Développement d'outils ou matériel d'enseignement de l'informatique

Développement d'un outil informatique démontrant visuellement la perception d'une liste pour un ordinateur

Collège du sud, Travail de maturité Version intermédiaire

Grégoire Geinoz

janv. 11, 2022

Table des matières

CHAPITRE 1

Introduction

1.1 Présentation de l'outil

L'objet de ce travail de maturité consiste en la programmation d'un outil permettant d'accompagner un professeur souhaitant développer une compréhension intuitive de la notion de liste à ses élèves. Il permet d'avoir une vision clair de ce qu'est une liste en programmation et quelles sont les manipulations qu'un ordinateur peut effectuer sur celle-ci.

Concrètement, une liste est représentée comme étant un "jeu de cartes". Chaque élément de la liste compose une carte avec sa valeur affichée d'un côté de la carte. Ainsi, tous les éléments de la liste forment à eux, le jeu de cartes au complet. Certains algorithmes requièrent des variables externes à la liste, ces variables sont créées via des appels de fonctions sur une API(à expliquer) et sont représentées dans une zone de l'écran spécifique. Cependant, des variables "spéciales", qui n'ont comme seul rôle, d'être un index qui parcours les éléments d'une liste (dans une "boucle for" par exemple), peuvent également être créées et sont représentées comme des flèches qui pointent sur l'élément de la liste qui correspond à l'index contenu dans la variable.

Le but étant de démontrer les actions opérées par un ordinateur sur une liste, les interactions entre les éléments de la liste entre eux ou avec une variable sont animées. Ainsi, pendant l'exécution d'un programme, il fera discrétement appel à l'API afin qu'une animation se déclenche et montre sur l'écran le comportement de la liste.

1.2 Intêret de l'outil

Les listes et les algorithmes sont des sujets importants à comprendre et à maîtriser dans le cadre de l'apprentissage de la programmation. Actuellement, ces concepts sont amenés aux étudiants et expliqués de manière bien trop abstraite. Il est nécessaire de comprendre ce que représente une liste pour un ordinateur et comment il peut interagir avec ; ainsi que de comprendre le fonctionnement des algorithmes, étape par étape.

Des plateformes d'enseignement de l'informatique tel que "code.org" ont un concept similaire au projet de ce travail de maturité : Développer une compréhension intuitive de concepts par l'expérimentation. Cependant, leur projet cible un public jeune pour leur inculquer des bases de programmations en posant brique par brique des blocs de code afin d'amener une cible jusqu'à son objectif. Le projet envisagé dans le cadre de ce travail de maturité cible un public plus mature capable de pousser un raisonnement plus en profondeur, en effet les listes sont une notion plus abstraite car

elles sont des conteneurs de valeurs quelconques. De plus, ce projet vise à enseigner des notions très spécifiques de la programmation : les listes.

1.3 Technologies utilisées

Le projet repose sur 2 technologies principales : Javascript et HTML5. Cependant pour alléger la quantité de travail, un framework de jeu 2D, Phaser, est utilisé pour gérer tout ce qui relève de l'affichage d'images, de la gestion de scènes et d'événements et autres qui impliquent la gestion d'objets (les cartes et les variables). Enfin, le projet étant écrit en Javascript principalement, un interpréteur de code peut être nécessaire si le langage étudié par l'outil est différent.

1.4 Configuration matérielle requise pour utiliser l'outil

Afin que l'utilisation de l'outil soit la plus accessible possible, les langages de programmation du Web ont été utilisé pour le développer. Ce qui signifie que pour utiliser l'outil, seul un accès à un ordinateur opérationnel étant doté d'un navigateur internet, et d'une connexion, suffit.

1.5 Connaissance requises pour utiliser l'outil

L'outil fait appel aux notions de variables ; éléments qui constituent la liste, ainsi qu'aux opérations de bases entre variables numérique (calculs, affectations de valeurs) et la notion de condition entre les variables utilisant des opérateurs de conditions.

1.6 Connaissances requises pour comprendre le fonctionnement de l'outil

Afin de comprendre comment l'outil fonctionne, il est nécessaire de comprendre le code qui le compose. C'est à dire qu'il faut avoir un niveau de base qui couvre tous les fondamentaux du Javascript. Le code étant largement commenté, il n'est pas forcément nécessaire de savoir utiliser le framework Phaser pour comprendre le code source.

CHAPITRE 2

Réalisation du projet

2.1 Familiarisation avec les outils de développement

Afin de pouvoir programmer le projet, il était essentiel d'être à l'aise avec les technologies nécessaire à son développement. Pour cela, j'ai réaliser un premier prototype afin de m'assurer que j'étais capable de rassembler toutes mes connaissances récemment acquises dans un seul projet.

Les cartes présentes sur la capture d'écran sont des objets Phaser déplaçables avec la souris. A ce stade j'étais capable de développer un environnement dans lequel je pouvais faire évoluer le projet final.

2.2 Problème majeur rencontré lors du développement

Lors du développement du projet, un problème majeur est intervenu. Le problème réside dans la gestion des déplacements des cartes en JavaScript. En effet, une animation de carte dure un certain temps et pendant ce laps de temps, le code doit être « bloqué ». C'est à dire que si une autre animtaion devait s'exécuter, elle devrait d'abord attendre que l'animation précédente soit terminée.

Or l'animation est gérée avec Phaser dans une fonction « update() » asynchrone, ce qui implique que la suite du code est exécuté en parallèle de l'animation, rien n'empêche donc de lancer plusieurs animations simultanément. De plus, le code principal n'est pas conscient qu'une animation est en cours, il ne peut donc pas attendre que l'animation soit finit avant d'exécuter la suite du programme.

Pour remédier à ce problème, après avoir été bloqué pendant plusieurs semaines, il m'a simplement fallut le poser par écrit. En écrivant précisément le problème, les solutions envisageables viennent plus facilement à l'esprit. Il se trouve que l'une d'entre elle m'est venu en écrivant.

En effet, sans se prendre la tête en utilisant les nouvelles normes « async/await » apparues de la version ES7 de Javascript, l'utilisation d'une « Promise » à l'air envisageable afin d'attendre une « promessse » retournée par l'animation avant d'exécuter la suite du code.

Une autre approche est à prendre en compte, il s'agit de stocker les animations dans une liste. Ainsi, il serait possible de jouer les animations les une après les autres, selon l'ordre dans lequel elles ont été ajoutés à la liste.

C'est sur cette dernière méthode que je me suis tourné.

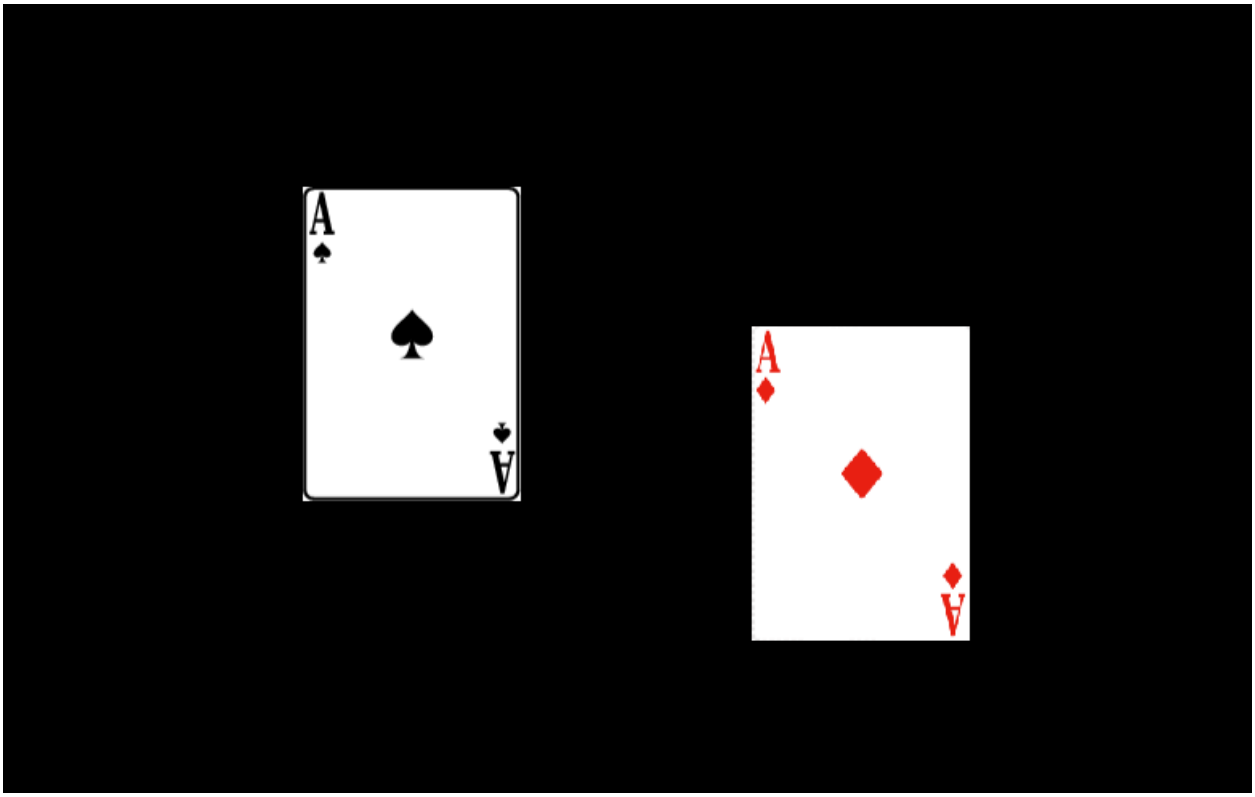

Fig. 1 – Capture d'écran du prototype

2.3 Titre 1

{ref}``

```
const variable = "Var1";
```

2.3.1 Titre 2

CHAPITRE 3

Titre du chapitre 2

3.1 Titre 1

3.1.1 Titre 2

CHAPITRE 4

Tutoriel Sphinx / MyST

En bref

Sphinx est le système de documentation du langage Python (https://www.sphinx-doc.org/en/master/).

Ce tutoriel, livré avec le kit de démarrage Sphinx pour la rédaction du travail écrit de votre TM, vous permet d'acquérir rapidement les bases de la syntaxe Sphinx.

4.1 Fonctionnement de la toolchain Sphinx

Avant de commencer, il faut comprendre le fonctionnement général de Sphinx. Il s'agit d'une « toolchain », à savoir un ensemble d'outils qui permettent de transformer des fichiers Markdown en documents formatés (soit HTML pour le Web ou LaTeX pour produire un PDF).

Pour les personnes qui connaissent un peu le LaTeX, Sphinx est un système un peu analogue : on écrit du code et un outil s'occupe de tout mettre en page et faire la reliure de manière professionnelle. D'ailleurs, Sphinx permet de générer du LaTeX pour générer un document PDF.

Au contraire de Microsoft Word, où l'on voit directement le résultat final lorsqu'on écrit le contenu, Sphinx utilise un paradigme semblable à celui utilisé en LaTeX : on écrit du code avec une syntaxe particulière qui est ensuite transformé dans le résultat final par la toolchain.

Fig. 1 – Processus de génération du HTML avec Sphinx

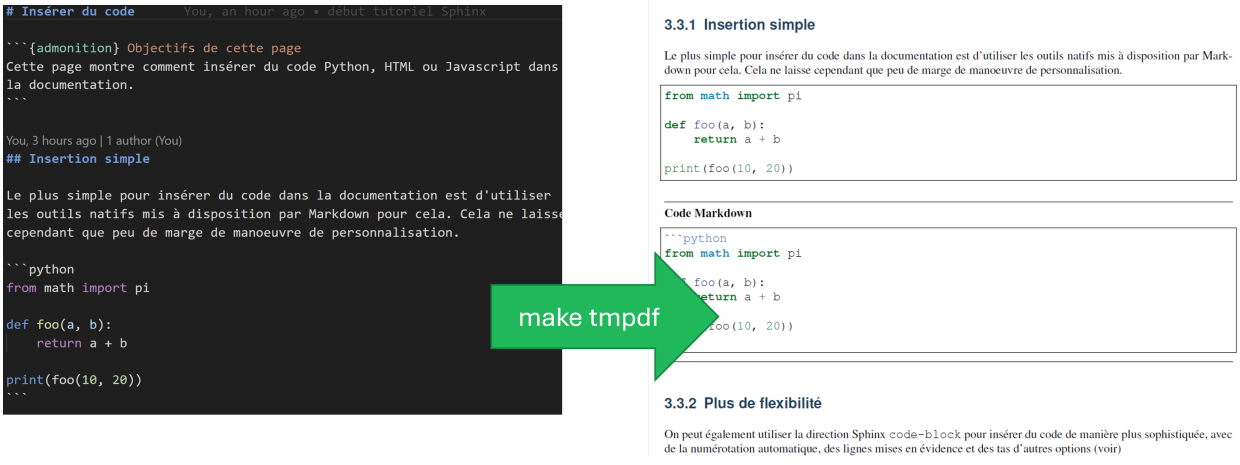

Fig. 2 – Processus de génération du PDF avec Sphinx

4.1.1 Les avantages de Sphinx par rapport à Word

Par rapport à Word, Sphinx présente les avantages suivants

- Sphinx est un outil Open-Source, ce qui signifie qu'on peut l'adapter sans problème à n'importe quel besoin. De plus, il n'y a aucun risque que le système ne soit tout-à-coup plus supporté, car une importante communauté s'occupe de le faire évoluer et de corriger les bugs détectés.
- On écrit de la documentation dans des fichiers qui ne contiennent que du texte, comme une sorte de « code source ».
- On écrit le contenu dans Visual Studio Code ou un autre éditeur. On peut donc utiliser toute la puissance de tels éditeurs pour optimiser la rédaction du contenu.
- Le contenu est très facilement versionalisable (utilisation de git et Github).
- Permet d'intégrer très facilement les éléments suivants qui sont très courants dans un travail de maturité en informatique
  - Des mathématiques
  - Du code source
  - Des éléments interactifs (quiz, programmes qui s'exécutent directement dans le navigateur, etc.)
- À partir d'une seule source, on peut générer des documents pour différentes cibles (Site Web, Document PDF, e-book pour une liseuse, . . .)
- Le système est extensible et facilement personnalisable
- Lorsqu'on écrit un document pour Sphinx, il n'est pas nécessaire de se préoccuper de la mise en page et de la mise en forme du contenu. On se contente d'écrire

4.1.2 Les désavantages de Sphinx par rapport à Word

- L'utilisation de Sphinx est moins intuitive que celle de Word au début.
- Word est un traitement de texte de type « WYSIWYG » (What You See Is What You Get). Lorsqu'on écrit le contenu dans Word, on voit directement le résultat final. Sphinx est un système de rédaction de type « WYSIWYM » (What You See Is What You Mean). Comme on ne voit pas directement le résultat final, il faut être attentif lors de la relecture du résultat final. On peut parfois avoir de sacrées surprises, en raison d'une erreur de syntaxe par exemple.
- La correction orthographique est moins facile qu'en Word, car il n'existe pas d'outil parfaitement intégré qui permette de faire la correction de manière automatique. Il est toutefois possible de passer du Sphinx à Word (avec un rendu dégradé), pour utiliser le correcteur automatique de Microsoft Word.

4.2 Les bases de la rédaction avec Sphinx

En cours de rédaction

Cette section est en cours de rédaction et sera complétée à l'avenir.

4.2.1 Fichiers Markdown ou RestructuredText

Le contenu à promprement parler est écrit dans des fichiers textes au format Markdown / MyST (voir)

Astuce : Pour écrire le TM, il est conseillé d'utiliser le format MyST qui

dispose d'une extension Visual Studio Code très pratique (https ://marketplace.visualstudio.com/items ?itemName=ExecutableBookProject.myst-highlight).

4.2.2 Bien démarrer

Pour le moment, il est recommandé de travailler sur Gitpod pour rédiger votre TM, à moins que vous ayez Linux installé sur votre machine et que vous soyez à l'aise avec.

Information

Nous allons apprendre à utiliser Linux ces prochains mois au cours d'OC.

En attendant que ce tutoriel soit plus fourni, reportez-vous à la documentation officielle de MyST

- https ://myst-parser.readthedocs.io/en/latest/sphinx/intro.html#intro-writing
- https ://myst-parser.readthedocs.io/en/latest/syntax/syntax.html#
- https ://myst-parser.readthedocs.io/en/latest/syntax/optional.html
- https ://myst-parser.readthedocs.io/en/latest/syntax/reference.html

Il peut également être utile de consulter la documentation de Sphinx pour savoir comment utiliser les fonctionnalités avancées.

- https ://www.sphinx-doc.org/en/master/index.html

4.3 Insérer du code

Objectifs de cette page

Cette page montre comment insérer du code Python, HTML ou Javascript dans la documentation.

4.3.1 Insertion simple

Le plus simple pour insérer du code dans la documentation est d'utiliser les outils natifs mis à disposition par Markdown pour cela. Cela ne laisse cependant que peu de marge de manoeuvre de personnalisation.

```
from math import pi

def foo(a, b):
    return a + b

print(foo(10, 20))
```

Code Markdown

````
```python
from math import pi

def foo(a, b):
    return a + b

print(foo(10, 20))
```
````

4.3.2 Plus de flexibilité

On peut également utiliser la direction Sphinx `code-block` pour insérer du code de manière plus sophistiquée, avec de la numérotation automatique, des lignes mises en évidence et des tas d'autres options (voir)

```
from math import pi

def foo(a, b):
    return a + b
```

Code Markdown

````
```{code-block} python
---
emphasize-lines: 1
linenos: true
---
from math import pi

def foo(a, b):
    return a + b
```
````

4.3.3 Insérer du code depuis un fichier externe

Il est également possible d'inclure du code dans la documentation depuis un fichier externe, au lieu d'avoir à écrire copier le code directement dans le fichier `.md`.

Titre 2

Titre 3

4.4 Fonctionnalités avancées

4.4.1 Encadrés

Encadré mis en évidence

Ceci est un encadré qui sort du fil du texte (on dit que cet encadré et « flottant », car il flotte sur le texte principal).

La directive Sphinx `sidebar` permet d'insérer certaines informations dans un encadré (dans le PDF) et dans une sorte de panneau flottant dans la version HTML.

$$f(x) = 2x^2 + 1$$

Il faut commencer par placer le `sidebar` dans le code Markdown et ensuite placer le contenu qui viendra se placer à gauche de la page.

Paragraphe est écrit de manière normale, au fil du texte.

— On pourrait remplir avec du contenu .... mais ce contenu n'est pas très intéressant. Ce contenu peut faire plusieurs lignes dans le fichier Markdown. C'est tout de même un peu long.

Avertissement : Il ne faut pas trop abuser des « sidebars », car le placement n'est pas du tout le même sur la version HTML en ligne et dans la version PDF.

Il vaut éviter également de mettre trop de contenu dans les `sidebar`.

4.4.2 Direction `raw`

Astuce : La directive `raw` permet de différencier le contenu présenté dans la version HTML et celui présent dans la version PDF. Elle permet également d'insérer du contenu tel quel pour profiter de toutes les fonctionnalités du format cible (par exemple le HTML ou le LaTeX).

Parfois, il est utile d'inclure un certain contenu (images, vidéos, ...) dans la version HTML en ligne, mais de ne pas inclure ce contenu dans la version PDF. C'est typiquement le cas pour des vidéos ou d'autres éléments interactifs.

En revanche, ce contenu n'apparaîtra que dans le LaTeX et pas dans le HTML.

Astuce : L'avantage de la directive `raw` est de pouvoir insérer n'importe quel code HTML, aussi exotique soit-il.

CHAPITRE 5

Indices and tables

— genindex
— search