PFE

N° d'ordre : PFE-INFO-2014-29

**Projet de fin d'études** présenté par

*Élève ingénieur de l'INSA de Rennes*
*Spécialité INFO*
*Année scolaire 2013-2014*

Amandine LE CAHAIN

# Développement d'une application web dite « second écran » pour interagir avec un jeu vidéo.

**Lieu du Projet de Fin d'Études**
Asobo Studio
**Tuteur du Projet de Fin d'Études**
Kévin CHOTEAU
**Correspondant pédagogique INSA**
Valérie GOURANTON

**PFE soutenu le 20/06/2014**

ASOBO STUDIO

Amandine LE CAHAIN

INSA de Rennes – 5INFO – 2014

# INTRODUCTION

En tant que passionnée de jeux vidéo, j'ai réalisé mon stage de fin d'études en informatique à **Asobo Studio**, un développeur indépendant de jeux vidéo pour PC et consoles. J'ai choisi ce stage dans l'opportunité de faire de mon métier **une passion au quotidien**. A l'origine à la recherche d'un stage en événementiel dans le jeu vidéo, j'ai accepté la proposition d'Asobo Studio pour mon deuxième intérêt, le **développement web**. Sur le temps de ces six mois de stages, j'ai été sollicitée sur différents besoins en développement web de l'entreprise. Ce rapport de stage de fin d'études s'efforcera au mieux de donner à son lecteur une idée précise de ma mission au sein d'Asobo pour les quatre premiers mois de stage. Dans une première partie, je présenterai **l'entreprise** ainsi que les différents corps de métier propres à l'industrie du jeu vidéo. Dans un second temps, j'aborderai **la mission** principale de mon stage, sa réalisation ainsi que les difficultés rencontrées et leurs solutions. Finalement, la dernière partie fera **le bilan** sur ces quatre premiers mois de stage et énoncera les futurs projets en développement web de l'entreprise.

# REMERCIEMENTS

Je remercie particulièrement mon maître de stage **Kévin CHOTEAU**, Lead Game Designer, pour ses conseils et son attention tout au long de mon stage.

Je remercie **toute l'équipe** du projet sur lequel j'ai travaillé pour leur bonne humeur, l'aide qu'ils m'ont apportée et le partage de leurs connaissances.

Je remercie également **Julie REGENTETE, Kévin PICQ et Sophie DOSIERE** pour leur accueil au sein de l'entreprise et leur attention toute particulière au bien-être de leurs stagiaires.

Finalement, je remercie **l'ensemble des employés d'Asobo Studio** pour leurs sourires et le partage de leur passion !

# TABLE DES MATIÈRES

# PRÉSENTATION DE L'ENTREPRISE

## 1. L'histoire d'Asobo Studio

**Asobo Studio** est un studio de développement français de jeux vidéo basé à Bordeaux et employant actuellement plus de 80 passionnés.

Reconnu pour la qualité de ses productions, Asobo Studio compte désormais **13 productions** à ce jour réparties en jeux à licences, jeux d'action et d'aventure et jeux de course.

L'histoire des 10 premières années d'Asobo est résumée dans la frise chronologique suivante:

| | |
|---|---|
| 2002 | • Création de l'entreprise par 12 associés passionnés de jeux<br>• **"Super Farm",** jeu sur PS2 |
| 2003 - 2005 | • Développement de 3 jeux en partenariats avec plusieurs grands éditeurs<br>• **"La Momie", "CT Special Forces", "Garfield 2"** |
| 2006 - 2008 | • Adaptation du film **"Ratatouille"** et du film **"Wall-E"**, première collaboration avec les studios Pixar<br>• Nouveaux locaux et 50 salariés |
| 2009 | • Développement du jeu de course **"Fuel"** reconnu pour son univers gigantesque<br>• Obtention du titre de **"Best Independant Developper"** au *Develop Industry Excellence Awards*<br>• Adaptation du film **"Là-haut"** |
| 2010 - 2012 | • Développement de **"Toy Story 3"** et **"Racket Sports"** sur PS3 et Wii pour Ubisoft<br>• Développement de **"Kinect Héros: Une aventure Disney - Pixar"** pour Microsoft |

**Figure 1: Les 10 premières années d'Asobo Studio**

*De façon plus anecdotique, le terme « Asobo » signifie « Jouons » en Japonais.*

# 2. Les corps de métiers

Asobo Studio compte actuellement plusieurs projets en cours de réalisation. Pour chacun de ces projets, les différents corps de métier suivant sont représentés:

### PRODUCER

Les **producteurs** ou « Producers » s'assurent du bon fonctionnement de l'équipe, du contact avec le client, du respect du planning et de la qualité du produit final.

### DESIGNER

Les **designers** ou « Game Designers » définissent et diffusent à l'équipe le contenu du jeu comme les interactions, les personnages, le scénario, le décor, les règles... Ils sont responsables de l'expérience finale du joueur.

### GRAPHISTE

Les **graphistes** ou « Level Builders » modélisent et créent les décors et plus généralement l'environnement du jeu. Ils sont également en charge de toute la conception graphique telle que l'interface utilisateur et les personnages du jeu.

### ANIMATEUR

Les **animateurs** donnent vie aux personnages du jeu préalablement créés par les « character artists ». Ils créent également les cinématiques du jeu.

### PROGRAMMEUR GAMEPLAY

Les programmeurs dits **« Gameplay »** implémentent tout ce qui a rapport à l'interaction dans le jeu comme le déplacement du joueur, l'intelligence artificielle des personnages et la logique du jeu dans son ensemble.

### PROGRAMMEUR ENGINE

Les programmeurs dits **« Engine »** conçoivent et développent les outils nécessaires à la production du jeu. Ils implémentent la technologie nécessaire pour utiliser les données produites.

### TESTEUR

Les **testeurs** s'assurent du bon fonctionnement du jeu comme défini par les designers. Ils s'assurent de la qualité du contenu et des versions intermédiaires livrées au client.

Le développement web ne fait donc pas parti des corps de métier principaux d'un projet de jeu vidéo. Avec l'évolution des tendances actuelles du web pour le jeu vidéo, j'ai intégré Asobo Studio pour l'ébauche d'un premier projet associant jeu vidéo et développement web. Mon statut en tant que programmeuse web se situe donc entre le statut de designer et de programmeur gameplay.

## 3. La clause de confidentialité

Pour ce stage de six mois au sein d'Asobo Studio, j'ai signé un **accord de confidentialité** m'obligeant à ne divulguer aucun détail des différents projets sur lesquels je suis intervenue. Asobo Studio travaille pour plusieurs grands éditeurs de l'industrie du jeu vidéo qui imposent une discrétion sur les projets confiés.

Le projet principal sur lequel j'ai été amenée à travailler n'a pas encore été rendu public. Selon cet accord de confidentialité, je ne suis pas en capacité de renseigner le lecteur sur le nom des jeux sur lesquels j'ai travaillé. Je ne pourrais également pas divulguer de détails précis sur mon travail en tant que développeuse web comme les designs des applications.

Néanmoins, je suis en capacité d'expliquer mon travail de façon générale pendant ces quatre premiers mois de stage ainsi que de renseigner les détails techniques du développement.

## 4. Ma mission au sein d'Asobo

En tant que « web developper » au sein d'Asobo Studio, j'ai été assignée à un projet en cours pour le développement d'une application web dite **« second écran »**. Cette application intervient en parallèle du jeu vidéo développé par l'entreprise et apporte de nouvelles fonctionnalités et opportunités au joueur. Dans la suite de ce rapport, je nommerai ce projet sur lequel j'ai passé la majorité de mon temps **« Projet Companion »**. Pour ce projet, j'ai donc été en charge du design de l'application et de sa réalisation complète. Ce projet étant une première pour l'entreprise, j'ai été autonome pour le développement de cette application.

Le « Projet Companion » étant mon projet principal, il a représenté la quasi-totalité de mon temps pour ces quatre derniers mois. Néanmoins, Asobo Studio ayant certains autres besoins en développement web, je suis également intervenue par moment sur un autre projet pour la réalisation d'un site vitrine avec implémentation d'une newsletter. J'aborderai rapidement mon travail sur cet autre projet puisqu'il ne représente pas un pourcentage conséquent de mon travail. Néanmoins, ce projet deviendra par la suite le projet principal sur lequel je serai amenée à travailler et il était à l'origine le projet définit dans ma convention de stage.

# DEVELOPPEMENT D'UNE APPLICATION "SECOND ECRAN"

## 1. Les « Companion Applications » ou « Second Screens »

### LES COMPANION APPS, UNE NOUVELLE TENDANCE

Les applications dites compagnonnes ou **« Companion Apps »**[1] sont des applications secondaires qui accompagnent l'utilisateur dans l'utilisation d'une application primaire, le plus souvent un jeu vidéo.

Cette application ne doit pas remplacer le jeu mais doit **enrichir l'expérience du joueur** avec de nouvelles fonctionnalités et opportunités. Le joueur pourra avoir accès plus rapidement et facilement à de nombreuses informations.

L'utilisation des « Companion Apps » reste encore une nouveauté dans le monde du jeu vidéo sur console. Certains jeux très connus proposent désormais des « Companion App » où le joueur pourra par exemple, consulter son profil, consulter une carte, visualiser des informations stratégiques et même interagir avec le jeu.


**Figure 2: Screenshot de la « Companion App » du jeu Assassin's Creed® IV**

[1] Le terme « Companion App » a été introduit en premier par Microsoft. Sony fait référence à son application compagnonne par le terme « PlayStation® App » et « Second Screen ».

Les « Companion Apps » ont pour but d'être disponibles gratuitement et accessibles à n'importe quel joueur possédant au minimum un smartphone. Elles se veulent donc multiplateformes ou **« cross-platform »** afin d'être téléchargeables sur tous types d'appareils indépendamment de la marque et du système d'exploitation.

### DES APPLICATIONS WEB CROSS-PLATFORM

Pour répondre à ce besoin d'indépendance de l'application vis à vis de la plateforme, quoi de mieux qu'une **application web**. Toutes les tablettes et smartphones possèdent leur propre navigateur internet ainsi que l'interpréteur JavaScript qui leur est associé. Dans cette volonté d'être accessibles à tous les utilisateurs, les « Companion Apps » ne sont ni plus ni moins des applications web communiquant en direct avec le jeu sur console.

La réalisation d'une application web se base sur l'utilisation d'HTML5/CSS3 pour le design et de JavaScript pour la dynamisation et le traitement des données avant leur affichage. Malgré l'intérêt majeur du web résidant en sa compatibilité sur tous les appareils, la difficulté de création d'une telle application vient du **manque de standardisation** dont souffre le web de nos jours[2].

La réalisation d'une application web représente certains avantages et inconvénients qu'il est possible de résumer rapidement selon le comparatif suivant:

| Avantages | Inconvénients |
|---|---|
| **Compatibilité** | Manque de **standardisation** |
| **Indépendance** vis-à-vis du système | **Différences** entre les différents rendus graphiques des navigateurs |
| Directement **disponible**, pas d'installation particulière | **Performances** liées à l'appareil |
| Possibilité de s'adapter à tous les **formats** d'écrans | |
| **Minimisation** du coût de développement (une application pour tous les appareils) | |

**Table 1: Avantages et inconvénients d'une application web en comparaison à une application « standalone »**

[2] Le **World Wide Web Consortium (W3C)** est un organisme de normalisation à but non lucratif s'efforçant de standardiser le web dans son ensemble. Néanmoins, chaque navigateur possédant son propre interpréteur JavaScript, de nombreuses différences apparaissent et ne sont standardisées que plus tard, après validation de leur intérêt.

## PLAYSTATION APP ET SMARTGLASS

Actuellement sur le marché, seules deux consoles permettent l'utilisation d'une application en second écran.


Figure 4: Aperçu de l'application « PlayStation®App »

**Sony**, avec l'arrivée de la **PlayStation 4**, offre la possibilité d'utiliser son application **« PlayStation®App »** et notamment la fonctionnalité « Second Screen » qui lui est associée. L'application web est hébergée sur la PS4 et mise à disposition avec le jeu. Le smartphone ou la tablette se connecte au même réseau local que la console et peut avoir accès à l'application relative au jeu.

**Microsoft** met aussi à disposition son application **« SmartGlass »** permettant de se connecter à la Xbox One avec un smartphone ou une tablette. La **« Companion App »** relative à un jeu est dans ce cas hébergée sur un serveur dédié par Microsoft et accessible via internet.

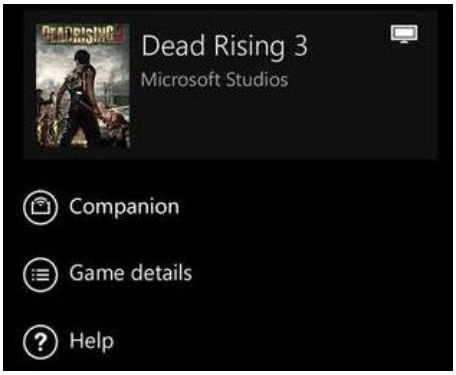



Figure 3: Aperçu de l'application "SmartGlass"

# 2. Une « Companion App » et bien plus

## PLUS QU'UN SECOND ECRAN, UNE MANETTE...

Dans le cadre de mon stage, la notion d'application compagnonne a été enrichie et l'application vise à prendre une place plus conséquente dans l'expérience du joueur. Tandis que le terme « Companion App » désigne une application secondaire qui apporte de nouvelles fonctionnalités sans être indispensable, l'application pour le « Projet Companion » a pour but de complètement **remplacer l'utilisation de la manette**.

Le joueur serait donc en capacité de connecter son **smartphone** ou sa **tablette** à la console et de jouer avec son appareil sans même avoir besoin d'une manette. Plus qu'un second écran utilisé pour l'affichage d'informations, l'application doit principalement relayer toutes les actions disponibles au joueur à tout moment de la partie.

La nouveauté réside donc dans le fait que l'application devient une manette à part entière tout en permettant également de consulter des informations qui ne sont normalement pas disponible au joueur à tout moment. Par exemple, dans un jeu au tour par tour, le joueur dont c'est le tour peut consulter ses informations et dans ce cas, sur console, il va monopoliser l'écran de la télévision. La « Companion App » permet alors aux autres joueurs qui attendent leur tour de consulter également leurs propres informations. Plus qu'une manette, l'application ajoute donc une toute autre dimension stratégique puisque le joueur qui attend son tour peut désormais préparer ses prochaines actions.

L'utilisation d'une « Companion App » en tant que manette permet également de **ne pas limiter le nombre de joueurs au nombre de manettes**. Chaque personne disposant au minimum d'un smartphone pourra donc rejoindre le jeu et accompagner les joueurs en possession d'une manette.

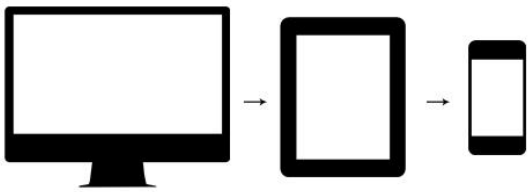
## SUIVRE LE « FLOW » DU JEU PRINCIPAL

L'implémentation d'une telle « Companion App » demande également un suivi rigoureux du **« flow »** du jeu. Le « flow » d'un jeu représente l'enchainement des différentes actions ainsi que les actions disponibles ou verrouillées à chaque instant. Le « flow » du jeu définit donc tout ce qu'un joueur est capable de réaliser à chaque instant et les conséquences de ses actions. Malgré l'ajout de nombreuses informations sur la « Companion App », celle-ci ne doit pas être capable faire « plus » que le jeu en termes d'actions. Elle doit donc être limitée par le jeu soit en recevant des informations permettant de bloquer ou débloquer certaines zones de l'application, soit en faisant en sorte que le jeu n'interprète que les actions disponibles. Une telle application demande un échange constant de données entre le jeu et le client web.

# 3. Une application "responsive" respectant le « mobile first design »

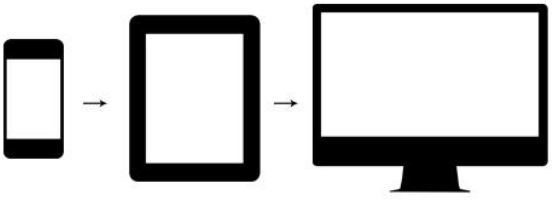
Une application dite **« responsive »** adapte son contenu à la taille d'écran de l'utilisateur afin de toujours offrir **une expérience de consultation optimale** facilitant la lecture et la navigation. Une « Companion App » doit pouvoir s'afficher sur tous les appareils disposant d'un navigateur et évidemment avoir un rendu graphique correct sur chacun de ces appareils. Le développement d'une application « responsive » requiert d'imaginer le design compatible à ce que les éléments puissent se placer différemment en fonction de la résolution. Au début de mon stage, j'ai effectué une première phase de design de l'application afin d'imaginer au mieux son apparence finale et le confort qu'aura le joueur à l'utiliser.

Accompagnant souvent le terme « responsive », le **« Mobile First Design »** est récemment apparu pour désigner les projets où la phase de design d'une application commence par les plus petits écrans (les smartphones) et non plus les grandes résolutions (écran d'ordinateur). Ce processus de design des applications web est apparu dans le but de remédier aux mauvaises adaptations mobiles de sites web déjà existants.



Le processus de design partant des plus grandes résolutions pour arriver au plus petite est parfois appelé **« Graceful Degradation »**. Ce cas de figure apparaît lorsqu'un site a été conçu pour des navigateurs d'ordinateur et utilise souvent des technologies et des performances à la hauteur de la plateforme primaire. Lorsque l'adaptation aux plus petits formats est réalisée, le site perd de nombreuses fonctionnalités ainsi que du contenu.

Le **« Progressive Enhancement »** ou « Mobile First Design » optimise l'application web en premier lieu pour les plus petits appareils. L'application sera donc composée de tous les éléments primordiaux pour le plus petit format. L'expérience utilisateur est alors enrichie et améliorée en passant sur de plus puissantes plateforme mais elle n'est jamais réduite.



# 4. La conception et l'implémentation

## LE DESIGN : HTML5/CSS3/LESS

Comme tout site web classique, le développement de l'interface de l'application se fait en **« markup »** HTML5. Désormais, chaque navigateur supporte la dernière version d'HTML qui intègre beaucoup plus de possibilités et différents types de balisages. CSS3 permet la définition de feuilles de style pour les différentes pages de l'application. CSS3 étant un langage **peu flexible et très répétitif**, j'ai choisi d'utiliser **LESS** pour automatiser et simplifier le code.

LESS étend le CSS avec un **comportement dynamique**, utilisant des variables, des classes abstraites, des opérations et des fonctions. LESS se présente sous la forme d'une librairie JavaScript et fonctionne aussi bien côté client que côté serveur avec l'utilisation de **Node.js**[3].

LESS

```
.button {
  &-ok {
    background-image: url("ok.png");
  }
  &-cancel {
    background-image: url("cancel.png");
  }

  &-custom {
    background-image: url("custom.png");
  }
}
```

CSS

```
.button-ok {
  background-image: url("ok.png");
}

.button-cancel {
  background-image: url("cancel.png");
}

.button-custom {
  background-image: url("custom.png");
}
```

**Figure 5: Utilisation du paramètre parent avec LESS**

Dans cet exemple, LESS permet d'automatiser directement la répétition de propriétés CSS ayant une partie de leur nom en commun.

## LE BACKEND OU « CODE BEHIND » : JAVASCRIPT

La réalisation de l'application est entièrement basée sur le **langage objet JavaScript**. JavaScript s'exécute côté client et permet l'affichage et le traitement dynamique de données.

Dans le cadre de ce projet, il nous était impossible de configurer un serveur web pour la préparation et la gestion des données. Le serveur étant directement hébergé sur la console pour la PlayStation 4, nous n'avions aucun moyen d'y installer nos propres outils. Le traitement des données est donc géré par le jeu et l'application cliente en JavaScript s'occupe de les traiter pour les afficher.

[3] Node.js est la plateforme construite depuis l'interpréteur JavaScript de Chrome permettant d'utiliser JavaScript côté serveur.

### DESSINER EN WEB, C'EST POSSIBLE !

L'application réalisée pour ce jeu utilise des technologies récentes du web comme notamment les **« Canvas »** qui sont apparus avec HTML5. La balise « Canvas » permet d'afficher des éléments grâce au code JavaScript.

HTML

```
<canvas id="myCanvas" width="200"
height="100"></canvas>
```

JAVASCRIPT

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
```

**Figure 6: Création d'un élément "Canvas" et dessin d'un rectangle rouge dans cet élément**

Les « canvas » peuvent également afficher des formes plus complexes, des dégradés, des ombres... Ils sont désormais très utilisés dans la réalisation de jeux en web. Un inconvénient néanmoins considérable concerne **l'historique des formes dessinées**. En dessinant une forme sur un « canvas », il n'est pas possible de la changer sans tout d'abord l'effacer puis la redessiner. Le « canvas » peut être vu comme un tableau blanc : les dessins se chevauchent les uns les autres, et pour un changer un, la totalité doit être effacée puis redessinée.

Pour pallier à ce problème, une autre technologie web a fait surface, le **« SVG »**[4]. Contrairement aux « canvas », le « SVG » garde en mémoire tous les éléments qui ont été inclus dans le dessin. Il permet donc d'y accéder à tout moment et de les éditer sans avoir à redessiner l'ensemble de la zone concernée. Néanmoins, l'utilisation du SVG représente une perte lourde en performance, puisque chaque élément est injecté au **DOM**[5]. Le traitement du DOM coûte énormément de performance à une application, et bien qu'il existe un populaire framework pour le gérer plus facilement (jQuery), l'insertion et la suppression d'éléments sont à éviter. Pour l'application compagnonne, j'ai choisi d'utiliser des « canvas » tout en gardant une trace des éléments dessinés grâce à leurs coordonnées.

### LE CODE OBJET

JavaScript est avant tout, il ne faut pas l'oublier, un langage orienté objet. Il est également définit de **langage orienté objet à prototype**, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe.

[4] **« Scalable Vector Graphics »** est un format de données conçu pour décrire des ensembles de graphiques vectoriels et basé sur XML. (Source : Wikipédia)
[5] DOM ou **« Document Object Model »** représente l'ensemble de la structure HTML d'un site web.

```
JAVASCRIPT

function Square(x, y, w, h) {
  this.x = x || 0;
  this.y = y || 0;
  this.width = w || 0;
  this.height = h || 0;
}

Square.prototype.draw(canvasContext) {
  canvasContext.fillRect(this.x, this.y, this.width, this.height);
}
```

**Figure 7 : Création de la classe Square avec une méthode draw**

JavaScript permet donc l'instanciation d'objet, `var square = new Square(0,0,10,10);` et utilise également le principe de **référence** sur les objets. En JavaScript, un objet est toujours passé par référence, sauf s'il s'agit d'un type natif comme un entier ou un string.

En JavaScript, lorsqu'une fonction ne définit pas une classe, elle est directement rattachée à l'**objet global**. Cet objet contient tous les éléments globaux de l'application comme les variables globales qui ne sont pas définies à l'intérieur d'une fonction ou bien même l'objet « window » qui représente la fenêtre de l'application. Une très mauvaise pratique, mais qui est bien souvent difficile à éviter, est d'utiliser JavaScript comme un réel langage de script, de définir les variables et les fonctions en tant que méthodes propres à l'objet global. JavaScript le permet comme il donne énormément de liberté sur sa syntaxe et sa structure. Selon cette même idée, lorsqu'une variable est appelée dans une fonction et n'a pas été définie auparavant par le mot clé « var », elle est directement rattachée à l'objet global et sort du « scope » local. Elle ne soulèvera aucune erreur à l'exécution.[6]

### LES ANIMATIONS ET LA RECONNAISSANCE DES GESTES

L'intérêt de l'utilisation d'un smartphone comme manette réside en la diversité des **« inputs » tactiles** qu'il est possible de réaliser. JavaScript gère les interactions avec l'application de façon évènementielle. Lorsqu'un joueur touche une zone de son écran, JavaScript peut récupérer ce toucher si auparavant un évènement avait été associé à l'élément touché. Le **framework JavaScript jQuery** permet la gestion de ces évènements très facilement. Dans le cadre de ce projet, nous avions à l'origine l'ambition d'intégrer beaucoup de types de touchers différents à l'application. De façon native, JavaScript ne sait pas reconnaitre les évènements de toucher mais seulement ceux de clic. jQuery permet de transformer ce clic en toucher et vice versa. Pour l'implémentation de gestes plus complexes tels que des **« Pinch »** pour zoomer, il a fallu trouver des librairies JavaScript capables de les réaliser. Néanmoins, le manque de standard sur l'interprétation de JavaScript a donné des résultats complètement hétérogènes d'un

[6] Le mot clé **« Use Strict »** permet de remédier à ce manque d'erreur de JavaScript. Dans ce cas, si la fonction qui oublie de définir la variable est en mode stricte, une erreur de compilation apparaitra lors de l'exécution du script.

système d'exploitation à l'autre. J'ai donc choisi d'implémenter moi-même les gestes nécessaires à l'application et limiter le nombre de gestes complexes. L'application compte au final des **touchers simples et du drag & drop.**

Les **animations** représentent aussi une fonctionnalité importante d'une telle application, le joueur souhaite avoir des retours visuels sur ce qu'il fait. jQuery permet d'animer facilement des éléments HTML grâce à sa fonction **« animate »**, qui n'existe pas en JavaScript. Néanmoins, animer des éléments du DOM reste encore une pratique très couteuse en performance pour l'application et là encore, j'ai été limitée par les différentes plateformes sur lesquelles l'application doit fonctionner. Les smartphones les moins performants ne supportent qu'à court terme des animations couteuses comme un **« Pulse »** avec changement de couleur d'une zone. J'ai préféré simplifier les animations pour qu'elles restent propres sur toutes les plateformes ce qui implique qu'elles soient limitées par le moins performant appareil.

Dans le cadre des animations, j'ai également réalisé un **tutoriel** pour l'application. Celui-ci doit apparaitre lorsque le jeu n'a pas encore commencé bien que l'environnement de l'application soit prêt. Ce tutoriel doit faire « pulser » différents endroits de l'application tout en faisant apparaitre un texte. L'application doit reprendre son fonctionnement habituel à la fin du tutoriel. Pour réaliser cet effet, j'ai eu recours à un objet très pratique de jQuery, l'objet **« Deferred »**. Cet objet permet l'exécution **asynchrone** d'une partie du script. Il faut savoir que JavaScript n'utilise **qu'un seul thread** puisqu'il utilise le thread d'exécution du navigateur. L'exécution simultanée de deux fonctions est donc simulée par exécution concurrente. L'objet « Deferred » possède trois états : **« Pending », « Resolved », « Rejected ».** A sa création, l'objet est dans l'état « Pending » et il est alors possible de passer cet objet en état « Resolved » au déclenchement d'un événement. En passant à l'état « Resolved », la fonction associée à l'événement **« Done »** de l'objet sera exécutée si elle existe.

```
JAVASCRIPT

var deferredObject = $.Deferred();

deferredObject.done(function() {
  alert("It is done!");
});

$("button").on("touchstart", function() {
  deferredObject.resolve();
});
```

**Figure 8: Création d'un objet "Deferred" et de ses évènements**

Cet objet est très pratique dans le cas où le script exécute régulièrement une fonction à l'aide d'un **« timer ».** L'objet « Deferred » permet d'interrompre ce « timer » et de le relancer à la fin de son travail. Pour cette application, l'utilisation de « timers » a été très importante pour le transfert de données avec le jeu.

## LA CONNEXION AVEC LE JEU : REQUETES HTTP ET JSON

La connexion avec le jeu pour la récupération des données à chaque instant a été programmée à l'aide de requêtes http envoyant régulièrement un **objet JSON**. Le JSON ou « JavaScript Object Notation » est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter des données structurées et remplace XML.

XML

```
<player id="player1">
  <color value="blue"/>
  <class value="archer"/>
</player>
```

JSON

```
{"player":
  {
    "id":"player1",
    "color":"blue",
    "class":"archer"
  }}
```

**Figure 9: Comparaison d'un objet "Player" au format XML et JSON**

L'application envoi toutes les 100ms l'état des données dans lequel elle se trouve en passant comme paramètre à la requête http l'objet JSON.

JAVASCRIPT

```
var XMLHttpRequest = new XMLHttpRequest();
XMLHttpRequest.open("POST", "/InGameDataStatic", "true");
XMLHttpRequest.send(jsonObject);
```

**Figure 10: Création d'un objet XMLHttpRequest et envoi de l'objet JSON**

Pour réaliser cet envoi toutes les 100ms, j'ai utilisé les « timers » de JavaScript. Un « timer » permet d'appeler une fonction à un intervalle de temps régulier. Chaque 100ms, l'application envoi au jeu l'état dans lequel elle se trouve et le jeu lui renvoi des différences s'il y en a. L'application met à jour son objet JSON avec les différences puis les traite pour les afficher.

JAVASCRIPT

```
var timer = setInterval(functionCallback, 100);
```

**Figure 11: Utilisation d'un timer en Javascript**

Cette méthode ne représente pas la meilleure stratégie de connectivité. Il aurait été beaucoup plus élégant d'implémenter un **système évènementiel**. A chaque changement dans le jeu, un évènement aurait pu être envoyé à l'application pour reproduire cette modification. De même, à chaque « input » de l'utilisateur, l'application aurait pu en informer le jeu grâce à un évènement. Ce principe n'existe que depuis peu en JavaScript, les **« Web Socket »**, mais il nécessite une configuration du serveur web. N'ayant pas accès au serveur web de la PlayStation 4, il était impossible de mettre en place une telle infrastructure pourtant bien plus performante.

# 5. Les difficultés rencontrées

## L'IMPLEMENTATION RESPONSIVE ET CROSS-PLATFORM

L'implémentation d'une application compatible sur tous les appareils et systèmes d'exploitation représentait une difficulté majeure tout au long du projet. De nombreuses différences graphiques ainsi que des différences d'interprétation du code JavaScript se constataient entre **iOS, Android et Windows**. Pour développer l'application et tester en parallèle les nouvelles fonctionnalités implémentées, j'ai utilisé l'outil de développement du navigateur Google Chrome. Cet outil permet notamment d'ajouter des **« breakpoints »** au code JavaScript au moment du « runtime » ainsi que d'émuler des environnements tactiles de tablettes ou smartphones. Malgré la simulation des différents environnements mobiles sur navigateur, des différences étaient toujours notables en situation réelle et de nombreux bugs n'étaient visibles qu'à la phase de test sur chacun des appareils.

## LA NOUVEAUTE DES COMPANION APP ET LE PEU DE DOCUMENTATION

Les « Companion App » ne sont que depuis peu de temps utilisées pour accompagner des jeux vidéo sur console. La première difficulté fut le manque de documentation. Les constructeurs Sony et Microsoft fournissent une documentation de leur environnement de développement mais qui malheureusement parfois ne répond pas à toutes les questions. La documentation de Sony pour son application « PlayStation®App » manquait notamment de références plus techniques à leur application ainsi qu'au serveur web associé. De même, très **peu d'informations techniques** se trouvent actuellement à ce sujet sur le web et les recherches précises étaient infructueuses. Le même problème s'est présenté avec « SmartGlass » où aucun détails techniques n'avaient été renseigné au niveau de la connectivité et du transfert des données ce qui nous a coûté beaucoup de recherche. Ce projet était le premier « Projet Companion » de l'entreprise, beaucoup de temps a été consacré à la découverte des deux plateformes.

## L'IMPLEMENTATION ORIENTEE OBJET AVEC JAVASCRIPT

La dernière difficulté rencontrée concernait les bonnes pratiques de développement avec JavaScript. N'ayant jamais réalisé auparavant une application d'une telle envergure, j'ai découvert au fur et à mesure du développement certaines fonctionnalités de JavaScript qui m'étaient auparavant inconnues. JavaScript étant un langage très **permissif**, j'ai progressivement découvert les bonnes et les mauvaises pratiques. J'ai donc assez régulièrement tout au long de mon stage refait certaines parties de l'application qui n'étaient pas codées en connaissance de ces bonnes pratiques. Le développement objet en JavaScript m'a également pris un peu de temps à cerner et mettre en place. Par ailleurs, le projet est globalement allé très vite et je n'ai pas eu le temps de réaliser une phase de conception. J'ai progressivement appris à maitriser ce langage et ses nombreuses libertés d'implémentation.

# CONCLUSION

## 1. Enseignements et bénéfices

### POUR L'ENTREPRISE

Asobo Studio n'avait jusqu'à ce jour pas eu de besoins conséquents en développement web. Avec l'apparition des « Companion App », le besoin en développement web de l'entreprise est devenu spécifique au jeu vidéo. Pour ce premier projet, Asobo Studio a tenté l'expérience de réaliser une « Companion App » et m'a confié cette tâche. Aucun autre projet auparavant ne faisait figurer d'application web et toute la première phase de découverte des plateformes était à faire.

En réalisant cette application, j'ai permis à l'entreprise d'apprendre à **planifier le coût et le temps de développement** d'une application compagnonne. J'ai également analysé et découvert les différentes **limites et contraintes** d'une telle application afin de rendre les futurs projets plus simple à préparer et planifier. Pour ce premier projet, à l'origine, beaucoup de fonctionnalités avaient été prévues et notamment beaucoup d'animations et de gestes pour interagir avec la tablette. Nous n'avions à ce moment pas conscience de ce que tout cela représentait en **temps de développement et perte de performance** sur l'application. Les prochaines applications seront donc pensées de façon à respecter ces différentes contraintes.

Sur ces premiers mois de stage, je pense également avoir fait preuve **d'autonomie, de motivation et de rigueur** dans le travail réalisé. J'ai essayé de réaliser au mieux mon travail en connaissance de l'importance de la fonctionnalité sur le produit final. En effet, les « Companion App » représentent un argument marketing très important et particulièrement dans le cadre de ce jeu.

### PERSONNELLEMENT

J'ai personnellement énormément appris de ces quatre premiers mois de stage. Par mon autonomie, je me suis considérablement améliorée d'un point de vue technique. Etant la seule développeuse web de l'entreprise, j'ai appris à régler les différentes difficultés par moi-même et à développer pour tous les environnements avec leurs contraintes. Tout au long de ce projet, j'ai également été entièrement **responsable** de l'avancement de mon travail et du bon fonctionnement de mon application. J'ai créé et tenu à jour un planning prévisionnel, j'ai également pris le temps de prioriser les différentes tâches. J'ai découvert le monde du jeu vidéo dans sa totalité et les différents corps de métier qui forme un projet. Ces quatre premiers mois de stage m'ont beaucoup appris et représentent une première expérience dans le monde du jeu vidéo me donnant très envie de continuer. Sur un plan plus humain, j'ai également adoré pouvoir partager ma passion pour le jeu vidéo au quotidien avec des gens tous aussi passionnés.

# 2. Les projets à venir

## DEVELOPPEMENT D'UNE SECONDE APPLICATION COMPAGNONNE

Dans la continuité du premier « Projet Companion », je vais développer une seconde application sur le temps de mon stage à partir du milieu du mois de Juin. Cette seconde application appartient au même jeu mais concerne un mode de jeu différent et apparaitra après la date de sortie du jeu principal dans une **DLC**.[7] L'application reprendra exactement les mêmes principes et technologies que la première. Elle sera donc réalisée plus facilement grâce au travail de recherche et aux difficultés rencontrées et solutionnées sur le premier projet. Elle me permettra d'approfondir mes connaissances acquises sur le développement de la première application. La date de sortie de cette seconde application se situe exactement à la fin de mon stage, c'est-à-dire mi-août.

**Compétences : HTML5 – CSS3 – JavaScript**

## DEVELOPPEMENT D'UN SITE VITRINE AVEC NEWSLETTER POUR UN JEU VIDEO

En parallèle des applications compagnonnes, je suis en charge d'un site internet vitrine pour un jeu vidéo. J'ai réalisé l'implémentation du design au tout début de mon stage. Désormais le site web doit faire figurer une **newsletter** et d'autres fonctionnalités arriveront au fur et à mesure. Je me chargerai donc progressivement de toutes ces nouvelles fonctionnalités en parallèle de mon travail sur la seconde application compagnonne.

Pour la réalisation de cette newsletter, j'ai choisi d'utiliser un **micro-framework PHP**, tel que **Silex**. Plus léger qu'un framework global comme **Symfony**[8], ce micro-framework apporte tous les composants nécessaires à la réalisation d'une petite application web.

**Compétences : HTML5 – CSS3 – JavaScript – PHP – Serveur Mail**

## DEVELOPPEMENT D'UNE APPLICATION WEB INTERFAÇANT AVEC UN JEU VIDEO

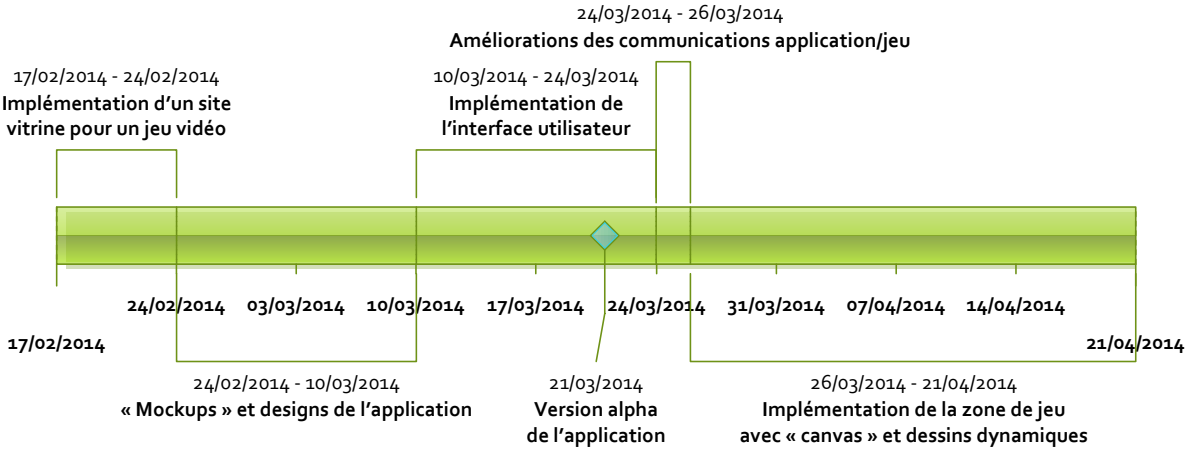
Finalement, un dernier projet sur lequel je serai susceptible de travailler concerne une application web beaucoup plus complexe. Cette application sera disponible sur ordinateur, tablette et smartphone et accompagnera également un jeu mais de façon beaucoup plus poussée. Elle permettra au joueur **d'interagir avec l'univers du jeu** sans même avoir eu besoin de le lancer. Cette application représente un très gros chantier et ne pourra pas être réalisée en parallèle des deux autres projets mais pourra peut-être faire l'objet d'un projet à la suite de mon stage.

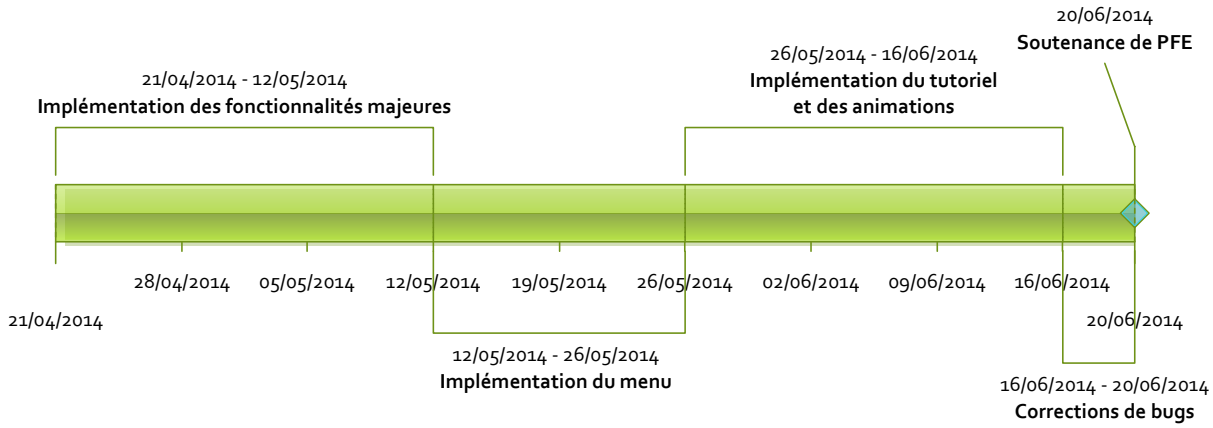
**Compétences : HTML5 – CSS3 – JavaScript – PHP/C# – Frameworks**

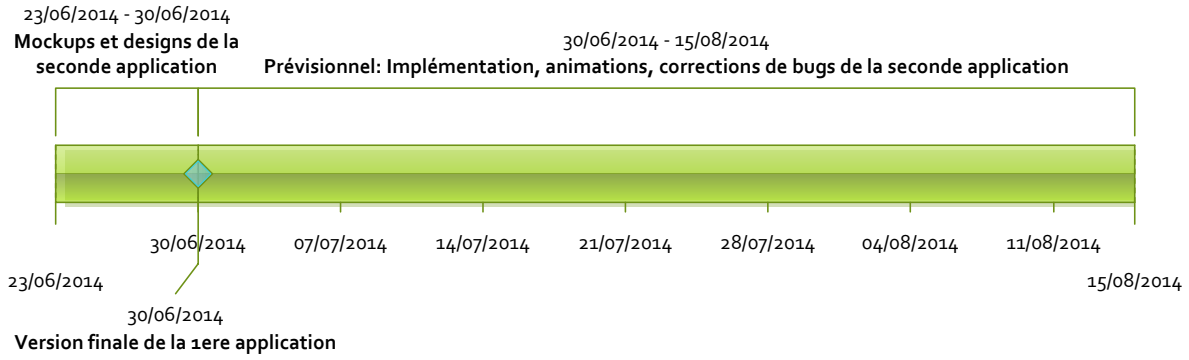
[7] DLC ou « Downloadable Content » désigne l'extension d'un jeu vidéo.
[8] Symfony est un framework PHP très connu pour l'implémentation d'application web très complexe.

# ANNEXE

## 1. Planning du stage



# TABLE DES ILLUSTRATIONS

## Résumé

En tant que passionnée de jeux vidéo, j'ai réalisé mon stage de fin d'études en informatique à **Asobo Studio**. Asobo Studio est un studio de développement français de jeux vidéo basé à **Bordeaux** et employant actuellement **plus de 80 passionnés**. L'entreprise compte actuellement plusieurs projets en cours de réalisation avec pour chacun de ces projets une équipe d'une quinzaine de personnes composée de producers, designers, programmeurs et artistes. Pour ce **stage de six mois** au sein d'Asobo Studio, j'ai signé un **accord de confidentialité** m'obligeant à ne divulguer aucun détail des différents projets sur lesquels je suis intervenue. Selon cet accord de confidentialité, je ne suis pas en capacité de renseigner le lecteur sur le nom et les détails des jeux sur lesquels j'ai travaillé.

En tant que **« web developper »** au sein d'Asobo Studio, j'ai été assignée à un projet en cours pour le développement d'une application web dite **« second écran ».** Les applications dites « second écran » ou **« Companion Apps »** sont des applications secondaires qui accompagnent l'utilisateur dans l'utilisation d'une application primaire, le plus souvent un jeu vidéo. Cette application ne doit pas remplacer le jeu mais doit enrichir l'expérience du joueur avec de nouvelles fonctionnalités et opportunités. Les « Companion Apps » ont pour but d'être disponibles **gratuitement** et accessibles à n'importe quel joueur possédant au minimum un smartphone. Elles se veulent donc multiplateformes ou **« cross-platform »** afin d'être téléchargeables sur tous types d'appareils indépendamment de la marque et du système d'exploitation. Dans le cadre de mon stage, la notion d'application compagnonne a été enrichie et l'application a pour but de complètement remplacer l'utilisation de la manette.

Pour le développement de cette application web, j'ai notamment utilisé **LESS**, une librairie JavaScript qui étend le CSS avec un comportement dynamique, utilisant des variables, des classes abstraites, des opérations et des fonctions. Le code de l'application a été réalisé en **JavaScript** supporté par sa célèbre librairie **jQuery**. Cette application tire parti au maximum des nouvelles technologies web datant de HTML5, notamment des « canvas ». L'application communique avec le jeu par **requêtes http** et transfert les données grâce au format **JSON**. Les majeures difficultés rencontrées viennent de l'implémentation d'une seule et unique application pour toutes les plateformes ainsi que du manque de documentation technique des « Companion App ».

Ce stage m'a fait considérablement progresser techniquement dans un univers qui me passionne et aura permis à l'entreprise d'apprendre à planifier le coût et le temps de développement d'une application compagnonne.

## Abstract

Being a video game player since my childhood, I wanted to find an internship in computer sciences within a video game company**. Asobo Studio** is a French independent developer located in Bordeaux. The company's size is around 80 people. Asobo is currently working on multiple projects, each one gathering a team of fifteen people such as producers, designers, developers and artists. For my **six months internship**, I had to sign a **non-disclosure agreement** forbidding me to give information about projects I worked on during my internship. For this report, I won't be able to provide the reader with any specific details, designs or names of the projects I've been working on.

As a **web developper** within Asobo Studio, I have been assigned the development of a web application known as a **companion application** for an on-going project. A companion application is a secondary application which provides a player with multiple features, screens, information while playing a video game. This application doesn't substitute the video game itself but it enhances the gaming experience. A companion application is most of the time **free of charges**, available for any players owning a smartphone and **cross-platform** so it can be used on any device without any constraint of brand or operating system. For this project, the companion application I had to develop was way more useful for the game since it had to completely replace a gamepad. This companion application can be used to play the game while visiting multiple information which could not be normally accessed in the main game at any time.

For the design part of the application, I used **LESS**, a pre-processor, meaning that it extends the CSS language, adding features that allow variables, mixins, functions and many other techniques allowing CSS code to be more maintainable. The application code has been developed in **JavaScript** help with jQuery, a well-known framework. This application has been done according to all HTML5 standards and using components such as "canvas". The application gets data from the game using **http requests**. Data are written in **JSON** format for an easier processing in JavaScript. Main difficulties were the cross-platform behavior of the application and the lack of technical documentation for companion app APIs.

I really improved my technical skills during this internship and I loved to work for creating video game with people sharing the same interest. My work helped the company gathering knowledge about companion applications and time needed to develop one.