UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

Departamento de Informática

Modelación y Simulación

Laboratorio 4

Diego Mellis

Andrés Muñoz

Profesor: Fernando Rannou

Ayudante: Francisco Muñoz

Santiago – Chile

2019

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1. INTRODUCCIÓN

La simulación mediante modelos matemáticos y técnicas de computación, para esta experiencia, permitirá evaluar cambios y comportamientos de una situación del mundo real dependiendo de ciertas condiciones definidas a medida que se estudia el comportamiento. De esta forma, la situación del mundo real, se proyecta en una simulación que permite una cercanía con el que estudia la situación y así logra manejar directamente las variables que están en juego. En este experiencia se asume que el sistema a simular corresponde a una bandada de aves, que se mueven en un plano, añadiendo también un líder, el cual las aves deben seguir si es que este se encuentra a cierta distancia y un disruptor, quien aleja a las aves a cierta distancia.

Para el desarrollo de esta experiencia, se utiliza *Processing*, un software que entrega las herramientas necesarias para la implementación de la simulación que está en juego en esta experiencia. *Processing* utiliza *Java*, un lenguaje de programación orientado a objetos, por lo que permite un desarrollo modular de los objetos que deben estar presentes en esta implementación, además de entregar métodos necesarios que facilitan el desarrollo.

El objetivo general de esta experiencia es:

1. Implementar una simulación de bandadas de aves utilizando *Processing*

Teniendo los siguientes objetivos específicos:

1. Implementar un líder que guíe a un conjunto de aves, definiendo una función de trayectoria, en base a fuerzas y distancias.
2. Implementar un disruptor que aleje a un conjunto de aves, definiendo una función de trayectoria, en base a fuerzas opuestas a las aves.

El siguiente informe presenta una descripción de la solución y cómo se implementó usando *Processing*. Finalmente, se presenta una conclusión con lo obtenido en esta experiencia.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN

En esta sección se describe el proceso de implementación de esta experiencia. *Processing*, en su página oficial (`https://processing.org`) contiene ejemplos de implementaciones, para esta experiencia se utiliza el ejemplo de *Flocking* desarrollado por Daniel Shiffman (Wikipedia, 2019), el cual muestra el comportamiento de un conjunto de aves moviéndose en un plano en base a ciertas reglas, que se mostrarán más adelante. Gran parte del código de esta experiencia es extraído de esta implementación, siendo la base para el desarrollo y la posterior implementación de un líder y un disruptor.

Para entender a grandes rasgos la forma en que está modelado el código, en la figura 2.1 se muestran las clases implementadas.

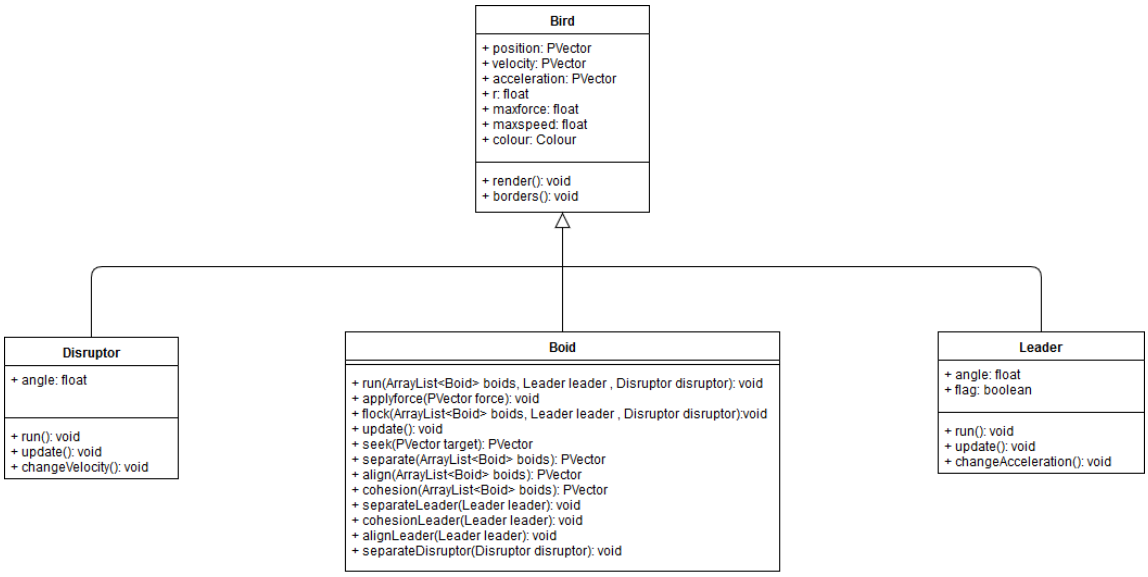

Figura 2.1: Diagrama de clases

Como se aprecia en la figura 2.1 existe la clase *Disruptor* y para dejar en claro su trayectoria, esta corresponde a una circunferencia fija y para definir esta trayectoria se utiliza el método de esta clase, que recibe el nombre de *changeAcceleration()* el cual define las variables respectivas para un movimiento circular uniforme (Ánonimo, 2015). La implementación de este método se muestra en la figura 2.2.

```
void changeAcceleration(){
  PVector centro = new PVector(width/2,height/2);
  PVector radio = PVector.sub(centro, position);
  float r = radio.mag();
  float v = velocity.mag();
  radio.normalize();
  acceleration = radio.mult((v*v)/r);
}
```

Figura 2.2: Método changeAcceleration() de Disruptor

La decisión de definir un movimiento circular uniforme para el disruptor, fue para que la visualización de la fuerza aplicada hacia el conjunto de aves fuese más notoria a nivel visual en la ejecución.

Por otro lado se tiene la clase *Leader* que también presenta una trayectoria definida, pero con naturaleza sinusoidal. Esta trayectoria varía con respecto a un ángulo, el cual presenta un rango para cambiar la trayectoria y así seguir en el plano de forma sinusoidal. Esta clase también presenta el método *changeVelocity()* el cual define la trayectoria en base a la variable *flag* que va variando con respecto a los ángulos que va describiendo el líder (clase *Leader*), la implementación se muestra a continuación en la figura 2.4.

```
void changeVelocity(){
  if(flag){
    angle += PI/720;
  }
  else{
    angle -= PI/720;
  }
  if(angle > PI/3){
    flag = false;
  }
  else if(angle < -PI/4){
    flag = true;
  }
  velocity.x = 1.8*cos(angle);
  velocity.y = 1.8*sin(angle);
}
```

Figura 2.3: Método changeVelocity() de Leader

Esta trayectoria logra transitar a través del tiempo, todo el plano, de esta forma se

logra ver el comportamiento que tienen las aves al tener cerca al líder.

Después de lo explicado anteriormente, es importante mencionar las tres fuerzas que están en juego en esta simulación: la cohesión, la alineación y la separación. A grandes rasgos se puede ver de la siguiente forma:

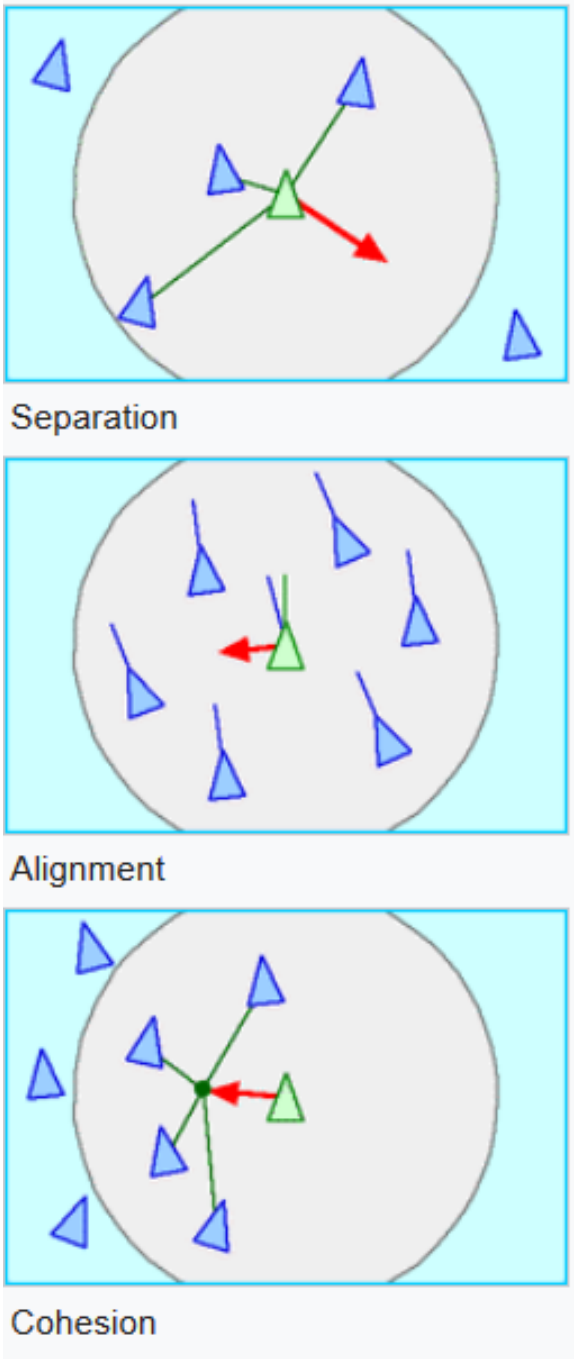

Figura 2.4: Reglas de fuerza para la simulación

Se ha optado por definir arbitrariamente estas fuerzas, pero con justificaciones respectivas. La idea principal es que el conjunto de aves sigan a un líder, pero manteniendo las reglas de fuerzas entre las aves que son del mismo tipo (ni líder, ni disruptor), es por esto que la fuerza de cohesión entre ellas debe mantener a las aves juntas. Pero al haber un líder es preciso que la cohesión hacia el líder predomine por sobre la del conjunto de aves, por lo tanto al aplicar estas tres reglas con respecto al líder las aves tienden a seguirlo, la forma de aplicar esto en la implementación, se muestra de la siguiente en la figura 2.5

```
25  void flock(ArrayList<Boid> boids, Leader leader, Disruptor disruptor) {
26    PVector sep = separate(boids);   // Separation
27    PVector ali = align(boids);      // Alignment
28    PVector coh = cohesion(boids);   // Cohesion
29    PVector sepLeader = separateLeader(leader); // acercarse al lider
30    PVector aliLeader = alignLeader(leader); // acercarse al lider
31    PVector cohLeader = cohesionLeader(leader); // acercarse al lider
32    PVector sepDisruptor = separateDisruptor(disruptor); // alejarse del disruptor
33
34    // Arbitrarily weight these forces p
35    sep.mult(2.25);
36    ali.mult(1.5);
37    coh.mult(1.5);
38    sepLeader.mult(2.0);
39    aliLeader.mult(1);
40    cohLeader.mult(1.2);
41    sepDisruptor.mult(2.6);
42    // Add the force vectors to acceleration
43    applyForce(sep);
44    applyForce(ali);
45    applyForce(coh);
46    applyForce(sepLeader);
47    applyForce(aliLeader);
48    applyForce(cohLeader);
49    applyForce(sepDisruptor);
50  }
```

Figura 2.5: Método flock que aplica las fuerzas a las aves (Boids)

El líder (*Leader*, como se mostraba en la figura 2.1 se extiende Boid, pero esto se hizo con la finalidad de independizar su movimiento en el plano y así solo aplicar fuerzas al conjunto de aves (boids) en base a su posición. Los escalares multiplicados a los vectores que se pueden apreciar desde las líneas 35 a 40 de la figura 2.5 se fueron ajustando mediante ensayo y error, para lograr el comportamiento deseado. Lo importante a destacar es que se fijaron estos escalares para evitar que las aves sobrepasen al líder, cumpliendo así el comportamiento deseado. Añadiendo a lo mencionado anteriormente, se definen distintos radios para las reglas (separación, cohesión y alineación). Para la separación se ha definido un radio de 25, este radio es pequeño en comparación al de las otras reglas, dado que solo es para evitar que las aves se sobrepongan a su lider, pero por otro lado se define un radio de 250 tanto para la cohesión como

para la alineación, principalmente para que más aves tomen el sentido del líder y se mantengan en conjunto una mayor cantidad de aves.

Por otro lado, para implementar el comportamiento de la ave disruptiva bastó con aplicar una fuerza de separación hacia las aves y al trabajar con vectores, lo importante es destacar que la posición del ave debe ser restada por la del disruptor, de esta forma se logra la lejanía. Esto se puede ver en la figura 2.5 en la línea 41. El radio para el ave disruptiva fue de 150, principalmente para hacer notoria la lejanía que experimentan las aves y de esta forma verlo de forma más natural.

Finalmente, con todo lo descrito anteriormente, al ejecutar la simulación implementada es posible notar los comportamientos requeridos, simulando el acto de que un conjunto de aves tengan un líder que los guía y un disruptor que busque desordenar el conjunto de las aves, viéndolo como una amenaza.

CAPÍTULO 3. CONCLUSIÓN

En esta experiencia queda demostrado que es posible simular situaciones del mundo real, aplicando ciertas reglas, variables y parámetros que definan la situación a simular. En este experiencia se simula el comportamiento de una bandada de aves, guiadas por un líder y alejadas por un ave disruptiva. *Processing* fue una herramienta sumamente útil para el desarrollo de esta experiencia, utilizando la implementación de ejemplo que se encuentra en la página oficial, fue posible reestructura el código y darle un uso para el enunciado planteado para esta experiencia.

Con respecto a los objetivos, tanto generales como específicos se cumplieron a cabalidad, ya que la simulación tiene la implementación de la situación, con el líder, con el disruptor y además con ambos juntos.

En base a lo aprendido durante esta experiencia, se llega a la conclusión que para estos casos, son necesarias ciertas leyes físicas, principalmente las que tiene que ver con movimientos y trayectorias en dos dimensiones, esto fue esencial para el desarrollo y se espera recibir retroalimentación respecto a lo desarrollado para seguir aplicando el concepto de simulación de manera correcta.

BIBLIOGRAFÍA

Wikipedia (2019). Boids. Recuperado desde https://en.wikipedia.org/wiki/Boids".

Ánonimo (2015). Movimiento circular uniforme. Recuperado desde https://www.profesorenlinea.cl/fisica/MovimientoCircular.html.