Ludovic D’ANjou-MAdore

Simon Lepage

Jérôme Pagé

Jonathan Simard

**Projet d'intégration en Sciences informatiques et mathématiques**

201-201-RE, gr.00001

Curved fractals

**Rapport final**

Travail présenté à

M. Walid Boulabiar

Département d'Informatique

Cégep Limoilou

Le 18 mai 2018

Table des matières

[1.Description du projet 2](#__RefHeading__789_1479271888)

[2.Développement 8](#__RefHeading__791_1479271888)

[3.Contribution personnelle 9](#__RefHeading__793_1479271888)

[4.Conclusion et perspectives du projet 10](#__RefHeading__795_1479271888)

[5.Annexes 12](#__RefHeading__797_1479271888)

# **Description du projet**

* 1. **Concepteurs : rôles, justifications et tâches**
     1. **Rôles :**

Directeur de produit : *Ludovic D'Anjou-Madore*

Scrum Master : *Jonathan Simard*

Secrétaire : *Simon Lepage*

Responsable des livrables : *Jérôme Pagé*

* + 1. **Justifications :**

*Ludovic D'Anjou-Madore* en tant que directeur de produit en raison de sa bonne vision d’ensemble de ce que le programme doit faire et pourrait faire. Est également celui qui en connaît le plus sur la géométrie riemannienne et les autres concepts mathématiques utilisés dans l’application.

*Jonathan Simard* en tant que Scrum master en raison de son expérience comme chef d’équipe à son travail.

*Simon Lepage* en tant que secrétaire en raison de ses bonnes capacités de rédaction.

*Jérôme Pagé* en tant que responsable des livrables en raison de son accessibilité et de son accès à Internet quasiment 24 heures sur 24.

* + 1. **Tâches (en plus de celles liées au rôle) :**

*Ludovic D’Anjou-Madore*: Travaille sur l’ensemble de l’application, surtout sur le modèle. Est responsable de la conception de l’algorithmie mathématique et de son développement (vérificateur de syntaxe pour les formules, calculs pour déformer la fractale, la déplacer et la zoomer, etc.). Travaille aussi sur les shaders openGL.

*Jonathan Simard* : Travaille sur l’intégration des différentes composantes nécessaires au bon fonctionnement de l’application (librairies, structure MVC). Contribue au code des shaders openGL.

*Simon Lepage*: Travaille sur l’ensemble de la documentation nécessaire pour le projet (backlogs de produit et de sprints, rapports) et sur l’interface de l’application (FXML, style CSS, intégration au code).

*Jérôme Pagé*: Travaille sur l’interface de l’application (FXML, style CSS, intégration au code, ajout de fonctionnalités) et sur l’utilisation faite par l’utilisateur de l’application (événements de la souris, interaction avec l’interface, …).

* 1. **Objectifs**

Un programme de visualisation de fractales dans un espace non-euclidien (doté d’une certaine courbure) permettrait d’introduire une compréhension de concepts mathématiques abstraits et complexes (courbure d’une variété riemannienne et l’influence d’une métrique sur celle-ci) en permettant à l’utilisateur de visualiser en temps réel la courbure qui découle d’une métrique spécifiée. Une compréhension des fractales est acquise en même temps, et l’application permet d’appliquer les principes de quelques concepts mathématiques (géométrie riemannienne, champ tensoriel métrique).

* Conception et réalisation d’une application programmée principalement en Java permettant l’affichage et la manipulation de fractales.
* Intégration de connaissances provenant de domaines connexes abordées par le programme d’études (ici, les mathématiques) à l’informatique.
* Utiliser les méthodes Agile et Scrum pour organiser et gérer quotidiennement la progression du projet.

* 1. **Description détaillée**

CurvedFractals est un logiciel libre et ouvert, développé dans le cadre du projet d’intégration du DEC en Sciences Informatiques et Mathématiques (SIM).  
Ce logiciel, ultime expérience esthétique, prétend humblement permettre de visualiser l’interaction entre deux objets mathématiques, soit les fractales et les variétés riemanniennes (au sens très faible du terme: la responsabilité de doter la variété d’une métrique riemannienne est laissé à l’utilisateur, ainsi que celle de s’assurer que la paramétrisation qu’il fournit donne effectivement une variété différentielle, mais le résultat s’affiche même si rien ne respecte les propriétés).

Effectivement, en plus d’afficher une fractale d’équation arbitraire spécifiée par l’utilisateur, celui-ci peut spécifier la structure (tenseur métrique et paramétrisation) de l’espace dans lequel cette fractale sera affichée. Évidemment, l’utilisateur peut aussi se déplacer et zoomer sur la fractale (de manière automatique ou manuelle). Les esthètes confirmés seront ravis de pouvoir pigmenter la fractale de la façon qu’ils le souhaitent, tandis que les technologues jouiront de l’affichage (presque) en temps réel grâce à l’utilisation active (OpenGL + shaders) de leur GPU.

* + 1. **Concepts présents dans l’application**

L’application met en application premièrement les concepts mathématiques suivants :

* Les Variétés Riemanniennes dans leurs constructions.
  + La spécification d’une métrique comme généralisation du produit scalaire
    - au point p
  + Calcul d’une immersion de l’espace qui contient les fractales dans l’espace écran
    - [https://en.wikipedia.org/wiki/Riemannian\_manifold#The\_pullback\_metric](#The_pullback_metric)

Les fractales seront transformés point par point en fonction de la métrique pour obtenir l'effet de courbure.

* + La transformation se fait par l'endomorphisme de courbure construit à partir du tenseur de courbure (tenseur métrique).

On remarque bien, à la complexité des équations en jeu, l'utilisation de concepts complexes, comme la dérivation covariante jusqu'au nombre effarent de terme, une fois la notation d'Einstein enlevée. La transmission de ceux-ci au GPU (carte graphique) sera un problème technique en soi. Sans cette transmission, la seule alternative est de les calculer sur le CPU (processeur), perdant beaucoup des avantages d'une carte graphique...

Ensuite, les concepts informatiques utilisés dans l'application sont principalement ceux reliés à l'imagerie par ordinateur. En effet, pour bénéficier du plein potentiel des cartes graphiques modernes, la majorité des calculs concernant le calcul de la fractale et la transformation de celle-ci en fonction de la courbure seront exécutés sur le GPU, par l'entremise des *shaders* (openGL), des petits programmes exécutés pour chaque sommet d'un modèle et pour chaque pixels de l’écran, parallélisés, qui permettent d'effectuer les transformations (ici appliquer la courbure sur la fractale) et effets (ici l'affichage de la fractale en soi) voulus de manière extrêmement efficace. De plus, envoyer une partie substantielle des calculs sur la carte graphique a des effets bénéfiques d'un point de vue technique par exemple, en permettant d'alléger grandement la structure du modèle MVC.

* + 1. **Fonctionnement**
       1. **Mécanismes internes**

Pour afficher une fractale en haute définition, il faut, sur chaque pixel de l’écran, itérer l’équation de la fractale et évaluer si cette suite tend à diverger (après environ 600 itérations, on peut approximativement considérer que si elle ne diverge pas maintenant, elle ne le fera pas). Donc, sur un écran d’une résolution de 1080p (le standard actuel), ça commence à être lourd, étant donné qu’il faut itérer sur 2 073 600 pixels… Heureusement, le progrès technologique étant ce qu’il est, une bonne partie des ordinateurs relativement modernes sont dotés d’une carte graphique accélérant beaucoup ce genre de calculs.

Toutefois, la seule manière de fournir les équations nécessaires aux shaders est de les écrire directement dans leur fichier. Sur le CPU, cela aurait été évité, mais cela aurait engendré la perte du temps réel lors de l’affichage. Notre approche nous rajoute un délai de compilation des shaders (les shaders étant assez légers, l’effet est négligeable), mais nous garde en temps réel ailleurs.

Donc, il est impératif que les fichiers genericShaderFrag.glsl, genericMat.j3md, initialFrag.glsl et initialMat.j3md restent dans /src/vue. De plus, les fichiers *.*j3md doivent rester inchangés, tandis que les fichiers .glsl doivent garder les commentaires qui indiquent la position des équations à inscrire, en plus de ne pas déranger le code des méthodes de lesquels ils sont contenus. Tout le reste peut être modifié sans altérer le fonctionnement de l’affichage. (Il est recommandé de ne pas changer les systèmes de coordonnées dans les shaders, c-à-d les normalisations des variables d’entrées, car cela rend la translation et le zoom inconfortables d’utilisation).

* + - 1. **Fonctionnement mathématique**

TODO : CTRL C+V du README.md sur GitHub, section Fonctionnement mathématique, FONCTIONS À CORRIGER (syntaxe Latex and shit)

* 1. **Comparatif avec l’existant et l’idée de départ**

L’application existante finale correspond à l’idée de départ du projet. Les fonctionnalités, l’interface et l’algorithmie conçues au début se trouvent dans l’application. Seulement une fonctionnalité a été enlevée du développement et ne fait pas partie de l’existant, soit l’utilisation d’une texture 3D. Cette fonctionnalité a d’ailleurs été remplacée par une fonctionnalité similaire permettant d’atteindre le même résultat, qui était d’utiliser le tenseur métrique afin de déformer les longueurs de la fractale affichée. Somme toute, nous avons bien évalué le temps nécessaire à la réalisation de l’ensemble de l’application. Peu de sacrifices ont été faits, et le fonctionnement global de l’application est satisfaisant.

* 1. **Technologies utilisées**
* **Outils et langages :** Eclipse, SceneBuilder, GitHub, GitHub Desktop, Trello, Notepad++, Visual Paradigm, Microsoft Office (Word, PowerPoint et Project), GPU avec support pour openGL 3.2+, Java, FXML, GLSL, CSS3
* **Librairies :** [jMonkey](https://github.com/jMonkeyEngine/jmonkeyengine) (openGL), [JavaFX](https://docs.oracle.com/javafx/2/api/index.html), [JME3-JFX](https://github.com/empirephoenix/JME3-JFX), [exp4j](https://github.com/fasseg/exp4j)

# **Développement**

**TODO**

* 1. **Méthodes de développement**
     1. **Vue d’ensemble du fonctionnement du cadre de développement**

Le fonctionnement de notre cadre de développement est simple. Nous utilisons Trello pour organiser les quelques Sprints qui ont lieu et les tâches à effectuer lors de chaque Sprint. Ces tâches sont déterminées selon les User Stories conçues au début du projet.

TODO – À CONTINUER

* + 1. **Déroulement des travaux**
    2. **Difficultés rencontrées**
    3. **Outils de développement**
  1. **Organisation du travail**
     1. **Fonctionnement de l’équipe**
     2. **Difficultés rencontrées et découvertes**
     3. **Forces, faiblesses et acquis**
  2. **Validation de l’application**

# **Contribution personnelle**

**TODO**

* 1. **Ludovic D’Anjou-Madore**
     1. **Apprentissages pour résoudre les problèmes que j’ai dû rencontrés**

1) Pour effectuer un zoom et une translation sur un ensemble de points sur la carte graphique, on ne fait pas une translation vectorielle et un zoom matriciel séparer, on trouve une matrice qui encode et le zoom et la transformation. Ça règle les problèmes de systèmes de coordonnées et ça améliore grandement les performances sur le shader.

2) Le tenseur métrique ne fait pas de déformation visible une fois que l’immersion est déjà spécifiée (l'immersion est exprimé en coordonnées qui compensent déjà la déformation du tenseur aussi exprimé dans ses coordonnées).

3) Les avantages/désavantages entre pré-calculer la fractal sur le CPU ou le faire en temps reel sur le GPU

CPU:possibilité zoom infini, mais pas temps reel

GPU : temps réel, mais pas zoom infini

* + 1. **Évaluation de ma contribution à l’équipe**

Ma plus grande contribution à l'équipe est l'apport en connaissances mathématiques et l'orientation du projet.

* + 1. **Évaluation de notre** ☭) **équipe et de mon expérience en équipe**

Équipe très sympathique, bon design. Je recommencerai le travail avec la même

équipe, le travail demandé est fait par les coéquipier au bon moment.

Chaque membre de l'équipe à quelque chose de spécifique et d'important à apporter. En plus, tous les membres sont conscient de l'importance de publier

open-source, ce qui n'est pas le cas de toutes les équipes...

* 1. **Jonathan Simard**
     1. **Apprentissages pour résoudre les problèmes que j’ai dû rencontrés**
     2. **Évaluation de ma contribution à l’équipe**
     3. **Évaluation de notre** ☭) **équipe et de mon expérience en équipe**
  2. **Simon Lepage**
     1. **Apprentissages pour résoudre les problèmes que j’ai dû rencontrés**
     2. **Évaluation de ma contribution à l’équipe**
     3. **Évaluation de notre** ☭) **équipe et de mon expérience en équipe**
  3. **Jérôme Pagé**
     1. **Apprentissages pour résoudre les problèmes que j’ai dû rencontrés**

1) Au départ, dans l’étape de conception du projet, nous voulions pouvoir ouvrir le menu de côté avec un clic gauche de la souris en plus de pouvoir translater dans la fractale à l’aide d’un dragg du clic gauche. Je me suis vite rendu compte que lorsqu’un dragg était effectuer, le menu s’ouvrait aussi. J’ai tenté de régler ce problème en posant une condition sur « l’event » reçu, mais puisque lorsqu’un drag est effectuer deux MouseEvent(BUTTON1, MOUSE\_DRAGGED) sont lancé, il est difficile de différencier un dragg d’un clic. J’ai réglé ce problème en changeant au clic droit pour ouvrir le menu. Par contre, il serait possible de détecté s’il y a eu déplacement de la souris un certain nombre de temps après le clic afin de déterminer si l’utilisateur faisait un clic ou un dragg.

2) Pour la fonctionnalité du zoom continu, je voulais changer l’image du bouton zoom en fonction de si le zoom était en cours ou non. Pour se faire, j’ai d’abord essayé d’utiliser un thread, j’ai ensuite réalisé en regardant de plus près l’API que le problème pouvait être réglé d’une autre façon. J’ai donc utilisé un service sur lequel je place des « listeners » permettant de savoir si le service est lancé, en cours ou terminé.

3) J’ai aussi approfondi mes connaissances en css, puisque nous voulions une interface plus intéressante que celles faites dans les sessions précédentes. J’ai appris comment changer l’apparence d’un bouton lorsque qu’il est « hovered » en plus d’apprendre comment changer l’apparence d’une Node de l’interface lorsqu’un event est détecté sur celle-ci.

* + 1. **Évaluation de ma contribution à l’équipe**

Ma plus grande contribution à l’équipe a été de permettre à mes autres coéquipiers d’utiliser une interface intuitive et de permettre le lien en l’interface (entrées, évents) et le modèle. Bien sûr, ma bonne humeur constante et ma ponctualité respectable on aussi fait de moi un très bon coéquipier de projet.

* + 1. **Évaluation de notre** ☭) **équipe et de mon expérience en équipe**

Je suis très content de mon choix d’équipe. Malgré de petites lenteurs en périodes d’examens, nous avons travailler à un bon rythme tout au long de la session. Le point le plus fort de l’équipe était notre la répartition du travail qui faisait en sorte que chacun pouvait progresser dans son travail indépendamment du travail fais par les autres membres de l’équipe. De plus, dans les autres cas où une collaboration était requise, tous les membres de l’équipe étaient à l’écoute des autres. Je suis fier du travail accompli par moi et mes coéquipiers.

# **Conclusion et perspectives du projet**

* 1. **Évaluation de la production**

Nous sommes globalement assez satisfaits de l’application que nous avons développée au courant de la session, CurvedFractals. Les fonctionnalités que l’on souhaitait implanter l’ont toutes été dans la version finale, sauf une seule minoritaire : le *scaling* de l’interface selon la résolution (1080p par défaut). Certains détails auraient pu être améliorés, mais le temps nous a manqué. Nous aurions aimé apporter des correctifs à la translation lorsque le zoom est élevé pour la rendre plus confortable d’utilisation. D’autres bugs mineurs auraient pu être corrigés : l’entrée occasionnelle du caractère « r » dans les zones d’entrées de l’interface lorsqu’on réinitialise la fractale et le zoom automatique qui effectue des zooms à l’infini. Il aurait été aussi intéressant d’ajouter un menu de réglages permettant de changer la résolution de l’application. Malgré ces moindres faiblesses, nous jugeons que notre production a bien été effectuée, et le développement s’est bien déroulé. C’est ce bon développement et la bonne conception de départ qui ont donné les forces à notre application :

* Excellente réactivité de notre application. Les calculs permettant d’afficher la fractale et de la déformer sont effectués dans des shaders openGL grâce à JMonkey. Cela permet une vitesse de calcul supérieure en utilisant les ressources du GPU au lieu du CPU. Toutes les transformations apportées à la fractale (forme, courbure, couleur et zoom) s’effectuent presque instantanément.
* Belle interface visuelle. L’application d’un style visuel CSS permet de doter l’interface d’un petit look futuriste inspiré de *The Matrix*. Un coup d’œil y est d’ailleurs fait dans l’application (voir le bouton du tenseur métrique). Son style minimaliste est inspiré du style visuel de Windows 10.
* Précision et rigueur mathématique. Toutes les déformations et les transformations effectuées sur la fractale affichée sont fidèles à la vérité.
  1. **Évaluation des acquis**

La réalisation de notre projet CurvedFractals nous a permis d’acquérir des connaissances dans les domaines suivants :

* Géométrie riemannienne;
* Programmation en GLSL (shaders openGL);
* Intégration de librairies au code et leur utilisation (jMonkey, JME3-JFX et exp4j);

Ces acquis, tant mathématiques qu’informatiques, nous ont permis de développer une application **intégrant** des connaissances de ces deux domaines connexes, satisfaisant ainsi le but du projet d’intégration.

* 1. **Avenir du projet**

La suite de notre projet pourrait être son utilisation et son amélioration par d’autres membres de la communauté informatique. En effet, tout le code source de CurvedFractals est ouvert et gratuit sur GitHub (<https://github.com/LudoDm/CurvedFractals>). Cela pourrait donc intéresser des chercheurs ou d’autres programmeurs qui désirent un outil du genre, ou désirent l’améliorer afin de l’intégrer dans d’autres logiciels.

# **Annexes**

* 1. **Diagramme UML**

**TODO – reverse code avec Visual Paradigm**

* 1. **Principales vues de l’interface**

**TODO – voir les captures d’écran dans le dossier \Rapport final\vues interface**

* 1. **Références et liens utilisés**

**Références :** Notes de cours,le grand monde qu’est Internet, connaissances acquises grâce aux précédents cours de programmation du DEC en SIM.

**Librairies et liens utilisés :**

[jMonkey](https://github.com/jMonkeyEngine/jmonkeyengine), [JavaFX](https://docs.oracle.com/javafx/2/api/index.html), [JME3-JFX](https://github.com/empirephoenix/JME3-JFX), [exp4j](https://github.com/fasseg/exp4j)

<https://emmanueldurand.net/spherical_projection/>

<https://www.shadertoy.com/view/MlfSz7>

<https://www.opengl.org/discussion_boards/showthread.php/199306-lat-long-spherical-projection-using-vertex-shader>

[http://www.madore.org/~david/programs/#prog\_kerr-image](#prog_kerr-image)

<http://acko.net/blog/how-to-fold-a-julia-fractal/>

<https://stackoverflow.com/questions/27356577/scale-at-pivot-point-in-an-already-scaled-node>

<https://stackoverflow.com/>

<https://trello.com>

<https://github.com>