Erode

Plan de tests logiciels

Version 1.1

Historique des révisions

Date	Version	Description	Auteur
2017-02-17	1.0	Première ébauche du plan de tests	Mathieu O. Wilson, Alexandre Paquet, Odric Plamondon, Jonathan Sauvé
2017-04-15	1.1	Mise à jour finale	Jonathan Sauvé

Table des matières

1. Introduction

Nous présenterons, dans ce document, les tests qui seront réalisés, afin d'assurance la qualité, la stabilité et le fonctionnement de notre jeu vidéo, Erode. Tout d'abord, il sera question d'énumérer toutes les exigences qui devront être testées. Ensuite, nous proposerons une stratégie de test en identifiant les types de tests que nous réaliserons, ainsi que les outils que nous utiliserons pour la réalisation desdits tests. Par la suite, nous aborderons les ressources, humaines et matérielles, nécessaires à la réalisation des tests. Finalement, nous établirons un jalon de réalisation pour chaque test.

2. Exigences à tester

En ce qui a trait aux exigences à tester, elles ont été énumérées dans le "Game Development Document" (GDD). Pour un bref rappel, les exigences à tester sont toutes les mécaniques du gameplay, c'est-à-dire le fonctionnement de la plateforme avec son désagrégement, les différents ennemis (Hunter, Charger, Shooter et Serpent), tous les obstacles (comètes, astéroïdes, vents solaires, barrières de force, trous noirs et impulsions électromagnétiques), le personnage principale et toutes ses mécaniques, ainsi que les power-up (super marteau, ralenti/pause du temps, multiplicateur de score, gros marteau et super jet pack). De plus, ils faut aussi tester les différentes vues de l'interface utilisateur, ainsi que les mécanique de jeu, tels que le début de partie, les conditions de fin de partie, ainsi que le calcul de score.

3. Stratégie de test

3.1 Types de test

3.1.1 Tests de jeu (“Playtest”)

Objectif de test:	Vérifier que la nouvelle composante possède le comportement adéquat.
Technique:	*Playtesting*
Critère de complétion:	La nouvelle composante n’introduit pas d’erreur dans Unity, et ne modifie pas le comportement d’autres composantes à moins que ce ne soit prévu.
Considérations spéciales:	Les effets visuels ne seront pas considérés comme important avant la fin du projet.

Ce type de test sera effectué à chaque fois qu’une nouvelle composante (fonctionnalité) sera ajoutée au projet. Dans un premier temps, la composante sera ajoutée seule au jeu, c’est-à-dire que que tous les autres éléments pouvant interagir avec elle (excepté le personnage) seront désactivés. Si elle semble bien s’intégrer, alors nous ajouterons le reste des éléments et vérifirons ses interactions avec ces derniers. Une composante qui fonctionne correctement et qui n’affecte pas le comportement normal d’autres composantes sera alors ajoutée au jeu.

3.1.2 Tests d’interface usager

Objectif de test:	S’assurer que la navigation entre les différents menus soit fluide et instinctive.
Technique:	*Model-based testing - State-Based model* : Tous les états du GUI doivent être traversés au moins une fois.
Critère de complétion:	Tous les états sont traversés au moins une fois, les objets sont toujours dans le bon état et les données ne sont pas corrompues par un changement d’état invalide.
Considérations spéciales:	La navigation sera parfois testée par le client. Le reste des critères sera vérifié par l’équipe.

Chaque fois qu'un nouvel élément d'interface est ajouté au projet, nous devrons effectuer cette stratégie de test. Cette dernière consiste à s'assurer que la navigation entre les différents menus n'est pas troublée ou rendue incohérente à cause de l'ajout du nouvel élément. De plus, on vérifiera que l'affichage se fait correctement et s'intègre bien au jeu. Un élément qui respecte ses conditions sera alors montré au client lors de la prochaine rencontre, et son avis décidera si on doit le modifier ou l'intégrer définitivement au jeu.

3.2 Outils

Les outils suivants seront utilisés au sein de la discipline de test:

Type de test	Outil
Test de jeu	Débogueur de Visual Studio & Unity
Test d'interface usager	Débogueur de Visual Studio & Unity

Le débogueur de Visual Studio sera utilisé pour tout ce qui touche directement au code source (pointeur *null*, problème de logique, etc.) tandis que Unity sera utilisé pour vérifier l'intégration de la nouvelle composante au jeu.

4. Ressources

Nous n'utiliserons pas beaucoup de ressources pour la discipline de test. Du côté matériel, seul Visual Studio et Unity seront utilisés pour trouver et régler les bogues présents dans notre jeu. Une manette de Xbox One sera aussi utilisée pour les tests de jeu, ainsi que les tests d'interface usager. Pour ce qui est des ressources humaines, chacun des membres de l'équipe effectuera les tests pour la composante qu'il a développé. De plus, certaines parties des tests d'interface usager pourront être réalisées par le client, Laurent Tremblay.

4.1 Équipe de test

Rôle	Membre de l'équipe	Responsabilités
Playtester	Interne : Toute l'équipe Externe : -	- Faire fonctionner la nouvelle composante dans l'environnement actuel de jeu - Régler les erreurs et avertissements lancés par Unity - Intégrer la nouvelle composante
GUI tester	Interne : Jonathan Externe : Laurent	- La navigation correspond aux exigences - La boucle de jeu n'est jamais placée dans un état invalide

4.2 Système

La discipline de test s'effectue sous Unity et Visual Studio. La version de l'environnement est 5.5.0f3 pour Unity. Le système d'exploitation utilisé dans notre cas est Windows 10.

5. Jalons du projet

Jalon	Effort (H-P)	Date de début	Date de fin
Documentation initiale	0	9 janvier 2017	27 janvier 2017
Prototype et documentation	12	28 janvier 2017	20 février 2017
Beta	20	21 février 2017	5 avril 2017
Produit final	40	6 avril 2017	18 avril 2017