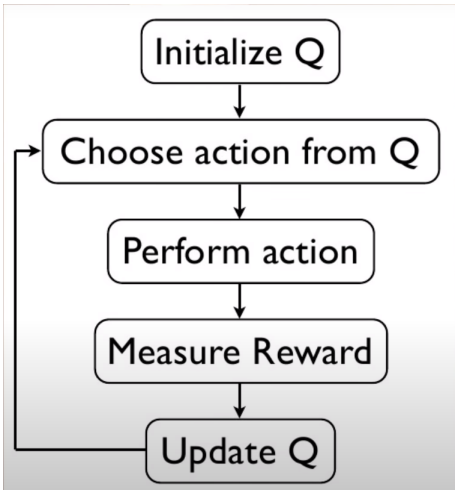


FIGURE 7 – Algorithme de génération d'une Q-table

Comme vous pouvez le voir, la Q-table est tout d'abord initialisée à 0. Ensuite commence la boucle d'apprentissage. L'agent regarde dans quel état il se trouve et doit choisir quelle action effectuer. Deux cas de figure sont alors possibles :

— Exploitation : consiste tout simplement à utiliser la Q-table de manière à maximiser la récompense. Il s'agit du comportement de base de l'agent.

— Exploration : l'agent choisit une action aléatoirement parmi celles possibles et ce quelque soit la valeur de la récompense générée.

Le choix entre exploitation et exploration se fait en fonction d'une valeur entre 0 et 1 correspondant à une probabilité tirée au sort. Si la valeur obtenue est inférieur au coefficient epsilon fixé, alors l'agent explore. Dans le cas contraire, il exploite. C'est le principe du epsilon greedy.

```
def take_action(st, Q, eps):
    # Take an action
    if random.uniform(0, 1) < eps:
        action = randint(0, 3)
    else:  # Or greedy action
        action = np.argmax(Q[st])
    return action
```

FIGURE 8 – Script utilisant le principe du epsilon greedy

Ce mécanisme est très important, car il force l'agent à explorer de nouvelles possibilités et lui évite de n'explorer que certains cas de figure, et permet ainsi de renforcer l'IA. Il est important de préciser cependant, que plus l'agent aura passé de temps dans l'environnement et plus le coefficient epsilon diminuera afin de privilégier l'exploitation de la Q-table à long terme (celle-ci étant suffisamment mise à jour).

Revenons à l'algorithme. Le choix de l'action a été fait. L'agent effectue donc l'action et récupère la récompense. Enfin, il met à jour la Q-table en fonction de la valeur de la récompense ainsi obtenue. Ce processus est itéré jusqu'à avoir une Q-table permettant une bonne prise de décision. L'objectif est qu'une fois cette phase d'apprentissage terminée, l'agent exploite systématiquement celle-ci pour prendre ses décisions et agir de manière efficace.

Sur nuage se trouvent quelques scripts python mettant en pratique ces concepts de Q-Learning. Voici des exemples de résultats obtenus :

```
1 [-0.1, -0.06, 0, -0.11084120283665608]
2 [-0.1147592600251134, 0.17329467147733355, -0.199, -0.07165128824682454]
3 [-0.06898967323170149, 0.7850557774978394, -0.3804996104578845, 0.07271786922626683]
4 [0.01862821529440456, 0.3204462409517161, -0.04799887837736719, 0.06580321351900376]
5 [0.07592657808269769, 0.0786445371311847, 0.02259333672284568, -0.00572211133228908]
6 [0.0, 0, 0.008536861791546618, 4.987487076729393e-08]
7 [-0.1, -0.057, -0.1716, 0.04328448621898391]
8 [-0.20695285172243952, 0.0985211947163041, -0.21334691362591496, 0.7296543402772749]
9 [0.052875913171815836, 0.2586157061934806, 0.24958470069367006, 0.598754261056799]
10 [0.00846169274156359, 0.11936861865331869, 0.5430737869320597, 0.9969567472778297]
11 [0, 0, 0, 0]
12 [0.0, 0, 0.1, 0.0]
13 [-0.114, -0.0016541541538623358, -0.16317793860250943, 0.10739806933877324]
14 [0.18239210563001682, 0.01251403996800695, -0.17664016015762865, 0.4239455840106698]
15 [0.8669604534558706, 0.40030313720567173, 0.18457991203541763, 0.08661691604692186]
16 [0.07518963846396692, 0.1282138575643058, 0.14112661523689, 0.47047708328476945]
17 [0.9721871610556306, 0.2525926372528924, 0.09143605582363973, 0.006476851013778772]
18 [0.0, 0.027974434453343182, 0, 0]
19 [-0.22830760143300002, 0.01646781349259434, -0.03354975578163652, 0.38185930591790757]
20 [0.04169283380455093, 0.048111406779838224, 0.06109114475585925, 0.7022602569237083]
21 [0.6193849619842399, 0.2901758025487867, 0.24992114611562155, 0.18218235538469812]
22 [0.05183808138630165, -0.06786966296484664, 0.24011622843942695, 0.6528855194577762]
23 [0.5722690408357715, 0.03478375673038973, 0.15355056417270385, 0.09223506664471992]
24 [0.0027609346086519236, 0.0, 0.5461897588424686, 0.0040565374091550005]
25 [0.05793638449785797, 0.03500383623040464, 0.039900178598190644, 0.2367987157365092]
26 [0.2138009660944654, -0.010182122569430574, 0.10529427815638831, 0.319245290351165]
27 [0.6748368973634621, -0.42362325749481067, 0.12325183834732432, -0.2211787906565272]
28 [0.18281918078996154, 0.009000000000000001, 0.011451890783468963, 0.004227803100000001]
29 [0.31531175128777567, 0, 0, 0.0]
30 [0.004552791705000001, 0, 0.0, 0]
31 [0.13592462100059943, 0.02914836564192535, 0.021873850900474737, 0.021452213809192842]
32 [0.1728690257223288, 0.004311841524982257, 0.0186871707370022, -0.33835752607649094]
33 [0.09913859704589309, -0.23209972834630488, -0.007837758556187508, 0.012097410431888615]
34 [0, 0.0171, -0.036584328534570956, 0.271]
35 [0, 0, 0, 0]
36 [0, 0, 0, 0]
```

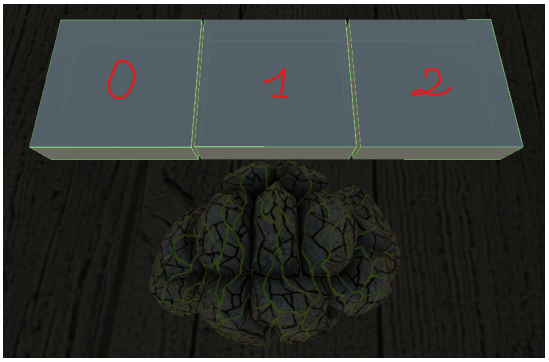
FIGURE 9 – Q-table après 100 itérations de l'algorithme

```
début test agent
tour 1
---------------------
-1  -0.6    -0.3    0   0   0
-0.6    0   0.5 0   1   0
-0.3    0   0   0   0   0
0   0   0.3 0   0.5 0
0   0   0   -0.3    0   0
X   0   -0.5    0   1   -1
tour 2
---------------------
-1  -0.6    -0.3    0   0   0
-0.6    0   0.5 0   1   0
-0.3    0   0   0   0   0
0   0   0.3 0   0.5 0
0   0   0   -0.3    0   0
X   0   -0.5    0   1   -1
tour 3
---------------------
-1  -0.6    -0.3    0   0   0
-0.6    0   0.5 0   1   0
-0.3    0   0   0   0   0
0   0   0.3 0   0.5 0
X   0   0   -0.3    0   0
0   0   -0.5    0   1   -1
tour 4
---------------------
-1  -0.6    -0.3    0   0   0
-0.6    0   0.5 0   1   0
-0.3    0   0   0   0   0
0   0   0.3 0   0.5 0
0   X   0   -0.3    0   0
0   0   -0.5    0   1   -1
```

Figure 10 – Agent évoluant dans son environnement en full exploitation

Pour ce qui est de l'implémentation sous Unity, nous avions pensé réaliser une forme d'interface en C# pour faire le lien entre le script Python mettant à jour la Q-table et l'environnement Unity. Malheureusement, Unity n'a actuellement pas de module stable permettant d'intégrer du python et nous n'avons pas eu le temps de ré-implémenter notre algorithme de Q-Learning sous Unity. Par conséquent, nous nous sommes concentrés sur la partie modélisation des états et actions dans notre scène. L'idée est de coder en dur la Q-table dans un premier temps.

(a) Notre Agent



(b) Modélisation du FOV de notre agent

Ici vous pouvez voir notre agent qui n'est autre qu'un golem. La partie qui nous intéresse ici le plus est la modélisation du FOV de ce golem. Il s'agit de trois triggers permettant de détecter les éléments de l'environnement et de les situer : gauche, centre et droite. De cette manière nous pouvons modéliser l'état du golem à un instant t sous la forme d'un tableau d'états de taille 3. Il existe 3 états composites pour ces "capteurs" : Obstacle, Joueur et Rien (respectivement 0, 1 et 2).

Ainsi, nous avons par exemple à l'instant t, l'état global suivant : [1,0,2] ce qui se traduit par le joueur est à gauche, il y a un obstacle devant et rien à droite.

Ensuite, en l'absence de génération de la Q-table, nous avons du attribuer à chaque état global possible une action à effectuer pour pouvoir créer un dictionnaire mappant états et actions.

Les actions sont au nombre de cinq :

- Continuer : le golem continue tout droit son chemin [0]
- Droite : le golem tourne à droite [1]
- Gauche : le golem tourne à gauche [2]
- Retourner : le golem se retourne [3]
- Suivre : le golem suit le joueur [4]

Il ne reste plus qu'à créer le dictionnaire avec par exemple les couples suivants :

- dico.Add(new Etat[2,2,0], Actions.Gauche) ;
- dico.Add(new Etat[0,2,2], Actions.Droite) ;

5.5 Liste de ce que nous n'avons pas pu implémenter

- Système d'inventaire : Le joueur pourra à tous moments ouvrir son inventaire et examiner son contenu.
- Gestion d'événements : Suite à des actions du personnage, on scriptera des événements scénarisés, tels que des portes qui se ferment/s'ouvrent ou des lumières qui s'éteignent.
- IA basée sur de l'apprentissage par renforcement : Faire en sorte que l'IA forme sa Q-table seule grâce au script Python.
- Audio : Est-ce que la gestion de sons et de musiques est complexe ? Mettre des sons et de la musique d'ambiance. On pourra éventuellement s'enregistrer en lisant à haute voix les notes du jeu pour renforcer l'aspect narratif/ambiance en jouant ces enregistrements lorsque le personnage examine les notes.

Certains éléments n'ont pas pu être implémentés suite à des problèmes de versionnage liés à notre utilisation non adaptée de Sharepoint. Nous avons tenté de travailler simultanément sur une même version de la scène principale en utilisant Sharepoint pour synchroniser nos travaux. Nous nous sommes finalement retrouvés avec de nombreux conflits et avons eu de gros problème vers la fin du projet.C'est pourquoi pour la suite du PAO nous recommandons que les membres codent sur des versions locales non partagées de la scène principale. L'utilisation d'un Git pourrait également être utile.

6 Sources

Vidéos :

- Chaine youtube française *TUTO UNITY FR*
- Tuto pour installer Unity
- Playlist de tuto pour développer un RPG sous Unity
- Tuto sur les charnières pour des portes
- Tuto sur les charnières pour des portes
- Tuto sur le Q-Learning
- Tuto sur le Q-Learning
- Tuto sur le Q-Learning

Ressources :

- Git pour l'installation de scripts pour utiliser NavMesh
- Git script python Q-Learning

Assets :

- Adventure Character
- Furniture Pack
- Flashlight
- Environmental Pack
- Old Medieval Chest
- Industrial building/Garage
- Medieval house
- Old Sea Port
- Medieval house modular lite
- Medieval Room
- Blacksmith's Forge
- Town Houses Pack
- Books, Scrolls and Other Things
- Animated Spiders
- Flashlight
- Medieval Tavern Pack
- Horror Starter Pack
- Rust Key
- Procedural fire