Télécom SudParis
Année scolaire 2017/2018

Projet Informatique 1ère Année
PRO3600 : Projet Informatique

Star Wars: Behind The Magic

AO John
DUBOC Aurélien
LAMOURET Etienne
PENCIOLELLI Valentin

Enseignant responsable : **TRAHAY François**

22 mai 2018

Sommaire

1 Introduction

Ce document est le rapport du projet informatique portant sur l'encyclopédie Star Wars: Behind The Magic.

Star Wars: Behind The Magic est une encyclopédie Star Wars distribuée en 1998 sous forme de deux disques CD-ROM. Elle porte sur la trilogie originale et l'*Univers Etendu*, actuellement *Légendes*.

Aujourd'hui, cette encyclopédie ne fonctionne plus convenablement sur les systèmes actuels. Rendre disponible cette encyclopédie sous forme de site web permettait d'en faciliter l'accès et la consultation, mais aussi de mutualiser les différentes versions existantes pour réaliser un site multilingue.

L'idée de ce projet en temps limité est de concevoir l'architecture du site et d'y ajouter les parties les plus intéressantes de l'encyclopédie pour permettre les itérations futures au-delà du temps imparti.

Le dépôt du projet se trouve dans l'organisation GitHub: https://github.com/BehindTheMagic, et le site sera hébergé à l'adresse suivante: https://behindthemagic.github.io.

2 Cahier des charges

L'objectif est d'adapter l'encyclopédie Star Wars: Behind The Magic sous forme de site web. Comme l'encyclopédie est conséquente, nous allons nous focaliser sur quatre parties de l'encyclopédie en particulier pour le livrable final:

- Technologie
- Evénements
- Scène par Scène
- Glossaire

Ce sont les quatre parties qui possèdent le plus d'interfaces utilisateurs différentes et diversifiées qui seront en outre réutilisables pour les autres parties (comme Personnages ou Univers Etendu).

Nous voulons respecter l'aspect et les fonctionnalités originales de l'encyclopédie:

- Les interfaces utilisateurs doivent suivre les interfaces originales
- La base de données textuelle, les images et les sons sont importés dans ce projet
- Les questions Trivia sur l'univers Star Wars sont disponibles.

Nous souhaiterions aussi rajouter ces fonctionnalités:

- Proposer un site multilingue basé sur les différentes versions de Behind The Magic (disponible anglais, français et allemand)

- Pour le glossaire: surligner le texte recherché dans les articles où les termes sont trouvés, et afficher des informations supplémentaires présentes dans la base de données mais non affichées dans la version originale.

2.1 Développement

2.2 Analyse du problème et spécification fonctionnelle

Les fonctionnalités à implémenter sont les suivantes :

- Architecture du site web pour facilement réaliser des vues (interfaces indépendantes), puis pour facilement naviguer entre elles

- Mise en place d'une page d'accueil, différente du menu principal de l'encyclopédie, pour le choix des langues et le lancement du logiciel

- Réalisation de la vue statique « menu principal » et de la barre de navigation omniprésente

- Mise en place d'un système de mise en cache côté client des ressources, pour réduire les temps de chargement

- Réalisation des parties citées dans le cahier des charges sous forme de plusieurs vues statiques et dynamiques chacune.

L'encyclopédie est composée de vues statiques et de vues dynamiques. Pour les vues statiques, celles-ci seront réalisées au plus proche de l'aspect original et contiendront les liens vers les autres vues de l'encyclopédie, en respectant l'ordonnancement original.

Pour les vues dynamiques, nous réaliserons une base de données dans un nouveau format par rapport à l'original, qui reprendra les données textuelles et pointera vers les ressources sonores et graphiques liées à chaque élément de l'encyclopédie. Le programme parcourra cette base de données afin de générer l'affichage en fonction du contenu fourni.

2.3 Conception préliminaire

Comme l'encyclopédie sera adaptée sous forme de site web avec plusieurs vues, et prévue pour être multilingue :

- Le site web sera statique sous forme de Single Page Application avec une librairie Javascript permettant de gérer des vues avec VueJS

- Les images, musiques et vidéos seront converties et optimisées pour une utilisation sur le Web, aussi bien en termes de taille pour la bande passante que de codecs pour une utilisation sur tous les navigateurs

- Nous utiliserons les interfaces de programmation i18n du navigateur (internationalisation, en anglais L10n: localization) pour ajouter des langues au projet)

Nous écrirons des scripts et des petits outils qui automatiseront et faciliteront la conversion des ressources originales afin de les utiliser sur le Web :

- Les fichiers sons peuvent être convertis manuellement avec Audacity, il doit être possible d'automatiser le procédé pour convertir tous les sons pour le Web

- Les vidéos peuvent être converties en utilisant ffmpeg. Cela nécessite plusieurs étapes qui peuvent être automatisées pour s'assurer d'obtenir du premier coup des vidéos de bonne qualité, compatibles Web (MP4 codec H264 et MP3, échantillonnage 4:2:0) et compressées

- La conversion des données textuelles se fera avec un petit outil qui permettra d'indiquer la forme de la base de données que l'on souhaite obtenir.

2.4 Conception détaillée

A partir de Vue, on implémente chaque composant sous la forme de vue dans des fichiers séparés. A ce niveau-là, les composants ont : « Menu Principal », « Technologie », « Glossaire », « Personnages »… Il faut à présent les afficher dans le contexte de l'application. Pour cela, on a besoin de créer un composant App qui contient toutes les données de l'application (barre de navigation, la vue par rapport au chemin (la route), lecteur vidéo…). Cet App est lui-même instancié lors de la création de l'instance VueJS dans le script principal qui permet de tout centraliser.

En pratique, toutes les vues ne sont pas indépendantes entre elles: il faut faire attention aux intéractions utilisateur et aux intéractions entre les vues. Par exemple, pour lancer une vidéo d'introduction à une sous-partie puis lancer une musique d'ambiance, il faut faire attention à ce que la vidéo soit bien finie avant de lancer la musique: il y a donc un soucis d'écoute d'événements, car il n'est pas évident de mettre des « $watch » ou d'écouter l'événement dans la partie « mounted » de la vue, voire même de créer des directives personnalisées communes à toutes les vues. Malheureusement, le manque d'exemples concrets sur internet a demandé de nombreuses tentatives pour trouver une solution fonctionnelle. Celle qui a été retenue est une directive vue personnalisée qui prend en compte qu'une vidéo en plein écran peut être jouée ou non, ce qui demande deux approches différentes. Ces deux approches sont : exécution directe de la vidéo ou observation d'une propriété inhérente à la vidéo avant de lancer l'audio.

En ce qui concerne les transitions entre les vue: la variable locale qui contient la chaine de caractères de la langue. Un problème d'héritage peut apparaitre, car il faut que cette variable soit créée le plus haut possible dans l'arbre des vues afin que les autres vues puissent en hériter. Cependant, certains composants vont avoir besoin de lire cette propriété directement à la racine, car même héritée, elle n'est pas forcément lisible dans le contexte du composant.

Implémentation du multilingue :

On utilise des dictionnaires clef-valeurs dans les différentes langues qui permettent d'associer à un identifiant ses différentes expressions dans lesdites langues. Pour afficher les textes dans l'interface on utilise une syntaxe particulière pour indiquer que l'on veut la valeur associée à un identifiant donne connaissant la langue qui est une valeur globale.

Conception d'une vue : exemple du menu principal [2]

- Le fond d'écran est extrait des ressources originales et est affiché avec la propriété CSS background-image
- On identifie ensuite les éléments interactifs de cette vue (boutons, liens vers les sous parties et l'animation vidéo)
- Pour chacun de ces éléments, on créer un élément dans le template que l'on stylise avec CSS pour se rapprocher de l'original et que l'on pose en absolu pour être au plus proche de l'original
- Ici il s'agissait de styliser les liens comme des boutons semi-transparents à fond noir et avec une police de caractère personnalisée et un effet au survol
- De plus, pour la vidéo que l'on avait préalablement positionnée en absolu, on indique la source vidéo qui est un fichier extrait et converti depuis les ressources originales pour être compatibles avec les navigateurs. C'est un élément qui produit de la musique, donc nécessite un traitement supplémentaire: il doit être lancé uniquement si tout autre audio est coupé (notamment celui du lecteur vidéo). Pour cela, on se repose sur l'écoute d'une propriété issue du composant principal App. Et en fonction de son état on procède de deux manières différentes (action directe ou attente du changement de l'état de la propriété écoutée).

Exemple supplémentaire : le glossaire [3]

- On importe les fichiers JSON correspondant au glossaire dans les différentes langues que l'on définit comme des données locales de la vue.

- A l'initialisation du glossaire, les sélecteurs sont automatiquement peuplés des différentes catégories et des différentes entrées du glossaire (toutes issues de la base de données de la langue courante). On utilise le système de binding de vue afin de refléter un changement de sélection dans un sélecteur sur les autres sélecteurs et la description de l'entrée (affichage principal)

- En changeant de catégorie, il faut filtrer les entrées pour ne garder que celles correspondant à cette entrée. On utilise pour cela la méthode filter des tableaux en JS pour le faire simplement et le retour de cette méthode permet d'actualiser le sélecteur des entrées.

Implémentation de la recherche dans le glossaire :

- On cherche l'entrée utilisateur dans la propriété « description » en parcourant les entrées de la base de données une par une et on s'arrête sur la première occurrence trouvée. A ce moment, on affiche à l'utilisateur cette entrée (et ses informations) tout en surlignant le mot clef recherche avec la balise « html mark ».

2.5 Codage

Vue permet d'écrire du code très lisible, et ce sans commentaire. Il n'a donc pas été jugé nécessaire de commenter le code en quantité d'autant plus que la compilation du travail avec VueJS fait que tous les commentaires sont éliminés du rendu final.

2.6 Tests

Les tests porteraient sur l'algorithmie, cependant, il n'y a pas de grosse algorithmie dans le code. De plus, l'algorithmie de ce code est très simple et ne peut être dissocié de l'interface graphique (accès DOM): on a besoin de voir son comportement pour juger de son fonctionnement.

Les véritables tests porteraient sur la fidélité du rendu graphique par rapport à l'original et ne se teste qu'en observant les deux versions. Après avoir fait des tests sur le glossaire, les deux versions répondent bien de la même façon. On considère ces tests comme des tests de validation. Cela a permis de rejeter des propositions d'implémentation de différentes vues.

3 Problèmes rencontrés et solutions éventuelles

Des difficultés sont apparues à toutes les étapes du projet.

Lors de la création de l'outil pour convertir les fichiers CSV en JSON, il nous fallait pouvoir travailler avec des formes et des structures CSV très différentes les unes des autres, faire une évaluation de code, rédiger des expressions régulières convenant à notre fichier de travail, et avoir un bon système d'export et de sauvegarde des fichiers JSON ainsi créer. Un compromis a été trouvé, mais il manque de performance.

De même, lors de la projection des données originales vers la structure que le développeur (nous) veut, c'est-à-dire les fichiers JSON en mode clef-valeur, une complication s'est fait sentir lors de la nécessité d'évaluer correctement le code pour ne pas avoir d'erreurs dans l'outil.

Lors de la conversion vidéo, il fallait un format compatible pour tous les navigateurs et si possible libre. Un compromis a été trouvé: h264 mp3, cependant il n'est pas libre mais toutefois correspond à nos attentes fonctionnelles. Le color sub-sampling 420 a dû être utilisé, car les alternatives n'étaient pas toutes lisibles dans tous les navigateurs.

La conversion de musique a engendrée des problèmes similaires à la conversion vidéo, mais plus facilement gérable. La conversion de samples audio a été faite à la main avec audacity au fur et à mesure des besoins, car nous n'avons pas trouvé de moyen d'automatiser la conversion de ces samples.

Les icones au format bitmap sont compressées de manière particulière, nous n'avons pas réussi à en ouvrir une seule. Il a donc été nécessaire de toutes les recréer afin de les utiliser.

Des soucis avec Vue se sont aussi fait ressentir, notamment lors de l'héritage de la propriété langue à travers tous les composants ; de même il y avait des incompatibilités entre les deux versions majeures de vue ce qui a entrainé des incompréhensions alors que nous faisions face à la documentation officielle. De plus, la documentation ne présentait pas de cas concrets proches de nos soucis et cela a donc demandé plusieurs heures de recherches, car les exemples trouvés n'étaient pas clairs ni dans la documentation fournie ni sur Stack Overflow.

Enfin, le système de routage a posé problème, car le système d'écouteur d'événements du changement de route a demandé de repenser la manière de faire les transitions entre les vues.

4 Manuel utilisateur

L'application en elle-même est facile d'utilisation ; il s'agit d'un point-and-click. C'est à l'utilisateur de découvrir les différentes fonctionnalités en balayant sur curseur sur les zones qui l'intéressent. De ce fait l'application est principalement faite pour une utilisation à la souris, mais il existe des raccourcis clavier:

- Espace pour mettre en pause une vidéo et la reprendre
- Echap pour passer une vidéo
- Flèche gauche et droite du clavier comme précédant et suivant
- Flèche haute, pour faire un retour (up) dans l'arborescence.

5 Conclusion

Les éléments suivants ont été implémentés pour le projet Star Wars: Behind the Magic:

- Un outil pour faciliter la création de contenu et permettre la conversion des fichiers CSV en JSON
- Un launcher, servant de page d'accueil du site permettant de sélectionner la langue et annonce sonore lors du démarrage
- Un lecteur vidéo plein écran, avec possibilité de mettre en pause et/ou passer la vidéo à l'aide de raccourcis clavier
- Un conteneur pour la vue selon la route prise et une barre de navigation omniprésente (précédent, suivant, home…)
- Une vue statique « menu principal » ainsi que les boutons pour sélectionner la catégorie de l'encyclopédie (Personnages, Scène par Scène, Technologies…)
- Un Glossaire fonctionnel en français et en anglais, avec :
    - affichage de l'entrée du glossaire
    - ajout des références (bonus par rapport à la Version Originale de l'encyclopédie)
    - fonctionnalité de trie par catégories des entrées
    - possibilité de naviguer entre entrées précédentes et suivantes
    - moteur de recherche avec surlignement du mot-clef recherché
- L'architecture du site web est telle que l'insertion des vues est facilitée (les interfaces sont indépendantes) et pour facilement naviguer entre elles
- Un système de mise en cache côté client des ressources pour réduire les temps de chargement.

Cependant, de nombreuses améliorations peuvent être apportées à ce projet, en voici les principales :

- Rajouter les icones dans la barre de navigation
- Meilleure gestion des transitions entre les vues; codes à simplifier, car actuellement on a besoin de nombreuses propriétés et écoutes sur des vues différentes

- Rajouter les catégories non mises en place telles que Personnages, Lieux, episode1… Le projet en l'état actuel permet d'ajouter facilement les vues nécessaires

- L'internationalisation. Pour ajouter des langues il suffit de rajouter les nouveaux drapeaux sur le launcher. La traduction se fait, dans les cas statique grâce à i18n et dans les cas dynamique en rajoutant le fichier database dans chacune des vues concernées, rien d'autre n'est à recoder; la traduction est faite automatiquement dès que les fichiers sont présents et complets.

6 Bibliographie

[1] Archive du site officiel de l'encyclopédie:

http://web.archive.org/http://www.lucasarts.com/products/btm/default.htm

[2] code du launcher:

https://github.com/BehindTheMagic/BehindTheMagic.github.io/blob/dev/src/components/Launcher/Launcher.vue

[3] code du glossaire:

https://github.com/BehindTheMagic/BehindTheMagic.github.io/blob/dev/src/components/Glossary/Glossary.vue

7 Annexes

7.1 Annexe 1 : Gestion de projet

Pour suivre le projet, nous avons fait 3 réunions avec François Trahay, notre enseignant responsable dont voici les compte-rendus:

Réunion du 01 Février 2018:

Cette réunion s'est axée autour de 3 points:

- Partie administrative, les informations sont toutes disponibles sur moodle ; trois livrables sont attendus: un premier rapport présentant le cahier des charges puis un prototype du sujet et enfin le produit final avec le rapport
- Des questionnements autour de :
  - VueJS ou Angular
  - Idée de faire de l'internationalisation, c'est-à-dire implémenter plusieurs langues
  - Quelles sont les fonctionnalités à implémenter, car le projet s'avère être conséquent
- François Trahay a émis une réserve quant à la légalité du projet vis à vis des droits d'auteur. C'est pourquoi dans le dépôt GitHub nous avons déposé un échange de mail entre Etienne de Vince Lee (directeur du projet chez LucasArt en 1998) pour qui le projet ne posait pas de problèmes à ce niveau.

Réunion du 19 Février 2018:

Cette réunion a permis de fixer les objectifs, et de présenter les réponses aux différentes questions évoquées lors de la réunion précédente: VueJS est préféré à Angular, car Angular est trop contraignant, malgré sa plus grande renommée.

Réunion du 19 Mars 2018:

Cette réunion a présenté les avancées et les points à continuer:

- L'outil pour convertir les fichiers CSV en JSON avec démonstration
- Soucis avec Vue qui est plus difficile à manier que prévu, mais le Menu principal a pu être implémenté avec une petite interface multilingue
- Il faut à présent retravailler sur Vue (il a été émis la possibilité de coder nous-même notre propre interface) et finir les conversions CSV en JSON pour s'attaquer aux différentes vues.