Procell

-

Projet Java

Robin Jungers
Chamseddine Kaddouri
Alice Jestin

2015 - 2016

I - Rappel de la commande

Nous devions réaliser un projet libre, mais couvrant néanmoins les notions abordées en cours. Le langage utilisé doit être Java, et en utilisant par exemple Processing. Tous les attributs des classes doivent être déclarés private (ou éventuellement protected) et *final* dans la mesure du possible. Les objets sont non mutables dans la mesure du possible. Sauf en phase de mise au point, aucune entrée/sortie dans les méthodes autres que celles de lecture/saisie et d'écriture/affichage n'est autorisée en mode console. Le code doit être commentée avec les annotations nécessaires à la production d'une Javadoc.

II - Le projet

Principe général :

Notre projet est un jeu d'exploration génératif en 2D créé à partir d'une recherche sur Twitter. Vous incarnez **Cell**, une petite cellule, et devez explorer l'environnement dans lequel vous vous trouvez. Tout en interagissant avec vous, les autres cellules que vous rencontrez vous donnent les derniers Tweet concernant la recherche Twitter que vous avez saisie. Il n'y a ni défaite ni victoire dans ce jeu.

Environnement :

L'environnement du jeu est d'inspiration sous-marine, dans un milieu abyssal. Il est construit à chaque ouverture de partie à partir du mot clé que l'on donne dans le menu de démarrage. Celui ci permet de recueillir des données sur Twitter. Avant chaque partie, le joueur renseigne un mot clé en fait interprété comme une recherche sur Twitter, et autour duquel sont récupérés des tweets, des retweets.

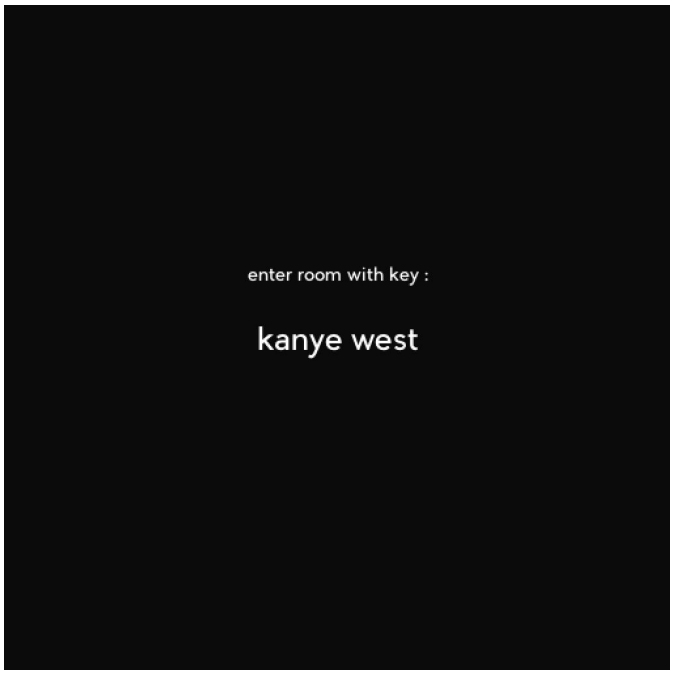


Figure 1 *: Menu de démarrage*

L'environnement, composé d'entités vivantes simples (bactéries, cellules, virus, microbes), récupère ces informations et donne à chaque entité (sauf Cell) un tweet associé. Ses dimensions sont construites à partir de la popularité du message en question ; il se créé donc une forme de jeu de découverte, à la recherche des plus grosses entités.

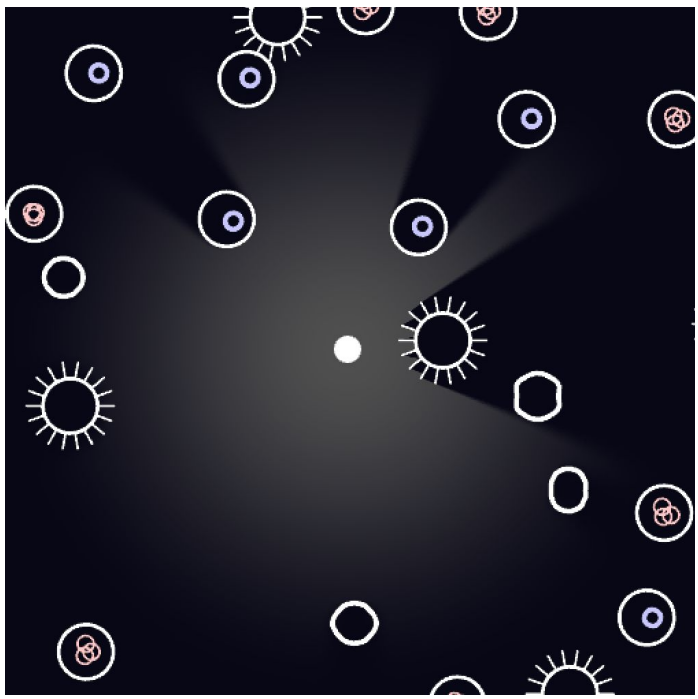


Figure 2 *: Environnement du jeu*

L’entité contrôlée par le joueur peut interagir avec cet environnement. A chaque fois qu’il touche une cellule, celle-ci va réagir, par exemple en grossissant, ou en se rétractant. La cellule touchée affiche alors un tweet en adéquation avec le mot-clé donné au départ.

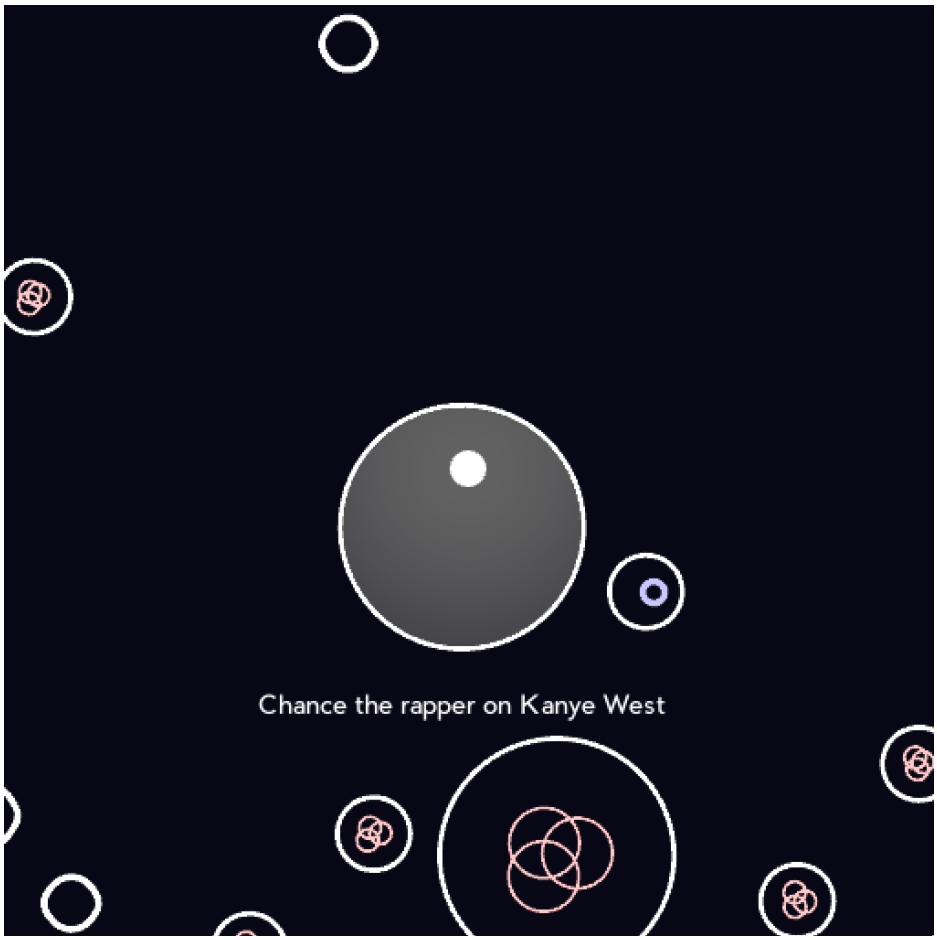
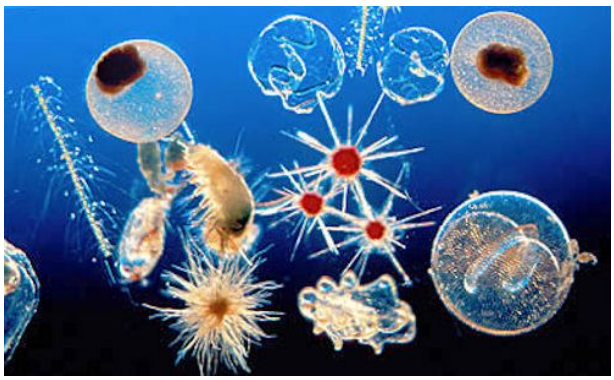


***Figure 3** : Affichage des Tweets associés aux cellules touchées*

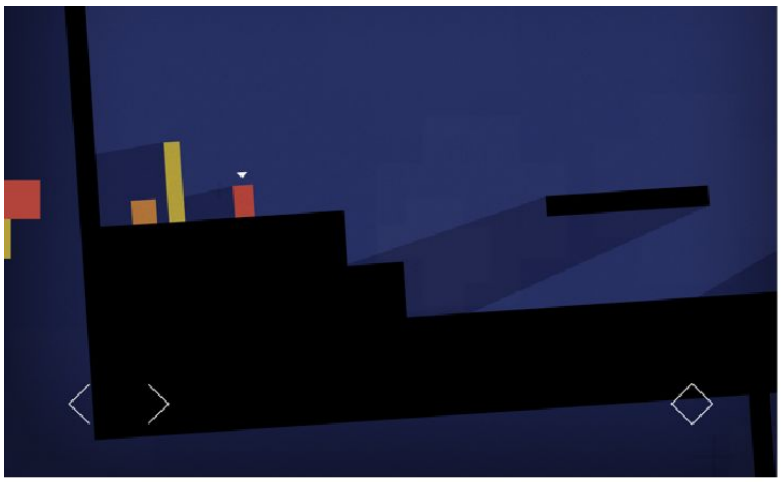
III - Inspirations

Nous souhaitions réaliser à la foi un projet technique mais aussi esthétique que nous pourrions présenter aux entreprises avec lesquels nous pourrions travailler à l'avenir. Pour ce faire, nous avons réalisé quelques recherches graphiques.

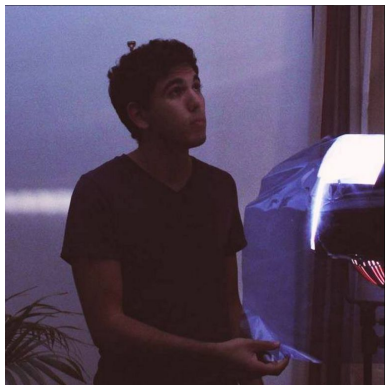
Nous nous sommes inspirés du monde vivant marins et de cellules réelles de planctons pour dessiner nos cellules, ainsi que du jeu *Cell.*



Pour la lumières nous nous sommes inspirés de différents jeu, notamment du jeu *Thomas Was Alone*, un jeu en Flat design avec une lumière multidirectionnelle très brut.



IV - Présentation de l’équipe


Robin Jungers, jeune artiste dans les milieux de la programmation et de l’image, appelé l’entremetteur, il marie l’art visuel aux techniques les plus pointus de la programmation, dans l’optique de toujours créer de nouvelles choses.

Chamses Kaddouri, alias le réseau social humain, c’est un bon communicant. Sa capacité a communiquer l’aide à cerner son publique, et à créer du contenu audiovisuel qui saura toucher et convaincre le plus grand nombre.

Alice Jestin, surnommée Colorgirl, elle aime tout ce qui touche à l’art numérique et la création graphique. Toujours un crayon en main, un projet qui lui plaît est un projet faisant appel à son imagination.

Pour l’organisation, nous avons travaillé ensemble afin de pouvoir nous aider mutuellement et pouvoir régler plus facilement les difficultés que les autres pouvait rencontrer. Nous avons utilisé *Github* pour partager les missions et mettre en commun les avancées.

V - Techniques employées

Technologies : *Java + Librairies Processing + API Twitter*

Le projet fait appel d'une part à des structures de données destinées à récolter les informations de la part de l'API Twitter, et d'autre part de structures de données d'ordre graphique et ludique, pour la gestion des entités du jeu et des interactions possibles.
L'ensemble du projet est monté sur la base du template Processing, fournissant des fonctions d'initialisation, de dessins, de paramètres d'affichages, etc.

Les classes personnages

Procell sert à mettre en place le jeu (fenêtre, personnages, dessins, lumière...)

Entity est un entité, possédant une taille, position, rapidité, destination … Elle possède plusieurs fonction :

- La fonction *update()* est une méthode d'**Entity** qui est redéfinie pour chaque entité. Elle sert à mettre à jour les attributs de l'entité.
- La fonction *applyPhysics()* permet de donner les principales propriété physique à chaque **Entity** : calcul de vitesse et de position, collisions avec les cellules trop proches.
- *setSurroundings()* est une methode qui créé un tableau qui permet de garder en mémoire les cellules autour de l'entité.
- *setReference()* permet de garder en mémoire l'entité **Cell** pour les possibles interractions.
- *displayOn()* permet d'afficher l'ensemble des graphismes d'une entitée sur une fenetre de type PGraphics.
- talk() est une methode qui écrit à l'écran le tweet auquel l'entité est associée.

Entity est en fait une classe maire, de laquelle héritent l'ensemble des autres :

Cell : C'est l'entité avec laquelle on joue, elle est unique. Elle possède une intensité de lumière, et elle se déplace par rapport à la position de la souris sur la fenêtre.

Microbe : Ce sont des Batterie de type cilié (avec des poils). Lorsque **Cell** s'approche d'elle, les cils proche d'elle se rétractent

Virus : C’est une cellule possédant 3 noyaux. Lorsque Cell passe dessus, ses noyaux éclatent. Lorsqu’elle s’écarte, ses noyaux réapparaissent doucement.

Grower : C’est une cellule qui grossi à l’approche de Cell. Si cette dernière est l'intérieur, sa vitesse est diminué.

Bacteria : Elle est très malléable, lorsque Cell s’approche d’elle, son ellispse se déforme de plus en plus.

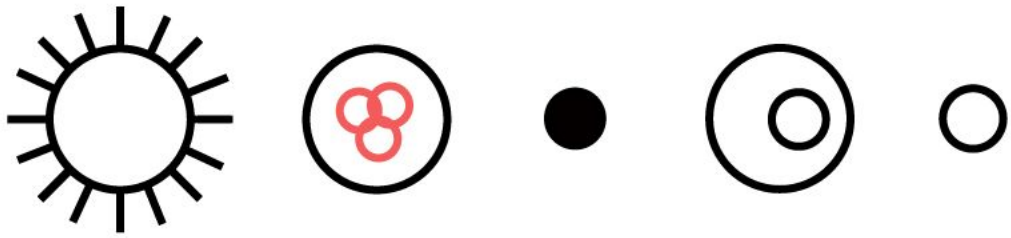


Figure 4 *: Virus, Microbe, Cell, Grower et Bacteria sans animation*

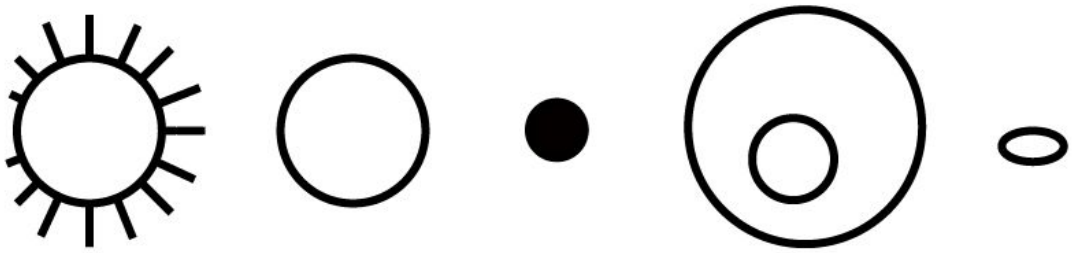


Figure 5 *: Virus, Microbe,Cell, Grower et Bacteria après animation*

La classe de gestion de Twitter

TwitterReader : Cette classe récupère les informations de Twitter;

Un des objectifs principaux de projet était la création d'un univers génératif, à partir donc d'une collection d'objets définie au dessus.
Pour ce faire, l'idée proposée ici est d'utiliser directement les tweets recueillis sur Twitter, autour d'un mot clé défini par l'utilisateur au début. Les entités sont alors construites portant les messages de ces tweets, et leur dimensions traduit la popularité du message.

L'outil permettant d'utiliser les services de Twitter est une libraire Java, Twitter4j. En effet, le réseau social propose bien une API disponible en ligne, mais il est nécessaire d'avoir recours à une authentification par un compte développeur au préalable ; c'est cette étape de sécurité qui est facilitée par Twitter4j.
Les messages sont ainsi extraits à la manière d'une simple recherche, desquels ont peut obtenir l'auteur, de nombre de "likes", etc.

Les classes de gestion du jeu

Game : Gère le menu, la configuration du jeu.

La Classe Game forme une base au jeu, bien qu'elle ne soit pas responsable de création des entités ou même des graphisme, mais permet l'affichage d'un menu en début de partie est l'appel au services Twitter, afin de lancer le jeu.

Animator : Elle gère les animations des entités.

Un des domaines important des interactions est l'animation ; ces sont elles qui fluidifient le jeu, et le rendant plus naturel que de simples changement d'états. Ces modifications nécessitent souvent une approche séquentielle, avec une condition d'entrée et de sortie ; la classe statique Animator est justement chargée de centraliser ces fonctions.
Une seule fonction est réellement instanciée dans le jeu ; les autres sont en fait basées sur un chronomètre, peu pratique ici, mais qui constituent potentiellement un axe d'amélioration pour des animations plus poussées, non linéaires dans le temps.

Rendu graphique et Optimisation

L'ensemble des graphismes du jeu est dessiné dans des fenêtres graphiques proposées par Processing, des objets PGraphics. Le processus est plus lourd qu'un affichage classique ; nous avions dans un premier temps implémenté un programme traitant toutes les entités à chaque boucle sans condition, ce qui faisait perdre un temps de calcul considérable par rapport aux réels besoins ; nous avons donc fait en sorte de ne traiter que les cellule visibles à l'écran, améliorant ainsi les performances.

L'utilisation de ses fenêtres graphique est en fait directement lié à l'implémentation du système de lumière. Celui ci est traité par des shaders sur la carte graphique (programmés en GLSL), puisque Processing fonctionne avec OpenGL ; Pour pouvoir appliquer les algorithmes et créer une carte des ombres, il est nécessaire d'avoir une texture de l'ensemble de la scène.

Ces shaders fonctionnent par paire ; un premier est chargé de former une "Shadow Map" d'une unique dimension, en passant d'un système polaire centré sur la lumière à un système cartésien, le second évalue les distances d'un pixel et déduit par comparaison avec la shadow map son éclairage.
Cet algorithme est basé sur celui proposé par Matt DesLauriers (https://github.com/mattdesl/lwjgl-basics/wiki/2D-Pixel-Perfect-Shadows).
Puisqu'étant appliqué pixel par pixel sur la carte graphique, le système est très efficace dans ses calculs mais reste potentiellement coûteux en ressources sur les machines les moins puissantes ; il a été nécessaire d'optimiser le processus, notamment en ne traitant qu'un pixel sur deux. Ceci peut mener parfois à un mauvais calcul de certains rayons de lumières, qui peuvent traverser les lignes les plus fines.
On applique pour finir un flou dépendant de la distance au centres aux pixels des ombres, pour créer un rendu plus proche de la réalité.

On obtient donc, à terme, une première couche des ombres, sur laquelle on ajoute la couche des dessins réels des éléments du jeu. En dessous, une fenêtre graphique basique forme un arrière plan.

VI - Améliorations

Nous aurions aimé apporter quelques améliorations pour compléter notre projet, notamment de la musique et des sons pour les interactions. Nous aurions aussi aimé pouvoir mieux développer les animations, en leur rajoutant par exemple une accélération.
Le jeu développé est totalement ouvert aux développement de nouvelles interactions graphiques. C'est ce qui constitue principalement un axe possiblement d'amélioration.

Il existe des également services, comme *sentiment140.org,* qui donnent une valeur ajouté a l'utilisation de Twitter. *Sentiment140* analyse par exemple les tweets et renvoie l'humeur qu'ils dégagent ; nous aurions pu utiliser cette API pour modifier l'environnement en fonction de l'humeur, mais les auteurs de l'outil demandent de s'enregistrer auprès d'eux auparavant.