E-ARTSUP.
DÉPARTEMENT DESIGN INTERACTIF & COMMUNICATION VISUELLE.

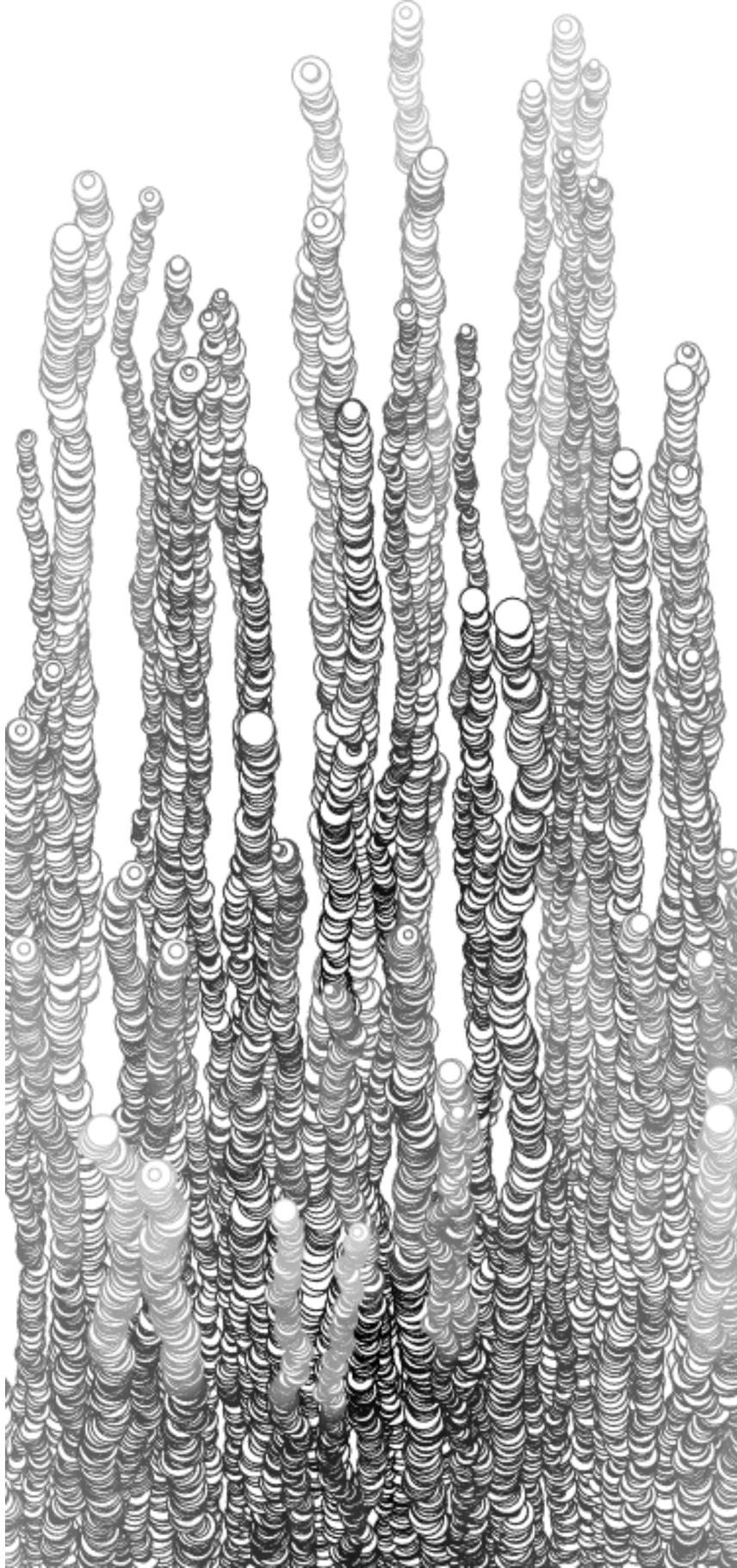
DEVELOPPEMENT ORIENTÉ OBJET [PARTIE 1/2]
(TABLEAUX ET CLASSES)

Digital Lab S.05 // Alexandre Rivaux arivaux@gmail.com

Département Design interactif & communication visuelle:

Enseignants:
Nicolas Baumgartner
Félicie d'Estienne D'Orves
Rémi Jamin
Wolf Ka
Jonathan Munn
Gustave Bernier
Alexandre Rivaux

1 - Programmation Orientée Objet (OOP)



La programmation orientée objet (POO) est un paradigme de programmation conçu par Ole-Johan Dahl et Kristen Nygaard au début des années 1960 et poursuivi par les travaux d'Alan Kay dans les années 1970. Un objet représente un concept, une idée ou toute entité du monde physique. Un objet permet d'encapsuler un nombre des propriétés et méthodes pour faciliter le traitement de groupe.

Un objet est directement inspiré du monde réel. En effet nous sommes entouré d'objets et chacun de ces objets à des propriétés propres. Par exemple nous avons les objets "téléphones". Chacun de ses objets "téléphones" sont définis par le fait qu'ils ont tous des variables communes, à savoir le fait de pouvoir passer des communications vocales par le biais d'un réseau téléphonique. Mais ils possèdent aussi des variables qui leur sont propre comme le fait de pouvoir envoyer des sms ou non, d'être tactile ou pas, d'avoir des poids et des tailles qui diffèrent… Bref nos téléphones sont des objets appartenant à une même classe, la classe "Téléphone.

2 - Les tableaux

Avant de se lancer dans le developpement orienté objet il est important de connaitre et comprendre une autre base du developpement : les listes ou tableaux.

Les tableaux, listes ou arrays font partis des fondamentaux des langages de programmation. Il s'agit d'une liste nous permettant de stocker plusieurs variables auxquelles on aura accès via un numero d'index. Sur le papier cela peut faire peur mais par un exemple c'est tout de suite plus simple.

Prenons une classe d'étudiants, chaque étudiant a un prénom. Nous avons donc un tableau d'étudiant de ce type

```
String nom de l'étudiant = {Pierre, Paul, Jacques};
```

Je pourrai donc alors dire que le nom de l'étudiant 1 = Pierre, le nom de l'étudiant 2 = Paul et le nom de l'étudiant 3 = Jacques

J'obtiens alors le tableau suivant

Index	**0**	**1**	**2**
String	**Pierre**	**Paul**	**Jacques**

Les tableaux servent à nous simplifier la vie. C'est le premier pas avant le développement orienté objet. Les tableaux vont nous permettre de stocker des valeurs "communes" pour les traiter plus facilement et par groupe.

Imaginons que nous voulions faire un sketch de 100*100 avec 3 balles sur scène. Nous aimerions qu'en fonction des touches du clavier a, z ou e cela change la couleur de la balle 1, 2 ou 3 et restore la couleur des autres balles à leur valeur d'origine. Pour cela, sans tableau nous aurions écris :

```
float c1=255;
float c2=255;
float c3=255;

void setup()
{
  size(500, 500, P2D);
}
```

2 - Les tableaux

```
void draw()
{
  background(255);
  ellipseMode(CORNER);
  fill(c1);
  ellipse(0, 0, 50, 50);
  fill(c2);
  ellipse(0, 50, 50, 50);
  fill(c3);
  ellipse(0, 100, 50, 50);
}

void keyReleased()
{
  if (key == 'a')
  {
    c1 = 0;
    c2 = 255;
    c3 = 255;
  }
  if (key == 'z')
  {
    c2 = 0;
    c1 = 255;
    c3 = 255;
  }
  if (key == 'e')
  {
    c3 = 0;
    c1 = 255;
    c2 = 255;
  }
}
```

Le résultat fonctionne mais si nous voulons appliquer cela à 10, 50, ou 200 balles alors cela devient très vite laborieux d'écrire 10, 50 ou 200 variables. C'est là où nos tableaux deviennent plus pratiques.

2 - Les tableaux

Il existe deux grand type de type de tableaux, les tableaux statique et les talbeaux dynamiques. Chacun de ses deux type permette de créer des liste des liste d'élement que nous pourront par la suite parcourir. Nous nous concentrerons d'abords sur les tableaux statiques puis verrons les tableaux dynamique dans un second temps.

Un tableau permet de stocker tout type de variables (char, String, float, int...) et se déclare de la manière suivante :

```
int[] table= {1, 2, 3}
```

où int est le type de variables remplissant notre tableaux, table est le nom du tableau, [] défini le tableau; {} ouvre et ferme et le tableau et 1, 2 et 3 sont les éléments de notre tableau.

Losque nous ne connaissons pas par avance les valeurs qui rempliront notre tableau (car créées de ma,ière dynamique). La déclaration du tableau changera. Ainsi nous aurons :


```
int[] table= new int[3]
```

où int est le type de variables remplissant notre tableaux, table est le nom du tableau, [] défini le tableau; et new table[3] permet de créer un nouveau tableau de 3 colonnes.

Une fois déclarer nous pourront définir les valeurs de notre tableau à l'aide d'une boucle for.

```
for (int i=0; i<3; i++)
  {
    table[i] = i;
  }
```

où table[valeur] permet de cibler la variable contenu dans la colonne «valeur»

Enfin, à tous moment il est possible de connaitre la taille d'un tableau à l'aide de la méthode suivante.

```
table.length
```

Ainsi utilisée dans une boucle for, nous ne prenons pas le risque d'interroger une colonnes n'existant pas au sein de notre tableau.

```
for (int i=0; i<table.length; i++)
  {
    table[i] = i;
  }
```

2 - Les tableaux

Reprenons maintenant notre premier exemple et recréons le à l'aide d'un tableau. Nous voulions 3 ellipse changeant de couleur en fonction des touches appuyées a, z et e.

Pour cela nous aurons besoins de déclarer deux tableaux :

```
float[] couleur;
char[] touche = {'a', 'z', 'e'};

int nbEllipse;
```

Les valeurs du tableau «touche» sont déjà initialisée mais il nous faut initialiser celles du tableau couleur à l'aide d'une boucle for. Nous effectuerons cela dans notre methode setup afin que notre tableau contienne les bonne valeurs à la création du sketch.

```
void setup()
{
  size(500, 500, P2D);

  nbEllipse = 3;
  couleur = new float[nbEllipse];

  for (int i = 0; i<couleur.length; i++)
  {
    couleur[i] = 255;
  }
}
```

Nous dessinons ensuite nos ellipse de couleur[i] à l'aide d'une simple boucle for nous permettant de parcourir notre tableau.

```
void draw()
{
  background(255);

  for (int i = 0; i<couleur.length; i++)
  {
    float taille = 50;
    float y = i*taille;

    fill(couleur[i]);
    stroke(0);
    ellipse(taille/2, y+taille/2, taille, taille);
  }
}
```

2 - Les tableaux

Appliquons maintenant cette même méthode à la partie interactive. La seule différence ici c'est qu'il va nous falloir savoir sur quelle touche nous avons appuyé.

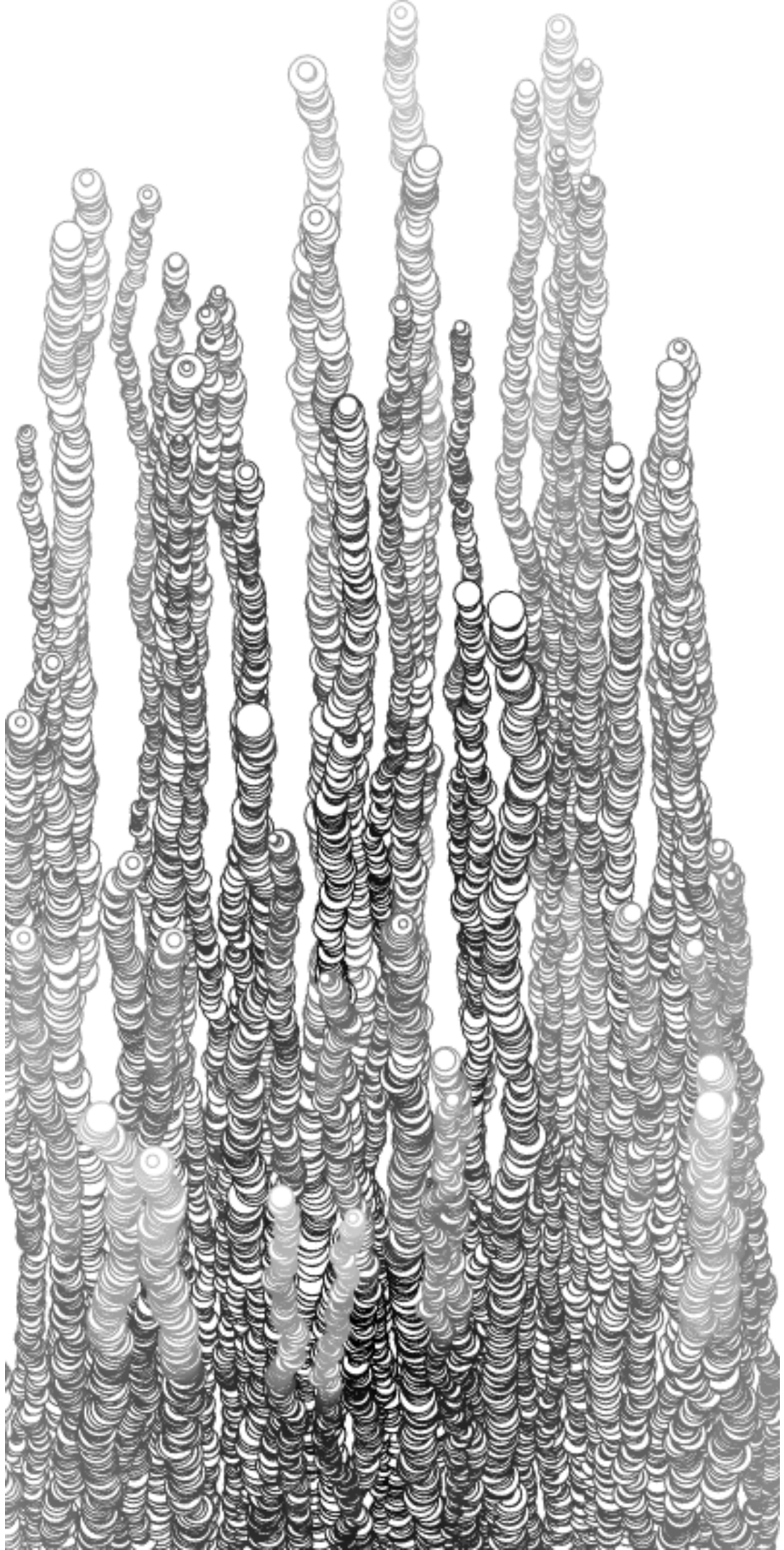
Précédemment nous utilisions des condition if(key == 'a').
Nous allons faire la même chose mais de manière dynamique à l'aide de notre boucle for et notre tableau touche.

```
void keyReleased()
{
  for (int i = 0; i<touche.length; i++)
  {
    if (key == touche[i])
    {
      couleur[i] = 0;
    }
    else
    {
      couleur[i] = 255;
    }
  }
}
```

Nous venons de voir comment utiliser un tableau dans le cas de trois éléments mais il est possible d'utiliser encore plus de tableau. Ainsi si nous souhaitons réaliser un sketch avec un certains nombre de balles tombantes nous pourrions nous servir des tableaux pour concerver les valeurs suivantes :

La position en x d'un cercles
La position en y d'un cercles
La taille minimale d'un cercle
La taille maximale d'un cercle
la vitesse en y d'un cercle

3 - Random Walker



Un random walker ou marcheur aléatoire est un objet, ici un cercle, se déplaçant de manière aléatoire. Imaginons le sketch suivant :

Nous aimerions avoir plusieurs cercles (50), effectuant un déplacement du bas vers le haut à une vitesse aléatoire et un déplacement latéral aléatoire entre -1 et 1 pixel.

Nous aimerions aussi que nos cercles aient une taille aléatoire et que cette même taille varie de manière aléatoire. Enfin nous aimerions que le contour de nos cercles soit plus ou moins noir en fonction de leur distance avec la souris.

Nous allons donc utiliser les tableaux afin de créer et concerver ces valeurs.

Contact

Alexandre Rivaux
Visual Designer & Partner Bonjour, interactive Lab

www.bonjour-lab.com
arivaux@gmail.com