|  |
| --- |
| Génération d'image fractale |

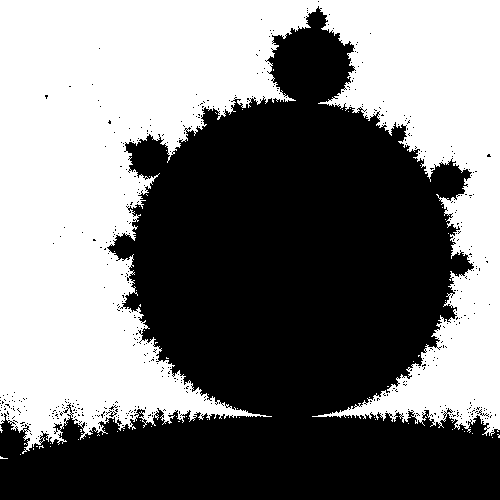


Image générée avec le programme  
(x : -0.3 à 0 ; y : 0.6 à 0.9 ; 150 itérations)

Auteur : HAUTIER Jonas

Email : [jonas.hautier@cpnv.ch](mailto:jonas.hautier@cpnv.ch)

Dernière modification : 31.05.2021

**Table des matières**

[1 Analyse préliminaire 3](#_Toc73093583)

[1.1 Introduction 3](#_Toc73093584)

[1.2 Objectifs 3](#_Toc73093585)

[1.3 Planification initiale 4](#_Toc73093586)

[2 Analyse / Conception 5](#_Toc73093587)

[2.1 Concept 5](#_Toc73093588)

[2.2 Stratégie de test 6](#_Toc73093589)

[2.2.1 Liste des scénarios 6](#_Toc73093590)

[2.3 Risques techniques 9](#_Toc73093591)

[2.4 Planification 9](#_Toc73093592)

[2.5 Dossier de conception 11](#_Toc73093593)

[3 Réalisation 14](#_Toc73093594)

[3.1 Dossier de réalisation 14](#_Toc73093595)

[3.1.1 Structure du répertoire « Code » 14](#_Toc73093596)

[3.1.2 Liste des paquets NuGet utilisés 14](#_Toc73093597)

[3.2 Résultat des tests effectués 14](#_Toc73093598)

[3.3 Erreurs restantes 14](#_Toc73093599)

[3.4 Liste des documents fournis 15](#_Toc73093600)

[4 Conclusions 16](#_Toc73093601)

[4.1 Objectifs 16](#_Toc73093602)

[4.2 Points positifs et négatifs 16](#_Toc73093603)

[4.3 Difficultés 16](#_Toc73093604)

[4.4 Suite pour le projet 16](#_Toc73093605)

[4.5 Comparaison entre la planification et le travail réalisé 16](#_Toc73093606)

[5 Glossaire 17](#_Toc73093607)

[6 Annexes 18](#_Toc73093608)

[6.1 Résumé du rapport du TPI 18](#_Toc73093609)

[6.2 Sources – Bibliographie 18](#_Toc73093610)

[6.3 Manuel d'installation 18](#_Toc73093611)

[6.4 Manuel d'utilisation 18](#_Toc73093612)

[6.5 Archives du projet 18](#_Toc73093613)

# Analyse préliminaire

## Introduction

L’objectif général de mon projet est d’implémenter un générateur d’image fractale.

Le programme sera développé en C#, utilisera SQLite comme système de gestion de base de données, sera versionné avec git et suivra la méthode de gestion de projet « Waterfall ».

Ce projet suivra les normes suivantes :

* La norme CPNV pour le MCD (Modèle Conceptuel de Données) et le MLD (Modèle Logique de Données) (accessible uniquement depuis la plateforme Moodle du CPNV)
* [Les normes de programmation et conventions de nommage en C#](https://github.com/ktaranov/naming-convention/blob/master/C%23%20Coding%20Standards%20and%20Naming%20Conventions.md) basées sur les recommandations de Microsoft

Ce projet est réalisé dans le condition du TPI, c.-à-d. :

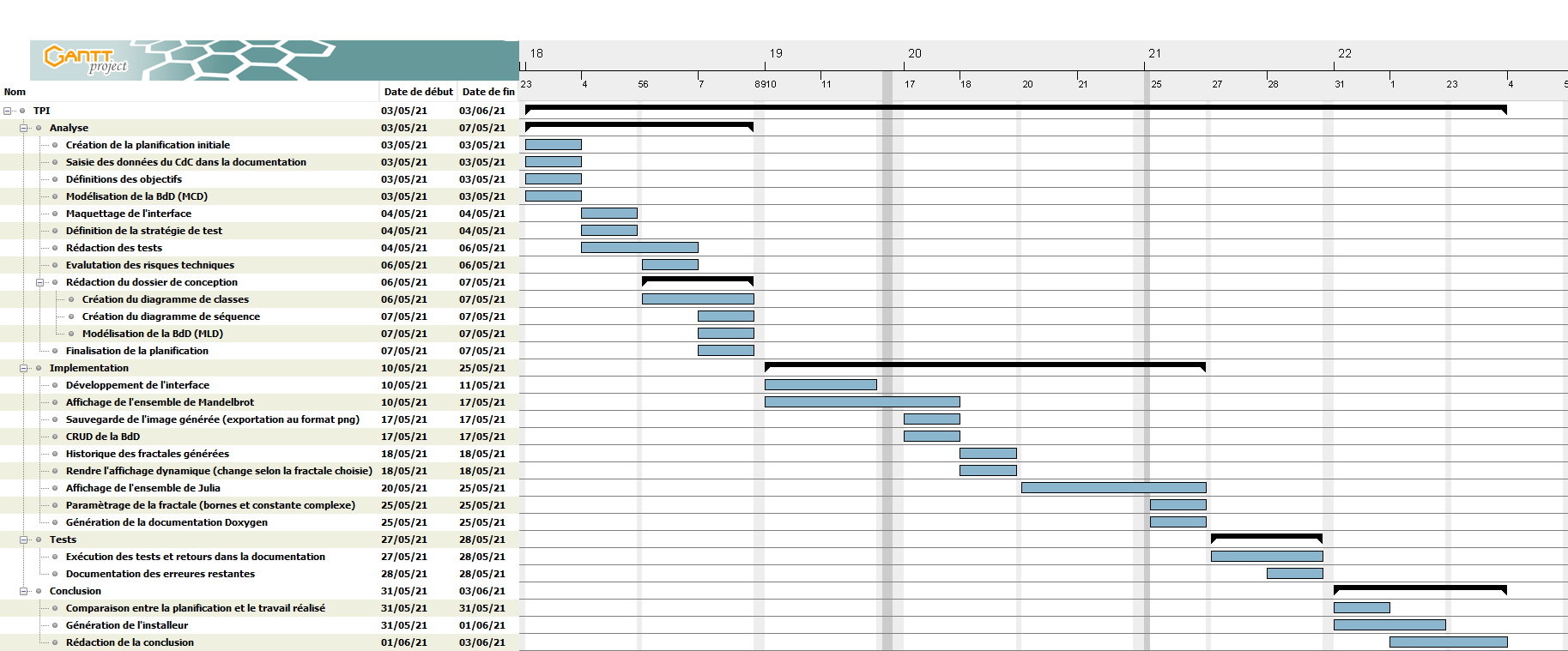
* Le projet est réalisé en 90h ;
* Le projet est individuel ;
* Le projet est commandé et évalué par le chef de projet ;
* Le rapport de projet est évalué par le chef de projet et deux experts ;
* La présentation et la défense sont évaluées par les experts.

Ce projet a pour but de confirmer que je suis capable d’exécuter un cahier des charges correspondant aux compétences d’un informaticien CFC.

## Objectifs

* Le programme est capable de générer l’une des fractales suivantes : l’ensemble de Mandelbrot et les ensembles de Julia de degré 2 ;
* Le programme donne à l’utilisateur la possibilité de définir les bornes réelles (x min et x max) et imaginaires (y min et y max) de la fractale générée ;
* Le programme adapte son affichage à la fractale choisie (c.-à-d., dans le cas des ensembles de Julia, il affiche un champ supplémentaire pour configurer *c*) ;
* Lorsque le programme a généré l’image de la fractale, il affiche le temps de génération de l’image ;
* Le programme donne à l’utilisateur la possibilité de sauvegarder l’image générée au format png ;
* Le programme est capable d’afficher un historique des derniers paramètres de génération utilisés ;
* Le programme donne à l’utilisateur la possibilité de régénérer une image à partir de l’historique ;

## Planification initiale



Ma planification initiale.  
J’ai prévu une semaine pour l’Analyse/Conception, deux semaines pour la Réalisation, deux jours pour les tests et trois jours pour la Conclusion.

# Analyse / Conception

## Concept

Voici les maquettes de mon programme :

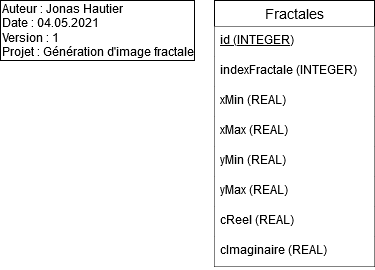
 

A gauche, la vue avec l’ensemble de Mandelbrot sélectionné.  
A droite, la vue avec l’ensemble de Julia sélectionné. Le paramétrage de « c » est visible.



La vue de l’historique. Régénère une entrée de l’historique quand on clique dessus.

Ci-dessous, le modèle conceptuel de données (MCD) de mon programme :



MCD

## Stratégie de test

Pour mon projet, j’ai choisi d’effectuer la majorité de mes tests après avoir réalisé tous les modules de mon programme.

Après avoir développé un module, je ferai uniquement les tests unitaires puis passerai à la suite s’il fonctionne. Une fois les modules développés, je referai les tests unitaires, pour m’assurer de la non-régression de mes modules, puis ferai les tests d’intégration et d’acception, basés sur les objectifs du projet.

Tous les tests seront exécutés sur mon poste de développement. Par précaution, j’effectuerai les tests d’acceptation également sur une autre machine du CPNV pour m’assurer que le programme valide les objectifs.

Pour les données de tests, je générerai une dizaine de fractales et vérifierai qu’elles sont correctes avec l’application XaoS ([lien de téléchargement de XaoS](https://github.com/xaos-project/XaoS/)).

Les tests seront faits par moi-même et ne seront pas automatisés.

### Liste des scénarios

J’ai groupé mes scénarios selon les modules de mon programme.

#### Interface

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Etant donné | Lorsque | Alors |
| 1.1 | Le programme vient d’être lancé | Le programme est affiché à l’écran | La fractale sélectionnée est « Mandelbrot » |
| 1.2 | La fractale sélectionnée est « Mandelbrot » | Je sélectionne la fractale « Julia » | Les champs « c réel » et « c ima. » deviennent visibles |
| 1.3 | La fractale sélectionnée est « Julia » | Je sélectionne la fractale « Mandelbrot » | Les champs « c réel » et « c ima. » deviennent invisibles |
| 1.4 | Le programme vient d’être lancé | Le programme est affiché à l’écran | Les champs suivants contiennent :  x min 🡪 -2  x max 🡪 2  y min 🡪 -2  y max 🡪 2 |
| 1.5 | La limite minimale est plus grande que la limite maximal dans l’axe des réels | Je clique sur « Générer ! » | La fractale a subi une symétrie axiale, avec l’axe de symétrie sur l’axe vertical |
| 1.6 | La limite minimale est plus grande que la limite maximal dans l’axe des imaginaires | Je clique sur « Générer ! » | La fractale a subi une symétrie axiale, avec l’axe de symétrie sur l’axe horizontal |
| 1.7 | Une fractale a été générée | Le programme a fini de générer la fractale | Le temps de génération de la fractale est indiqué |

#### Génération de l’ensemble de Mandelbrot

Pour tous les scénarios de cette section, la fractale sélectionnée sera toujours « Mandelbrot ».

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Etant donné | Lorsque | Alors |
| 2.1 | Tous les champs sont remplis avec des données valides | Je clique sur « Générer ! » | La fractale est générée dans les limites données |
| 2.2 | Un des champs suivants n’a pas été rempli : « x min », « x max », « y min » ou « y max » | Je clique sur « Générer ! » | Le programme affiche une pop-up informant l’utilisateur du champ qu’il a omis |
| 2.3 | La fractale sélectionnée est « Mandelbrot » et un des champs suivants a été rempli avec des données invalides : « x min », « x max », « y min » ou « y max » | Je clique sur « Générer ! » | Le programme affiche une pop-up informant l’utilisateur du champ dans lequel une valeur incorrecte a été saisie |

#### Génération de l’ensemble de Julia

Pour tous les scénarios de cette section, la fractale sélectionnée sera toujours « Julia ».

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Etant donné | Lorsque | Alors |
| 3.1 | Tous les champs sont remplis avec des données valides | Je clique sur « Générer ! » | La fractale est générée dans les limites données |
| 3.2 | Un des champs suivants n’a pas été rempli : « x min », « x max », « y min », « y max », « c réel » ou « c ima. » | Je clique sur « Générer ! » | Le programme affiche une pop-up informant l’utilisateur du champ qu’il a omis |
| 3.3 | Un des champs suivants a été rempli avec des données invalides : « x min », « x max », « y min », « y max », « c réel » ou « c ima. » | Je clique sur « Générer ! » | Le programme affiche une pop-up informant l’utilisateur du champ dans lequel une valeur incorrecte a été saisie |

#### Historique des générations

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Etant donné | Lorsque | Alors |
| 4.1 | Le programme vient d’être installé et exécuté | Je regarde dans l’historique | L’historique est vide |
| 4.2 | Une fractale vient d’être générée | Je regarde dans l’historique | Les paramètres de génération de la fractale ont été ajouté à l’historique |
| 4.2.1 | Une fractale « Mandelbrot » vient d’être générée | Je regarde dans l’historique | Les paramètres de génération sont sous cette forme :  *Type : Mandelbrot x :* **xMin** *à* **xMax** *y :* **yMin** *à* **yMax** |
| 4.2.2 | Une fractale « Julia » vient d’être générée | Je regarde dans l’historique | Les paramètres de génération sont sous cette forme :  *Type : Julia x :* **xMin** *à* **xMax** *y :* **yMin** *à* **yMax** *c :* **cReel** *+/-* **cIma.** *i* |
| 4.3 | J’ai cliqué sur le bouton « Générer ! » mais il y avait un problème dans les paramètres de génération | Je regarde dans l’historique | L’historique contient les mêmes données/!\ |
| 4.4 | L’historique contient déjà des données | Je ferme le programme puis le relance | L’historique contient les mêmes données |
| 4.5 | L’historique est affiché et contient des données | Je clique sur l’une des entrées de l’historique | Le programme régénère la fractale sur laquelle j’ai cliqué |

#### Sauvegarde de la fractale générée

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Etant donné | Lorsque | Alors |
| 5.1 | Une fractale a été générée et est affichée dans la zone de dessin | Je clique sur la zone de dessin | Le programme ouvre une fenêtre pour enregistrer l’image générée |
| 5.2 | Une fractale a été générée et sauvegardée sur l’ordinateur | Je cherche le fichier sur mon ordinateur | Je trouve le fichier dans le répertoire défini à l’enregistrement et le nom de fichier est celui demandé |
| 5.3 | Une fractale a été générée et sauvegardée sur l’ordinateur | J’ouvre l’image générée | L’image correspond à la fractale générée |
| 5.4 | Le programme me propose d’enregistrer l’image de la fractale générée | Je saisi un nom de fichier invalide (dans un système NTFS) | La fenêtre « Enregistrer Sous » refuse le nom saisi et demande pour . |

## Risques techniques

Ci-dessous, vous trouverez les risques techniques que je prévois pour mon projet. Après chaque risque décrit se trouvent les solutions mises en place pour réduire et surveiller les potentiels impacts du risque.

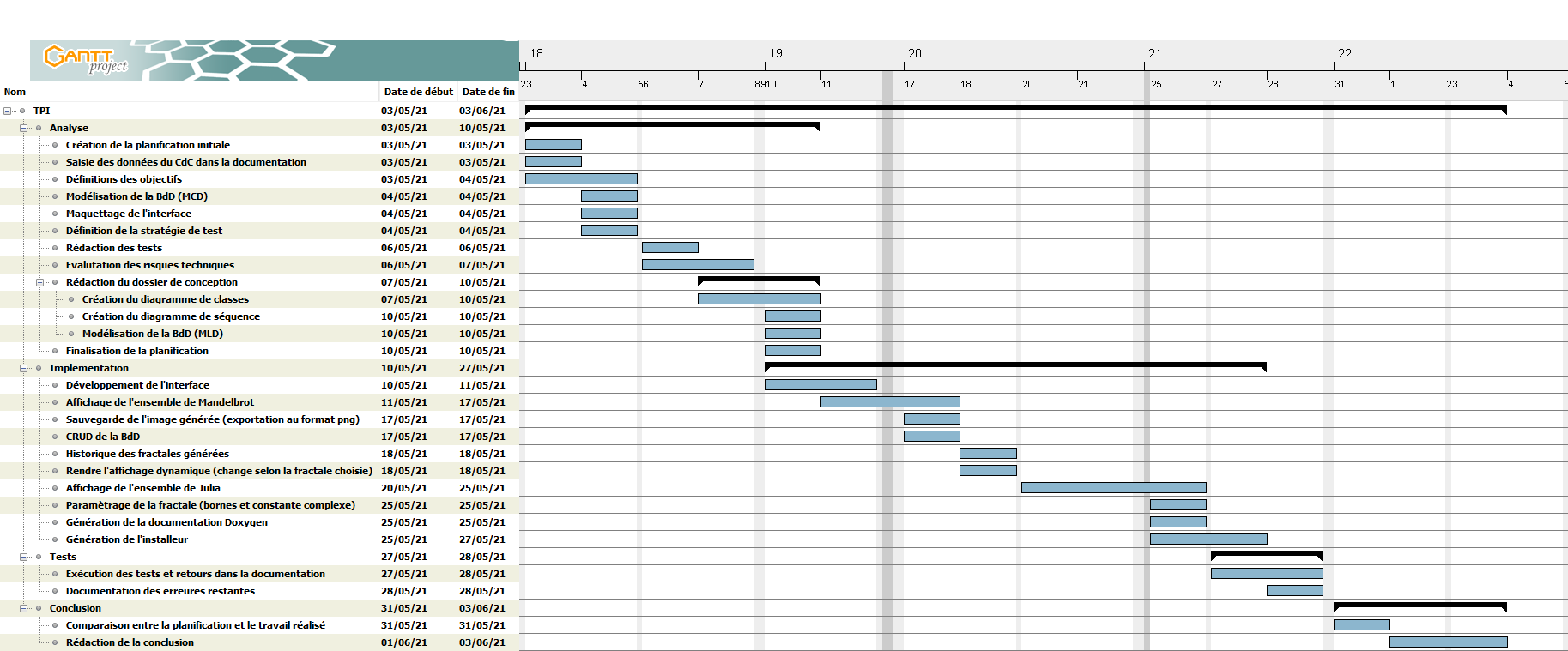
* Manipuler des nombres complexes en C# et les connecter correctement à mon parser :  
  C# a une structure pour gérer des nombres complexes et le parser que je compte utiliser n’accepte pas les nombres complexes. J’ai effectué des tests de mon côté pour m’assurer que je pouvais les connecter en décomposant les nombres complexes.
* Vitesse de génération de la fractale :  
  Récemment, je me suis documenté sur le parallélisme pour pouvoir l’implémenter dans ce projet.
* Mauvaise gestion du temps dans l’exécution du CdC :  
  Création d’une copie de la planification que je tiens à jour durant le déroulement du projet. Ce fichier, « TempsReel.gan », représente le temps réellement passé sur chaque tâche contrairement à la planification.

## Planification

J’ai choisi la méthode en cascade (Waterfall) comme méthodologie de gestion de projet car :

* je me sens plus à l’aise avec elle qu’avec la méthode Agile ;
* elle est plus adaptée à ma manière de travailler et de fonctionner : définir clairement ce qui va être fait avec un cadre bien délimité ;
* elle correspond parfaitement au cadre du TPI, c.-à-d. : elle est faite pour des projets où le client attend une seule livraison et où ses besoins ne vont pas de changer.

Voici la planification finale de mon projet :



Planification finale

## Dossier de conception

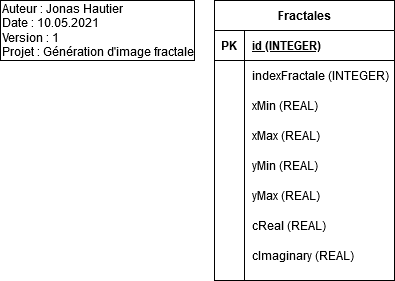
Pour le matériel hardware, j’utiliserai un poste de 4e année du CPNV :

|  |  |
| --- | --- |
| Propriété | Valeur |
| OS | Windows 10 |
| Architecture 32/64 bits | 64 bits |
| RAM | 16 Go |
| Nombre de processeurs logiques | 8 |

Ci-dessous, vous trouverez la liste des logiciels utilisés dans mon projet avec l’utilisation qui en sera faite et la version utilisée.

|  |  |  |
| --- | --- | --- |
| Logiciel | Utilisation | Version |
| 7-Zip | Créer des archives .zip | 16.04 |
| DB Browser for SQLite | Tester et modéliser ma BDD | 3.12.1 |
| diagrams.net (anciennement draw.io) | Faire les différents diagrammes (BDD et code) | En ligne |
| Doxygen | Générer la documentation logicielle | 1.9.1 |
| Excel 2019 | Tenir mon journal de travail | MS Office 2019 |
| GanttProject | Faire les planifications initiale et finale | 2.8.1 |
| GitHub Desktop | Publier sur GitHub et versionner avec git | 2.7.1 |
| Pencil | Faire les maquettes du programme | 2.0.5 |
| Visual Studio 2019 | Réaliser le programme et les tests | 16.4.2 |
| Word 2019 | Rédiger ma documentation | MS Office 2019 |
| XaoS | Tester la génération de fractales | 4.2.1 |

Voici le modèle logique de données (MLD) de mon projet :



MLD

Voici le diagramme de classes du programme :

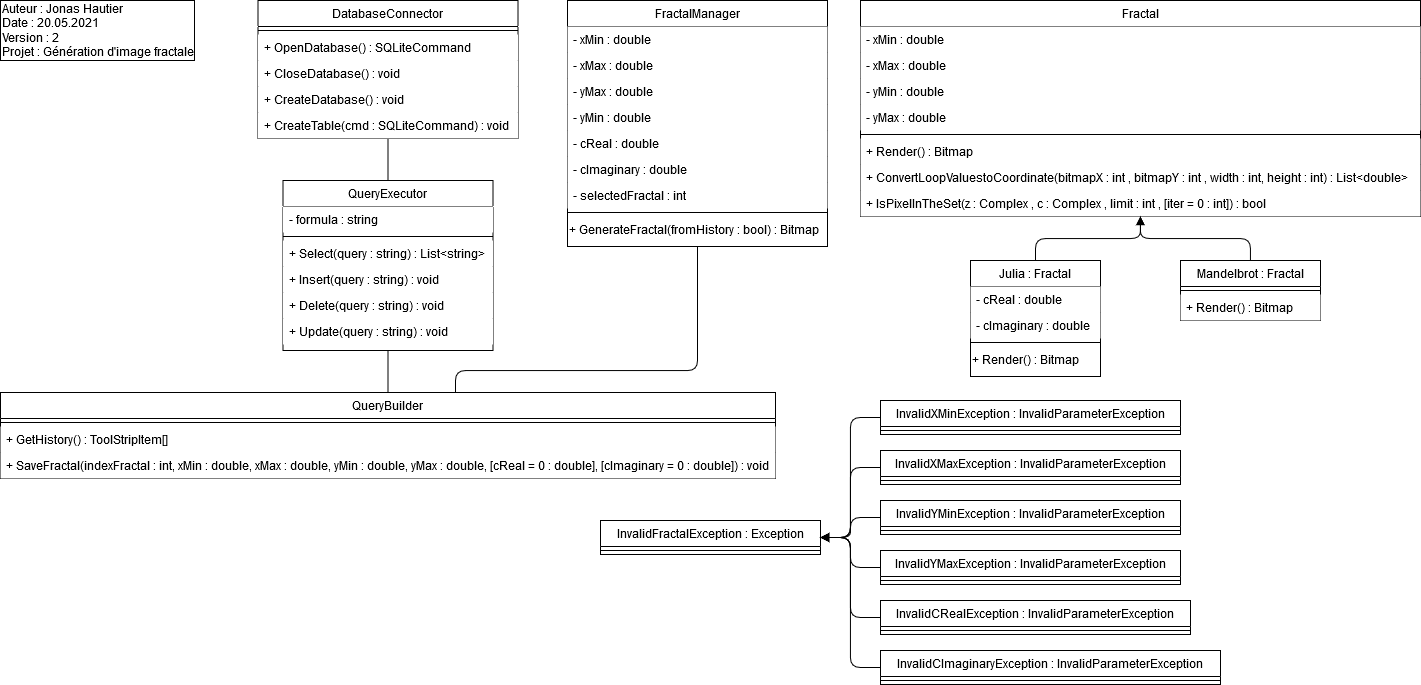


Diagramme de classes

Voici le diagramme de séquence du cas « On génère l’ensemble de Mandelbrot » :

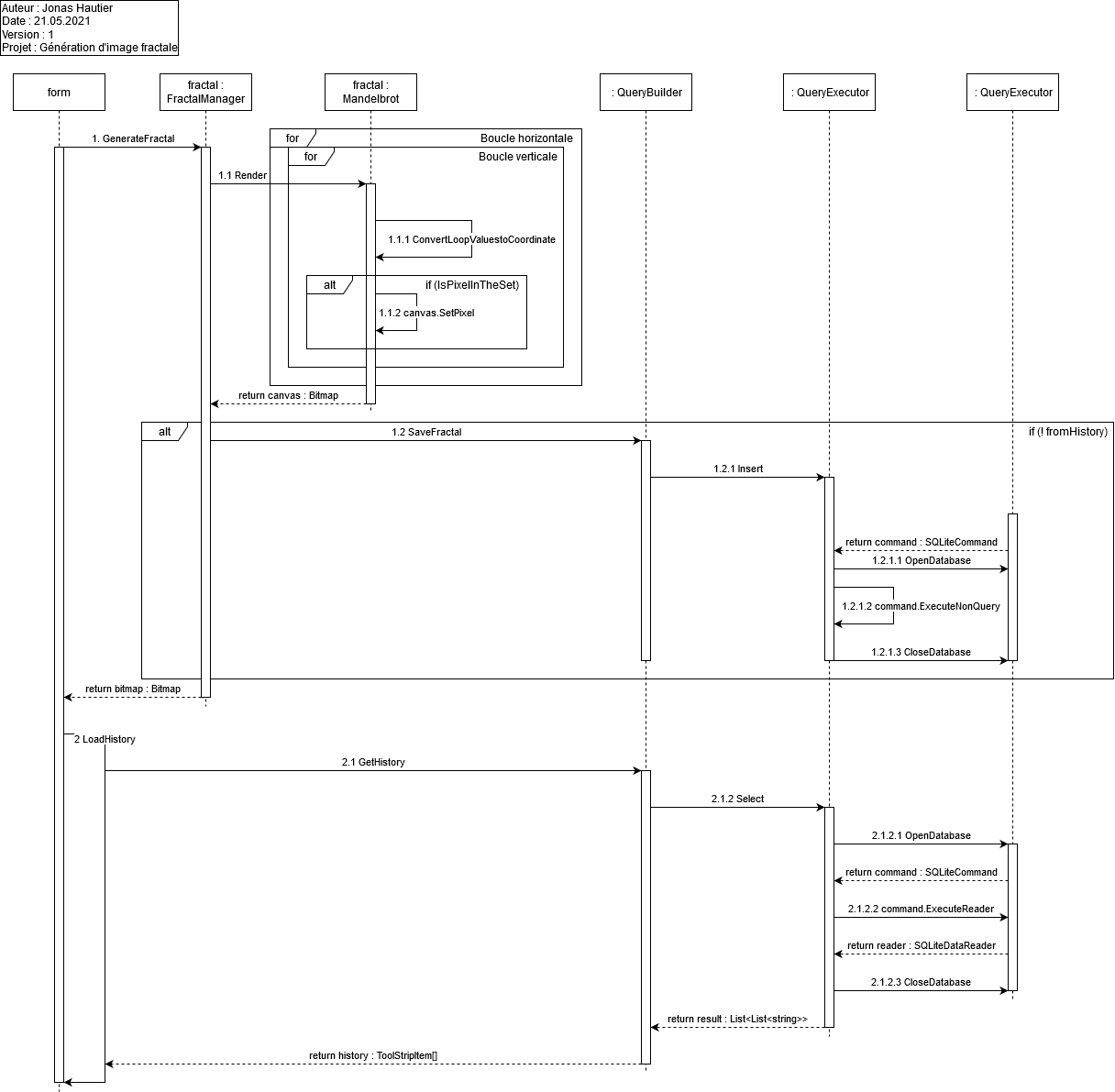


Diagramme de séquence du cas « On génère l’ensemble de Mandelbrot »

# Réalisation

## Dossier de réalisation

### Structure du répertoire « Code »

A côté de chaque fichier ou dossier importants se trouve une description de son utilité. Les dossiers sont en italiques pour mieux les différencier des fichiers. Des points de suspension indiquent que les fichiers ou dossiers se trouvant dans le répertoire ne nécessitent pas de description.

|  |  |  |  |
| --- | --- | --- | --- |
| Fichier / dossier | | | Description |
| *GenerationImageFractale* | | | Projet Visual Studio du logiciel |
|  | *Properties* | | Les propriétés du projet |
|  |  | ... |  |
|  | App.config | | Un des fichiers de configuration du logiciel |
|  | DatabaseConnector.cs | | Classe gérant la connexion à la base de données |
|  | Exception.cs | | Exception personnalisée pour le logiciel |
|  | Fractal.cs | | Classe représentant une fractale |
|  | FractalManager.cs | | Classe de gestion des fractales |
|  | GenerationImageFractale.csproj | | Fichier de projet du logiciel |
|  | GenerationImageFractale.sln | |  |
|  | Julia.cs | | Classe représentant une Julia |
|  | Main.Designer.cs | | Crée les composants de la fenêtre du logiciel |
|  | Main.cs | | La fenêtre du programme |
|  | Main.resx | |  |
|  | Mandelbrot.cs | | Classe représentant un Mandelbrot |
|  | Program.cs | | Point d’entrée du programme |
|  | QueryBuilder.cs | | Classe créant les requêtes pour la base de données |
|  | QueryExecutor.cs | | Classe exécutant les requêtes à la base de données |
|  | packages.config | | Fichier de configuration des packages NuGet. |
| *Setup* | | | Projet Visual Studio de l’installeur |
|  | Setup.vdproj | | Fichier de projet de l’installeur |

### Liste des paquets NuGet utilisés

|  |  |  |
| --- | --- | --- |
| Logiciels | Auteur(s) | Version |
| EntityFramework | Microsoft | 6.3.0 |
| MathParser.org-mXparser | Mariusz Gromada | 4.4.2 |
| Stub.System.Data.SQLite.Core.NetFramework | SQLite Development Team | 1.0.113.3 |
| System.Data.SQLite | SQLite Development Team | 1.0.113.7 |
| System.Data.SQLite.Core | SQLite Development Team | 1.0.113.7 |
| System.Data.SQLite.EF6 | SQLite Development Team | 1.0.113 |
| System.Data.SQLite.Linq | SQLite Development Team | 1.0.113 |

## Résultat des tests effectués

|  |  |  |
| --- | --- | --- |
| Module | ID du test | Test passé ? |
| Interface du programme | 1.1 | Oui |
| 1.2 | Oui |
| 1.3 | Oui |
| 1.4 | Oui |
| 1.5 | Oui |
| 1.6 | Oui |
| 1.7 | Oui |
| Génération de l’ensemble de Mandelbrot | 2.1 | Oui |
| 2.2 | Oui |
| 2.3 | Oui |
| Génération de l’ensemble de Julia | 3.1 | Oui |
| 3.2 | Oui |
| 3.3 | Oui |
| Historique des générations | 4.1 | Oui |
| 4.2 | Oui |
| 4.2.1 | Oui |
| 4.2.2 | Oui |
| 4.3 | Oui |
| 4.4 | Oui |
| 4.5 | Oui |
| Sauvegarde de la fractale générée | 5.1 | Oui |
| 5.2 | Oui |
| 5.3 | Oui |
| 5.4 | Oui |

## Erreurs restantes

Ci-dessous, vous trouverez la liste des erreurs restantes identifiées dans le programme :

* Il est possible de générer une fractale avec une borne inférieure plus grande que la borne supérieure.

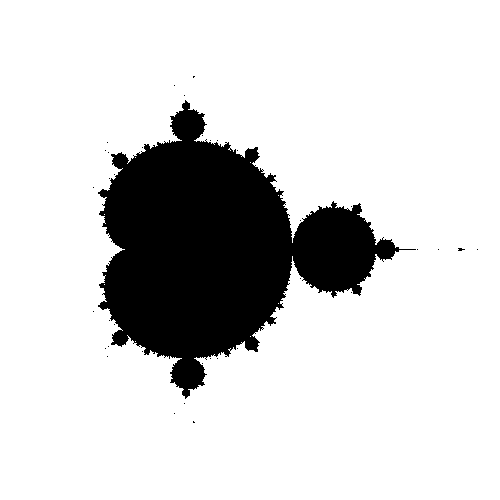
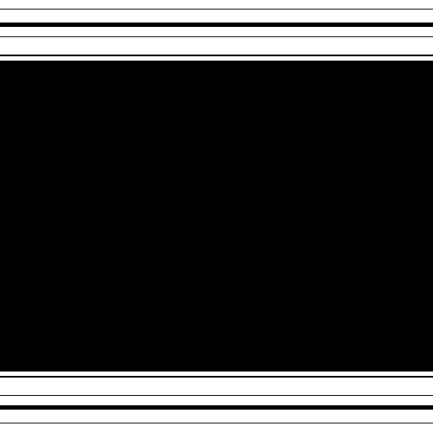
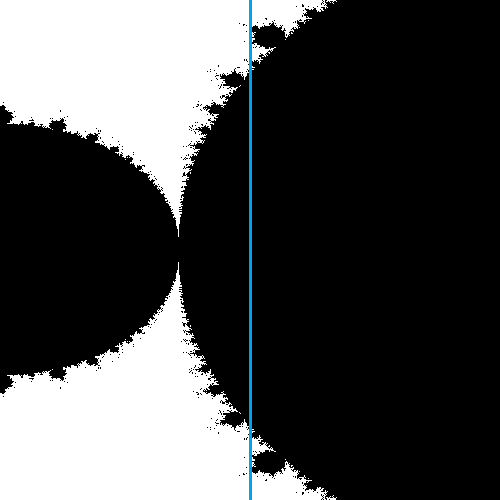


Image illustrant la symétrie d’axe, générée avec pour valeurs « x : 1 à -2 ; y : -1,5 à 1,5

Cela a pour conséquence d’effectuer une symétrie axiale sur la fractale, l’axe où les valeurs n’ont pas été inversée étant l’axe de symétrie. J’ai laissé cette fonctionnalité car il me semble qu’elle peut être utile.

* Il est possible de générer une fractale avec la même valeur comme borne inférieure et supérieure.

A gauche, un exemple de « tranche » du Mandelbrot. A droite, la « tranche », indiquée par la ligne bleue, dans l’ensemble de Mandelbrot. Voici les valeurs utilisées pour la génération :  
Image de gauche : « x : -0.65 à -0.65 ; y : -0.5 à 0.5 »  
Image de droite : « x : -1 à -0.3 ; y : -0.5 à 0.5 »

Cela a pour conséquence d’afficher une « tranche » de la fractale. Je ne pense pas que cette fonctionnalité puisse servir à beaucoup de monde, mais je ne vois pas non plus de raison d’empêcher ce cas.

## Liste des documents fournis

* Documentation
* Journal de travail
* Résumé du rapport du TPI

/\* à mettre :

* Les documents fournis au rendu (documentation, journal de travail, rapport du projet)

\*/

# Conclusions

/\* à mettre :

* Conclusion(s) générale(s) concernant le projet et sa réalisation

\*/

## Objectifs

/\* à mettre :

* Réflexion sur les objectifs atteints et, s’il y en a, les objectifs non-atteints

\*/

## Points positifs et négatifs

/\* à mettre :

* Réflexion sur les points positifs et négatifs du projet

\*/

## Difficultés

/\* à mettre :

* Réflexion sur les difficultés rencontrées, comment j’y ai réagi et comment j’aurais pu améliorer ma méthodologie
* Comparaison avec les risques techniques imaginés durant l’analyse

\*/

## Suite pour le projet

/\* à mettre :

* Les évolutions ou améliorations que je pense apporter au projet par la suite

\*/

## Comparaison entre la planification et le travail réalisé

/\* à mettre :

* Comparaison et réflexion sur les différences entre la planification et le travail réalisé

\*/

# Glossaire

# Annexes

/\* à mettre :

* Lister les documents annexes avec une description

\*/

## Résumé du rapport du TPI

/\* à mettre :

* 3 paragraphes : situation de départ, mise en œuvre et résultats
* Se référer aux critères d’évaluation pour plus de détails

\*/

## Sources – Bibliographie

Aide externes :

* 06.05.2021 – Louis Richard (camarade de classe) :  
  Louis m’a aidé pour détecter des cas qui n’auraient pas été pris en compte dans mes scénarios.
* 20.05.2021 – Frédérique Andolfatto (cheffe de projet)  
  Madame Andolfatto m’a aidé pour résoudre un problème de structure de classe. J’ajoutais une gestion de clic dans une classe liée à la base de données et cela rendait la classe inutilisable car elle devait être statique et non-statique à la fois.

Sites web :

* 18.05.2021 – <https://complex-analysis.com/content/julia_set.html>  
   <https://youtu.be/oCkQ7WK7vuY> (vidéo)  
  Ces sources m’ont permis de mieux comprendre comment je devais adapter mon algorithme de génération du Mandelbrot pour pouvoir générer l’ensemble de Julia.

/\* à mettre :

* Les sources utilisées au format :
  + Pour les livres 🡪 titre, auteur, date
  + Pour les sites web 🡪 URL, date de consultation
  + Pour les articles 🡪 revue, titre, auteur, date
* Les aides externes au format : noms, raison, date

\*/

## Manuel d'installation

/\* à mettre :

* Comment installer le programme à partir de l’archive livrée

\*/

## Manuel d'utilisation

/\* à mettre :

* Comment utiliser les fonctionnalités du programme

\*/

## Archives du projet

/\* à mettre :

* Structure de l’archive
* Média utilisé pour livrer l’archive

\*/