**« Backlog » de sprint #4**

Produit : Curved Fractals

Conçu par : JeeGo

## Nom des membres :

**(JP) : Jérôme Pagé** – Secrétaire

**(SL) : Simon Lepage** – Responsable des livrables

**(JS) : Jonathan Simard** – Scrum master

**(LDM) : Ludovic D’Anjou-Madore** – Directeur de produit

## Échéancier :

**Période** **:** 6 au 20 avril 2018

**Nombre d’heures estimé** **:** 30h

## Légende :

* Vert, indique que ces items sont réalisés.
* Jaune, indique que ces items sont en cours de réalisation.
* Rouge, problème ou questionnement important qui demande une rencontre d’équipe.
* Aucune couleur, indique que ces items ne sont pas encore faits ou commencés, **on peut toujours les enrichir mais il faut le consentement de toute l’équipe**.

## « Backlog » de sprint

|  |  |
| --- | --- |
| **3** | |
| Acteur ou rôle : | Utilisateur |
| Scénario ou story : | **(US3)** En tant qu’utilisateur, je veux pouvoir spécifier les composantes du champ tensoriel métrique et voir les effets de la courbure de celui-ci sur une fractale |
| Détail ou description : | 1. Recevoir les chaînes de caractères de l’utilisateur représentant les composantes du champ tensoriel métrique.    1. Qui et temps       1. Nom(s) : JP, SL       2. Temps : 2h    2. Préconditions       1. Présence d’un onglet dans l’interface permettant la saisie des composantes du champ tensoriel métrique.    3. Règles d’affaires       1. Stocker les chaînes entrées par l’utilisateur dans les TextField correspondants dans des variables String.    4. Règles d’affaires alternatives       1. Pas de bugs ou autre. Rien à corriger.    5. Tests d’acceptation       1. Vérifier que la saisie des composantes s’effectue correctement et qu’elles sont facilement accessibles par le shader.    6. Post-conditions       1. Je suis capable de spécifier les composantes à utiliser pour le champ tensoriel métrique. 2. Valider la structure et la syntaxe des composantes du tenseur métrique et les interpréter pour construire le champ tensoriel métrique en Java.    1. Qui et temps       1. Nom(s) : JS, LDM       2. Temps : 2h    2. Préconditions       1. Méthode permettant de construire le champ tensoriel métrique en Java dans le modèle de l’application (dans une nouvelle classe qui manipule et représente le tenseur métrique).    3. Règles d’affaires       1. Calculer la nouvelle longueur du vecteur normal selon la surface affichée (fractale).       2. Déformer la fractale selon le nouveau vecteur normal (transforme la longueur des segments).    4. Règles d’affaires alternatives       1. Pas de bugs ou autre. Rien à corriger.    5. Tests d’acceptation       1. Vérifier que le champ tensoriel métrique construit est valide : le vecteur normal est changé et on peut voir sur la fractale que les longueurs des segments ont été changées.    6. Post-conditions       1. L’application peut construire le champ tensoriel métrique en Java selon les composantes que j’ai spécifiées. 3. ~~Construire une texture 3D~~     1. ~~Qui et temps~~       1. ~~Nom(s) : JS, LDM~~       2. ~~Temps : 4h~~    2. ~~Préconditions~~       1. ~~Être capable d’accéder à un champ tensoriel métrique valide, stocké dans une variable.~~    3. ~~Règles d’affaires~~       1. ~~Stocker la texture 3D dans une variable qui sera envoyée au shader.~~       2. ~~Utiliser des 2D Array Textures ou un tableau 2x2 de matrices 2x2 sur OGL 4.3 et +. En évaluant le tenseur à chaque pixel (les composantes x et y de la texture représentent la position de chaque pixel et le niveau (pour le 2DTextures Array) représente la nième composante du tenseur a un point (pour le tableau 2x2, en fixant la position, on obtient une matrice, la représentation locale du tenseur métrique).~~    4. ~~Règles d’affaires alternatives~~       1. ~~To update…~~    5. ~~Tests d’acceptation~~       1. ~~Vérifier qu’une texture 3D peut être construite et que ses attributs correspondent à ceux attendus selon les composantes du champ tensoriel.~~    6. ~~Post-conditions~~       1. ~~L’application est capable de construire une texture 3D selon le champ tensoriel métrique.~~ 4. Envoyer l’équation de la courbure au shader et calculer la déformation pour chaque pixel grâce à JMonkey.    1. Qui et temps       1. Nom(s) : JS, LDM       2. Temps : 4h    2. Préconditions       1. ~~Être capable de construire une texture 3D.~~    3. Règles d’affaires       1. Recevoir l’équation écrite par l’utilisateur via l’interface graphique (TextField)       2. L’équation est envoyée au shader et ce dernier calcule la déformation pour chaque pixel de la fractale.    4. Règles d’affaires alternatives       1. Pas de bugs ou autre. Rien à corriger.    5. Tests d’acceptation       1. Vérifier visuellement que la fractale se déforme selon la structure de données reçue.       2. Vérifier dans les shaders temporaires (runtime) si l’équation de la courbure est interprétée et le calcul de déformation se fait.    6. Post-conditions       1. Je suis capable de voir la déformation causée par la texture 3D sur la fractale affichée à l’écran. Cette déformation change selon les composantes du champ tensoriel. |
| Tests d’acceptation : | 1. Afficher l’ensemble de Mandelbrot et confirmation visuelle de la déformation.    1. Vérifier que les composantes du champ tensoriel métrique entrées par l’utilisateur construisent un tenseur métrique qui transforme la fractale affichée (transforme les longueurs selon le tenseur).    2. Vérifier que l’équation pour la paramétrisation sphérique entrée par l’utilisateur déforme la fractale en la courbant (vérifier que la courbure agit comme un globe terrestre vu du dessus). |
| Complexité : | 7 |
| Effort : | 3j/personne **ou** 4 |
| Commentaires : | -Lors du changement de tenseur spécifié par l’utilisateur, on recalcule une texture et on met à jour la référence dans le shader  -Lors d’un zoom, on doit « zoomer » aussi les coordonnés d’accès à la représentation en mémoire du tenseur. Donc, si on utilise une texture 3d, l’interpolation se fait automatique pour chaque niveau. Si on utilise un tableau de matrice, dans chaque direction, on doit interpoler entre les composantes. (par exemple : voir l’interpolation linéaire dans la description des concepts) |