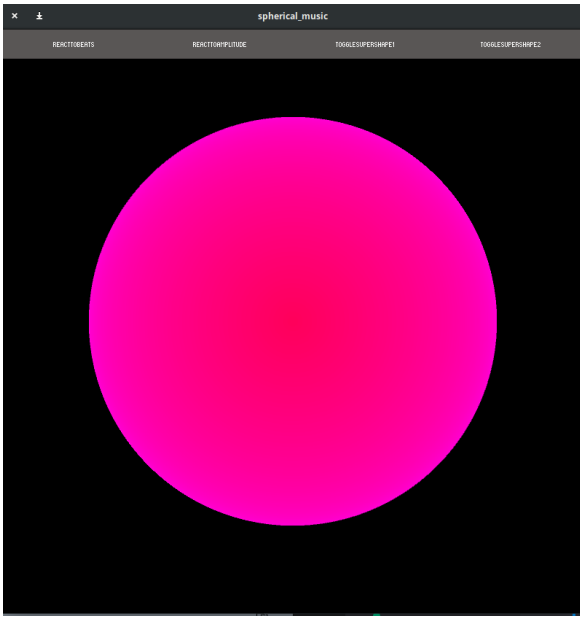
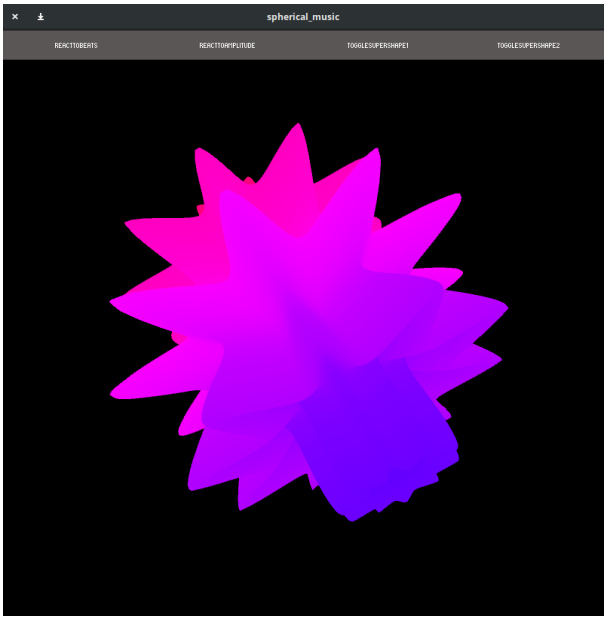
# Création Numérique

**CHABANE HUGO**
**BILLAUD MAEL**
**486N**



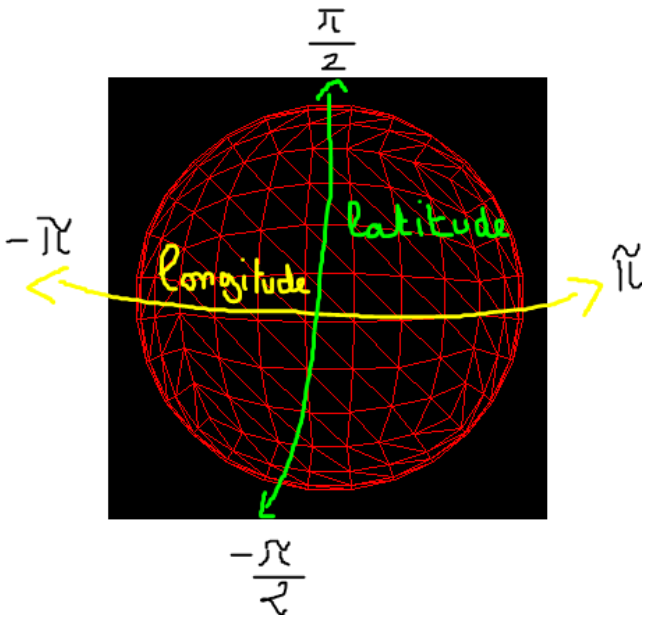

**Année: 2020-2021**
**Université de Nantes, L2  Informatique**

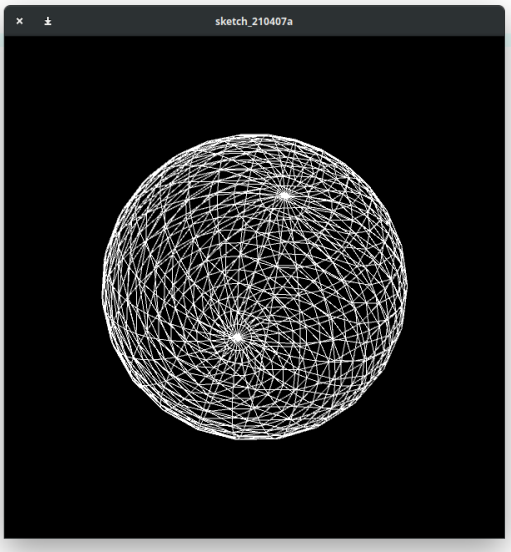
UNIVERSITÉ DE NANTES

*Dans le monde qui nous entoure de nombreux objets abordent une forme sphérique ou des courbures. Lors de ce projet nous avons décidé de comprendre comment une sphère peut être représentée informatiquement dans un espace à 3 dimensions. Cette sphère doit pouvoir se déformer en SuperShape en fonction de l'amplitude ou des beats ("pulsation") d'une musique. Une SuperShape (SuperFormula) est une généralisation de la formule de la superellipse qui a été proposée par Johan Gielis. Il suggère que la formule puisse être utilisée pour décrire de nombreuses formes et courbes complexes que l'on trouve dans la nature. Pour ce projet nous avons utilisé Processing qui est un environnement de développement libre adapté à la création plastique et graphique interactive et au graphisme de données.*

## 1 - Réalisation d'une Sphère dans un Espace 3D

La sphère étant la base essentielle de notre projet nous nous sommes documentés afin de comprendre comment la réaliser sous Processing. Le logiciel nous fournissait une fonction permettant de créer une sphère mais nous ne pouvions pas agir sur ses divers paramètres. Cependant nous pouvons remarquer que cette **sphère était composée de séries de triangles** reliés par leurs arêtes ou coins communs. Les triangles étant connectés entre eux, **l'ordinateur n'a pas besoin de spécifier plusieurs fois les trois vertex** pour chaque triangle. En effet, les "bandes de triangles" (**triangle strips**) peuvent être utilisées en informatique pour spécifier des objets complexes de manière à **utiliser efficacement la mémoire et le temps de traitement**. La sphère n'est qu'une illusion pour notre œil, les triangles créant des primitives ( ensemble de vertex qui forme une entité 3D unique) dont les surfaces semblent être des courbes lisses.




Nous avons dû trouver des équations permettant de définir une sphère en fonction de certains angles. Pour définir des coordonnées sur le globe nous utilisons une longitude et une latitude. Nous avons donc décidé d'utiliser deux boucles imbriquées afin de calculer les points de **nos triangles en fonction de deux coordonnées polaires représentant la longitude (comprise entre -PI et PI) et la latitude (comprise entre -PI/2 et PI/2)** et d'un paramètre représentant le degré de précision pour notre sphère soit le nombre de triangles qui la composeront. Il est possible de faire varier cette précision pour se rendre compte qu'il faut de nombreux triangles afin d'obtenir une courbure naturelle.

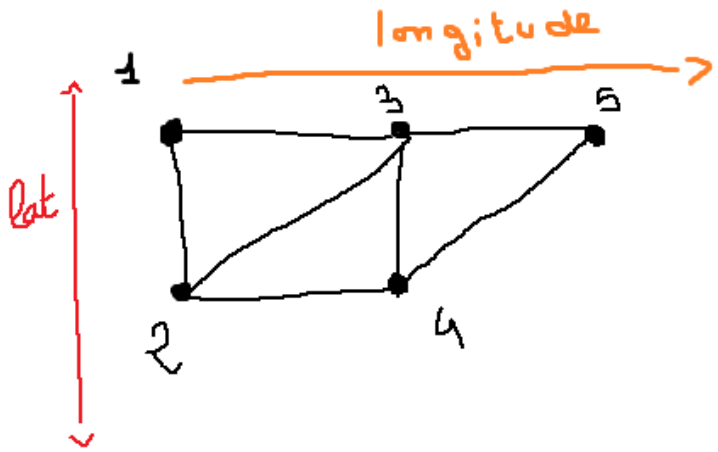
```
for(int i = 0; i < TRIANGLES_AMOUNT+1; i++)
{
  // lat = theta  long = phi
  float theta = map(i, 0, TRIANGLES_AMOUNT, -HALF_PI, HALF_PI);

  for(int j = 0; j < TRIANGLES_AMOUNT+1; j++)
  {
    float phi = map(j, 0, TRIANGLES_AMOUNT, -PI, PI);
```

Il nous a fallu ensuite à partir de ces angles et du rayon de la sphère désirée obtenir les coordonnées cartésiennes (x,y,z) situant les points dans un espace à 3 Dimensions.

$$x = r\cos\varphi\ \sin\theta,$$
$$y = r\sin\varphi\ \sin\theta,$$
$$z = r\cos\theta.$$

*Dans cette formule l'angle Theta représente la latitude et l'angle Phi représentant la longitude*

A partir des ces coordonnées cartésiennes il nous est alors possible de relier les divers points afin de créer **une série de triangles horizontaux. Pour chaque latitude nous allons donc devoir créer une nouvelle série de triangles** pour laquelle chaque triangle sera **composé d'un premier point sur une longitude suivi d'un autre sur la latitude suivante puis de la longitude suivante et ainsi de suite (voir schéma ci-dessous).**




Nous avons donc décidé d'utiliser un tableau à 2 dimensions de PVector pour stocker les valeurs (x,y,z) des différents points de la sphère. Le premier indice correspond aux points sur la latitude et le second ceux sur la longitude. Afin de ne pas consommer trop de temps de calcul nous avons décidé de séparer les calculs des points de la sphère de la fonction la dessinant graphiquement. Ainsi nous pouvons calculer les points de notre sphère seulement quand nous en avons besoin. Pour que notre sphère apparaisse correctement et entière il a fallu ensuite faire en sorte que notre **série de triangles se termine où nous l'avons commencé.** Pour cela nous avons dû ajuster les indices de notre tableau et ceux des boucles lorsque nous dessinons notre sphère.

Par la suite, afin de transformer notre Sphère en Supershape à l'aide de nos angles Phi et Thêta nous nous sommes basés sur les formules du site de Paul Bourkes.
http://paulbourke.net/geometry/supershape/.

Definition of supershape in 2 dimensions.

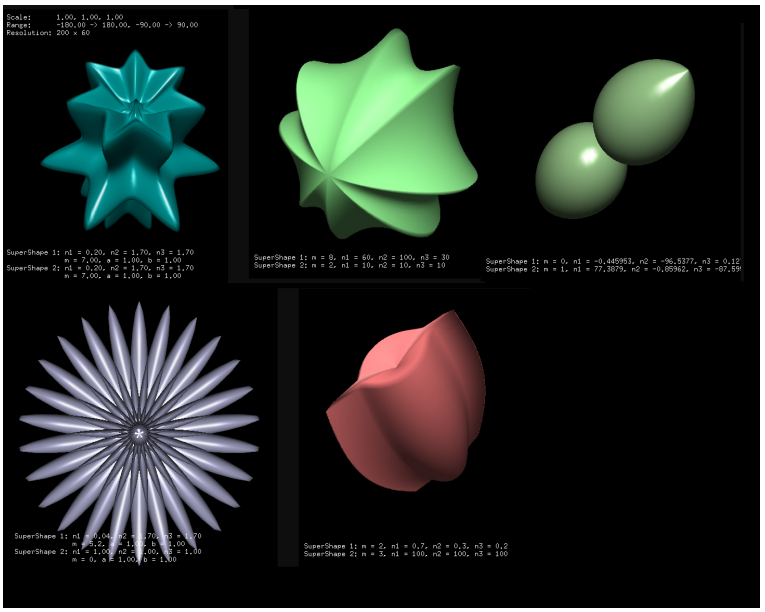
$$r(\phi) = \left[ \left| \frac{1}{a} \cos(m\,\phi/4) \right|^{n2} + \left| \frac{1}{b} \sin(m\,\phi/4) \right|^{n3} \right]^{-1/n1}$$

Extension to 3D using the spherical product.

$$x = r1(\theta)\cos(\theta)\, r2(\varphi)\cos(\varphi)$$
$$y = r1(\theta)\sin(\theta)\, r2(\varphi)\cos(\varphi)$$
$$z = r2(\varphi)\sin(\varphi)$$

$-\pi/2 \le \varphi \le \pi/2$ eg: latitude
$-\pi \le \theta \le \pi$ eg: longitude

Afin d'utiliser cette formule nous avons renommé par la suite nos variables Phi et Thêta. Dans cette formule **Phi correspondra à la latitude et Thêta à la longitude.** En fonction des valeurs n1, n2 et n3 nous pouvons obtenir diverses SuperShape.



Nous avons ensuite créé une fonction p**ermettant d'obtenir la valeurs des radius R1 et R2** en fonction des constantes fournies pour réaliser une SuperShape.

Nous savions d'après la formule que R1 dépendrait de Thêta (longitude) et R2 dépendrait de Phi (latitude). Il nous a fallu par la même occasion effectuer les changements nécessaires sur le calcul des coordonnées (x,y,z).

Il nous suffit alors d'ajuster nos constantes (N1 , N2, N3 et M) afin d'obtenir différentes SuperShape. Dans notre cas, nous devrions être en mesure d'obtenir une sphère puis de la déformer. Il fallait donc être capable de jouer avec les paramètres de la Supershape. Ces paramètres seraient par la suite modifiés en fonction de la musique. En relisant l'équation définissant une SuperShape en 2D et permettant de calculer R1 et R2 nous avons remarqué que **si la valeur du paramètre M est proche de 0 et que A et B tendent vers 1** nous obtiendrons **des rayons (R1 et R2) qui vaudraient également 1** et par conséquent tendront à se rapprocher de notre sphère. Nous allions donc pouvoir jouer avec ce paramètre M.

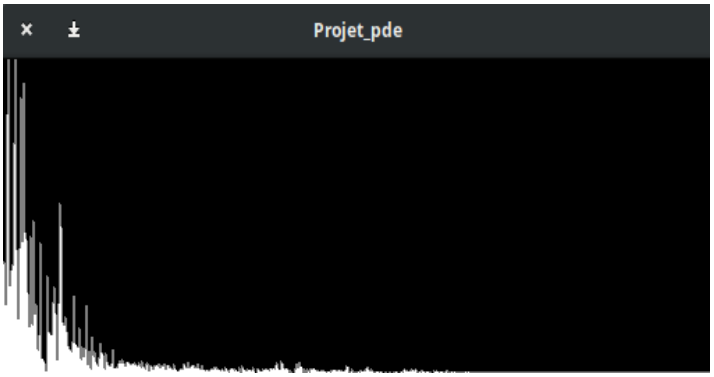
## 2 - Implémentation du module musique

Dans un second temps nous avons rendu notre SuperShape réactive à la musique. Pour cela nous avons utilisé la librairie Minim. Une question se posait alors: comment allons nous utiliser la musique comme entrée pour modifier notre paramètre M ?
La musique est perçue via des ondes sonores. Un haut parleur produit une vibration dans l'air à une certaine fréquence et amplitude ce qui frappe notre tympan puis est traduite en signal électrique à notre cerveau qui l'interprète en son. La fréquence de cette vibration détermine si nous percevons la musique comme un son grave ou aigu.
Dans la vraie vie, de nombreuses fréquences différentes frappent notre tympan en succession rapide ce qui signifie qu'en fin de compte, u**n son peut être décomposé en une collection de fréquences avec leur volume et amplitude.** L'amplitude est une autre caractéristique importante d'un son. L'intensité perçue dépend (entre autres) de l'amplitude. Dans l'air, **l'amplitude correspond aux variations de pression de l'onde** tandis que l'intensité sonore représente la variation de la pression du milieu dans lequel s'est produit l'onde acoustique.

Numériquement cela peut être représenté en divisant notre musique à un moment donné en plusieurs bandes de fréquences. **Chaque bande contient les informations d'amplitude pour cette fréquence spécifique.**
Dans processing il existe deux manières principales de récupérer des données au sujet d'une musique:avec l'amplitude et le FFT (Fast Fourier Transform Algorithm). La première méthode nous donnera un nombre représentant le volume global de la musique à un moment donné. La deuxième méthode quant à elle utilise les informations provenant des bandes de fréquences individuelles énoncés précédemment.

Dans notre cas, nous avons décidé de créer deux modes: un ou notre SuperShape réagirait en fonction des "beats" d'une musique et un second en fonction du **changement d'amplitude entre deux instants T.**
Dans le premier mode nous avons utilisé la classe *BeatDetect* qui permet de détecter de grandes variations d'énergie sonore. C'est ce que nous réalisons en appelant la fonction *GetMFromBeats()* qui nous retourne une valeur pour le paramètre M qui permettra de transformer notre sphère en SuperShape comme expliqué précédemment. Il est facile de se rendre compte de l'utilité de cette fonction avec une musique rythmée ou la pulsation est facilement audible par un humain. **Il faudra cependant baisser la valeur de la variable *FPS_TO_GENERATE_POINTS* selon la musique.** En effet, pour éviter de recalculer trop souvent des points nous avons décidé de limiter le calcul à un nombre d'images par secondes, cependant si les variations d'énergies sont trop fréquentes les points de la forme ne seront pas forcément recalculés si la valeur de cette variable est trop importante. Par exemple avec divers morceaux on remarque qu'il a fallu modifier cette variable pour obtenir un résultat convenable.

Pour le second mode nous avons décidé de jouer avec l'amplitude. **Pour un résultat idéal** il **faudrait lire la musique une première fois afin de récupérer l'amplitude maximale et minimale** puis de déformer notre forme en fonction de ces deux paramètres. Cependant nous désirions réaliser quelque chose de plus global qui fonctionnerait à la première lecture. Nous avons donc décidé de faire évoluer notre forme en fonction du pourcentage séparant la somme des amplitudes de chaque bandes de fréquences. **Plus le pourcentage augmente, plus notre sphère se déforme en SuperShape et inversement une SuperShape se transformera en sphère lors d'une diminution.**
Ce snippet issu de la documentation de la librairie Minim nous a permis de comprendre que la somme de l'amplitude des bandes de fréquence sera plus importante lorsque la musique atteindra un pic d'intensité: http://code.compartmental.net/minim/fft_method_getband.html. (Voir photo ci dessous). L'inconvénient de cette méthode est que selon la bande sonore, l'amplitude peut augmenter d'un coup au début de la piste audio lorsque la musique se lance seulement mais reste constante selon les différentes musiques.




Ce projet nous a fait prendre conscience qu'il est complexe de traiter une musique informatiquement et nous a permis d'en découvrir les bases. En raison des contraintes de temps, il était malheureusement difficile d'approfondir ce sujet afin d'obtenir un résultat plus précis.

## 3 - Ajout de bouton / slider dans pour interagir avec la SuperShape

Pour compléter notre projet, nous voulions ajouter quelques interactions avec la shape pour l'utilisateur.
Nous avons donc pensé à différents moyens de changer la déformation de la SuperShape ou encore sa couleur.
Pour créer des interactions, quoi de mieux que des boutons ?
Nous nous sommes donc servis de la librairie ControlP5 pour créer des boutons ainsi que des sliders. Cependant, leur utilisation posait un problème : comme nous utilisons PeasyCam pour la vue 3D, lorsque l'on tourne la forme, ces derniers tournent avec et ne sont donc pas cliquables… .
Pour les faire passer en arrière plan et les rendre utilisables, nous avons dû importer une autre librairie, une sous classe de processing, opengl. Nous avons ensuite dû passer en 3ème paramètre de  la fonction *size()* OPENGL au lieu de P3D et utiliser plusieurs fonctions dans *DrawGui()* et *InitGui()*.

Nous avons ainsi pu créer deux boutons pour changer le mode de déformation de la SuperShape :

- *React_To_Beats* (sélectionné par défaut) qui induit une déformation en fonction du beat de la musique
- *React_To_Amplitude* qui induit une déformation en fonction de l'amplitude de la musique

Nous avons ensuite ajouté deux autres boutons pour changer le mode de couleurs associé à la SuperShape :

- *Rainbow_Mode* (sélectionné par défaut) qui lui donne un effet arc-en-ciel. 3 modes d'arc-en-ciel sont implémentés, on peut passer de l'un à l'autre en cliquant à nouveau sur ce bouton.
- *RGB_Mode* qui la colore en fonction de 3 slider, un pour le rouge, un pour le vert et un pour le bleu.

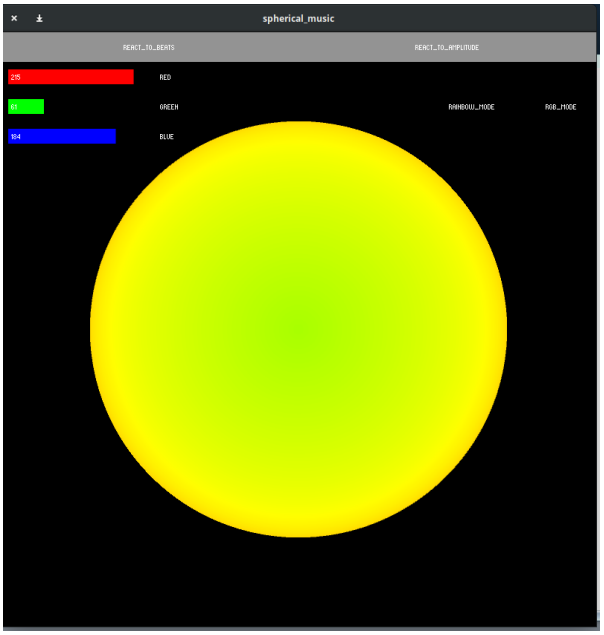
Pour finir, nous avons donc créé 3 sliders prenant des valeurs comprises entre 0 et 255 qui permettent de changer en temps réel la couleur de la SuperShape quand le bouton RGB_Mode à été cliqué.

Avec ControleP5, pour associer des boutons à des actions spécifiques, il faut simplement créer des fonctions avec un nom identique à celui des boutons et implémenter les actions voulues dans ces dernières.
Un des problèmes de ce fonctionnement est qu'il nous est donc impossible de créer des boutons dont le nom contient un espace. Nous avons donc dû nous affranchir de certaines conventions concernant les noms de fonction pour que nos boutons restent à la fois compréhensibles et lisibles lors de l'exécution du programme.

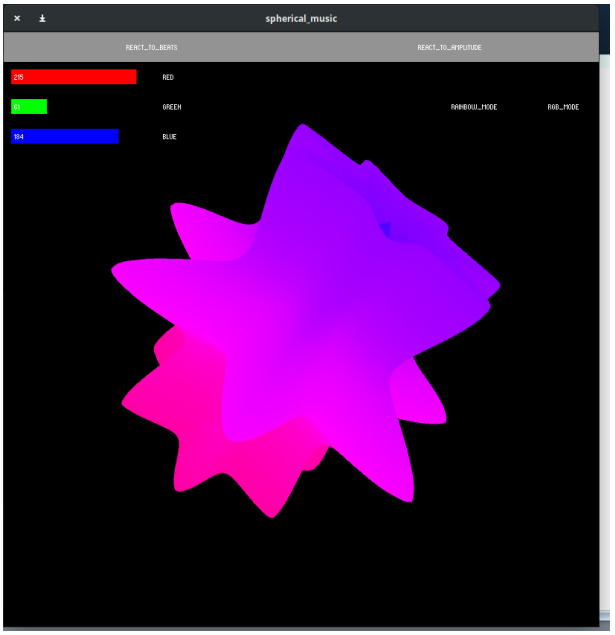
Pour que notre code soit le plus modulable possible, nous avons créer les variables de fonction *widthButton*, *heightButton*, *widthSlider, heightSlider,* pour que la position et la taille des boutons / slider s'adapte à celle de la fenêtre tout comme le reste de notre programme.

Pour conclure, tout au long de notre projet nous avons découvert une partie des nombreuses possibilités qu'offre la programmation sous processing dans le milieu artistique. Mettre en relation le domaine de la musique et celui du visuel était réellement intéressant.

Nous avons pu trouver les formules mathématiques permettant de construire une sphère en 3 dimensions capable de se transformer en SuperShape et la faire évoluer dans ce même espace mais aussi la faire interagir avec un fichier audio. Nous regrettons seulement de ne pas avoir eu le temps d'approfondir les  notions de traitement musical pour faire évoluer notre sphère de manière plus fluide. Ce projet nous a fait toucher du doigt des notions plus complexes telles que l'acoustique (les ondes sonores). Mettre en “partition” informatique, mathématique et physique demande d’approfondir nos connaissances et reste très stimulant.




Sphère en Rainbow mode




Sphères déformées en SuperShape