

Processing

Movimentos Físicos

Movimentos Físicos

- Biblioteca Externa
 - Physics
- Quatro Partes:
 - [“The ParticleSystem](#), which basically takes of care of everything
 - [Particles](#), which move around in space according to the forces acting on them
 - [Springs](#), each which acts on 2 particles
 - [Attractions/repulsions](#), which act on 2 particles”

ParticleSystem

- Classe ParticleSystem
- Construtor
 - `ParticleSystem(float gravityY, float drag)`
 - OU
 - `ParticleSystem(float gx, float gy, float gz, float drag)`
- Gere o “mundo”
 - Conhece todas as partículas e forças
 - Calcula as forças, velocidades, posições
 - Permite definir gravidade e atrito

Exemplo_1

```
import traer.physics.*;

// Representa o mundo das particulas
ParticleSystem mundo;

Particle p;

void setup() {
  size(400, 400);

  // gravidade 0.1 e atrito 0.1
  mundo = new ParticleSystem(0.1, 0.1);

  p = mundo.makeParticle(5, 200, 0, 0);
}

void draw() {
  background(0);
  fill(255);
  stroke(255);

  ellipse(p.position().x(),
    p.position().y(), 10, 10);

  // calcula todas as forcas,
  //velocidades e posicoes novas
  mundo.tick();
}
```

Exercício #1

- Modifique o Exemplo #1 de forma a que a partícula comece por aparecer na base do ecrã e suba, em vez de descer...

Particles

- Representam “pontos de massa”
 - Objectos sem dimensão (pontos) mas com massa
- As Particles têm massa, posição, velocidade e idade
- A posição é actualizada automaticamente pelo ParticleSystem
 - O nosso programa apenas tem de saber qual a posição actual
 - `Particle.position().x()`, `Particle.position().y()`, `Particle.position().z()`
- Podemos fixar algumas partículas
 - A sua posição nunca é alterada pelo ParticleSystem, mas influencia as outras
 - O nosso programa pode mudar a posição da partícula

Exemplo_2

```
import traer.physics.*;

// Representa o mundo das partículas
ParticleSystem mundo;

// A nossa partícula
Particle p[];

void setup() {
  size(400, 400);

  // gravidade 0.1 e atrito 0.1
  mundo = new ParticleSystem(0.1, 0.1);

  p = new Particle[200];

  for (int i = 0; i < 200; i++) {
    // Massa 5, e posicao (200, 0, 0)
    p[i] = mundo.makeParticle(random(0.1, 5), random(400), random(-500, 100), 0);
  }
}

void draw() {
  background(0);
  fill(255);
  stroke(255);

  for (int i = 0; i < 200; i++) {
    // Desenhar uma ellipse no lugar da partícula
    ellipse(p[i].position().x(), p[i].position().y(), 5, 5);

    if (p[i].position().y() > 400) {
      p[i].moveTo(random(400), random(-500, 0), 0);
    }
  }

  // calcula todas as forças, velocidades e posições novas
  mundo.tick();
}
```

03-06-2007

Jorge Cardoso

Exercício #2

- Modifique o Exemplo #2, de forma que os “flocos” tenham um tamanho proporcional à massa da partícula

Spring

- Representam molas a ligar pontos de massa
- Têm um comprimento de descanso
 - Qualquer alteração ao comprimento gera uma força aplicada às partículas ligadas pela mola
- A força aplicada pode variar
- `makeSpring(Particle a, Particle b, float strength, float damping, float restLength)`

Exemplo_3

```
import traer.physics.*;

// Representa o mundo das particulas
ParticleSystem mundo;

// A nossa partícula
Particle p;

Particle rato;

void setup() {
  size(400, 400);

  // gravidade 0.1 e atrito 0.1
  mundo = new ParticleSystem(1, 0.1);

  // criar uma partícula que vai estar sempre na posicao do rato
  rato = mundo.makeParticle(1, 0, 0, 0);
  rato.makeFixed();

  // criar uma partícula ligada 'a outra por uma mola
  p = mundo.makeParticle(5, 0, 0, 0);
  mundo.makeSpring(rato, p, 2.0, 0.051, 100);
}
```

```
void draw() {
  background(0);
  fill(255);
  stroke(255);

  // mover a partícula para a posicao do rato
  rato.moveTo(mouseX, mouseY, 0);

  // desenhar a outra partícula e a mola
  ellipse(p.position().x(), p.position().y(), 5, 5);
  line(p.position().x(), p.position().y(), mouseX, mouseY);

  // calcula todas as forcas, velocidades e posicoes novas
  mundo.tick();
}
```

Exercício #3

- Modificar o Exemplo #3, de forma a ter várias partículas ligadas ao rato
 - Ligar 50 partículas
 - Usar massas diferentes para as partículas

Exercício #4

- Modificar o Exemplo #3, de forma a ligar 5 partículas da seguinte forma:
rato – p1 – p2 – p3 – p4

Attractions/Repulsions

- Representam forças de atracção/repulsão entre partículas
- `makeAttraction(Particle a, Particle b, float strength, float minimumDistance)`

Exemplo_4

```

• import processing.opengl.*;
• import traer.physics.*;
• // Representa o mundo das particulas
• ParticleSystem mundo;
• Particle rato;
• Particle afasta;
• Particle atrai;
• void setup() {
•   size(800, 600);
•   // gravidade 0.1 e atrito 0.1
•   mundo = new ParticleSystem(0.1, 0.6);
•   rato = mundo.makeParticle(1, 0, 0, 0);
•   rato.makeFixed();
•   atrai = mundo.makeParticle(50, 400, 300, 0);
•   mundo.makeAttraction(rato, atrai, 1000, 10);
•   afasta = mundo.makeParticle(5, 400, 300, 0);
•   mundo.makeAttraction(rato, afasta, -1000, 10);
• }
• void draw() {
•   background(0);
•   fill(255);
•   stroke(255);
•   rato.moveTo(mouseX, mouseY, 0);
•   // Desenhar a particula que atrai
•   fill(255);
•   ellipse(atrai.position().x(), atrai.position().y(), 10, 10);
•   // Desenhar a que afasta
•   fill(100);
•   ellipse(afasta.position().x(), afasta.position().y(), 10, 10);
•   // Manter as particulas dentro do ecrã
•   if (atrai.position().x() < 50 || atrai.position().x() > width-50) {
•     atrai.setVelocity( -0.9*atrai.velocity().x(), atrai.velocity().y(), 0 );
•   }
•   if (atrai.position().y() < 50 || atrai.position().y() > height-50) {
•     atrai.setVelocity(atrai.velocity().x(), -0.9*atrai.velocity().y(), 0 );
•   }
•   atrai.moveTo( constrain( atrai.position().x(), 50, width-50 ), constrain(
•     atrai.position().y(), 50, height-50 ), 0 );
•   // Manter as particulas dentro do ecrã
•   if (afasta.position().x() < 50 || afasta.position().x() > width-50) {
•     afasta.setVelocity( -0.9*afasta.velocity().x(), afasta.velocity().y(), 0 );
•   }
•   if (afasta.position().y() < 50 || afasta.position().y() > height-50) {
•     afasta.setVelocity( afasta.velocity().x(), -0.9*afasta.velocity().y(), 0 );
•   }
•   afasta.moveTo( constrain( afasta.position().x(), 50, width-50 ), constrain(
•     afasta.position().y(), 50, height-50 ), 0 );
•   // calcula todas as forcas, velocidades e posicoes novas
•   mundo.tick();
• }

```

Exercício #5

- Modifique o Exemplo #4, de forma a ter mais duas partículas