POLITECNICO
MILANO 1863

IMAGE AND SOUND
ISPG
PROCESSING GROUP

CREATIVE PROGRAMMING AND COMPUTING

Lab: MultiAgent systems

MULTIAGENT SYSTEM

- We will see different multiagent systems based on their type of interaction
  - No Interaction
  - Personal Goal
  - Collective Goal

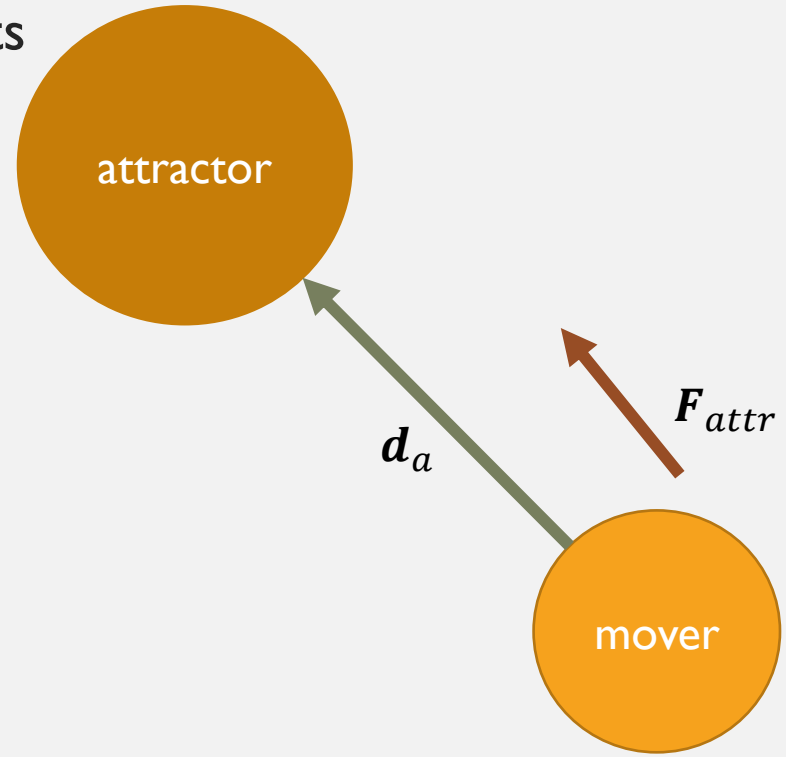
- First we will adapt some of the agents we have seen in the previous lesson to the multiagent system (without interaction).

GRAVITY AND ATTRACTION

GRAVITY AND ATTRACTION

- We will start from moving_ball_attractor → moving_balls_attractor
- Instead of just defining one Agent, we will define several agents
  - Skip the OSC sound

```
# moving_balls_attractor.pde
int N_AGENTS=20; AgentMover[] movers;
void setup(){
  movers=new AgentMover[N_AGENTS];
  for(int i=0; i<N_AGENTS; i++){
     movers[i]=new AgentMover(random(5,30));
  }
  /*...*/}
void draw(){
  /* ...*/
  PVector force;
  for(int i=0; i<N_AGENTS; i++){
    force = computeGravityForce(movers[i]);
    /* your code */
  }}
```



GRAVITY AND ATTRACTION

- Let's try to sonify this.
- We can connect it to SuperCollider, but here we will use Processing Audio Library
- Install Sound Library in your Processing.
- We will use a few selected operations, but feel free to check for more advanced tools.
- Go to www.mynoise.net

GRAVITY AND ATTRACTION

- Go to mynoise.net
- It's a website where you can listen to relaxing sounds
- Each slider controls one sound sample that is played in loop
- Users can choose their mix, or choose to make their gains change dynamically
- Let's make something like this, but changing our gains dynamically

GRAVITY AND ATTRACTION

1. Collect some sounds you like from freesounds or take what I collected
2. Put the files in a directory named «sounds» inside your sketch directory
3. Now let's see how we can edit the file using SoundFile https://processing.org/reference/libraries/sound/SoundFile.html
4. Start from moving_balls_attr.pde

Link for files https://drive.google.com/file/d/1GXKFAdq6jyU7fOfKy6j2SAr8EWs1tLEn/

GRAVITY AND ATTRACTION

1. Collect some sounds you
2. Put the files in a directory
3. Now let's see how we car
   https://processing.org/refe
4. Start from moving_balls_a

```
# moving_balls_attr_sounds.pde
import processing.sound.*;
import java.util.Date;

int N_AGENTS;
AgentMover[] movers;
SoundFile[] samples;
void setup(){
  String path=sketchPath()+"/sounds";
  File dir = new File(path);
  String filenames[] = dir.list();
  N_AGENTS=filenames.length;
  movers=new AgentMover[N_AGENTS];
  samples=new SoundFile[N_AGENTS];
  for(int i=0; i<N_AGENTS; i++){
     movers[i]=new AgentMover(random(100,200));
     samples[i] = new SoundFile(this, path+"/"+filenames[i]);
     samples[i].amp(0); samples[i].loop();
  }}
```

GRAVITY AND ATTRACTION

1. Collect some sounds you
2. Put the files in a directory
3. Now let's see how we car
   https://processing.org/refe
4. Start from moving_balls_a

```
# moving_balls_attr_sounds.pde
import processing.sound.*;
import java.util.Date;

int N_AGENTS;
AgentMover[] movers;
SoundFile[] samples;
void setup(){
  String path=sketchPath()+"/sounds";
  File dir = new File(path);
  String filenames[] = dir.list();
  N_AGENTS=filenames.length;
  movers=new AgentMover[N_AGENTS];
  samples=new SoundFile[N_AGENTS];
  for(int i=0; i<N_AGENTS; i++){
     movers[i]=new AgentMover(random(100,200));
     samples[i] = new SoundFile(this, path+"/"+filenames[i]);
     samples[i].amp(0); samples[i].loop();
  }}
```

Import audio library and utils

Get all the namefiles in the directory "sounds"

Create a mover for each audiofile, setting the volume to zero and play it on loop

GRAVITY AND ATTRACTION

1. Collect some sounds you
2. Put the files in a directory
3. Now let's see how we car
   https://processing.org/refe
4. Start from moving_balls_a

```
# moving_balls_attr_sounds.pde

void draw(){
  rectMode(CORNER); fill(0,20);  rect(0,0,width, height);
  fill(200, 0, 200, 40);
  ellipse(pos_attractor.x, pos_attractor.y,
          radius_attractor, radius_attractor);

  PVector force_a;
  for(int i=0; i<N_AGENTS; i++){
    force_a = computeGravityForce(movers[i]);
    movers[i].applyForce(force_a);
    changeAmp(i);
    movers[i].update();
    movers[i].draw();

  }
}
```

Place the attractor in the middle of the screen

Write the function
`void changeAmp(int i)`

GRAVITY AND ATTRACTION

1. Collect some sounds you like from freesounds or take what I collected from here
2. Put the files in a directory named «sounds» inside your sketch directory
3. Now let's see how we can edit the file using SoundFile
   https://processing.org/reference/libraries/sound/SoundFile.html
4. Start from moving_balls_attr.pde
5. Write the function `void changeAmp(int i)`

GRAVITY AND ATTRACTION

Write the function `void changeAmp(int i)`

1. The function must set a different gain to `samples[i]` given some property of `movers[i]`
2. You can use any mapping you want. I suggest to use

$$A_i = \left\lfloor \frac{1}{1 + \alpha\, d_i} \right\rfloor_{A_{min}}$$

$\lfloor x \rfloor_y$ = `max(x,y)`

With

- A_i the amplitude of the i-th samples,
- d_i the distance from the i-th mover to the attractor,
- α a constant to rescale (I choose 0.5)
- A_{min} the minimum allowed amplitude, so that samples are always audible (I choose 0.02)

GRAVITY AND ATTRACTION

Write the function `void changeAmp(int i)`

1. The function must set a different gain to `samples[i]` given some property of `movers[i]`
2. You can use any mapping you want. I suggest to use

$$A_i = \left\lfloor \frac{1}{1 + \alpha\, d_i} \right\rfloor_{A_{min}}$$

$\lfloor x \rfloor_y$ = `max(x,y)`

When $d_i = 0 \rightarrow A_i = \frac{1}{1+\alpha}$ maximum value, when $d_i \rightarrow \infty \rightarrow A_i = A_{min}$

The gain increases when the mover is closest to the attractor

GRAVITY AND ATTRACTION

1. Collect some sounds you like from freesounds or take what I collected from here
2. Put the files in a directory named «sounds» inside your sketch directory
3. Now let's see how we can edit the file using SoundFile
   https://processing.org/reference/libraries/sound/SoundFile.html
4. Start from moving_balls_attr.pde
5. Write the function `void changeAmp(int i)`
6. Test the method: it is probably too fast for your application.
7. Homework: adapt your physics model in order to slow it down

GRAVITY AND ATTRACTION

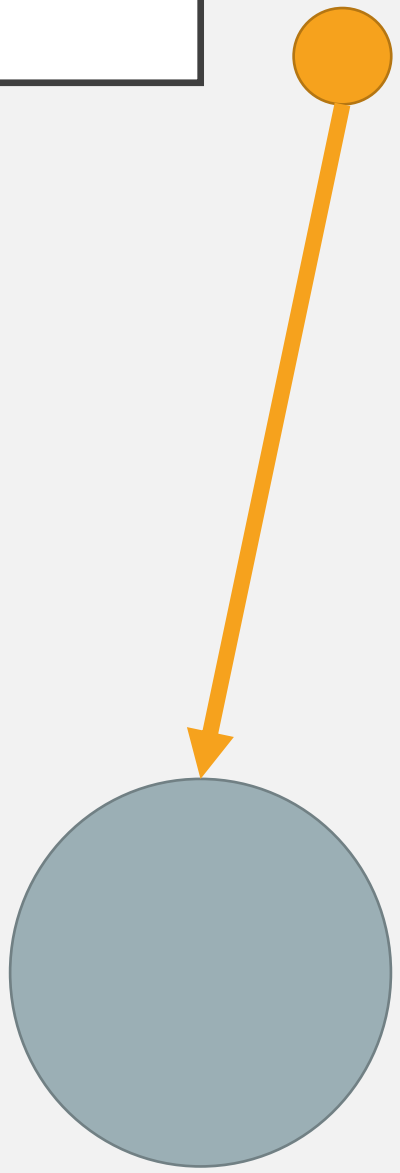
Possible extension

1. Slow down the system: how? (increase masses, multiply the computed forces for a constant to reduce its magnitude, etc.)
2. Play with the animation: change colors, shapes, background dynamically following the movement of the AgentMover
3. Combine this application with the feature visualizer we coded during our first lesson: change colors, shapes, background dynamically following the audio content of the AgentMover.

GRAVITY AND ATTRACTION

Challenge

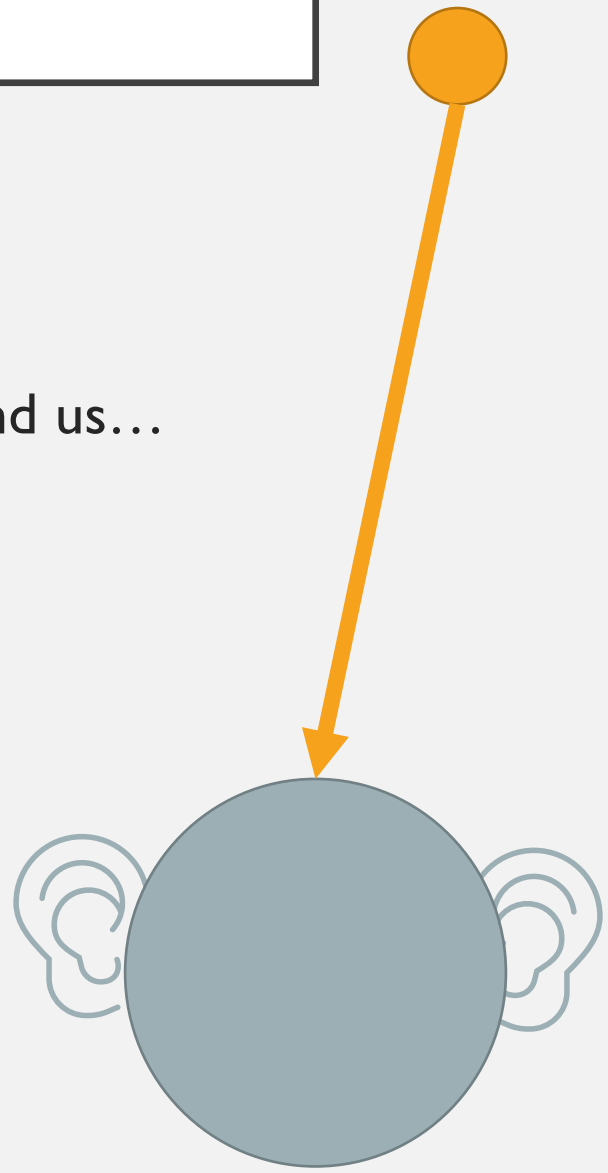
- We made the amp change with the distance from the attractor



GRAVITY AND ATTRACTION

Challenge

- We made the amp change with the distance from the attractor
- Suppose the attractor is our head, and the 2D plan is the space around us…



GRAVITY AND ATTRACTION

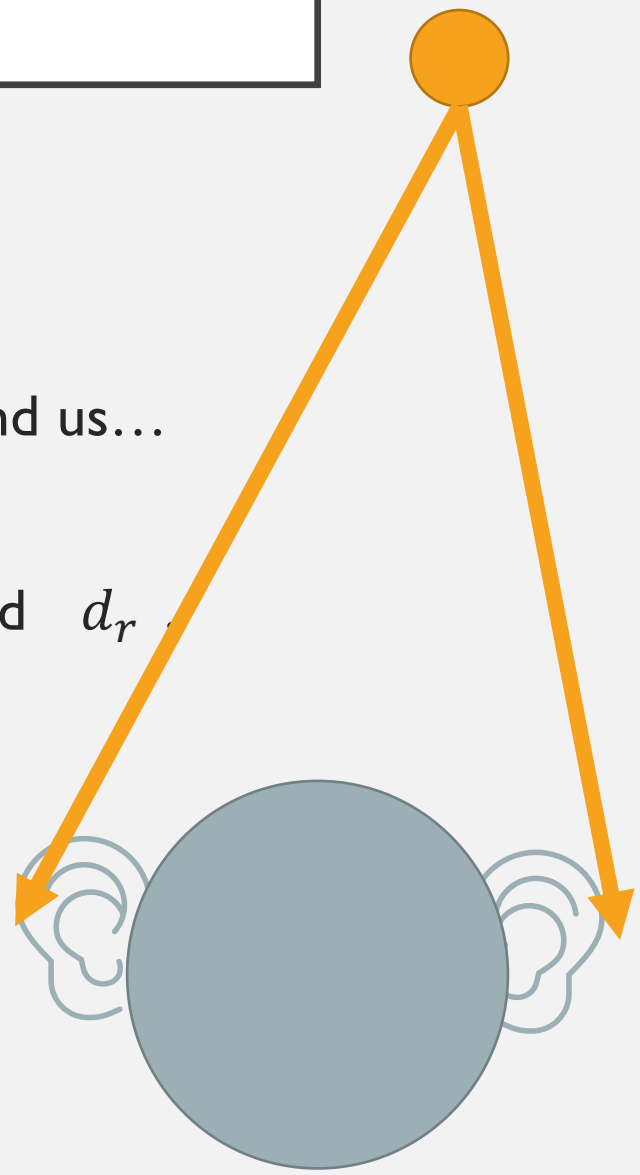
Challenge

- We made the amp change with the distance from the attractor
- Suppose the attractor is our head, and the 2D plan is the space around us…
- So we can make amp change with the distances from our «ears»
-

GRAVITY AND ATTRACTION

Challenge

- We made the amp change with the distance from the attractor
- Suppose the attractor is our head, and the 2D plan is the space around us…
- So we can make amp change with the distances from our «ears»
- Given the two distances from left and right ears, d_l and d_r , map them into the pan of the samples

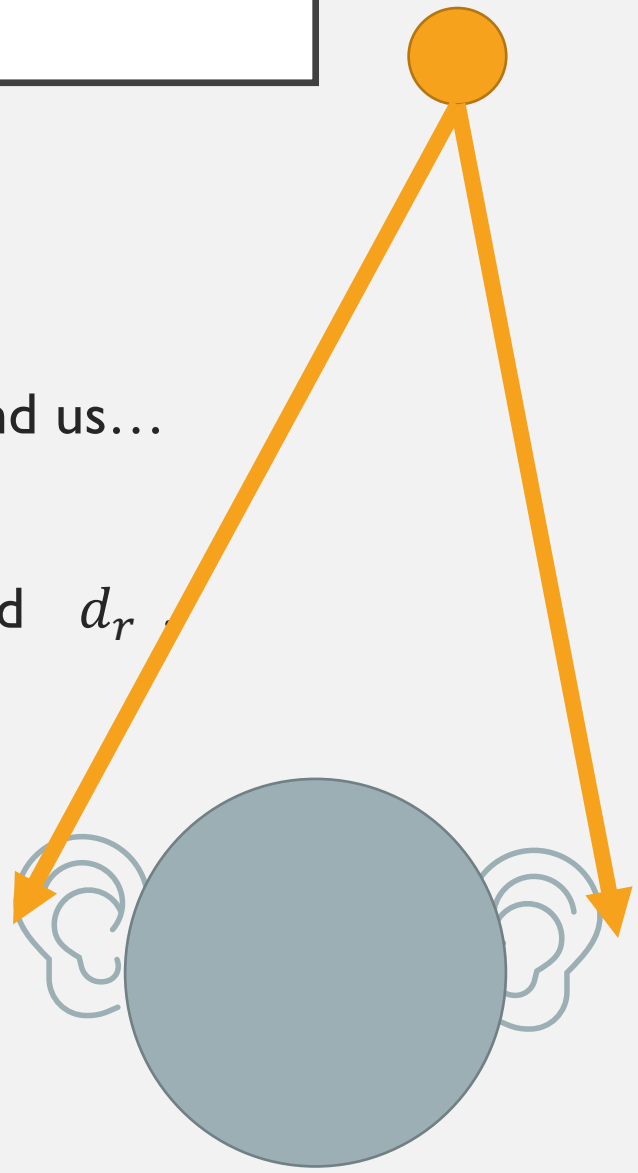


GRAVITY AND ATTRACTION

Challenge

- We made the amp change with the distance from the attractor
- Suppose the attractor is our head, and the 2D plan is the space around us…
- So we can make amp change with the distances from our «ears»
- Given the two distances from left and right ears, d_l and d_r , map them into the pan of the samples
  - When $d_l = d_r$, the pan is 0; when $d_l \gg d_r$ the pan is 1, and viceversa
  - Make your sound a «3D» sound

https://processing.org/reference/libraries/sound/SoundFile_pan_.html



PARTICLE SYSTEMS

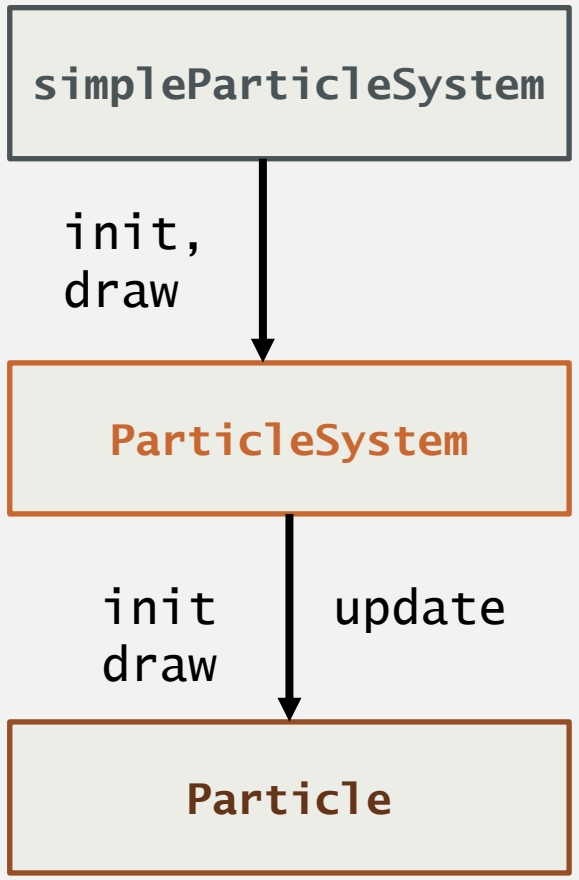
PARTICLE SYSTEM

- *"A particle system is a collection of many many minute particles that together represent a fuzzy object. Over a period of time, particles are generated into a system, move and change from within the system, and die from the system."*
  —William Reeves, "Particle Systems—A Technique for Modeling a Class of Fuzzy Objects," ACM Transactions on Graphics *2:2 (April 1983), 92.*
- We define a Particle System as a… system of particles
  - Each particle move with a certain behavior, both rule-based and randomic
  - The system is the Agent that collects and acts on all of them
- Let's first create a single Particle, not very differently from our mover

PARTICLE SYSTEM

Our architecture is composed by:

- The main script simpleParticleSystem.pde, which creates a Particle System
- A class ParticleSystem that creates, updates and controls an array of Particles
- A class Particle representing each object



PARTICLE SYSTEM

- Let's first create a single Particle, not very differently from our mover
- We neglect the mass and, instead, we add a *lifespan* attribute
  - It starts with a given value and decreases at each update
  - When it gets to 0, it means the particle is dead, and it should be removed
  - Lifespan can be mapped into alpha, to make the particle fade out with time

```
# Particle.pde
class Particle{
  PVector loc, vel, acc;
  float radius, lifespan;
  AgentMover(PVector pos, float r,
                          float ls){
    this.pos= pos.copy();
    this.vel = new PVector();
    this.acc = new PVector();
    this.radius=r;
    this.lifespan=ls}
  void update(){
    this.vel.add(this.acc);
    this.loc.add(this.vel);
    this.acc.mult(0);}
  void applyForce(PVector force){
    this.acceleration.add(force);}
  void draw(){
    /* draw */ }
}
```

PARTICLE SYSTEM

- Now we create a class for ParticleSystem that organizes particles
  - We will make use of ArrayList and function overloading
- ArrayList are an advanced type in Java that support easy adding, removing and iteration
  - particles.add(), particles.get(int i), particles.remove(int i), particles.size()
- Function overloading means to define a function several times with different parameters
  - In the example, either an origin is specified, or it is automatically defined as the middle of the screen

```
# ParticleSystem.pde
class ParticleSystem{
  ArrayList<Particle> particles;
  PVector origin;
  ParticleSystem(){
    this.particles = new ArrayList<Particle>();
    this.origin=new PVector(width/2, height/2);
  }
  ParticleSystem(PVector origin){
    this.particles = new ArrayList<Particle>();
    this.origin=origin.copy();
  }
```

PARTICLE SYSTEM

```
# ParticleSystem.pde
class ParticleSystem{
  ArrayList<Particle> particles; PVector origin;
  ParticleSystem(){
    this.particles = new ArrayList<Particle>(); // here we store the particles
    this.origin=new PVector(width/2, height/2); // this is the origin of the system
  }
  ParticleSystem(PVector origin){ /*see prev. slide*/}
  void addParticle(){
    this.particles.add(new Particle(this.origin, 10, random(0,255)));}
  void draw(){
    for(int i=this.particles.size()-1; i>=0; i--){
      Particle p=this.particles.get(i);
      /* your code */
      p.draw(); p.lifespan-=0.3;
      if(p.isDead()){particles.remove(i); this.addParticle();}
    }
  }
}
```

PARTICLE SYSTEM

```
# ParticleSystem.pde
class ParticleSystem{
  ArrayList<Particle> particles; PVector or
  ParticleSystem(){
    this.particles = new ArrayList<Particle
    this.origin=new PVector(width/2, height
  }
  ParticleSystem(PVector origin){ /*see prev. sl
  void addParticle(){
    this.particles.add(new Particle(thi  rigin, 10, random(0,255)));}
  void draw(){
    for(int i=this.particles.size()-1; i>=0; i--){
      Particle p=this.particles.get(i);
      /* your code */
      p.draw(); p.lifespan-=0.3;
      if(p.isDead()){particles.remove(i); this.addParticle();}
    }
  }
}
```

In this loop, we are dynamically changing the size of the ArrayList, removing particles when their lifespan has expired and adding new particles.

In order to be sure to read all particles, we invert the loop: start from the last particle and go back

PARTICLE SYSTEM

```
# ParticleSystem.pde
class ParticleSystem{
  ArrayList<Particle> particles; PVector
  ParticleSystem(){
    this.particles = new ArrayList<Partic
    this.origin=new PVector(width/2, heig
  }
  ParticleSystem(PVector origin){ /*see p
  void addParticle(){
    this.particles.add(new Particle(thi
  void draw(){
    for(int i=this.particles.size()
      Particle p=this.particles.g
      /* your code */
      p.draw(); p.lifespan-=0.3;
      if(p.isDead()){particles.remove(i); this.addParticle();}
    }
  }
}
```

This will make sure at some point the lifespan will make particle to die.
We use 255 as maximum value (so we can map it to alpha) and remove a value z (in this case, 0.3) at each iteration.
The final maximum lifespan of a particle is
Number of loops=255/z
Seconds = 255/z/refresh_rate
Usually refresh_rate is 60Hz, so particles can leave at maximum 255/0.3/60 = 14.166666 seconds

PARTICLE SYSTEM

```
# ParticleSystem.pde
class ParticleSystem{
  ArrayList<Particle> particles; PVector origin;
  ParticleSystem(){
    this.particles = new ArrayList<Particle>(); // here we store the particles
    this.origin=new PVector(width/2, heig
  }
  ParticleSystem(PVector origin){ /*see p
  void addParticle(){
    this.particles.add(new Particle(this
  void draw(){
    for(int i=this.particles.siz          i>
      Particle p=this.particl      get(i);
      /* your code */
      p.draw(); p.lifespan-=0.3;
      if(p.isDead()){particles.remove(i); this.addParticle();}
    }
  }
}
```

We must implement a isDead() method for Particle that returns a Boolean (true/false) value whether the particle is dead (lifespan lower than 0).
In this case the system removes it from the set and add a new one

PARTICLE SYSTEM

Let's create a simple source of particles:

- origin following the mouse
- Apply a random (small) acceleration to each particle
- Implement isDead method to the particle class
- Use lifespan for the alpha value for each particle

```
  }
  Part      em(PVector origin){ /*see pr
  void a    icle(){
    this.   icles.add(new Particle(this.o
  void dra  ){
    for(int i=this.particles.size()-1; i>=
      Particle p=this.particles.get(i);
      /* your code */
      p.draw(); p.lifespan-=0.3;
      if(p.isDead()){particles.remove(i);
    }
  }
}
```

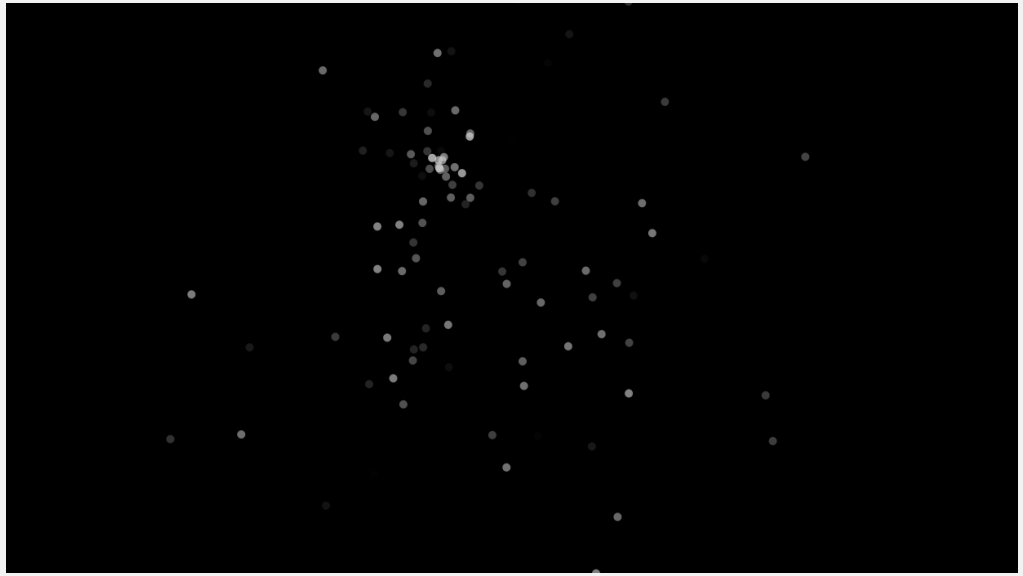
```
# simpleParticleSystem.pde
ParticleSystem ps;
int Nparticles=100;
void setup(){
  size(1280,720);
  ps=new ParticleSystem();
  for(int p=0; p<Nparticles; p++){
    ps.addParticle();
  }
  background(0);
}

void draw(){
  background(0);
  ps.origin=new PVector(mouseX, mouseY);

  ps.draw();
}
```

PARTICLE SYSTEM

Let's create a simple source of particles

What if you change color?

Play with it changing parameters



PARTICLE SYSTEM + TEXTURES

- Particle systems are a useful way to handle multiple agents under one system
- Plotting circles or other shape is nice, but very limiting
- Particles are extremely effective when combined with textures
- Even by just replacing circles with fuzzy-edge circles is a great evolution
- See lab4/data/texture.png
- How does it change our code for using textures?

PARTICLE SYSTEM + TEXTURES

- How does it change our code for using textures?
- In main script:
  - We add the texture in an image
  - Use additive Blender to add layers with each other → glowing effect

```
# simpleParticleSystem.pde
ParticleSystem ps;
int Nparticles=100;
void setup(){
  size(1280,720);
  // ...
}
void draw(){
  background(0);
   // ...}
```

```
# textureParticleSystem.pde
// ...
PImage img;
void setup(){
  size(1280,720, P2D);
  img=loadImage("texture.png");
   // ...
}
void draw(){
  blendMode(ADD);
   // ...
}
```

PARTICLE SYSTEM + TEXTURES

- How does it change our code for using textures?
- In Particle:
  - We just render an image instead of drawing a circle

```
# Particle.pde
class Particle{
  // ...
  void action(){
    this.planning();
    fill(255, this.ls);
    ellipse(this.location.x,
            this.location.y,
            this.radius_circle,
            this.radius_circle);
}
}
```

```
# Particle.pde
class Particle{
  // ...
  void action(){
    this.planning();
    imageMode(CENTER);
    tint(255, this.lifespan);
    image(img, this.location.x,
                this.location.y); }
}
```

PARTICLE SYSTEM + TEXTURES

- Let's create a smokey effect
- Use the provided texture + the previous particle system
- Requirements:
  - Origin is at 0.75*width, height
  - Velocity of each particle is set at $\mathcal{N}(\boldsymbol{\mu}, \sigma)$, i.e., a value from a normal distribution with mean $\boldsymbol{\mu} = [0, -1]$ and standard deviation $\sigma = 0.3$.
- At each step, apply to every particles a horizontal wind force, i.e., a Pvector with y =0
- For the x, we use a Microphone input (for macOS, a recording of the wind), and we extract the energy as we did in the first lab
  - `Pvector wind= new PVector(-audio.Energy(), 0);`
- Use 1000 particles

Processing provides a function randomGaussian() that outputs random values drawn from $\mathcal{N}(0,1)$.

Remember that
$\mathcal{N}(\boldsymbol{\mu}, \sigma) = \mathcal{N}(0,1)\sigma + \mu$

PARTICLE SYSTEM + TEXTURES

```
# textureParticleSystem.pde
AudioIn audio;
boolean song_mic=true;
void setup(){
  /* ... */
  audio=new AudioIn(song_mic, this);
}

PVector computeWind(){
  float energy= audio.getEnergy();
  // your code
}

void draw(){
  blendMode(ADD);
  background(0);
ps.action(computeWind());
}
```

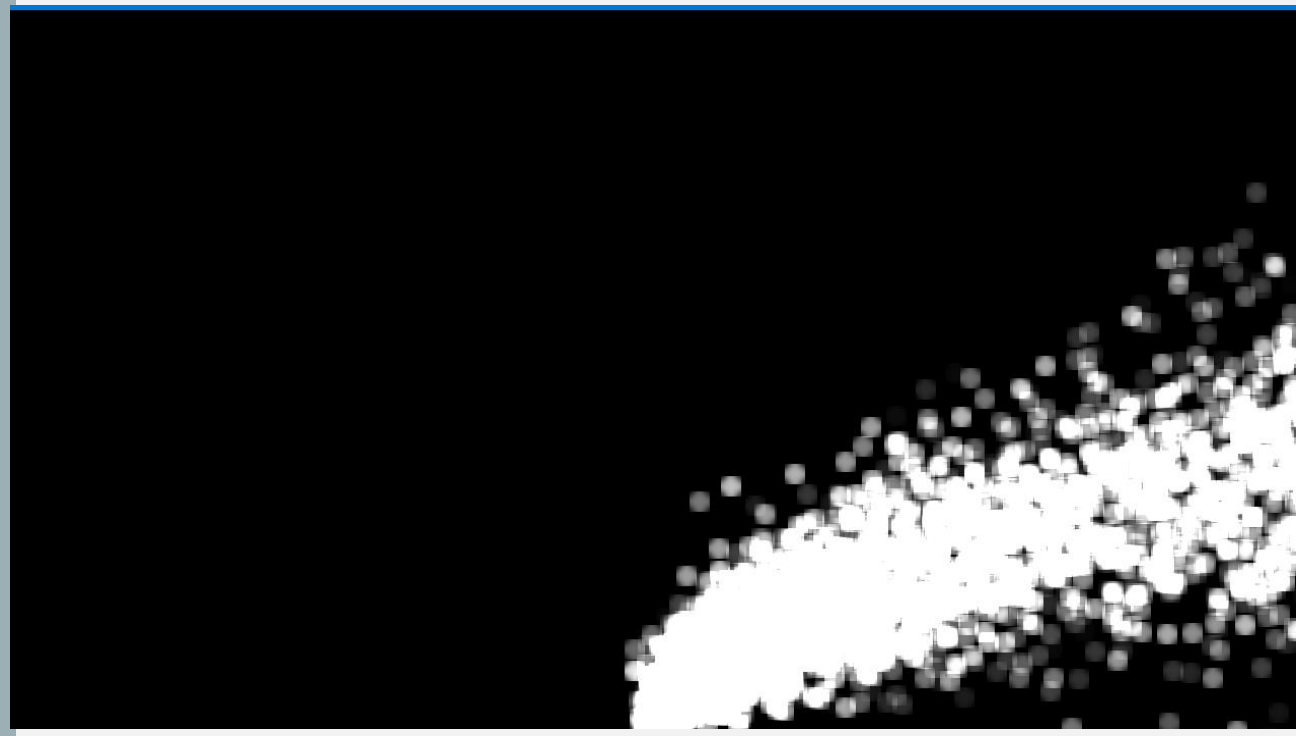
```
# AudioIn.pde
import ddf.minim.*; import ddf.minim.analysis.*;
int frameLength = 1024;
String path="../../../data/wind.mp3“;
class AudioIn{ // variables...
   AudioIn(boolean song_mic,
            textureParticleSystem app){
     this.minim= new Minim(app);
     this.song_mic=song_mic;
     if(this.song_mic){ // load a file }
     else{// use mic input}}
  float getEnergy(){
     if(this.song_mic){this.fft.forward(
                                    this.song.mix);}
     else{this.fft.forward(this.mic.mix);}
    float energy = 0;
    /* your code */
    energy=map(energy,0, this.fft.specSize(), 0, 1);
    this.energy= this.energy*0.1+energy*0.9;
    return this.energy;
  }
}
```

PARTICLE SYSTEM + TEXTURE

Now the wind is controlled by your mouse movement

Can you control the wind with some musical features?

Can you combine wind + sinusoidal motion?



BOIDS & BOX2D

BOIDS & BOX2D

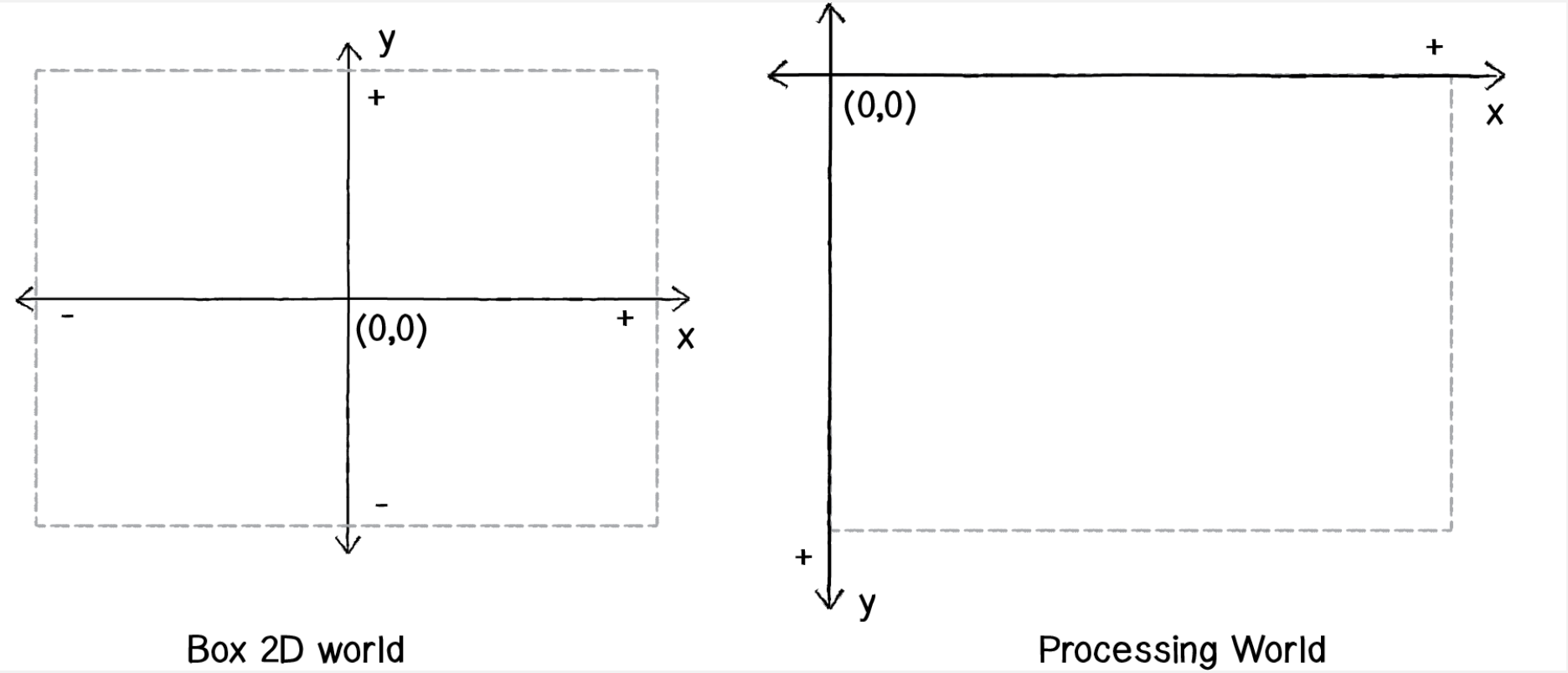
- With *boids* we refer to bird-like objects that act in a multiagent system and they are aware of the surrounding
- Boids' flocking behavior is the result of three rules:
  - Avoid the other boids, i.e., do not collide with them
  - Align their direction with nearby boids
  - Approach to distant boids
- More rules can be defined to, for example, avoid predators
- Online we can find several examples of boids that implement such rules
- Let's instead join the concept of boid with another one: a **physical engine**

BOIDS & BOX2D

- A **physical engine** is a set of rules which emulate Physics
- In the second lab we have seen a small amount of a physical engine by looking at how the object can follow rule of physics
- However, each agent was moving on their own and they may even overlap
- We want to implement **collisions**
- Instead of implementing a complex system of collisions and what happen in that case, we will use the box2d physical engine, which will do that for us
- The box2d engine is different from the system we made with PVector
- Install **box2d on Processing**
- https://pub.dev/documentation/box2d_flame/latest/box2d/box2d-library.html

BOIDS & BOX2D

- There is a main difference on how we define the world in Processing and in Box2D
  - (there are function to make the convertion for us)



BOIDS & BOX2D

- Instead of Pvector, in Box2D we use Vec2,
  - The syntax is slightly different as well

PVector	Vec2D
PVector a = new PVector(1,-1); PVector b = new PVector(3,4); a.add(b);	Vec2D a = new Vec2D(1,-1); Vec2D b = new Vec2D(3,4); a.addLocal(b);
PVector a = new PVector(1,-1); PVector b = new PVector(3,4); PVector c = PVector.add(a,b);	Vec2D a = new Vec2D(1,-1); Vec2D b = new Vec2D(3,4); Vec2D c = a.add(b);

PVector a = new PVector(1,-1);
float m = a.mag();
a.normalize();
	Vec2D a = new Vec2D(1,-1);
	float m = a.length();
	a.normalize();

BOIDS & BOX2D

- When using box2d, we need to create a box2d world

```
# example.pde
import org.jbox2d.collision.shapes.*;
import org.jbox2d.common.*;
import org.jbox2d.dynamics.*;
import org.jbox2d.dynamics.contacts.*;
Box2DProcessing box2d;
void setup(){
  box2d = new Box2DProcessing(this);
  box2d.createWorld(); // create world
  box2d.setGravity(0, 0); // no gravity
  ...
}
```

BOIDS & BOX2D

- When using box2d, we need to create a box2d world
- Then create a body with a certain body definition and shape as its *fixture*
  - A DYNAMIC body will move, while a STATIC body is used to draw boundaries or terrain
  - The body automatically implements a function *applyForce*, so we don't have to write it

```
# example.pde
import org.jbox2d.collision.shapes.*;
import org.jbox2d.common.*;
import org.jbox2d.dynamics.*;
import org.jbox2d.dynamics.contacts.*;
Box2DProcessing box2d;
void setup(){
  box2d = new Box2DProcessing(this);
  box2d.createWorld(); // create world
  box2d.setGravity(0, 0); // no gravity
  bd= new BodyDef(); // body definition
  bd.type= BodyType.DYNAMIC;
  cs  = new CircleShape();
  cs.m_radius = P2W(RADIUS_CIRCLE/2);
  bd.linearDamping=0;
  Vec2 position=P2W(new Pvector(witdh/2,
                                height/2));
  bd.position.set(position);
  body = box2d.createBody(bd);
  body.m_mass=1;
  body.createFixture(ps, 1);
}
```

BOIDS & BOX2D

- When using box2d, we need to create a box2d world
- Then create a body with a certain body definition and shape as its *fixture*
  - A DYNAMIC body will move, while a STATIC body is used to draw boundaries or terrain
  - The body automatically implements a function *applyForce*, so we don't have to write it
- we need to convert positions and dimensions from the Pixel domain to the box domain

```
# example.pde
import org.jbox2d.collision.shapes.*;
import org.jbox2d.common.*;
import org.jbox2d.dynamics.*;
import org.jbox2d.dynamics.contacts.*;
Box2DProcessing box2d;
void setup(){
  box2d = new Box2DProcessing(this);
  box2d.createWorld(); // create world
  box2d.setGravity(0, 0); // no gravity
  bd= new BodyDef(); // body definition
  bd.type= BodyType.DYNAMIC;
  cs  = new CircleShape();
  cs.m_radius = P2W(RADIUS_CIRCLE/2);
  bd.linearDamping=0;
  Vec2 position=P2W(new Pvector(witdh/2,
                                height/2));
  bd.position.set(position);
  body = box2d.createBody(bd);
  body.m_mass=1;
  body.createFixture(ps, 1);
}
```

BOIDS & BOX2D

- We need to convert positions and dimensions from the Pixel domain to the box domain
- I defined two functions for you, using function overloading
  - P2W: from Pixels to World
  - W2P: from World to Pixels

```
# example.pde
Vec2 P2W(Vec2 in_value){
  return box2d.coordPixelsToWorld(in_value);}

Vec2 P2W(float pixelX, float pixelY){
  return box2d.coordPixelsToWorld(pixelX, pixelY);}

Vec2 W2P(Vec2 in_value){
  return box2d.coordWorldToPixels(in_value);}

Vec2 W2P(float worldX, float worldY){
  return box2d.coordWorldToPixels(worldX, worldY);}

float P2W(float in_value){
  return box2d.scalarPixelsToWorld(in_value);}

float W2P(float in_value){
  return box2d.scalarWorldToPixels(in_value);}
```

BOIDS & BOX2D

- We need to convert positions and dimensions from the Pixel domain to the box domain
- I defined two functions for you, using function overloading
  - P2W: from Pixels to World
  - W2P: from World to Pix

These functions refer to the Vector worlds, i.e., they convert coordinates from Pixel to World and viceversa

```
# example.pde
Vec2 P2W(Vec2 in_value){
  return box2d.coordPixelsToWorld(in_value);}

Vec2 P2W(float pixelX, float pixelY){
  return box2d.coordPixelsToWorld(pixelX, pixelY);}

Vec2 W2P(Vec2 in_value){
  return box2d.coordWorldToPixels(in_value);}

Vec2 W2P(float worldX, float worldY){
  return box2d.coordWorldToPixels(worldX, worldY);}

float P2W(float in_value){
  return box2d.scalarPixelsToWorld(in_value);}

float W2P(float in_value){
  return box2d.scalarWorldToPixels(in_value);}
```

BOIDS & BOX2D

- We need to convert positions and dimensions from the Pixel domain to the box domain
- I defined two functions for you, using function overloading
  - P2W: from Pixels to World
  - W2P: from World to Pixels

These functions refer to the scalar worlds, i.e., they convert sizes from Pixel to World and viceversa

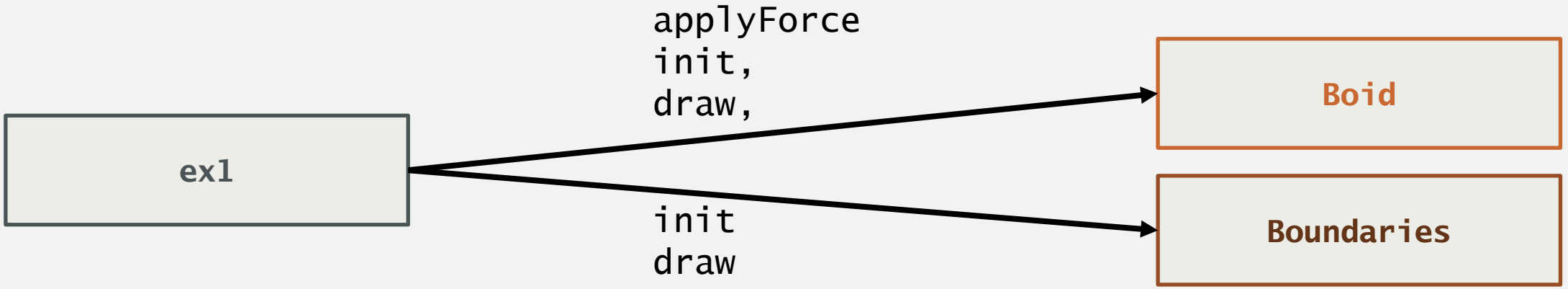
```
# example.pde
Vec2 P2W(Vec2 in_value){
  return box2d.coordPixelsToWorld(in_value);}

Vec2 P2W(float pixelX, float pixelY){
  return box2d.coordPixelsToWorld(pixelX, pixelY);}

Vec2 W2P(Vec2 in_value){
  return box2d.coordWorldToPixels(in_value);}

Vec2 W2P(float worldX, float worldY){
  return box2d.coordWorldToPixels(worldX, worldY);}

float P2W(float in_value){
  return box2d.scalarPixelsToWorld(in_value);}

float W2P(float in_value){
  return box2d.scalarWorldToPixels(in_value);}
```

BOIDS & BOX2D

EX 1: Let's start to make together our first «stupid» boid

- We start initializing a Box2D world
- Every time we left-click on the screen, a new boid is created
- We give them a random force at the beginning and a force whenever we press the right click button
- We include boundaries (STATIC bodies) at the sides of our screen
- Apart for that, they are free to move wherever they want
- The engine is in charge of collisions

BOIDS & BOX2D



Look at the code and let's fill

`ex1->mousePressed()`

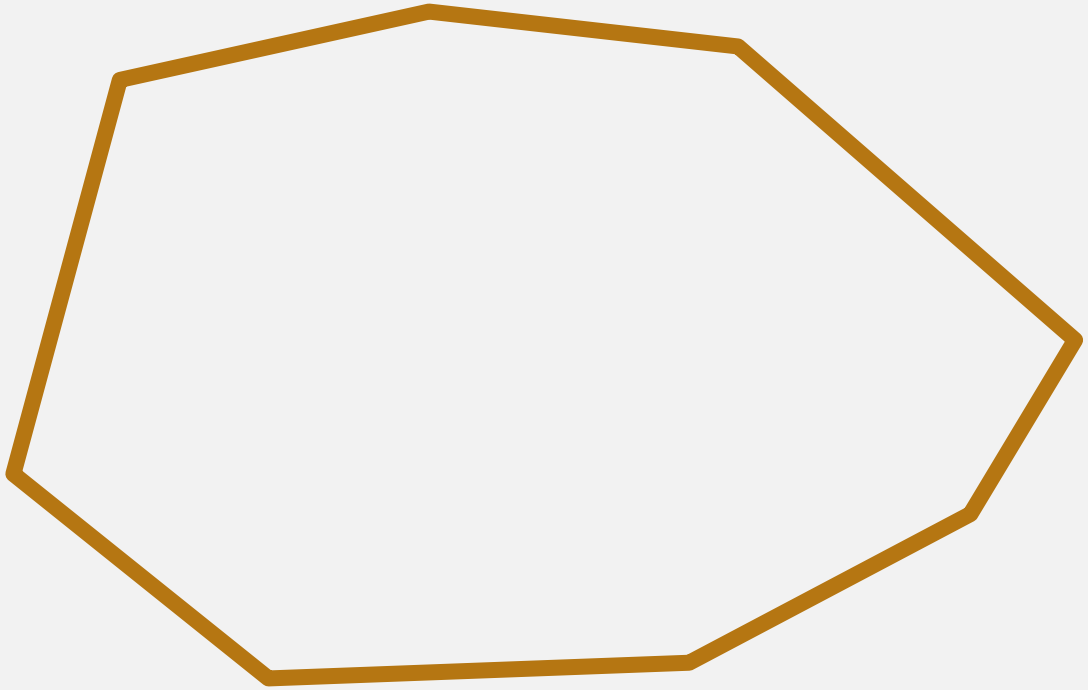
`Boid->draw()`

```
# ex1.pde
void draw() {
  fill(0,50);
  rectMode(CENTER);
  rect(width/2, height/2, width, height);
  box2d.step(); // THIS makes the world update
  boundaries.draw();
  for (Boid b : boids) {
    b.draw();
  }
}
```

BOIDS & BOX2D

EX 2: Let's make our boid follow a Path:

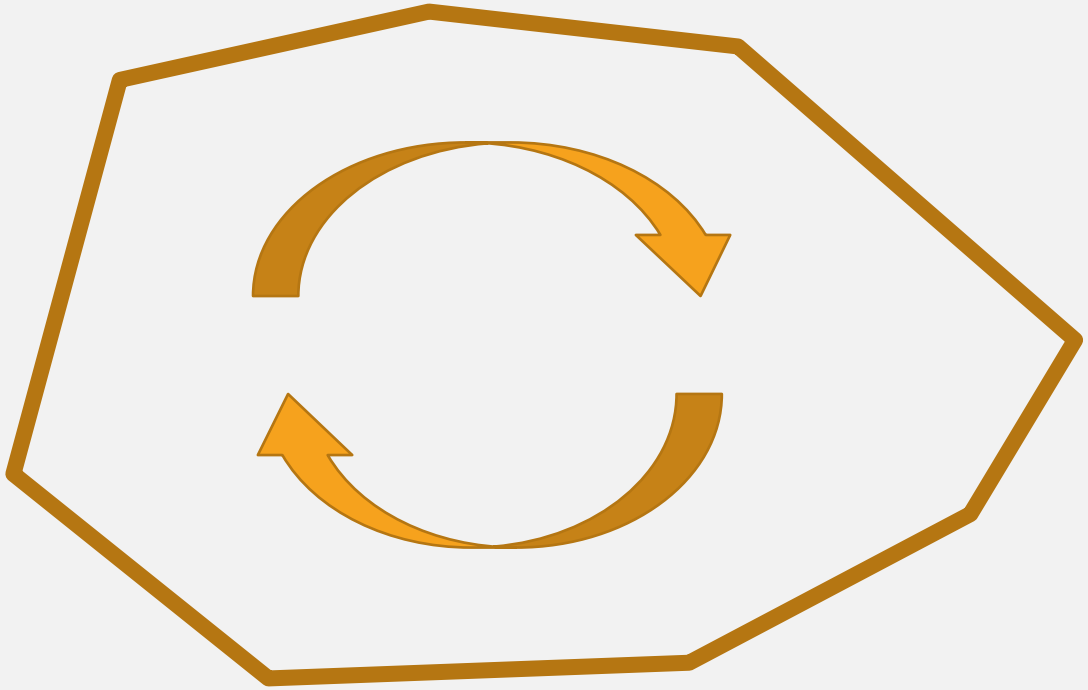
- We draw a Path as a set of corners (points)
- At every instant the boid steers to the closest point
- Once it gets close enough to the point, we mark it as «passed» and steer toward the following point



BOIDS & BOX2D

EX 2: Let's make our boid follow a Path:

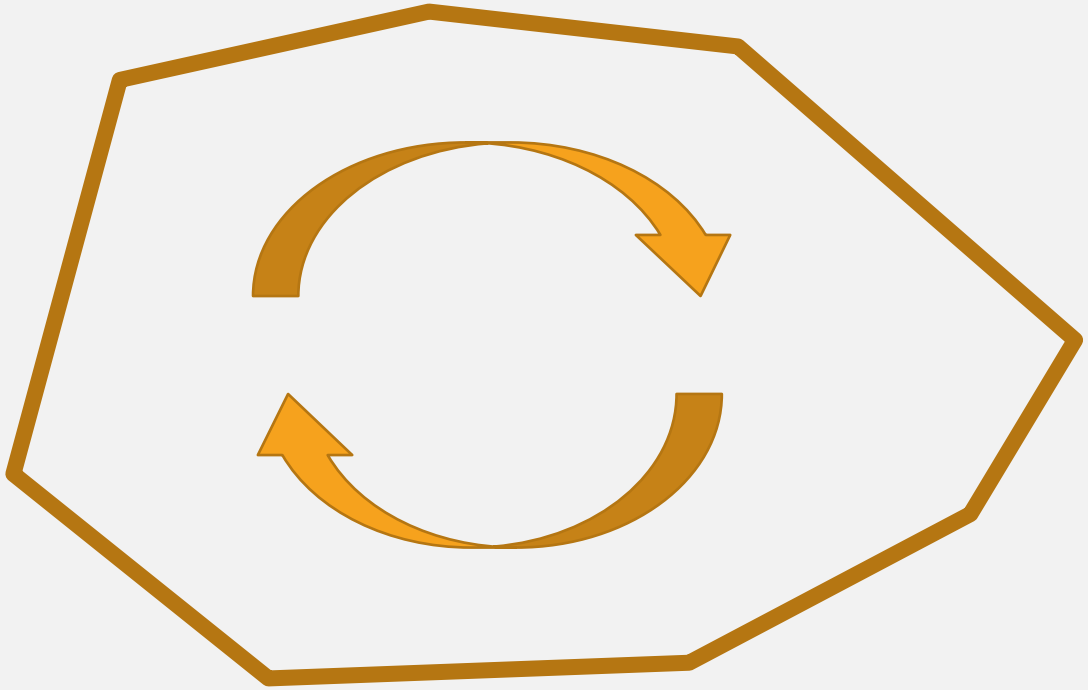
- We draw a Path as a set of corners (points)
- At every instant the boid steers to the closest point
- Once it gets close enough to the point, we mark it as «passed» and steer toward the following point
- Let's make it clockwise



BOIDS & BOX2D

EX 2: Let's make our boid follow a Path:

- Basic idea for the initialization: use N angles θ_i equally spaced between 0 and 2π
- Find a random scale factor f_i as a distance between the middle for each centre
- Compute each point as
  - $\boldsymbol{p}_i = \left[\frac{w}{2} + f_i w \cos(\theta_i), \frac{h}{2} + f_i h \sin(\theta_i)\right]$



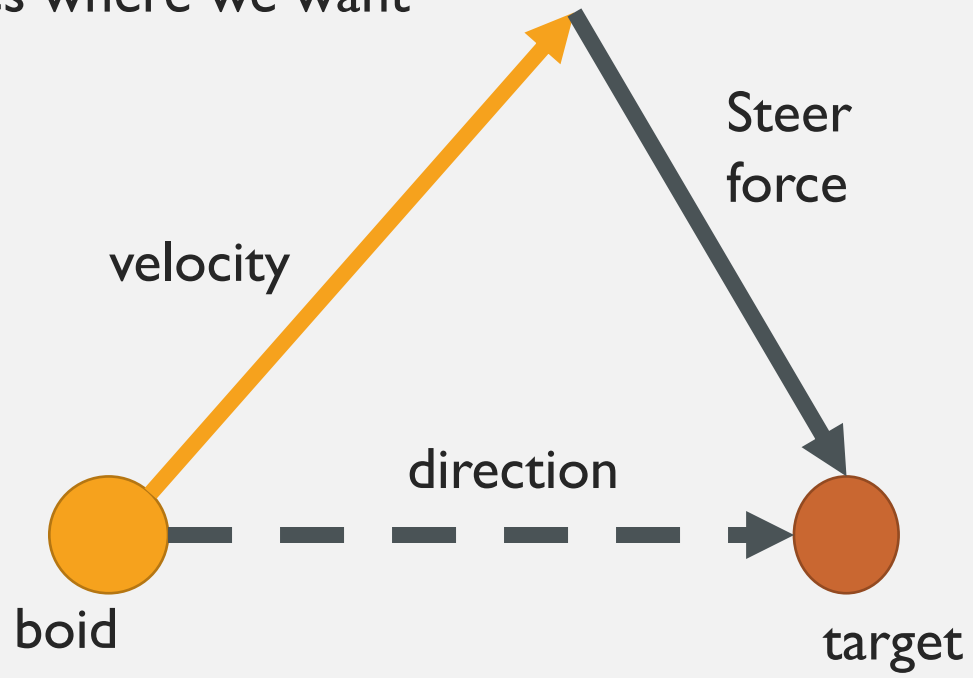
BOIDS & BOX2D

```
# Path.pde
class Path{  Vec2[] pointsP, pointsW; int num_points; float alpha=0.4;
  Path(int num_points, float min_fact, float max_fact){
    this.num_points=num_points; float angle; float fact=0.5*(min_fact+max_fact);
    this.pointsW = new Vec2[this.num_points]; this.pointsP = new Vec2[this.num_points];
    for(int i=0; i<this.num_points; i++){
      angle=map(i, 0, this.num_points, 0, 2*PI);
      fact=this.alpha*random(min_fact, max_fact)+(1-alpha)*fact
      this.pointsP[i]=new Vec2(width*(0.5+fact*cos(angle)),height*(0.5+fact*sin(angle)));
      this.pointsW[i]=P2W(this.pointsP[i]);
    }
  }
  void draw(){ # your code: draw lines}
  int closestTarget(Vec2 posW){
    # your code: must return the index of the closest point to posW
  }
  int nextPoint(int i){return (i+1)%this.num_points;}
  Vec2 getDirection(Vec2 posW, int i){ return this.pointsW[i].sub(posW);}
}
```

BOIDS & BOX2D

EX 2: Let's make our boid follow a Path:

- We want to «steer» the boid, i.e., push it so that it goes where we want
- Given the velocity vector, we use a force as
- Steering_force=direction – velocity
- We limit the force it in order not to overshoot



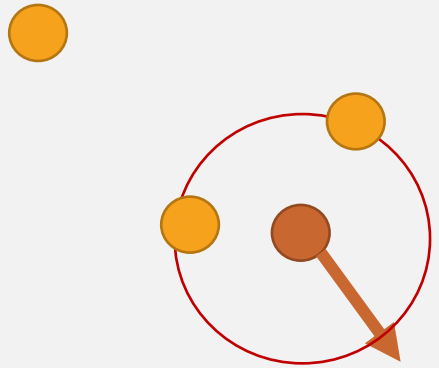
BOIDS & BOX2D

```
# ex2.pde
Vec2 computeForce(Boid b){
  Vec2 posW= b.body.getPosition();
  Vec2 direction1= path.getDirection(posW, b.nextPoint);
  Vec2 direction2= path.getDirection(posW, path.nextPoint(b.nextPoint));
  Vec2 direction;
  if(direction.length() < P2W(DIST_TO_NEXT) ||
     direction2.length() < direction1.length()){
    b.nextPoint=path.nextPoint(b.nextPoint); direction= direction2;
  }else{direction=direction1;}
  // your code: compute steering
  Vec2 velocity=b.body.getLinearVelocity();
  return steering;
} ...
void draw() {
  if(DRAW_PATH){  path.draw();}
  box2d.step(); boundaries.draw();
  for (Boid b : boids) {
    b.applyForce(computeForce(b));
    b.draw();}
}
```

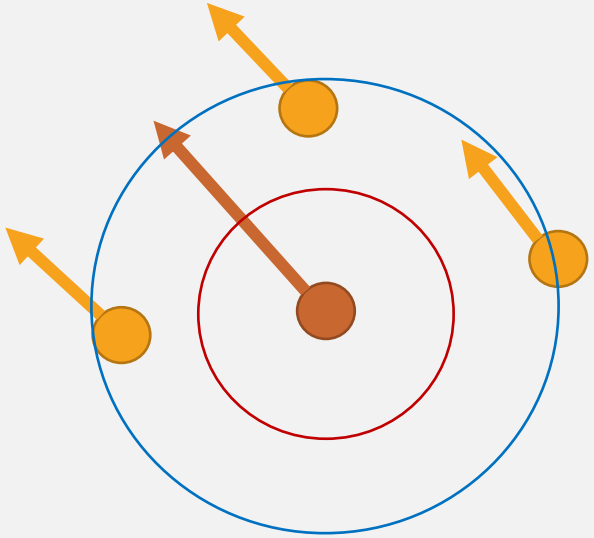
BOIDS & BOX2D

EX 3: Make the boids behave like boids

- (try to) **avoid collisions**: whenever other boids are closest than AVOID_DIST, apply a force that is the opposite of the direction toward them, in order to make space



- **align**: steer to align to the direction of boids who are closest than ALIGN_DIST
  - (but further than AVOID_DIST)



- Implement the method `update(ArrayList<Boid> boids);`

BOIDS & BOX2D

```
# Boid.pde
float AVOID_DIST=6;
float ALIGN_DIST=25;
Class Boid{//...
  void update(ArrayList<Boid> boids){
    Vec2 myPosW=this.body.getPosition(); Vec2 otherPosW;
    Vec2 myVel=this.body.getLinearVelocity(); Vec2 otherVel;
    Vec2 direction; float dist;
    Vec2 align_force=new Vec2(0,0); Vec2 avoid_force=new Vec2(0,0);
    for(Boid other: boids){
    // avoid considering the boid itself
      if(this.body==other.body){continue;}
      // your code }
    // your code
    if(avoid_force.length()>0){
      this.applyForce(avoid_force);}
    if(align_force.length()>0){
      this.applyForce(align_force);}
  }
```

```
# ex3.pde
void draw() {
  //...
  for (Boid b : boids) {
    b.applyForce(computeForce(b));
    b.update(boids);
    b.draw();
  }
}
```

- Implement the method `update(ArrayList<Boid> boids);`

BOIDS & BOX2D

EX 4: Behavior during collisions!

- We want to listen to collisions, i.e., know when a collision happen
- We want the boid to *react* to the collision by
  - Playing a sound
  - Briefly changing color
- Use the Python script to generate sounds we will use for this script
  - Execute `create_sounds.py`

BOIDS & BOX2D

```
# create_sounds.py
#...
env=np.zeros((int(DUR*sr),));
env[:N]=np.sin(np.linspace(0, np.pi, N)) # envelope

if __name__=="__main__":
  for f, freq in enumerate(freqs):
     fn_out="sounds/%.2fHz.wav"%(freq)
     T=int(sr/freq);
     if Osc_type=="square":
        osc=np.zeros((T,))-1;
        osc[int(T/4):int(-T/4)]=1
     elif Osc=="saw":
        osc=np.concatenate([np.linspace(-1,1,int(T/2)),
                            np.linspace(1,-1,T-int(T/2))])
     sample=np.tile(osc, (1+int(DUR/freq),))
     sample=sample[:int(sr*DUR)]
     sample*=env
     sf.write(fn_out, 0.707*sample/np.max(np.abs(sample)), sr)
```

Creating an envelope to avoid abrupt attack

Period in sample corresponding to frequency

Creating the basic shape

Repeat and apply envelope

Write

BOIDS & BOX2D

EX 4: Behavior during collisions!

- How to make box2d react to collisions?

We ask box2d to listen for Collision

We retrieve the Boid connected with a Body by setting the Boid as "User Data" of the body

Every time a new collision occurs, the function beginContact will be called

```
# ex4.pde
void setup() {
  //...
  box2d.createWorld();
  box2d.listenForCollisions();
  //... }

void beginContact(Contact cp) {
  Body body1 = cp.getFixtureA().getBody();
  Body body2 = cp.getFixtureB().getBody();
  Boid b1 = (Boid) body1.getUserData();
  Boid b2 = (Boid) body2.getUserData();
  if (b1!=null) {b1.play(); b1.changeColor();}
  if (b2!=null) {b2.play(); b2.changeColor();}
}
```

```
# Boid.pde
Boid(/*...*/){
  this.box2d = box2d;
  bd.position.set(position);
  /*...*/
  this.body.setUserData(this);
  /* ... */
}
```

BOIDS & BOX2D

EX 4: Behavior during collisions!

- Implement Boid.play and Boid.changeColor()
- we change the constructor of the Boid

```
# ex4.pde
void beginContact(Contact cp) {
  Body body1 = cp.getFixtureA().getBody();
  Body body2 = cp.getFixtureB().getBody();
  Boid b1 = (Boid) body1.getUserData();
  Boid b2 = (Boid) body2.getUserData();
  if (b1!=null) {b1.play(); b1.changeColor();}
  if (b2!=null) {b2.play(); b2.changeColor();}}
```

```
# Boid.pde
class Boid{
  Body body; Box2DProcessing
  box2d; int nextPoint;
  color defColor = color(200, 200, 200); // default color
  color contactColor; // color during contact
  float time_to_color,time_index;
  SoundFile sample;
  Boid(Box2DProcessing  box2d, CircleShape ps, BodyDef bd,
       Vec2 position, SoundFile sample, int nextPoint){
    /*...*/
  }
```

BOIDS & BOX2D

EX 4: Behavior during collisions!

- Implement Boid.play and Boid.changeColor()
- we change the constructor of the Boid

```
# ex4.pde
void beginContact(Contact cp) {
  Body body1 = cp.getFixtureA().getBody();
  Body body2 = cp.getFixtureB().getBody();
  Boid b1 = (Boid) body1.getUserData();
  Boid b2 = (Boid) body2.getUserData();
  if (b1!=null) {b1.play(); b1.changeColor();}
  if (b2!=null) {b2.play(); b2.changeColor();}}
void setup(){  //...
  String path=sketchPath()+"/sounds";
  File dir = new File(path);
  filenames= dir.list();} // wavfiles
```

```
# Boid.pde
class Boid{
  Body body; Box2DProcessing
  box2d; int nextPoint;
  color defColor = color(200, 200, 200); // default color
  color contactColor; // color during contact
  float time_to_color,time_index;
  SoundFile sample;
  Boid(Box2DProcessing  box2d, CircleShape ps, BodyDef bd,
       Vec2 position, SoundFile sample, int nextPoint){
    /*...*/
  }
```

BOIDS & BOX2D

EX 4: Behavior during collisions!

- Implement Boid.play and Boid.changeColor()
- we change the constructor of the Boid

```
# Boid.pde
class Boid{
  Body body; Box2DProcessing
  box2d; int nextPoint;
  color defColor = color(200, 200, 200); // de
  color contactColor; // color during contact
  float time_to_color,time_index;
  SoundFile sample;
  Boid(Box2DProcessing  box2d, CircleShape ps,
       Vec2 position, SoundFile sample, int ne
     /*...*/
  }
```

```
# ex4.pde
void beginContact(Contact cp) {
  Body body1 = cp.getFixtureA().getBody();
  Body body2 = cp.getFixtureB().getBody();
  Boid b1 = (Boid) body1.getUserData();
  Boid b2 = (Boid) body2.getUserData();
  if (b1!=null) {b1.play(); b1.changeColor();}
  if (b2!=null) {b2.play(); b2.changeColor();}}
void setup(){  //...
  String path=sketchPath()+"/sounds";
  File dir = new File(path);
  filenames= dir.list(); // wavfiles }
void mousePressed() {
  //insert a new Boid
  Boid b = new Boid(box2d, cs, bd,
                    P2W(mouseX, mouseY),
                    new SoundFile(this,
                       "sounds/"+filenames[i]),
                    p); }
  }
```

GRAVITY AND ATTRACTION

How to change the color in changeColor

1. Define the color you want to use during a collision in `Boid.contactColor;`
2. Define the time (in frames) you want the new color to be active in `Boid.time_to_color`
   - time in frames=time in seconds * frameRate
3. Use the function `lerpColor` to move from one color to the other
   - `Color c=lerpColor(color Color0, color Color1, float value);`
   - when value=0 → `c =Color0;` whenvalue=1 → `c =Color1`
   - When 0<value<1 → c is the mix off the two

GRAVITY AND ATTRACTION

How to change the color in changeColor

1. Define the color you want to use during a collision in `Boid.contactColor;`
2. Define the time (in frames) you want the new color to be active in `Boid.time_to_color`
   - time in frames=time in seconds * frameRate
3. Use the function `lerpColor` to move from one color to the other
   - `Color c=lerpColor(color Color0, color Color1, float value);`
   - when value=0 → `c =Color0;` whenvalue=1 → `c =Color1`
   - When 0<value<1 → c is the mix off the two
4. Use a value related to time in frames and a time index
   - How do you need to update time index with respect to the collision?