
Simulador de um Ambiente Virtual Distribuído Multiusuário para Batalhas de Tanques 3D com Inteligência Baseada em Agentes BDI

Acadêmico: Germano Fronza
gfronza@inf.furb.br

Orientador: Dalton Solano dos Reis
dalton@furb.br

julho de 2008

Roteiro da Apresentação

1. **Introdução:** Contextualização e Objetivos
2. **Fundamentação Teórica**
 - (a) AVD, SMA, JME
 - (b) Trabalhos Correlatos
3. **Desenvolvimento do Simulador**
 - (a) Principais Requisitos
 - (b) Especificação
 - (c) Implementação
 - (d) Resultados e Discussão
4. **Conclusão e Extensões**

Introdução

Contextualização

- Responsabilidade dos simuladores.
- Ambientes 3D e o aumento da imersão nos simuladores.
- Usuários remotos → o surgimento dos AVDs.
- IA contribuindo para o aumento da realidade nos simuladores.
- Simulador provendo um ambiente atrativo para desenvolvimento e testes de agentes BDI.

Objetivos do Trabalho

- Desenvolver um simulador 3D de um AVD para atuar em uma rede local.
- Objetivos específicos:
 - ★ Definir um cenário a ser utilizado no simulador (ambiente, objetos, percepções e comportamentos);
 - ★ Utilizar a engine gráfica JMonkey Engine para construção do ambiente gráfico do AVD;
 - ★ Permitir que os tanques sejam controlados por avatares ou tenham um comportamento autônomo.

Fundamentação Teórica

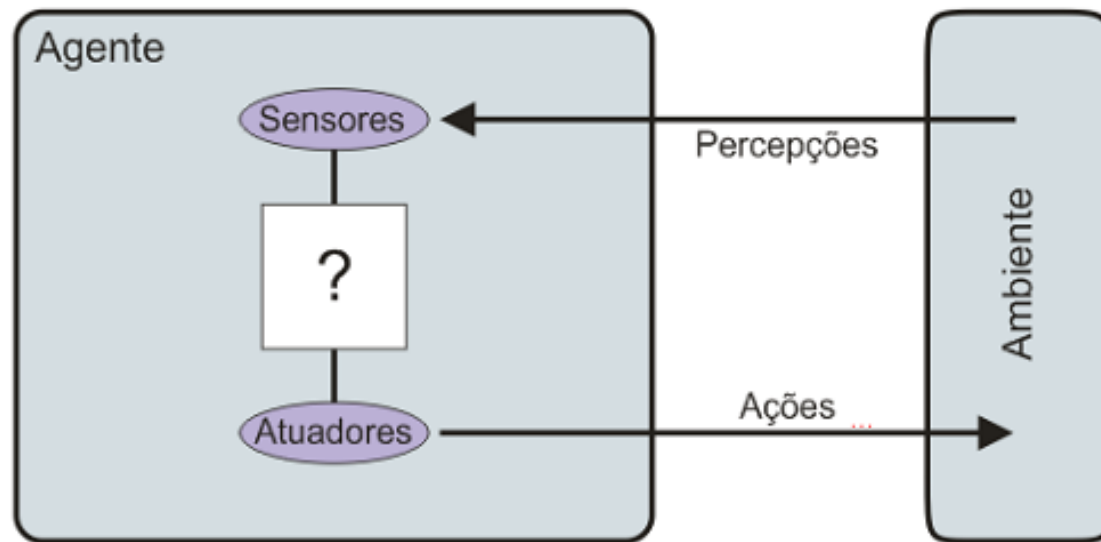
Ambiente Virtual Distribuído

- Primeiros AVDs desenvolvidos:
 - ★ Simulator Network (SIMNET);
 - ★ Naval Postgraduate School Network (NPSNET).
- Resultados obtidos a partir desses projetos.
- Modelos gerais de comunicação:
 - ★ Centralizado;
 - ★ Distribuído.

-
- Java Game Networking (JGN)
 - ★ *Open-source*.
 - ★ Maior **abstração** na implementação do módulo de comunicação.
 - ★ Mensagens definidas como **classes Java**.
 - ★ Faz uso dos protocolos TCP e UDP.
 - ★ Suporta UDP *broadcasting*.
 - ★ Falta de documentação → ponto negativo.

Sistemas MultiAgentes (SMA)

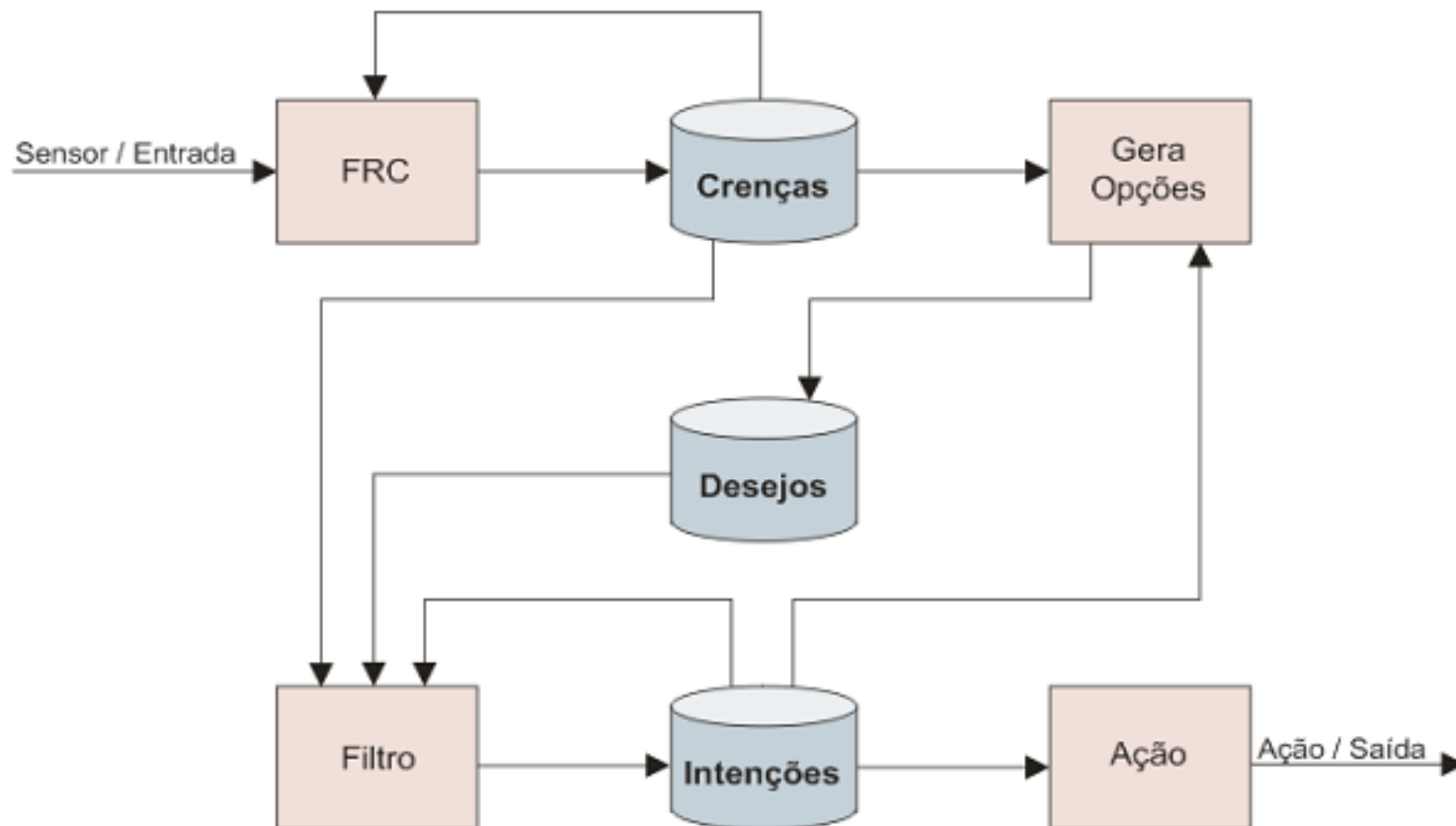
” Os **sistemas multiagentes** são sistemas compostos por multiplos elementos computacionais interativos, conhecidos como **agentes**.” (WOOLDRIDGE, 2002)



- Autonomia, Pró-atividade, Reatividade e Habilidade social.

Arquitetura BDI

- Atitudes mentais.
- *Belief, Desire and Intentions* (BDI).



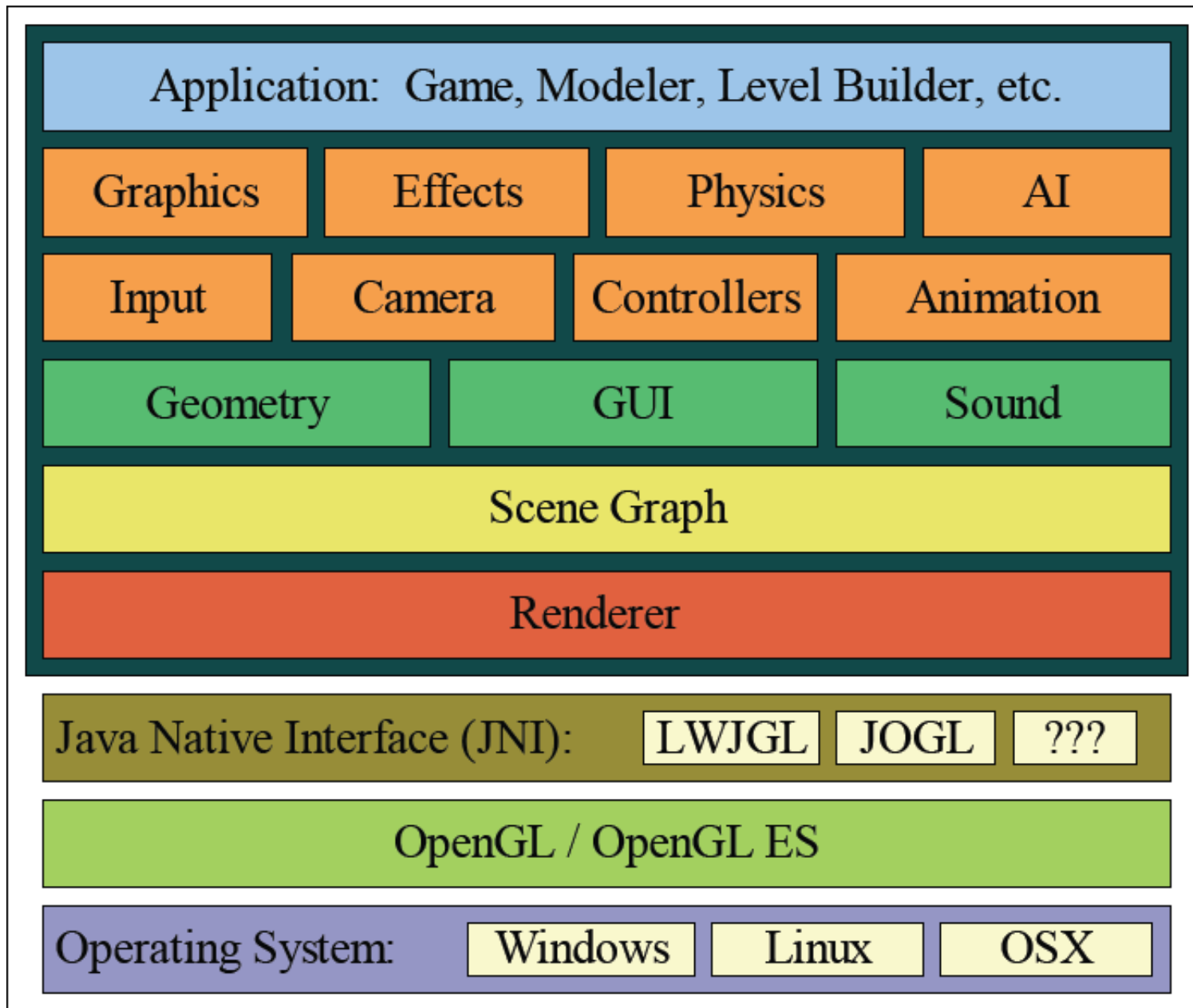
Linguagem AgentSpeak(L) e o interpretador Jason

- Sintaxe abstrata.
- Linguagem de lógica, semelhante ao Prolog.
- Código de um agente AgentSpeak(L) é composto por um conjunto de crenças iniciais, metas e planos.

```
1
2  /** Crenças iniciais */
3  likes(orquestraSinfonica).
4  likes(robertoCarlos).
5  likes(coralDeNatal).
6
7  /** Planos */
8  +concert(A, V) : likes(A)
9      ← !book_tickets(A, V).
10
11 +!book_tickets(A, V) : ¬busy(phone)
12     ← call(V);
13     ...;
14     !choose_seats(A, V).
15
```

JMonkey Engine (JME)

- *Open-source.*
- Escrita totalmente em Java delegando tarefas de renderização à biblioteca nativa OpenGL.
- Baseada em Grafo de Cena.
- Sem acoplamento com uma implementação específica de OpenGL para Java (através de JNI)
- Física usando o *framework* JME Physics.
- Conceito de *Game States*.



Trabalhos Correlatos

- Robocode.
- Sistema multiagentes utilizando a linguagem AgentSpeak(L) para criar estratégias de armadilha e cooperação em um jogo tipo PacMan.
- Protótipo de um ambiente virtual distribuído multiusuário.

Desenvolvimento do Simulador

Principais Requisitos

Funcionais:

- Permitir a escolha do mapa da batalha.
- Possibilitar a entrada de novos modelos de mapas.
- Disponibilizar os dois modelos de tanques de guerra.
- Fornecer um conjunto de ações, percepções e crenças para os agentes.
- Permitir que os tanques sejam controlados por usuários ou tenham um comportamento autônomo.
- Prover um AVD com arquitetura híbrida para LAN.

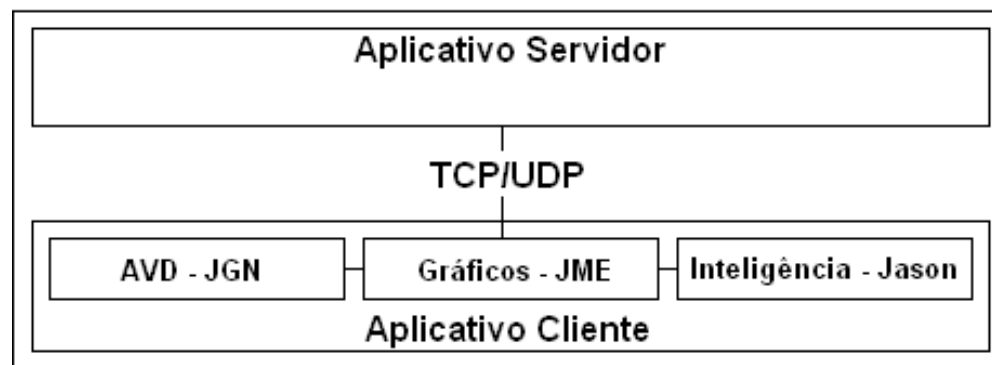
Não-Funcionais:

- Tratar no máximo seis tanques de guerra em cada time.
- Garantir que nenhum usuário se conecte ao servidor após o início da batalha.
- JavaSE 6.0, JMonkey Engine, JME Physics, JGN e Jason.

Especificação

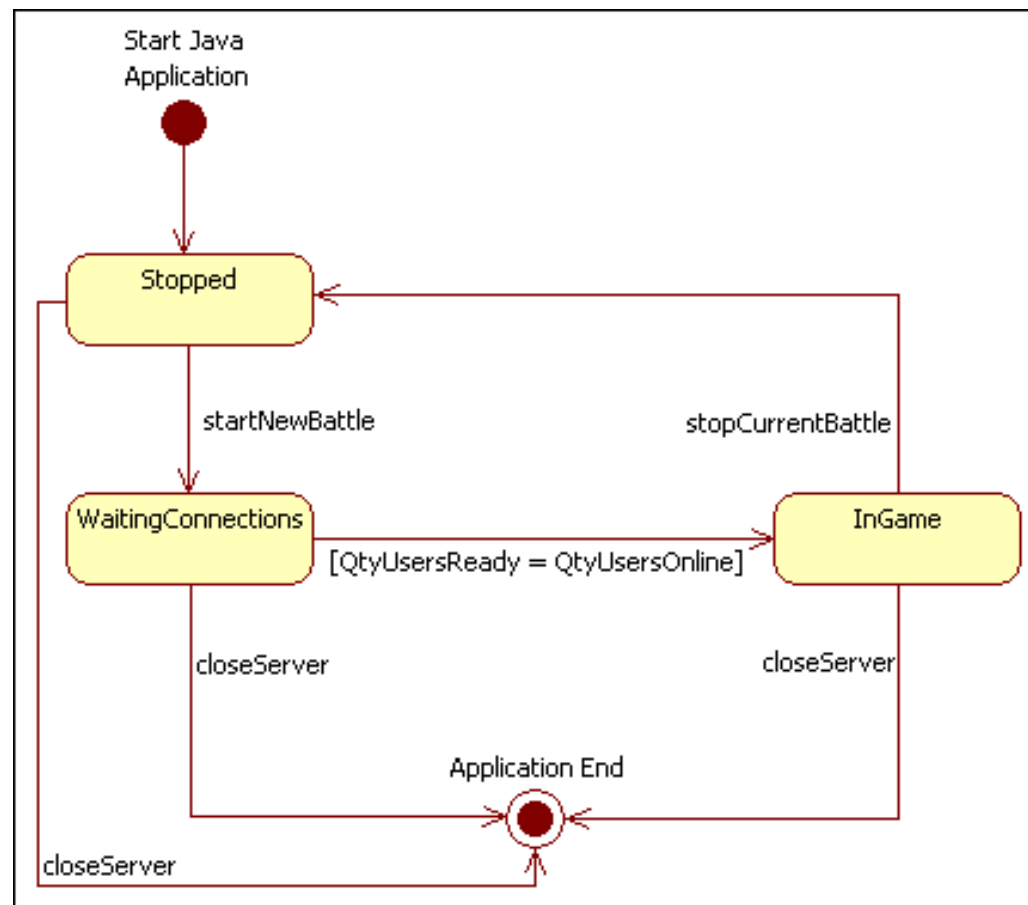
Arquitetura Híbrida: Características de arquitetura centralizada combinadas com características de arquitetura distribuída.

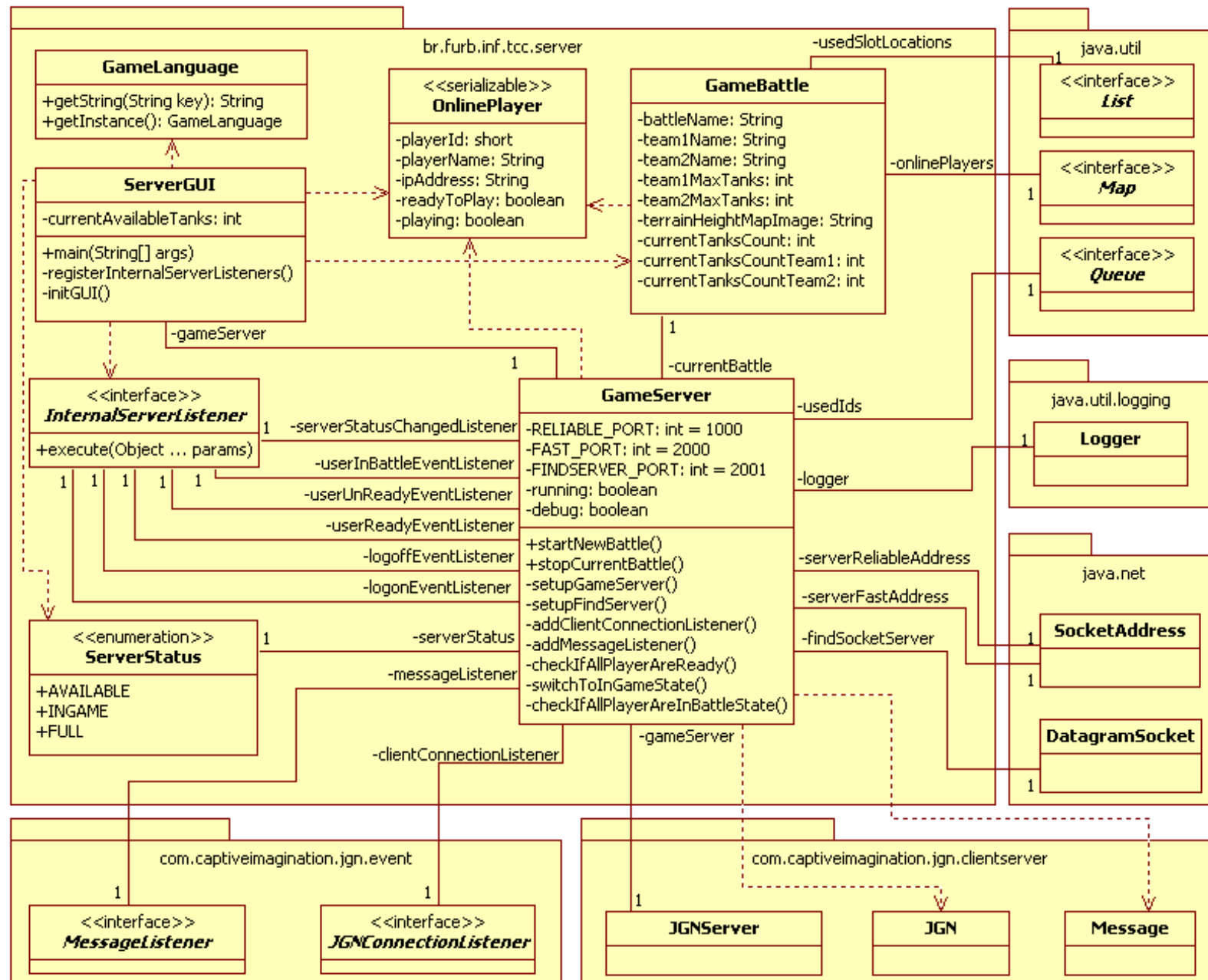
- O Aplicativo Servidor;
- O Aplicativo Cliente.



O Aplicativo Servidor

Estados possíveis:



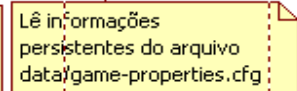


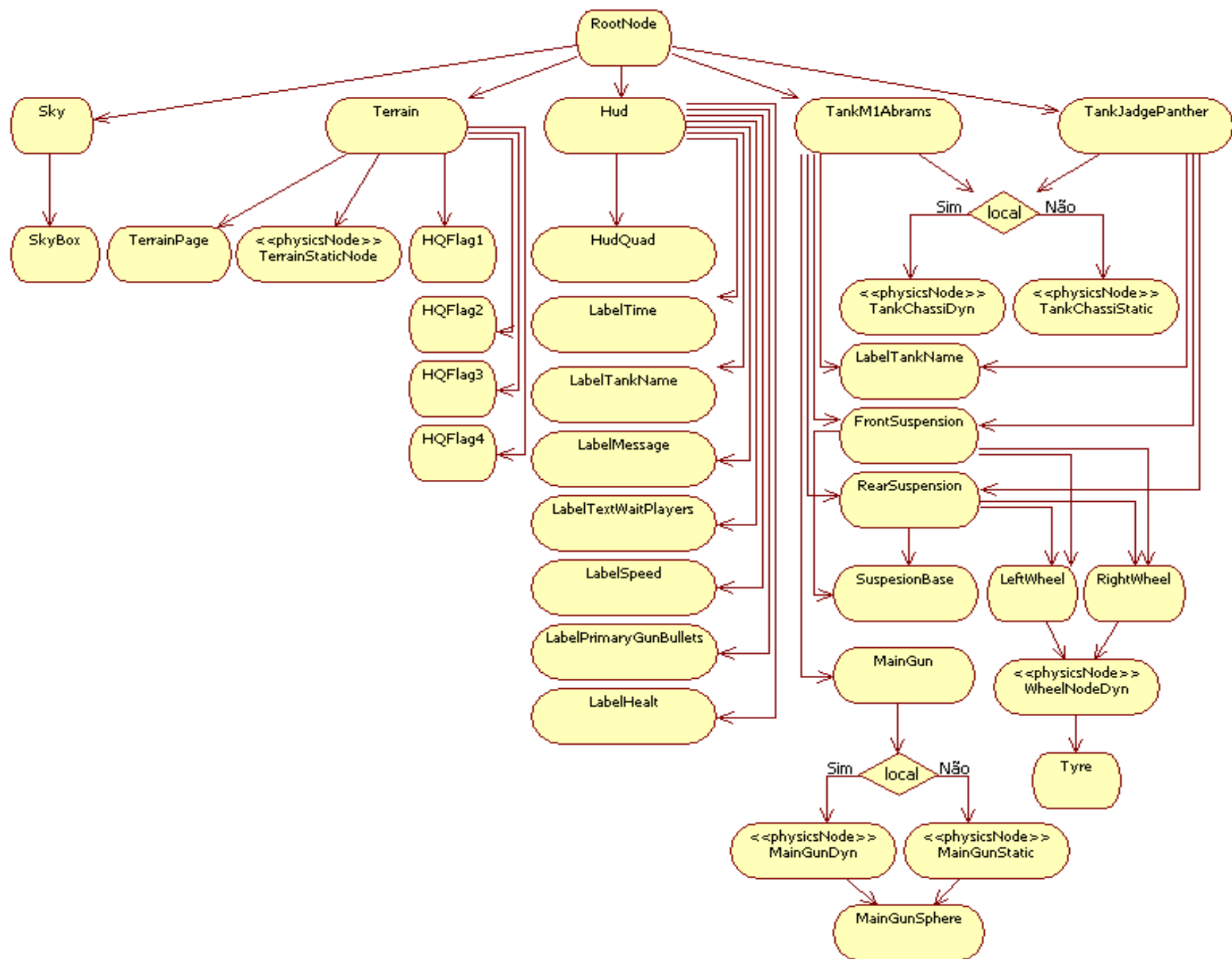
O Aplicativo Cliente

- Representação gráfica do estado do AVD.
 - ★ *Middleware* Java Game Networking (JGN).
- Interação com o aplicativo servidor.
 - ★ *Engine* gráfica JMonkey Engine (JME).
- Integração com o interpretador Jason.
 - ★ Interpretador Jason.

Representação gráfica do estado do AVD

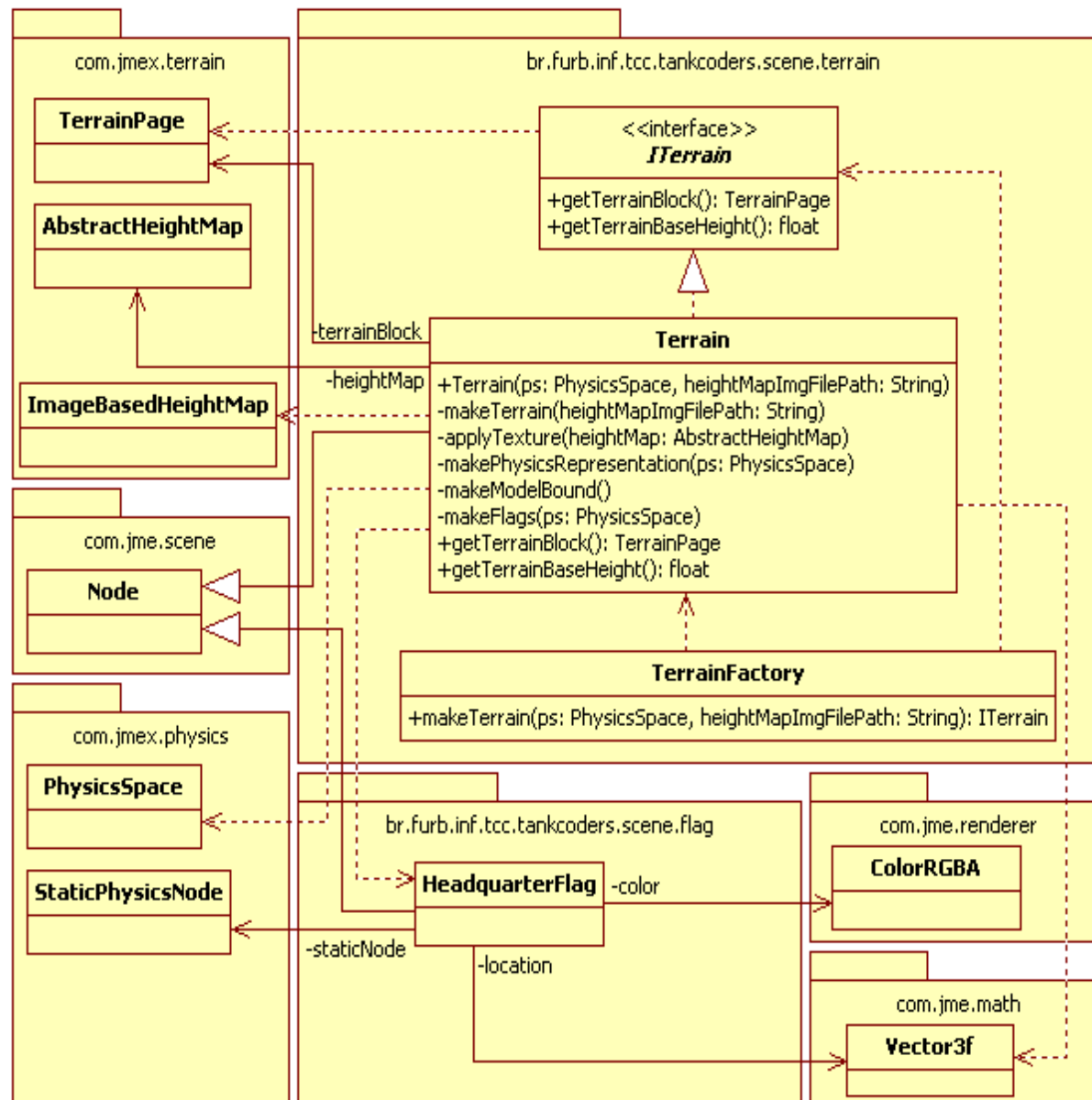
- *Game States* do simulador:
 - ★ GameMenuState;
 - ★ InGameState.
- Somente um *game state* ativo no GameStateManager.





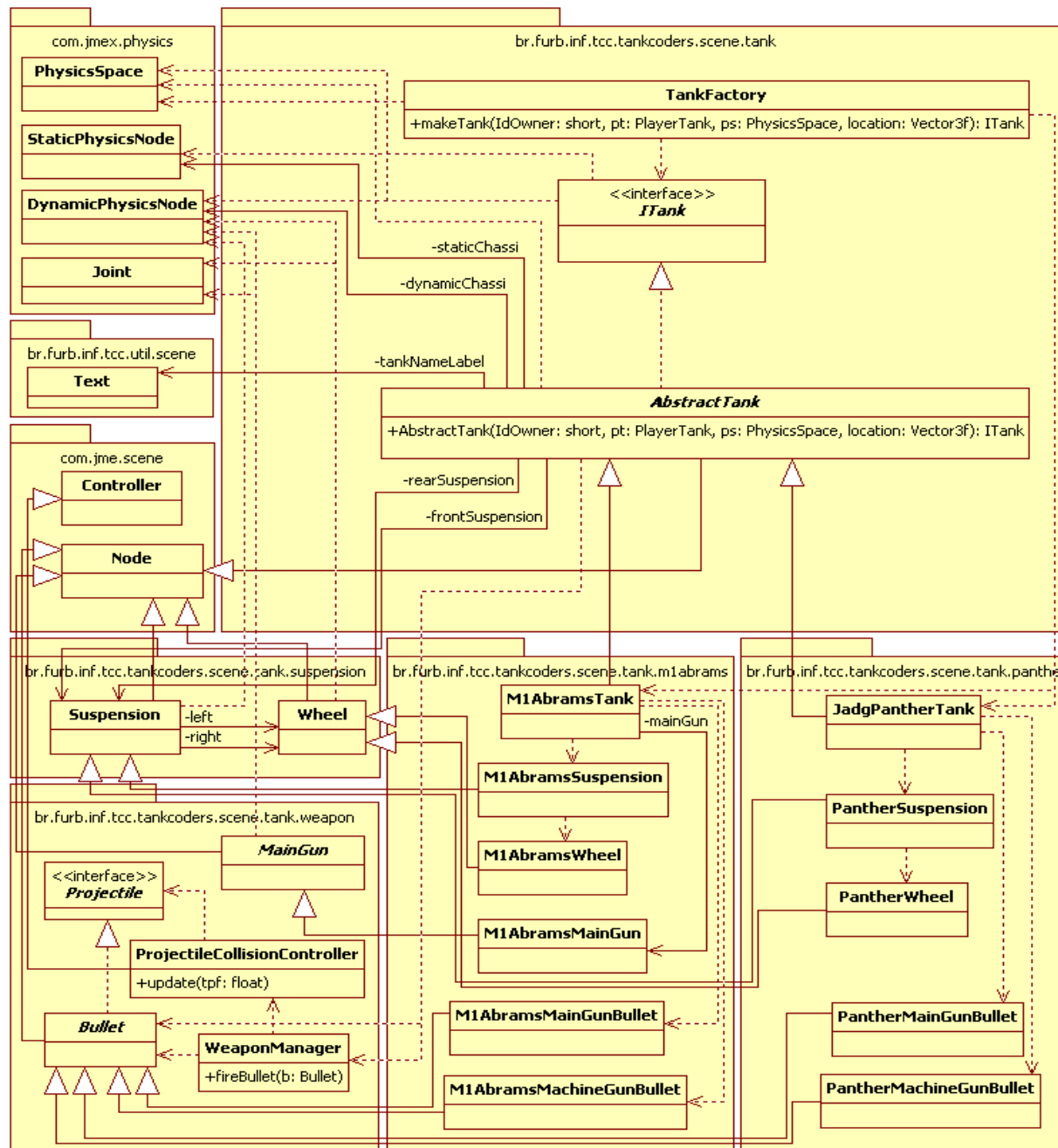
Terreno

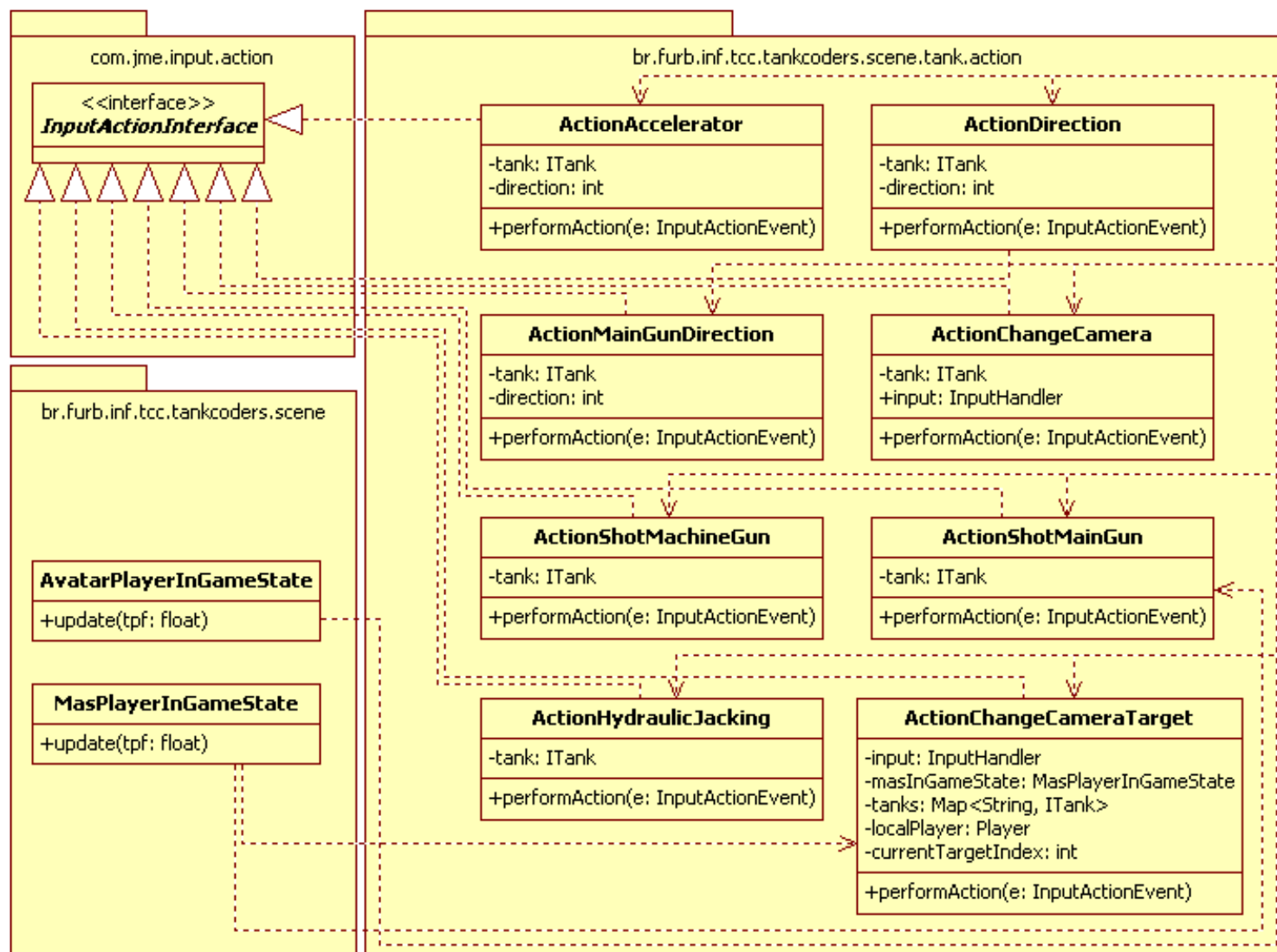
- Criação através da classe `TerrainFactory`
- Composto por uma malha de polígonos gerada a partir de um **mapa de alturas**
- Mistura de texturas para representar um terreno de areia
- Possui um **Quartel General (QG)** para cada time
- Delimita o QG com bandeiras (`HeadquarterFlag`)



Tanques

- Criação através da classe TankFactory
- Pode ser integrado a um agente
- Pode ser controlado por um usuário remoto
- Pode ser controlado via teclado e mouse (*Actions*)
- Composição:
 - ★ Suspensões
 - * Rodas
 - ★ Arma principal (Móvel ou Estática)
 - ★ Metralhadora (Fixa na base da arma principal)
 - ★ Chassi

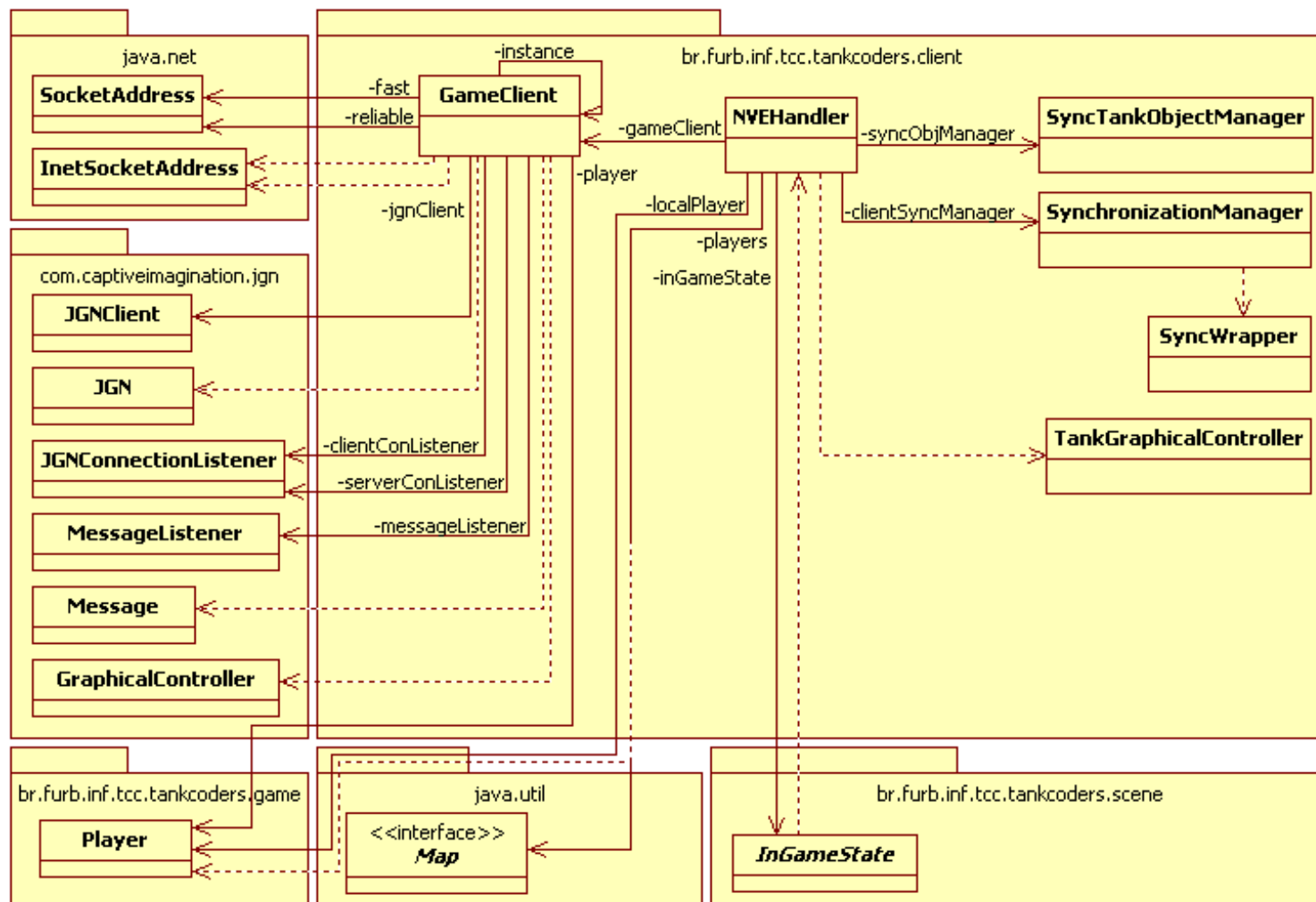




Interação com o aplicativo servidor

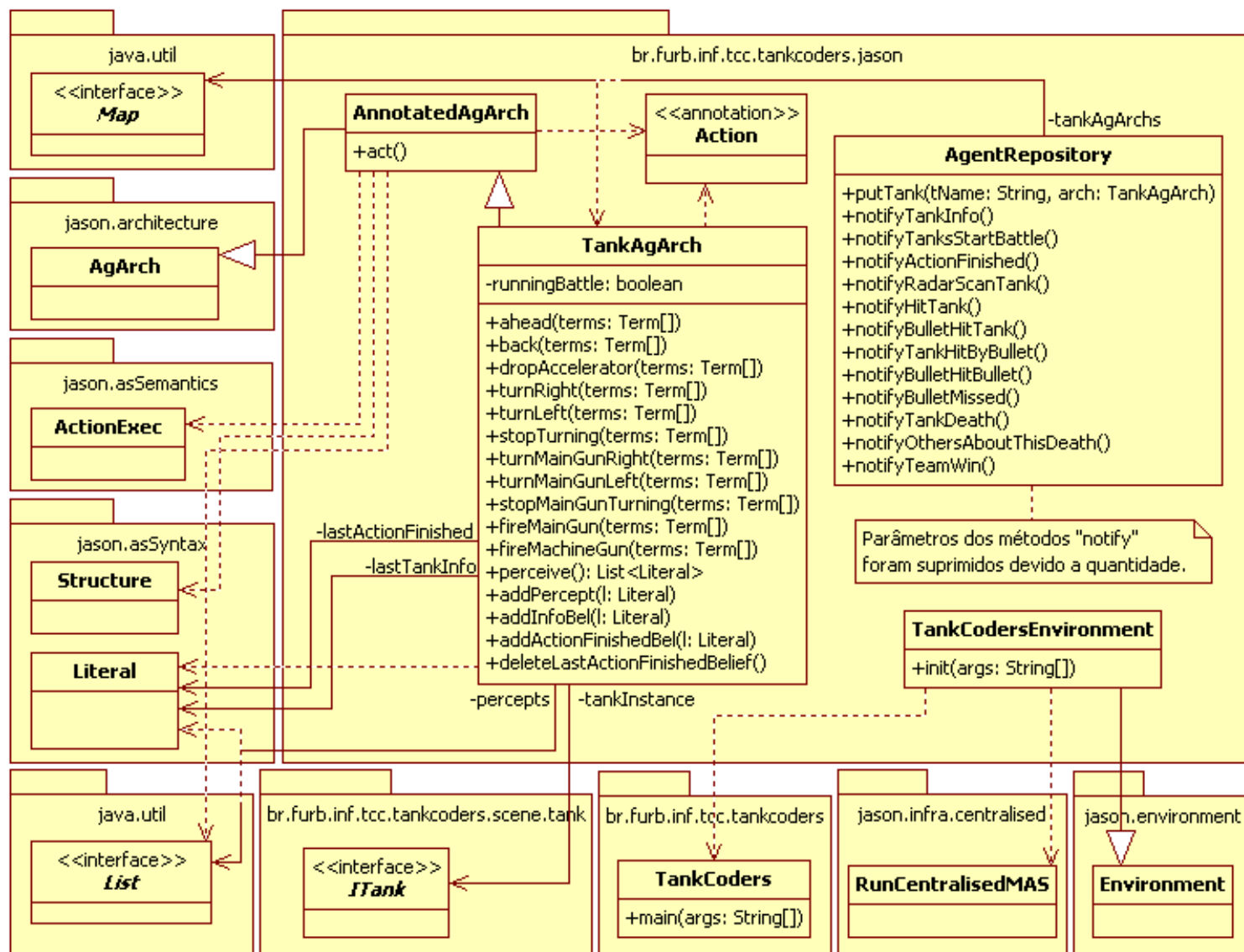
- Utilizando o *middleware* JGN
- Mensagens representadas por classes
- Todas as mensagens, exceto as de sincronização, derivam de `RealtimeMessage` (protocolo UDP)
- Mensagens classificadas de acordo com o *game state* em que ocorrem
- Uso de UDP *broadcasting* para mensagens comuns a todos os jogadores

CLASSE	CLASSIFICAÇÃO	SENTIDO
<i>AbstractGameMessage</i>	Ambos	Ambos
AllPlayersAreInGameState	Em jogo	Servidor -> Cliente
AnotherPlayerChangeModelResponse	Menu	Servidor -> Cliente
AnotherUserLogonResponse	Menu	Servidor -> Cliente
ChangeModel	Menu	Cliente -> Servidor
ChangeTeam	Menu	Cliente -> Servidor
InvalidChangeTeamResponse	Menu	Servidor -> Cliente
PlayerChangeTeamResponse	Menu	Servidor -> Cliente
PlayerInBattle	Em jogo	Cliente -> Servidor
PlayerLeftGame	Em jogo	Servidor -> Cliente
StartBattle	Menu	Servidor -> Cliente
Synchronize3DMessage	Em jogo	Cliente -> <i>broadcast</i>
SynchronizeArticulatedObject3DMessage	Em jogo	Cliente -> <i>broadcast</i>
SynchronizeCreateTankMessage	Em jogo	Cliente -> <i>broadcast</i>
TankActionShotMachineGun	Em jogo	Cliente -> <i>broadcast</i>
TankActionShotMainGun	Em jogo	Cliente -> <i>broadcast</i>
TankBulletHit	Em jogo	Cliente -> <i>broadcast</i>
TankDead	Em jogo	Cliente -> <i>broadcast</i>
TeamLeftGame	Em jogo	Cliente -> <i>broadcast</i>
UserLogoff	Menu	Cliente -> Servidor
UserLogon	Menu	Cliente -> Servidor
UserLogonFailedResponse	Menu	Servidor -> Cliente
UserLogonResponse	Menu	Servidor -> Cliente
UserReady	Menu	Cliente -> Servidor
UserUnready	Menu	Cliente -> Servidor



Integração com o interpretador Jason

- Jason 1.1.1
- Componentes do Jason personalizados:
 - ★ AgArch
 - ★ Environment



Implementação

Técnicas e Ferramentas Utilizadas

- IDE Eclipse 3.3 → Java.
- Blender 2.43 → Modelos dos Tanques 3D.
- Adobe Photoshop CS → Texturas do terreno.
- Assembla Space → Gerenciamento do projeto.
- Apache Ant → Tarefas repetitivas.
- Padrões de Projeto: *Singleton*, *Command*, *Factory Method* e *Transfer Object*.

Operacionalidade da Implementação

Passos:

- Abrir aplicativo servidor, configurar e iniciar a batalha.
- Abrir um aplicativo cliente:
 - ★ Jogador **Avatar** ou **MAS**.
- Conectar-se ao servidor criado.
- Preparar os tanques.
- Definir-se como **Ready**.

Servidor notifica inicio da batalha → clientes carregam cenário (estado AVD local)

Jogador Avatar

- Teclas:
 - ★ Movimentar Tanque: **W, S, A e D.**
 - ★ Atirar Arma Principal: **Barra Espaço.**
 - ★ Atirar Metralhadora: **Ctrl.**
 - ★ Movimentar Arma principal: **Direita e Esquerda.**
 - ★ Macaco hidráulico: **J**
 - ★ Modo Wireframed: **O**

Jogador MAS

- Teclas:
 - ★ Trocar foco da camera: **TAB**
 - ★ Modo Wireframed: **O**

Resultados e Discussão

- *Middleware* JGN → performance e abstração.
- *Engine* JME → Grafo de cena = performance, otimização e abstração.
- Cenário 3D com bom nível de imersão:
 - ★ Gravidade, massa, colisão, suspensões e disparo de projéteis.
- *Chase Camera* melhor visão para o jogador Avatar.
- Jason 1.1.1 → desempenho nas ações.
- Abstração para o desenvolvedor AgentSpeak(L).
- **Orientação a Objetos** em aplicações gráficas.

Conclusão

Conclusão

- Simulador **TankCoders** e o estudo de POA.
- *Middlewares* e *frameworks* utilizados:
 - ★ JGN: pouca documentação e exemplos;
 - ★ Jason e plug-in Eclipse: Hübner, Bordini e Fronza;
 - ★ AgentSpeak: agentes cognitivos de forma declarativa e elegante;
 - ★ JME: Abstração e performance.

Extensões

- Desenvolver modelos 3D mais leves.
- Audio no ambiente.
- Efeitos de explosão.
- Novos modelos de terreno.
- Ações internas genéricas:
 - ★ *Pathfinding* por exemplo.
- Notificação de obstáculos do terreno aos agentes.
- Rodar sobre redes WAN (Internet).

Estatísticas

- Total de arquivos .java: 135
- Total de linhas: 16478
 - ★ Linhas de código: 81,19 %
 - ★ Comentários: 2,18 %

"O único lugar onde o sucesso vem antes do
trabalho é no dicionário."
Albert Einstein

Suspensão

