ENSEIRB-MATMECA

Filière informatique

–Projet de programmation fonctionnelle–
–Semestre 6–

Rapport de projet
Génération Procédurale

Théo GOMICHON Clément SACCOCCIO

Hugo LANGLAIS Faustin BOITEL

Encadré par Mr. Morandat Floréal

2020 — 2021

Introduction

Ce rapport est le document explicatif du projet de programmation fonctionnelle de semestre 6 réalisé par l'équipe composée de Théo Gomichon, Clément Saccoccio, Hugo Langlais et Faustin Boitel.
Il porte sur le développement d'une bibliothèque de génération et traitement d'images en Javascript.

Ce rapport est structuré dans une logique d'approfondissement progressif. Nous commencerons ainsi par expliquer brièvement les tenants et aboutissants du sujet. Nous analyserons ensuite l'architecture de notre projet en réalisant une observation des différentes structures et interfaces mises en oeuvre. Puis nous plongerons au coeur des détails techniques en présentant certains exemples d'implémentations. Enfin nous conclurons par une prise de recul globale sur le projet, en présentant ses limites ainsi que les différents enseignements que nous avons pu tirer de ce travail.

Table des matières

TRANSPARENCE REFERENTIELLE + fond vert

1 Description du projet et son organisation

1.1 Génération procédurale

La création d'objets manuellement peut être fastidieuse, c'est pourquoi la génération procédurale, est utilisée afin de créer des objets automatiquement. Cette méthode utilise des outils algorithmiques, il est donc très facile de produire des objets variés et en grand nombre sans passer des heures sur un outil de création numérique. La génération procédurale est beaucoup utilisé pour la génération de textures afin de colorier des grandes zones 3D comme des forêts ou des routes. Dans ce projet, nous allons développer une bibliothèque de génération procédurale d'image.

1.2 Choix de l'équipe

Nous avons décidés de réaliser notre projet en TypeScript, qui est pour nous un bon compromis entre la flexibilité du JavaScript et la lisibilité d'un langage typé.

Pour l'organisation, nous avions décidés de tenir un cahier de bord, dans lequel nous notions notre avancée à chaque fin de séance. L'objectif était d'avoir une trace de notre avancée afin de simplifier la rédaction du rapport. Après quelques semaines, le cahier de bord n'était plus mis à jour, ce qui avec le recul était une erreur.

Au niveau du développement, nous avons défini les éléments de base tous ensembles (image, couleur, générateur et filtre) afin que chaque membre ai bien compris le fondement de la bibliothèque. Nous nous sommes ensuite réparti les taches entre l'implémentation des différents filtres et générateurs et le développement des interfaces Node et web.

2 Structure des éléments

Le travail en équipe conjugué à l'utilisation d'un langage à typage fort tel que TypeScript nous a obligé à définir rapidement et de façon claire les différents éléments constituants le projet. Il a ainsi fallu caractériser les briques de base que sont les images et les couleurs qui les composent, puis les générateurs et les filtres composants notre bibliothèque, mais également les différentes interfaces nous permettant d' utiliser ces derniers correctement.

2.1 C'est quoi une couleur ?

Nous définissons une couleur suivant la notation RGBA. C'est un tableau de 4 valeurs entières allant de 0 à 255. Une couleur représentera un pixel de l'image.

$$Couleur = \begin{cases} rouge \in [0\ ;\ 255] \\ vert \in [0\ ;\ 255] \\ bleu \in [0\ ;\ 255] \\ alpha \in [0\ ;\ 255] \end{cases}$$

Nous implémentons des fonctions de manipulations de couleur au fur et à mesure des besoins rencontrés lors du développement de la bibliothèque.

2.2 C'est quoi une image ?

Il existe de nombreuses manière de représenter une image numériquement. Nous avions décidés de retenir deux implémentations : - représenter l'image par un tableau 2D - représenter l'image par une fonction
Après réflexion, c'est finalement la dernière implémentation qui à été retenue. Elle semblait pour nous plus pratique à manipuler, bien que moins intuitive. Nous avions besoins de stocker des informations sur l'image comme ses dimensions. Notre image est donc un objet ayant comme attribut sa largeur, sa hauteur et une fonction qui étant donné des coordonnées retourne une couleur.

$$Image = \begin{cases} width \in \mathbb{N} \\ height \in \mathbb{N} \\ f : \mathbb{N} \times \mathbb{N} \to Couleur \\ (x, y) \mapsto c \end{cases}$$

2.3 C'est quoi un générateur ?

Un générateur est un constructeur d'image. Étant donné une largeur et une hauteur, il retourne une image. D'autres arguments propres au générateur peuvent être fournis, par exemple des couleurs.

$$Generateur = \begin{cases} f : \mathbb{N} \times \mathbb{N} \times \cdots \to Image \\ (x, y, \dots) \mapsto img \end{cases}$$

2.4 C'est quoi un filtre ?

On peut définir simplement un filtre comme une boîte noire prenant en entrée une image et renvoyant en sortie une autre image. Cette dernière est issue de l'image originale à laquelle on aura appliqué divers transformations, pouvant jouer sur ses couleurs et également sa taille.

$$Filtre = \begin{cases} f : Image \times \mathbb{N} \times \mathbb{N} \times \cdots \to Image \\ (img, x, y, \dots) \mapsto img \end{cases}$$

Différents questionnement se sont posés sur la nature de ces transformations. Tout d'abord, étant dans une logique fonctionnelle, l'image renvoyée devait être une entité différente de l'image d'entrée afin de limiter les effets de bord. L'implémentation d'une image sous forme fonctionnelle vu en section 2.2 a permis de réaliser cela facilement. En effet, l'image étant caractérisé par la fonction renvoyant
–> insister sur la différence et ptn communs

2.5 Interfaces node et web

Nous avons implémenté les interfaces node et web. Même si Les deux interfaces sont très différentes, elle ont tout de même des bases communes, comme par exemple la définition d'une image. Nous avons voulu limiter au maximum la duplication de code. Une image peut être décrite par un fichier JSON, qui contient la suite de filtres et générateurs à appliquer afin d'avoir l'image finale.

Afin de simplifier le développement, nous avons mit en place des script de ...

L'interface node fonctionne à partir de la ligne de commande. Nous pouvons passer en argument le fichier JSON qui décrit l'image à créer. Le résultat est ensuite sauvegardé en PNG dans le dossier public.

La version web sert principalement aux tests, afin d'avoir un visuel

–> diff + ptn communs centralisation –> création à partir d'un json –> mise en place des règles npm, parcel

3 Analyse technique

–> Exemples d'implémentations

3.1 Générateurs

–> donner 4 exemples de géné

3.2 Filtres

–> donner 4 exemples de filtres

3.3 Comment combiner efficacement tout ça ?

-> accent sur la combinaison fonctionnelle

annexe : exemples d'images

3.4 Difficultés / limites du projet

Conclusion

Goodbye world