TELECOM SudParis

Projet Informatique 1ère Année
PRO 3600

Bash In Time

HERMIER Chloé
SANSANÉ Maxime
SCHNEIDER Thomas
PUJAS Paul-Emmanuel
ROUQUETTE Gaston

Enseignant responsable : Michel SIMATIC

22/05/2023

Sommaire

Chapitre 1 : Introduction

Dans ce projet informatique, l'objectif est de créer un jeu d'enquête point and click pédagogique 2D sur Unity avec pour thème principal le langage de système d'exploitation bash (basé sur le début du cours de CSC 3102) .

L'élément principal du gameplay est un ordinateur rempli de fichiers (avec une structure type linux). Ce dernier comporte des énigmes obligeant le joueur à utiliser les connaissances sur le langage qu'il découvre au fil du jeu (50% bash, 50% énigmes physiques).

Le second objectif du jeu est de permettre d'apprendre ou de réviser les premières leçons du cours "Introduction aux systèmes d'exploitation", le premier étant d'offrir une expérience stimulante et plaisante au joueur.

Ce document a pour but d'expliquer, de manière précise et ordonnée les objectifs du projet informatique "Bash In Time". Dans le chapitre suivant cette introduction, nous détaillerons l'aspect créatif lié à la création du jeu. Le chapitre 3 quant à lui se concentrera sur les aspects plus techniques ainsi qu'aux difficultés que nous avons rencontrées lors du développement. Nous finirons par un post-mortem résumant ce que nous avons appris lors de ce projet, autant en tant qu'équipe qu'individus.

Chapitre 2 : Game Design Document (Cahier des charges)

2.1) Public cible

Le jeu se destine vers un public ayant déjà une certaine connaissance du monde de l'informatique sans pour autant en être un expert. Nous souhaitons notamment viser les personnes ayant déjà eu des expériences vidéoludiques et qui souhaitent s'initier au bash.

2.2) Synopsis

Le joueur se réveille amnésique dans une chambre en désordre, sombre avec beaucoup de matériel informatique. Il devra utiliser ce matériel afin de comprendre son environnement. Au bout d'un certain temps (entre 15 et 30 minutes, cela reste à définir) la chambre explose et le joueur retourne dans le temps, c'est-à-dire au début du jeu.

Le joueur se met dans la peau de Chell (en référence à l'héroïne de Portal et un jeu de mot évident avec le shell) qui évoluera dans cet espace clos, cherchant un moyen de sortir. Elle aura pour cela accès à un ordinateur disposant d'un terminal bash lui permettant de trouver des indices et à éventuellement s'échapper.

2.3) Gameplay

Ce jeu a pour objectif de faire apprendre des commandes basiques de Bash à l'utilisateur. Il n'y a pas à avoir de prérequis sur les bases de ce langage et de manière plus générale dans le fonctionnement d'un système d'exploitation. Ainsi, tout au long de sa progression, le joueur découvrira des commandes souvent utilisées en bash pour lui fournir une base d'apprentissage.

Exemples d'énigmes à résoudre :

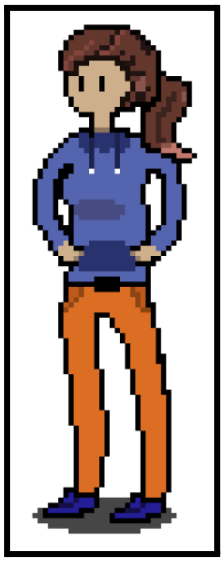
1) Le joueur trouve des images dans les fichiers du jeux, toutes les images sont différentes mais ont un point commun qui permet de les trouver
2) Le joueur imprime les images (dans le jeu) et les stockent dans son inventaire
3) En associant les différentes pièces le joueur découvre une image
4) Il obtient ainsi une image sur lequel il devine un code qui permet d'ouvrir un cadenas ou de déverrouiller une autre énigme

Les commandes qui sont à implémenter : **cd**, **ls**, **echo**, **read**, **rm**, **man**, **pwd**, **grep**, **mkdir**, **cat**

Les commandes que l'on aimerait implémenter : **touch**, **cp**, **find**, **flux**, **ssh**, **mv**, **./script.sh**, **sudo**, **head**, **tail**

2.4) Design

2.4.1) Le personnage : Chell



Le personnage principal du jeu se nomme Chell. C'est une jeune femme née le 8 juin 1989. Son apparence ressemble au personnage du même nom dans Portal.

Chell est une Doctoresse, elle travaille sur les fluctuations spatio-temporelles et comment les modifier. Suite à une découverte majeure dans son domaine, elle a attiré l'attention de l'Organisation, un groupe hors du temps et de l'espace qui a pour but de contrôler tous les axes temporels.

Son nom complet est Chell Bachelle.

Trait de caractère : précautionneuse, calme et responsable.

2.4.2) L'Organisation

On ne rencontre pas directement l'Organisation dans le jeu, mais elle est bien présente et surveille tout changement dans les axes temporels.

Chapitre 3 : Processus de développement

3.1) État de l'art des jeux similaires déjà existants

Avant de commencer notre projet, nous avons testé quelques jeux similaires au niveau du concept / des fonctionnalités afin de voir ce qui existe déjà et de nous inspirer. Parmi les jeux que nous avons testés, il y a eu notamment :

Elevator Saga[1] : L'un des deux seuls jeux qui n'a pas de rapport direct avec bash, il a pour but d'apprendre la programmation et plus particulièrement l'optimisation de flux.

Untrusted[2]: Surement le jeu plus éloigné du concept du projet mais avec de très fortes influences dans le domaine informatique/réseau qui pourraient nous être utiles pour l'ambiance du jeu.

Wargames[3]: Les joueurs apprennent les commandes de base telles que cd, ls et certaines options des commandes. Nous avons trouvé le jeu vite lassant, car il oblige le joueur à aller chercher des éléments précis de la documentation pour résoudre les énigmes.

Terminus Game[4]: Les joueurs apprennent à interagir avec le monde avec les commandes **ls** (voir le monde, liste des locations, items), **cd** (voyager) et **less** (interagir avec des items). Les joueurs doivent également apprendre des commandes supplémentaires telles que **mv**, **touch**, **rm**, **cp**, **man**, **pwd**, etc. Cependant, nous avons trouvé que le jeu est un peu frustrant car les commandes apprises ne sont pas disponibles partout.

Clmystery Game[5]: Les joueurs sont amenés à utiliser diverses commandes et opérations :
- **cat** (pour voir les indices et la solution ou afficher les fichiers)
- **cd**
- **grep**/**find** (recherche dans un document des chaîne de caractères; information pertinente);
- **grep + option -A, -B, -C** pour récupérer un bloc de ligne ;
- **head -n N ./ex | tail -n 1** (permet de choisir une ligne d'un doc);
- recherche d'une ligne précise dans un fichier (avec un pipe et un grep en plus);
- recherche d'une ligne commune entre plusieurs fichiers.

---

[1] Lien : https://play.elevatorsaga.com/
[2] Lien : https://store.steampowered.com/app/1502660/Untrusted/
[3] Lien : https://overthewire.org/wargames/bandit/
[4] Lien : https://github.com/mprat/terminus
[5] Lien : https://github.com/veltman/clmystery

Suite aux tests de ces jeux, nous avons remarqué que certains d'entre eux étaient vite lassants car ils ressemblaient à un enchaînement d'exercices comme par exemple Wargames qui nous demande constamment d'aller chercher dans la documentation. Nous avons donc décidé d'apporter une vraie dimension ludique à l'apprentissage du bash, en apportant directement les commandes dans le jeu au joueur, de manière simple, synthétique et progressive. Ainsi, le joueur ne sera pas perdu dès le début du jeu parmi des milliers de possibilités et de documentations à parcourir pour résoudre les énigmes. Il suffira alors pour le joueur de combiner les commandes qu'il a déjà apprises et celles qu'il découvre au fur et à mesure. Selon nous, cela permettrait donc de ne pas effrayer le joueur tout en proposant une dimension pédagogique.

Aussi, nous avons apprécié dans certains jeux comme Terminus Game la dualité entre l'espace des commandes et entre un monde visuel/physique. Nous voulons nous inspirer de ce concept avec la possibilité de connecter des objets physiques trouvés dans le monde du jeu avec l'ordinateur comme par exemple une clé USB que l'on pourra brancher sur l'ordinateur et explorer les fichiers, le tout dans le jeu.

Finalement, nous avons surtout noté que beaucoup de ces jeux étaient toujours exclusivement focalisés sur des énigmes fonctionnant avec bash (ou d'autres langages). C'est pourquoi nous pensons apporter une touche d'originalité avec un vrai mélange entre des énigmes utilisant le bash et d'autres mécaniques comme le voyage dans le temps ou des puzzles physiques présents dans des jeux vidéo plus classiques. Ainsi, nous pensons que cela permet d'amorcer un apprentissage progressif du bash chez un public qui a déjà l'habitude de résoudre des énigmes classiques dans les jeux.

3.2) Choix technologiques et de gestion

Afin de fluidifier le déroulement du développement, nous avons décidé de procéder avec une méthode agile, plus adaptée au développement de jeu.

Comme moteur, nous avons déjà choisi Unity, un moteur à la fois graphique et physique que nous avons déjà utilisé précédemment. Nous pouvons ainsi nous concentrer sur le reste du développement.

Le style graphique du jeu sera en pixel art pour des raisons de temps et de capacité.

Nous avons finalement décidé d'utiliser un véritable système de fichiers cachés pour les commandes, tout en les interceptant nous-mêmes sur Unity. Une

fausse racine sera mise en place pour utiliser uniquement des fichiers présents dans un dossier spécifique des données du jeu.

Cependant, nous avons souligné que le système peut poser des problèmes de sécurité en ce qui concerne l'interpréteur bash. Les chemins absolus ne posent pas de problèmes, mais les chemins relatifs ne doivent pas remonter au-delà du répertoire. Nous avons convenu d'examiner attentivement tous les ".." dans le système surtout en cas d'utilisation de certaines commandes comme **rm** par exemple. Nous pensons donc qu'il est important de retenir que même si les fichiers du jeu peuvent être réinitialisés sans problème, il faut nous assurer qu'il n'y ait pas de problèmes en dehors de ceux-ci.

Pour exécuter les commandes, nous avons besoin d'une console implémentant le langage bash. Celle-ci est présente par défaut sur les systèmes Linux, mais pas sur Windows. Or, par souci de simplicité, ayant tous les membres de l'équipe ayant Unity sur Windows, nous préférons développer sur Windows pour la praticité du développement. Nous avons donc pris la décision d'utiliser git bash pour Windows.

En effet, nous avons effectué quelques essais préliminaires d'exécution des commandes bash avec git bash, et celles-ci fonctionnent toutes (sauf man et sudo que l'on pourra prendre en charge nous-mêmes). De plus, le langage C# présent avec Unity permet d'utiliser des fonctionnalités systèmes comme écrire dans des fichiers ou lancer des processus. Ainsi, nous avons pu tester d'écrire une commande dans un fichier, et l'exécuter dans la console git bash le tout depuis Unity.

3.3) Aides au joueur

3.3.1) Pages du manuel

Le joueur doit pouvoir avoir accès à un manuel explicatif des commandes et qui résume les options disponibles pour chaque fonction. Dans un terminal Bash, cette fonction est remplie par la commande **man**. Le problème étant que ces pages de manuel sont trop longues et trop fournies pour une bonne expérience du joueur.

Le but fût de proposer au joueur une version raccourcie du **man** synthétisant les options usuelles et importantes. Pour cela, nous avons directement repris les manuels pouvant être trouvés sur Ubuntu, puis avons procédé à un long processus de relecture et de synthèse.

3.3.2) Palais mental

Le palais mental est un outil qui permet au joueur de sauvegarder des informations importantes. Il s'actualise progressivement au fur et à mesure que le joueur avance dans le jeu et trouve les solutions des énigmes.

Les données sont sauvegardées grâce aux PlayerPrefs de Unity de manière permanentes.

Grâce à un shader fait sur mesure et à un ensemble de particules, une ambiance mystérieuse est créée.

3.4) Ordonnancement des éléments à implémenter

Afin d'être sûr de fournir un jeu jouable à la fin de ce projet en tenant compte du temps de travail que l'on possède et de la relative imprévisibilité des différentes tâches lors du développement, il convient tout d'abord d'ordonnancer les éléments selon leur importance.

Importance	Description de la tâche
MUST	- Une application Unity comportant un menu simple et une scène de jeu exécutable sur un environnement Windows(avec git bash) et Linux - Un petit monde simple et la possibilité de se déplacer dans l'environnement - Un design simple du monde et du personnage - L'accès à un terminal Bash depuis le jeu - Une dizaine de commandes fonctionnelles tout en garantissant l'intégrité de l'ordinateur - La présence de 2-3 énigmes physiques simples, dont des codes pouvant être utilisées sur le terminal - Mécanique de retour dans le temps
SHOULD	- Possibilité d'interagir avec des objets en 3D pour les observer - Un design avancé du monde et du personnage, présence d'animation, celui-ci doit paraître dynamique avec du polish - Présence de quasiment toutes les commandes du GDD - Une dizaine d'énigmes physiques - Palais Mental pour aider le joueur - Interactions entre le terminal et les éléments physiques (pouvoir brancher des périphériques par exemple) - Cinématique de mort/retour au début - Présence d'un inventaire pour les objets physiques
COULD	- Implémenter des easter egg pour rendre l'expérience de jeu plus fun - Ajouter une seconde pièce dans le jeu - Contrôler d'autres ordinateurs en SSH - Système de clé publique, clé privée - Cinématique d'introduction, et de fin de jeu - Adapter le jeu pour qu'il aborde des notions plus avancées du cours de bash comme les tubes anonymes par exemple. - Menu très bien design - Possibilité de jouer sur MacOS
WOULDN'T	- Pas d'interface WebGL - Porter le jeu en Lisp - Porter le jeu en Blank - Pas Développements de scripts bash complets par l'utilisateur - Pas de concurrence et de processus

3.5) Estimation du plan de charge

En partant de cet ordonnancement, nous avons établi un plan de charge pour estimer le temps de travail de chaque tâche majeure. Aussi, nous ajouterons le temps de travail réel passé sur chacune de celles-ci après les avoir réalisées. De cette manière, nous pouvons planifier notre travail et apprendre à estimer plus précisément le temps nécessaire dans la réalisation de tâches dans le jeu vidéo après comparaison entre l'estimation et la réalité.

Description de l'activité	Charge en H	Charge réel
— Gestion de projet —	38h	60h
Réunions	20h	20h
Rédaction	18h	40h
--POLISH & BUILD--	30h	30h
Polish	20h	20h
Build final	10h	10h
— Fonctionnalités MUST —	91h	103h
Une application Unity comportant un menu simple et une scène de jeu exécutable sur un environnement Windows (avec git bash) et Linux	7h	5h
Un petit monde simple et la possibilité de se déplacer dans l'environnement	3h	5h
Un design simple du monde et du personnage	6h	10h
L'accès à un terminal Bash depuis le jeu (Programmation de l'UI)	10h	10h
Une dizaine de commandes fonctionnelles tout en garantissant l'intégrité de l'ordinateur	50h	50h
Interaction avec les objets (mise en avant visuelle)		8h
La présence de 2~3 énigmes physiques simples dont des codes pouvant être utilisées sur le terminal -> Système d'intéraction + Boîte de dialogue	10h	20h
Mécanique de retour dans le temps (Reset du jeu)	5h	3h
— Fonctionnalités SHOULD —	210h	106h
Possibilité de jouer sur MacOSs	5h	
Possibilité d'interagir avec des objets en 3D pour les observer	15h	
Un design avancé du monde et du personnage, présence d'animation, celui-ci doit paraître dynamique avec du polish	20h	20h
Présence de quasiment toutes les commandes du GDD	50h	50h
Une dizaine d'énigmes physique	50h	
Palais Mental pour aider le joueur	20h	30h
Interactions entre le terminal et les éléments physiques (pouvoir brancher des périphériques par exemple)	20h	

Cinématique de mort/retour au début	10h	6h
Présence d'un inventaire pour les objets physiques	20h	
— Fonctionnalités COULD —	150h	2h
Implémenter des easter egg pour rendre l'expérience de jeu plus fun	10h	2h
Ajouter une seconde pièce dans le jeu	20h	
Contrôler d'autres ordinateurs en SSH	20h	
Système de clé publique, clé privée	10h	
Cinématique d'introduction, et de fin de jeu	20h	
Adapter le jeu pour qu'il aborde des notions plus avancées du cours de bash.	50h	
Menu très bien design	20h	
-- TOTAL --	**539h**	**309h**

3.6) Builds intermédiaires

Dans cette partie, on discutera les résultats et fonctionnalités implémentées après chaque build intermédiaire du jeu ainsi qu'éventuellement les bugs à fixer ultérieurement. On y mettra aussi les grandes décisions au niveau de la structure principale du code de jeu.

Build intermédiaire du 16/03: Le premier build intermédiaire a été réalisé dans le cadre d'une réunion de projet avec notre tuteur. Ce build comporte le sprite de Chell, d'autres sprites d'objets ainsi qu'un décor de chambre en placeholder, créé par IA. En fonctionnalités, il comportait un shell bash quasi fonctionnel, avec encore quelques bugs à fixer et des fonctionnalités "quality of life" à implémenter pour l'améliorer. Les deux cadenas, les dialogues, les événements, l'horloge, les mouvements sont déjà implémentés et fonctionnels.

Les prochaines étapes qui ont été décidées après ce build ont été de réfléchir à l'histoire et aux énigmes à implémenter, écrire le **man** des commandes, faire le système de reset et le palais mental ainsi que l'animation des sprites.

Build intermédiaire du 17/05: Ce build intermédiaire est le dernier avant le build final. Il a été réalisé pour la dernière réunion de projet avec notre tuteur, et fait office de build quasi final, sans compter les corrections de bugs et implémentation de dernières fonctionnalités minimes à réaliser pour le build final.

La dernière étape avant le build final après ce build est principalement du polish, le but étant, après les playtests, de corriger tous les bugs rencontrés.

Build final du 22/05: Ce build a essentiellement consisté à la correction des bugs trouvés lors des playtests. On a également apporté des ajustement dans l'emplacement des objets dans la scène de jeu, et changé la durée d'un cycle.

3.7) Compte-rendus de playtest

3 playtest ont été effectués:

Un 3A Pierre Cornette, un élève international en master Charbel RIZKALLAH et un développeur en intelligence artificielle. Tous les trois possèdent des bases de bash, mais n'en sont pas expert, c'est un niveau comparable à un élève venant de finir les premiers cours et ayant moyennement suivi.

Plusieurs remarques ont été émises, notamment des bugs d'affichage, des problèmes de syntaxe ou de formulation des énigmes.

Pour le build final, tous les problèmes rencontrés lors des playtests ont été fixés.

Chapitre 4 : Post-mortem (Conclusion)

Dans ce chapitre, nous détaillerons dans les prochains livrables, les principaux choix, erreurs et succès que nous avons expérimenté. Ainsi, nous pourrons gagner en expérience dans nos prochains projets, en apprenant de nos réussites et de nos échecs.

Il est encore trop tôt pour faire un vrai Post-mortem, mais voici les conclusions que nous avons eues, suite à la finalisation de ce projet.

Points positifs :

- Nous nous sommes grandement améliorés dans notre utilisation de GIT, très peu de merge conflict ont eu lieu.

- Le projet est mieux structuré, les différentes fonctionnalités sont rangées dans des scripts clairs. Ce qui a rendu plus facile la modification du code, surtout à la fin du projet.

- Certains ont appris de nouvelles compétences comme Unity, les shaders ou ont approfondi leur connaissance en Bash (notamment Thomas qui a lu l'intégralité des manuels des fonctions utilisables).

Points négatifs:

- Le clavier ne semble toujours pas naturel, s'il était à réécrire, il faudrait ajouter un delay entre la première et la deuxième action pour les touches de combo (flèches et delete). Après un temps d'adaptation, cette manipulation devient plus naturelle.

- Le scénario est un peu maigre, trop peu de temps y a été consacré, et même si une base a été écrite, la traduire en énigmes amusantes et originales s'est révélée être plus complexe que prévu.

- Des problèmes d'optimisation, notamment avec la manière dont les commandes sont exécutées ; une solution serait de garder le shell ouvert en permanence pour éviter de le relancer. En effet, pour des raisons de protection de l'ordinateur exécutant le jeu, git bash est lancé pour chaque commande.