Modelação e Simulação de Sistemas Naturais

Trabalho Prático 2

52553 - Artur Assis
52864 - Tiago Teixeira

Licenciatura em Engenharia Informática e Multimédia
Semestre de Inverno 2025/2026
23/11/2025

Índice

Introdução

Este relatório apresenta o desenvolvimento e a análise de um conjunto de simulações no âmbito da disciplina de Modelação e Simulação de Sistemas Naturais, realizadas com a biblioteca do Processing. O trabalho está organizado em quatro blocos principais, cada um explorando conceitos fundamentais da modelação de sistemas físicos e comportamentos inteligentes.

No primeiro bloco, “Queda Livre com Atrito”, foi simulada a dinâmica de um corpo em queda sob a ação da gravidade e de forças de atrito aerodinâmico. Este modelo permitiu analisar sistemas não lineares e o papel dos ciclos de feedback no comportamento dinâmico.

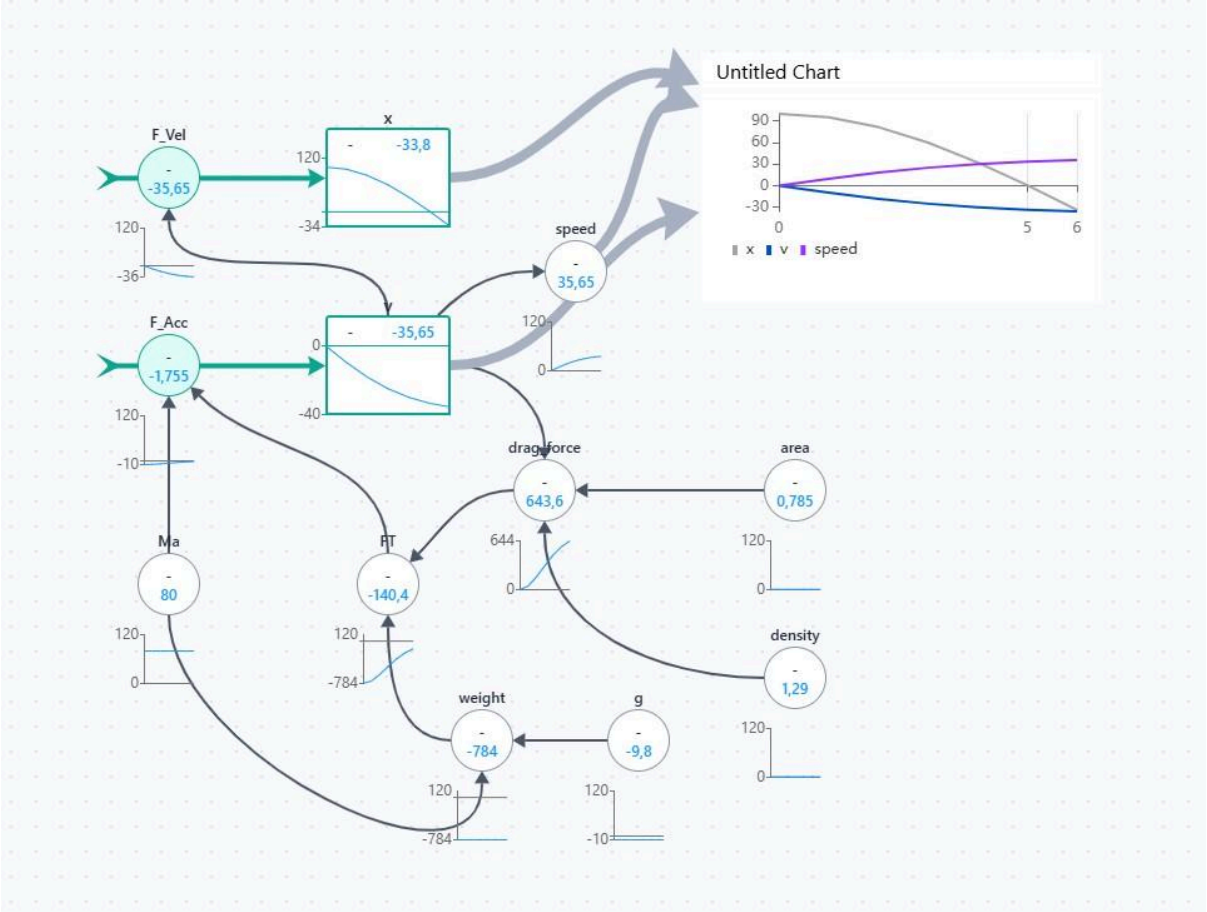
No segundo bloco, “Sistema Solar & Sistemas de Partículas”, implementou-se uma simulação de N corpos celestes interagindo gravitacionalmente de acordo com a Lei da Gravitação Universal de Newton. Foram explorados cenários com um e múltiplos planetas, bem como a utilização de sistemas de partículas para representação visual de fenómenos astronómicos.

O terceiro bloco, “Agentes Autónomos – Comportamentos Individuais”, focou-se na implementação de comportamentos básicos para um Boid, como seek, arrive, patrol e wander, bem como na comutação entre comportamentos consoante estímulos do ambiente.

Por fim, no quarto bloco, “Agentes Autónomos – Comportamentos de Grupo (Flocking)”, foram modelados comportamentos coletivos, incluindo flocking com predador, Boids explosivos, liderança e ferramentas de debugging, demonstrando a emergência de padrões complexos a partir de regras simples.

Cada secção inclui uma discussão crítica dos resultados, realçando as opções de implementação e o comportamento observado nas simulações.

B. Queda Livre Com Atrito (Sheetless)



Neste exercício analisamos a queda livre considerando o peso e uma força de arrasto proporcional ao quadrado da velocidade. Como essa força depende da própria velocidade, o sistema torna-se não linear e não possui solução analítica simples, sendo necessário o uso do Sheetless para simulação.

O diagrama de Stocks & Flows mostra um feedback negativo: quanto maior a velocidade, maior o arrasto; e quanto maior o arrasto, menor a prosperidade. Isso leva naturalmente ao aparecimento de uma velocidade terminal, visível nos gráficos da simulação.

Para um corpo de 70 kg com área aproximada de 0,5–0,8 m², uma simulação mostra que:

- o tempo de queda aumenta em relação à queda livre ideal;
- A velocidade de impacto é muito menor devido ao arrasto crescente.

Quando simulamos um paraquedas (10 kg, diâmetro 6 m), o arrasto aumenta drasticamente, ganhando velocidade quase imediatamente após a abertura e tornando o impacto seguro. Já no caso de queda na água (densidade muito maior), a desaceleração no impacto é extremamente brusca, mostrando que a água, nessas velocidades, se comporta quase como um sólido.

C. Sistema Solar & Sistemas de Partículas

Estrutura Principal:

Classe SolarSystemApp: Controla a simulação completa do sistema solar
Classe Body: Representa corpos celestes com massa, posição, velocidade e aceleração
Sistema de Coordenadas: Utiliza SubPlot para gestão de viewports e transformação de coordenadas.

Implementação da Física Gravitacional:

```
public PVector attraction(Body other) {
   PVector r = PVector.sub(other.pos, this.pos);
   float d = r.mag();
   float force = (G * mass * other.mass) / (d * d);
   r.normalize();
   return r.mult(force);
}
```

Características do Sistema Implementado:

- 8 Planetas + Sol: Parâmetros realistas (massa, distância, velocidade orbital);
- Cometas Dinâmicos: Gerados proceduralmente com caudas de partículas;
- Campo Estelar: Fundo com 2500 estrelas posicionadas aleatoriamente;
- Sistema de Zoom: Navegação entre diferentes escalas do sistema solar;

Controlo de Visualização:

```
private void zoomToPlanetOrbit(PApplet p, int planetIndex) {
   float orbit = planetOrbits[planetIndex];
   double range = orbit * 1.1f;
   window[0] = -range; window[1] = range;
   window[2] = -range; window[3] = range;
   plt = new SubPlot(window, viewport, p.width, p.height);
}
```

Resultados e Análise

Comportamentos Observados:

- Órbitas Estáveis: Planetas mantêm trajetórias elípticas consistentes;
- Influência Gravitacional: Cometas exibem trajetórias hiperbólicas realistas;
- Escala Multi-nível: Transição suave entre visão planetária e sistema completo;

Validação Científica:

- Períodos orbitais consistentes com a 3ª Lei de Kepler;
- Velocidades orbitais decrescentes com a distância ao Sol;
- Comportamento de cometas realisticamente modelado;

D. AA - Comportamentos Individuais

Arquitetura do Sistema

Hierarquia de Classes:

text
IBehaviour (Interface)
  ↳ Behaviour (Classe Abstrata)
    ↳ Seek, Flee, Arrive, Wander, Pursuit, Evade, Separate, Align, Cohesion

Sistema de DNA Parametrizável:

```
public class DNA {
    public float maxSpeed;          // Velocidade máxima
    public float maxForce;          // Força máxima de steering
    public float visionDistance;    // Alcance visual
    public float visionAngle;       // Ângulo de visão
    public float radiusArrive;      // Raio para comportamento arrive
    public float deltaTWander;      // Parâmetro de wander
    // ... outros parâmetros
}
```

Comportamentos Implementados

Seek & Flee:

```
public PVector getDesiredVelocity(Boid me) {
    PVector vd = PVector.sub(me.eye.target.getPos(), me.getPos());
    return vd; // Seek
    // return vd.mult(-1); // Flee
}
```

Wander (Versão Reynolds):

```
public PVector getDesiredVelocity(Boid me) {
    PVector center = me.getPos().copy();
    center.add(me.getVel().copy().mult(me.dna.deltaTWander));
    PVector target = new PVector(me.dna.radiusWander * cos(me.phiWander),
                                 me.dna.radiusWander * sin(me.phiWander));
    me.phiWander += random(-me.dna.deltaPhiWander, me.dna.deltaPhiWander);
    return PVector.sub(target, me.getPos());
}
```

Sistema de Percepção (Eye):

Campo de Visão: Ângulo e distância configuráveis;

FarSight: Detecção de corpos no campo de visão;

NearSight: Deteção para comportamentos de colisão;

Target Selection: Alvo primário para seek/flee;

Mecânica de Steering

Cálculo de Velocidade Desejada:

```
public void applyBehaviours(float dt) {
    PVector vd = new PVector();
    for (Behaviour behaviour : behaviours) {
        PVector vdd = behaviour.getDesiredVelocity(this);
        vdd.mult(behaviour.getWeight() / sumWeights);
        vd.add(vdd);
    }
    move(dt, vd);
}
```

Aplicação de Forças:

```
private void move(float dt, PVector vd) {
    vd.normalize().mult(dna.maxSpeed);
    PVector fs = PVector.sub(vd, vel);
    applyForce(fs.limit(dna.maxForce));
    super.move(dt);
    // Wrap-around nas bordas
}
```

Resultados e Análise

Comportamentos Observados:

- Seek/Flee: Movimento direto e responsivo;
- Arrive: Paragens suaves sem oscilações;
- Wander: Padrões orgânicos e não repetitivos;
- Pursuit/Evade: Interceptação inteligente de alvos móveis;

Parâmetros Críticos:

- Peso de Comportamentos: Balanceamento entre objetivos concorrentes;
- MaxForce: Controla a "agressividade" das manobras;
- Vision Parameters: Define a consciência ambiental do agente;

Aplicações Demonstradas:

- Agentes reativos a estímulos do ambiente;
- Comutação dinâmica entre comportamentos;
- Sistemas predador-presa básicos;

E. AA - Flocking

Implementação do Modelo de Reynolds

Três Comportamentos Fundamentais (SAC):

Separate (Separação):

```
public PVector getDesiredVelocity(Boid me) {
    PVector vd = new PVector();
    for (Body b : me.eye.getNearSight()) {
        PVector r = PVector.sub(me.getPos(), b.getPos());
        float d = r.mag();
        r.div(d * d); // Força inversamente proporcional ao quadrado da distância
        vd.add(r);
    }
    return vd;
}
```

Align (Alinhamento):

- Calcula velocidade média dos vizinhos;
- Suaviza mudanças de direção do grupo;
- Mantém coerência de movimento;

Cohesion (Coesão):

- Move-se em direção ao centro de massa dos vizinhos;
- Mantém o grupo unido;
- Balanceia com Separate para evitar colisões;

Sistema de Flocking Avançado

Arquitetura do Flock:

```
public class Flock {
    private List<Boid> boids;

    public Flock(int nboids, float mass, float radius, int color,
                 float[] sacWeights, PApplet p, SubPlot plt) {
        // Inicialização com pesos configuráveis para Separate, Align, Cohesion
    }
}
```

Configuração de Pesos Típica:

```
float[] sacWeights = {1.5f, 1.0f, 1.0f}; // Separate, Align, Cohesion
```

Funcionalidades Avançadas

Liderança e Controlo Humano:

- Um Boid controlado por WASD influencia todo o flock;
- Transição suave entre modos autónomo e dirigido;
- Manutenção de coesão mesmo com liderança externa;

Sistema Predador-Presa:

```
// Predador usa Pursuit para caçar
pursuiter.addBehaviour(new Pursuit(1f));
// Presas usam Evade para fugir
prey.addBehaviour(new Evade(1f));
```

Debugging e Visualização:

- Vetores de velocidade visíveis;
- Campos de visão representados graficamente;
- Destaque colorido para Boids no campo de visão;
- Monitorização individual de agentes específicos;

Boids Explosivos:

- Remoção dinâmica baseada em eventos (timer, clique, zona);
- Efeitos visuais com sistemas de partículas;
- Reorganização automática do flock após explosões;

Conclusões

B. Queda Livre Com Atrito

A simulação da queda livre com força de arrasto permitiu observar que o comportamento real de um corpo em queda é profundamente inspirado pela resistência do meio. O feedback negativo criado entre velocidade e força de atrito impede que a velocidade cresça indefinidamente, originando uma velocidade terminal.

C. Sistema Solar & Sistemas de Partículas

A implementação do sistema solar demonstrou sucesso notável na modelação de sistemas gravitacionais complexos. O modelo N-corpos, baseado na Lei da Gravitação Universal, produziu órbitas estáveis e visualmente convincentes para todos os 8 planetas implementados. A integração numérica mostrou-se suficientemente precisa para manter a estabilidade orbital em escalas de tempo prolongadas, mesmo com as enormes diferenças de escala entre as distâncias planetárias.

D. Agentes Autónomos - Comportamentos Individuais

A arquitetura modular de comportamentos revelou-se extremamente flexível e extensível. O sistema baseado em pesos permitiu combinações sofisticadas de comportamentos, com o Arrive destacando-se pela sua capacidade de eliminar oscilações indesejadas presentes no Seek básico. O comportamento Wander implementado segundo o modelo de Reynolds produziu movimentos orgânicos e naturais, superando abordagens mais simples baseadas em waypoints aleatórios.

O sistema de DNA parametrizável provou ser uma solução elegante para personalizar o comportamento dos agentes, enquanto a classe Eye forneceu uma base sólida para sistemas de percepção realistas. A comutação dinâmica entre comportamentos funcionou de forma fluida e responsiva, abrindo caminho para máquinas de estado mais complexas.

E. Agentes Autónomos - Flocking

O modelo de flocking de Reynolds confirmou a sua eficácia na geração de comportamentos coletivos complexos a partir de regras simples. A tríade Separate-Align-Cohesion produziu padrões emergentes convincentes que mimetizam comportamentos naturais observados em cardumes e bandos de aves.

A implementação de liderança controlada pelo utilizador demonstrou a robustez do sistema, mantendo a coesão do grupo mesmo sob influência externa. O sistema predador-presa evidenciou a versatilidade da arquitetura, permitindo interações complexas entre diferentes tipos de agentes.