# Universitus
## Game Design Document

## Présentation

Universitus est un jeu sur navigateur de type point & click, où le joueur est amené à entrer des commandes de type unix dans un terminal émulé afin de progresser.

### Histoire :
Dans une université où magie et technologie cohabitent, les étudiants sont formés depuis des décennies à l'usage d'un pouvoir extrêmement rare appelé l'Unixité.
Vous êtes un des nouveaux étudiants, malheureusement le début de votre année est marquée par la libération d'un processus daemon particulièrement redoutable, qui se met à répandre des processus zombies dans tout le campus.
Il vous faudra parcourir l'université de fond en comble et acquérir les connaissances nécessaires afin de mettre un terme à cette terrible menace, et peut-être de découvrir qui se cache derrière tout ça.

### Type :
Point & click style old-school en terminal émulé

### Public cible :
Nouveaux étudiants en informatique

## Inspirations

Terminus pour l'interaction avec l'utilisateur par le biais d'un terminal.
Harry Potter pour l'univers.

# Fonctionnalités

## Déplacements et map

Durant l'aventure, le joueur est amené à traverser l'université en long et en large, de ses plus sombres souterrains à ses étages les plus hauts. Concrètement, les divers environnements qu'il parcourt sont représentés sous forme de **lieux**.

Un **lieu** est défini par son nom, sa localisation dans l'université, sa description, la liste des lieux qui lui sont adjacents et la liste des objets, des pnjs, des ennemis et des quêtes qu'il contient.

Afin d'interagir avec le lieu dans lequel il est situé, le joueur a à sa disposition une liste de commandes décrites ci-dessous.

**cd :** C'est la commande de base, la plus simple et une des commandes que le joueur pourra utiliser dès le début du jeu. Elle permet de passer du lieu courant au lieu spécifié.

Elle doit être entrée sous la forme : **< cd lieu >** où le paramètre lieu est le nom du lieu où le joueur souhaite se déplacer. Lorsque le joueur entre dans un nouveau lieu, la description de ce nouveau lieu s'affiche dans le terminal.

Cette commande permet également une interaction avec l'inventaire, mais nous en parlerons dans la section appropriée.

**ls :** Une commande qui permet de lister dans le terminal tous les objets, les pnjs, les ennemis potentiels et les lieux adjacents du lieu courant. L'inventaire apparaît également au joueur dans la liste.

Elle doit être entrée sous la forme **< ls >** et ne prend aucun paramètre.

Cette commande est aussi utilisable dans l'inventaire, mais ce point sera abordé dans la section appropriée.

**pwd :** Une commande qui permet d'afficher dans le terminal une représentation visuelle du lieu courant.

Elle doit être entrée sous la forme **< pwd >** et ne prend aucun paramètre.

**Exemple :**

Ici nous allons montrer un exemple typique d'interaction avec le lieu courant.
Le joueur se trouve dans le lieu **Salle de musique.**

Il entre la commande **pwd** afin d'observer les lieux :
**> pwd**
Une image du lieu s'affiche dans le terminal.
Puis il entre la commande **ls** afin de lister le contenu du lieu :
**> ls**
La liste apparaît :
**Couloir Placard**
Il choisit de changer de lieu et d'aller dans le couloir :
**> cd couloir**
Le joueur se trouve désormais dans le couloir ! Une description du lieu apparaît :
**Vous vous trouvez dans un couloir étroit et obscur. Des yeux brillants semblent vous observer à travers une fissure dans le mur.**

## Objets et inventaire

Tout au long de son périple, le joueur est amené à trouver dans les lieux qu'il parcourt une multitude d'**objets** avec lesquels il pourra interagir de diverses manières. Ils peuvent apparaître sous forme de clés à prendre et utiliser, de portes à ouvrir, d'armes etc. Les objets sont généralement des entités simples que le joueur pourra éventuellement ramasser et placer dans son **inventaire**.

L'inventaire du joueur est extrêmement important, c'est là qu'il pourra entreposer tous les objets qu'il a acquis au cours du jeu. A l'aide des commandes qui lui sont accessibles, le joueur peut organiser son inventaire exactement comme il l'entend, créer des poches et des sous-poches afin de le trier et d'accéder rapidement aux éléments qu'il recherche.

De la même façon que pour les lieux, le joueur peut interagir avec les objets et l'inventaire avec les commandes suivantes :

**cd :** Il s'agit de la même commande que celle permettant de se déplacer de lieux en lieux, elle est utilisée aussi afin de parcourir l'inventaire. Pour ouvrir l'inventaire depuis un lieu, il suffit de remplacer le paramètre **lieu** par le nom de l'inventaire.

A partir de là, le joueur peut se déplacer librement entre ses poches comme dans un dossier unix, à l'aide de deux versions de la commande : **< cd poche >** avec le paramètre poche indiquant la poche désirée (accessible depuis la poche courante ou en entrant le chemin jusqu'à cette poche) ; ou bien **< cd .. >** pour revenir dans la poche parent de la poche courante.

Pour sortir de l'inventaire, deux options sont offertes : se déplacer dans la poche nommée "Quitter" ou revenir à la racine puis entrer **< cd .. >**. Cela aura pour effet de renvoyer le joueur dans le lieu où il se trouvait.

**ls :** La version de cette commande accessible depuis l'inventaire fonctionne de la même manière que celle accessible depuis un lieu : en entrant **< ls >** dans l'inventaire, une liste des objets contenus dans la poche courante est affichée dans le terminal.

**mkdir :** Cette commande permet de créer une poche dans la poche courante. Elle est surtout utile afin d'organiser son inventaire.

Elle doit être entrée sous la forme **< mkdir nom_poche >** où "nom_poche" représente le nom de la poche que l'on souhaite créer.

**mv :** Cette commande permet de déplacer un objet. L'objet peut être déplacé depuis le lieu courant dans l'inventaire, depuis une poche de l'inventaire au lieu courant, d'une poche à une autre poche, voire d'une poche ou d'un lieu à un objet/pnj dans certains cas particuliers (par exemple déplacer une clé d'une poche à une porte). L'usage de cette commande est sujet à des restrictions en fonction de l'objet ou de l'endroit où l'on souhaite placer l'objet (Une porte ne peut par exemple pas être déplacée, au contraire d'un livre).

Elle doit être entrée sous la forme **< mv objet destination >** où "objet" représente l'objet que l'on souhaite déplacer et "destination" l'endroit où l'on souhaite déposer l'objet.

**cp :** Cette commande permet de cloner un objet et de le placer à l'endroit voulu. Elle s'utilise de la même façon que la commande **mv,** à l'exception que cette commande agit **sur une copie de l'objet sélectionné** et non sur l'objet en lui-même. Cette commande est en revanche sujette à des restrictions extrêmement fortes, seul un petit nombre d'objet peut être affecté.

Elle doit être entrée sous la forme **< cp objet destination >** , les paramètres étant les même que pour la commande **mv**.

**file :** Cette commande permet d'afficher la description d'un objet.

Elle doit être entrée sous la forme **< cat objet >** où "objet" représente l'objet que l'on souhaite examiner.

Cette commande est aussi utilisable sur les ennemis et les pnjs, mais nous verrons cela dans la section dédiée.

**rm :** Cette commande permet de détruire des objets, cependant des restrictions particulières sont appliquées (les objets principaux par exemple ne peuvent pas être détruits).

Elle doit être entrée sous la forme **< rm objet >** où “objet” représente l’objet à détruire.

**./ :** La commande particulière d’exécution. Cette commande permet d’activer un objet (par exemple une potion). La plupart du temps les objets ne seront pas affectés par cette commande.

Elle est entrée sous la forme **< ./objet >** où objet représente l’objet à activer.

Cette commande est aussi utilisée sur les pnjs et certaines entités.

## PNJs et dialogues

Au cours du jeu le joueur aura l’occasion d’interagir avec de nombreux pnjs. Certains seront d’une importance capitale et feront avancer l’histoire, d’autres peuvent donner des quêtes secondaires à accomplir afin d’obtenir des récompenses, quelques-uns pourront être combattus, enfin certains ne seront là que pour rendre le monde un peu plus vivants et n’offrir que quelques lignes de dialogue au joueur.

Le joueur peut interagir avec les différents pnjs via différentes commandes : (cf via un exécutable (./ mon_pnj)). Une description du pnj peut aussi être obtenue grâce à la commande cat.

Lors d’un dialogue, les touches “pages suivante” et “page précédente” peuvent être utilisées afin de passer d’un bloc de lignes à l’autre, ou bien les flèches haut et bas afin de lire ligne par ligne. Lorsqu’un choix est proposé, il faut entrer le numéro du choix puis appuyer sur la touche entrée.

Lorsqu’un pnj arrive à la fin de ses lignes de dialogue, une réplique finale indiquant bien que la conversation est terminée est répétée à chaque fois que le joueur interagit avec lui.

## Combats contre les processus

De nombreux ennemis menaceront le joueur tout le long du jeu, en particulier les zombies libérés par le puissant daemon. Si un ou plusieurs ennemis se situent dans un lieu lorsque le joueur y entre, le joueur est déporté dans une zone spécifique où le combat se déroulera. A l’issue du combat, si le joueur est toujours en vie, il est renvoyé dans la zone où il a rencontré les ennemis. Si le joueur est mort durant le combat, le jeu le renvoie à la dernière zone où il avait enregistré sa progression.

En combat, une entité est définie par son nom, son pid, ses points de vie, sa force, sa défense, sa vitesse et ses différentes commandes d'actions, ainsi que les récompenses en objets et expérience dans le cas d'un ennemi.

**Exemple :**

Zombie de nom Hollow
pid : 16284
20 hp  10 str  10 def  4 spe
Actions :  agripper  < cible >:  Inflige 100% * str dégâts
gémir :  Ne fait rien
Récompenses :  50 xp
10%  Essence d'Hollow

Un combat se déroule au tour par tour, selon la vitesse des différents intervenants. Les ennemis utilisent une commande par tour au hasard. Le joueur peut utiliser une commande par tour.

Afin de détruire un processus ennemi, il convient de trouver son pid et d'utiliser la commande **kill.** Cependant passé un cap, les ennemis seront capables de cacher leurs informations et il faudra vider leur réserve de vie avant d'espérer pouvoir trouver leur pid.

Le joueur peut combattre de différentes façon. Il peut utiliser les quelques commandes disponibles en combat (cependant ces commandes montreront rapidement leurs limites face aux ennemis les plus puissants) ou bien il peut utiliser des processus tiers pour l'aider.

Les commandes de bases ne seront utiles que pour vaincre les ennemis les plus faibles. En voici la liste :

**top :** Cette commande permet de lister tous les processus courants dans le combat et leur pid. Elle est utilisée en combinaison avec la commande kill.

**kill :** Cette commande permet de tuer un processus passé en paramètre.

Elle doit être utilisée sous la forme **< kill pid >**. Des restrictions sur les processus tuables peuvent être appliquées.

**cat :** Cette commande permet d'afficher les informations de base de l'ennemi choisi, c'est-à-dire son nom, ses points de vie, sa force, sa défense et sa vitesse.

Sinon, le joueur peut utiliser des "invocations". Concrètement, une invocation est un programme que l'on peut exécuter avec **./** et éventuellement des arguments. Par exemple, l'invocation **Horloger** peut être exécutée avec la commande **./horloger stop x** qui permet d'empêcher les ennemis d'agir durant plusieurs tours.

Pour lancer une invocation, le joueur doit posséder l'objet rattacher à l'invocation et l'activer (par exemple l'objet **Montre mystique** dans le cas de

l'**Horloger**). Un processus est alors créé et accompagne le joueur tout le long du combat.

Le joueur ne peut posséder que deux invocations actives à la fois. Pour lancer une nouvelle invocation si deux sont déjà actives, il doit d'abord en supprimer une à l'aide de la commande **kill**.

**Altérations d'état :**

Certaines actions entreprises par les personnages sont capables d'infliger des bonus ou des malus aux personnages, ce que l'on nomme des **altérations d'état.** Une altération d'état est un bonus ou malus qui affecte les statistiques d'un personnage de manière temporaire. L'altération d'état **stop** par exemple réduit la vitesse d'un personnage à zéro durant un certain laps de temps.

## Quêtes

Dès le début du jeu, un fichier **quest.txt** sera présent dans l'inventaire et recensera toutes les quêtes possibles du jeu. Afin de visualiser le contenu de ce fichier la commande **cat** sera utilisée. De plus le joueur pourra débloquer un alias afin d'y accéder directement.

Accomplir une quête permettra au joueur de gagner de l'XP afin de débloquer de nouvelles commandes, de nouveaux lieux, etc … ainsi que de l'argent.

Une quête est définie par son nom, sa description, ses conditions d'obtention et de validation, ainsi que ses récompenses.

Par exemple, en parlant à un étudiant au comportement étrange assis sur un banc devant le bâtiment A22, le joueur peut obtenir la quête "Commercial". Pour la valider, le joueur doit trouver une dose de cannabis dans l'Université et lui apporter. Une fois cette condition remplie, le joueur obtient une petite bourse ainsi que la commande **cp.**