Aps’Game

Spécifications

**N°2020\_XXXXX**

Table des Matières

1 Validité du Document 3

1.1 Historique des Révisions 3

1.2 Approbation 3

1.3 Destination 3

1.4 Documents de référence 3

1.5 Acronymes 4

2 Objet du document 5

3 Description générale 6

3.1 Contexte 6

3.2 Domaine d’application 6

4 SPECIFICATIONS FONCTIONNELLE 7

4.1 Splash screen 7

4.2 Module de mise à jour 7

4.3 Module de connexion/inscription 7

4.4 Menu du jeu 7

4.5 Compte utilisateur 8

4.6 Scores 8

4.7 Sauvegarde 8

4.8 Partie 9

5 SPECIFICATIONS TECHNIQUES 10

5.1 Choix technologique 10

5.2 Protocoles de communication 10

5.3 Gestion de configuration 10

5.4 Maintenance et évolutions 10

5.5 Validation 11

6 Planning 12

# Validité du Document

## Historique des Révisions

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Motif de l’évolution** |
| 01 | 19/04/2020 | ***Création*** |
| 02 |  |  |
| 03 |  |  |
| 04 |  |  |
| 05 |  |  |

## Approbation

|  |  |  |
| --- | --- | --- |
| **Rédigé par** | **Vérifié par** | **Approuvé par** |
| Alexandre RIFFAUD  ***Ingénieur Etude et développement*** |  |  |

## Destination

|  |
| --- |
| **Destinataires** |
| **ESIMED**  Henri Michelon |

## Documents de référence

Les documents ci-dessous ont servi à élaborer la proposition.

|  |  |  |
| --- | --- | --- |
| **#** | **Libellé** | **Référence** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Dans la suite des présentes spécifications, Apside pourra faire référence à ces documents grâce aux repères entre crochets.

## Acronymes

| **Libellé** | **Signification** |
| --- | --- |
| VR | Virtual Reality. |
| API | Application Programming Interface. |
| IHM | Interface Homme Machine. |
| Qt | API orienté objet et développée en C++, framework de développement. |
| QML | QML est un langage de balisage d'interface utilisateur. Il s'agit d'un langage déclaratif pour la conception d'applications centrées sur l'interface utilisateur. |
| QtQuick | Framework permettant d’écrire des applications en QML |
| MinGW | Minimalist GNU for Windows |
| Clang | Clang est un [compilateur](https://fr.wikipedia.org/wiki/Compilateur) pour les [langages de programmation](https://fr.wikipedia.org/wiki/Langage_de_programmation) [C](https://fr.wikipedia.org/wiki/C_(langage)), [C++](https://fr.wikipedia.org/wiki/C%2B%2B) et [Objective-C](https://fr.wikipedia.org/wiki/Objective-C). |
|  |  |

# Objet du document

Ce document a pour but de détailler les spécifications liées au projet de développement d’un jeu vidéo cross-plateforme pour ESIMED.

Chaque exigence est classifiée et indexée afin d’assurer une traçabilité et une couverture pour le développement de l’outil de rejeu.

Il fait suite à la réunion organisée le lundi 19 avril 2020 dans le cadre du projet de fin de cursus.

# Description générale

## Contexte

L’idée de développer un jeu vidéo en collaboration avec APSIDE est née il y a deux ans lors de la création du plateau industriel de l’agence d’Aix-en-Provence. Le but était de réaliser un projet commun avec l’équipe socle afin d’apprendre à travailler en équipe. L’évolution rapide du plateau et le développement de la suite logicielle pour les services d’APSIDE n’a pas laisser assez de temps pour réaliser ce projet.

Pour concrétiser les trois années d’alternance au sein d’APSIDE, j’ai décider de reprendre ce projet de le développer dans le cadre de mon dernier module à ESIMED.

Le jeu vidéo devait au départ être en VR (une représentation de l’agence d’Aix-en-Provence). APSIDE ne pouvant garantir la fourniture du matériel à temps et la disponibilité de celui-ci durant les périodes de développement. Il a été décider de réaliser un jeu vidéo de type « Beat’em All » en QML avec la librairie Felgo.

## Domaine d’application

Ce document constitue la réponse d’Alexandre RIFFAUD concernant le développement d’un jeu vidéo cross-plateforme pour le projet de fin d’année.

Ce document est constitué de notre dossier de spécifications fonctionnelles et techniques. Nous pensons que cette proposition répond aux objectifs exprimés dans la consultation et restons à votre entière disposition pour apporter tout complément d’information concernant notre démarche.

# SPECIFICATIONS FONCTIONNELLE

Seules les fonctionnalités listées par les exigences ci-dessous seront réalisées.

## Splash screen

1-001 : L’application doit démarrer par un splash screen

1-002 : Le splash screen est composé du logo et de l’auteur du jeu.

## Module de mise à jour

2-001 : La mise à jour du jeu doit être semi-transparente pour l’utilisateur

2-002 : Lorsqu’une mise à jour est disponible au lancement du jeu, une barre de progression et un message doit signaler à l’utilisateur qu’une mise à jour est en cours.

2-003 : La mise à jour doit se faire via l’API

## Module de connexion/inscription

3-001 : Chaque utilisateur a son propre compte

3-002 : Un nouvel utilisateur peut créer son compte

3-003 : Une fois le chargement/mise à jour terminée l’utilisateur doit s’authentifier ou s’inscrire s’il ne possède pas de compte

3-004 : L’utilisateur reste connecté le temps de la session de jeu. La déconnexion se fait uniquement lorsqu’il quitte le jeu

3-005 : Pas de connexion via Facebook ou Google

## Menu du jeu

4-001 : Le menu du jeu a un background simple

4-002 : Le logo du jeu doit être présent

4-003 : Le menu est composé de :

* Commencer une partie
* Reprendre une partie
* Compte
* Scores
* Crédits

4-004 : Aucune animation dans le menu

## Compte utilisateur

5-001 : L’utilisateur devra renseigner sa date de naissance, son prénom, son pseudo, son adresse mail

5-002 : L’utilisateur peut changer son pseudo, et son adresse mail via le menu compte

5-003 : Pas de confirmation de l’adresse mail

5-004 : L’utilisateur peut choisir la langue du jeu :

* Anglais
* Français

## Scores

6-001 : L’utilisateur a la possibilité d’enregistrer son score à la fin d’une partie

6-002 : Chaque score enregistré de chaque joueur est public

6-003 : Les scores sont consultables via le menu Scores

6-004 : L’utilisateur peut filtrer les scores pour ne voir que les siens

6-005 : Deux affichages sont disponibles dans la vue Score :

* Par niveau
* Total

6-006 : Les scores sont calculés en fonction du temps

## Sauvegarde

7-001 : Un joueur peut sauvegarder sa progression de niveau en niveau

7-002 : Un joueur peut reprendre sa partie au niveau où il a sauvegardé

## Partie

8-001 : Un joueur peut commencer une nouvelle partie

8-002 : Un joueur peut avancer, reculer, sauter

8-003 : Un joueur ne peut pas plus reculer que ce que la caméra lui permet

8-004 : Le joueur possède une jauge de vie à 100

8-005 : Un joueur perd 20 points de vie s’il est touché par un ennemi

8-006 : Un joueur peut ramasser un objet qui lui donne 50 points de vie

8-007 : Un joueur possède deux armes :

* Un fusil d’assaut
* Un couteau

8-008 : Les ennemis peuvent se déplacer

8-009 : Il y a un nombre d’ennemi définit par niveau

8-010 : Chaque niveau est chronométré

8-011 : L’utilisateur peut mettre en pause la partie

8-012 : L’utilisateur peut quitter la partie et revenir au menu du jeu

8-013 : Le jeu se passe uniquement sur un plan horizontal

# SPECIFICATIONS TECHNIQUES

## Choix technologique

9-001 : Le langage de développement utilisé sera le C++, en version C++17 en utilisant les compilateurs MinGW 8.1 et Clang

9-002 : Le développement de L’IHM sera réalisé en QtQuick 2 avec la librairie Felgo

9-003 : Les environnement cible sont Windows 10 et Android 8.1

9-004 : L’API sera développée en JAVA EE avec le framework Spring boot

9-005 : L’API sera déployée sur un serveur distant

9-006 : Sur Windows 10, le jeu aura un installateur pour le déploiement

9-007 : Sur Android, le jeu sera déployé sur l’App store de Google

9-008 : Le projet suivra l’arborescence type d’un projet APSIDE : répertoire développement, documentation etc…

## Protocoles de communication

10-001 : Le protocole de communication utilisé sera le TCP

## Gestion de configuration

11-001 : L’Aps’Game sera mis en gestion de configuration SVN sur github.

## Maintenance et évolutions

12-001 : L’architecture doit être simple, modulable, évolutive et suivre le principe de la Programmation Orienté Objet

12-002 : Le code doit être documenté au format Doxygen

12-003 : L’Aps’Game doit être fiable et stable : pas de crash, pas de latence

## Validation

13-001 : Pour chaque exigence « testable », un test unitaire sera rédigé

13-002 : La validation sera faite en suivant des tests courts et longs

# Planning

Le planning présenté ci-dessous est à titre informatif.

|  |  |
| --- | --- |
| 19/04/2021 au 23/04/2021 | Rédaction des spécifications et de la conception |
| 26/04/2021 au 30/04/2021 | Monté en compétence sur Felgo |
| 03/04/2021 au 07/05/2021 | Mise en place du projet (github, API, QML) |
| 10/05/2021 au 14/05/2021 | Réalisation l’API et de l’application (hors partie jeu) |
| 07/06/2021 au 25/06/2021 | Réalisation de la partie jeu |
| 28/06/2021 au 02/07/2021 | Tests et déploiement |