JaRTS: Java RTS Simulator

Vicente V. Filho José C. M. Júnior Renan T. Weber

Geber L. Ramalho Patrícia C. A. R. Tedesco

Universidade Federal de Pernambuco, Centro de Informática, Brasil

Resumo

Simuladores, tais como os da Robocode [Robocode] têm auxiliado de forma significativa os avanços da Inteligência Artificial (IA). Em jogos, particularmente em jogos de estratégia em tempo real (RTS), com alta aplicação potencial de técnicas de IA, o mesmo avanço poderia ser facilitado com ambientes de simulação. Atualmente já existem simuladores para encapsular a complexidade da simulação do jogo e permitir aos pesquisadores se concentrarem somente na questão da IA. No entanto essas ferramentas pecam em diversos aspectos tais como documentação, facilidade de uso e foco. O objetivo desse trabalho é propor o desenvolvimento de um novo simulador de RTS, denominado JaRTS, mais simples de usar e com foco na pesquisa de Inteligência Artificial.

Palavras-chave: entretenimento digital, inteligência artificial, jogo de estratégia, ambiente de simulação

Contato dos autores:
{vvf, jcmj, rtw, glr, pcart}@cin.ufpe.br

1. Introdução

O gênero RTS foca no planejamento cuidadoso e na habilidade de gerenciar recursos de maneira a alcançar a vitória. Os jogos desse gênero são jogos nos quais a estratégia deve ser montada e executada a cada instante, em tempo real. Exemplos de jogos desse gênero são: Warcraft, Starcraft, Command and Conquer, Total Annihilation e Age of Empires.

Os jogos RTS são excelentes aplicações em termos de IA por exigirem coordenação multiagente, tomadas de decisão multi-critério e combinação de decisões estratégicas e planejamento com decisões táticas e reatividade [Schwab 2004].

As pesquisas da Inteligência Artificial nesse gênero, assim como nos demais, encontram uma barreira: a ausência de simuladores que acarreta na falta de um ambiente para testar técnicas e comparar resultados. Aliado a isso existe o fato de que os jogos RTS comerciais são fechados e não permitem que pesquisadores conectem módulos de IA.

Devido a isso, há atualmente um esforço para a construção de simuladores e ferramentas para validar as pesquisas na área. Dentre as ferramentas livres existentes destacam-se o ORTS [Buro 2005], Stratagus [Stratagus], Boson [Boson] e Glest [Glest]. No entanto, essas ferramentas possuem problemas como documentação incipiente, instabilidade e falta de foco para simulação de IA.

Por esse motivo propomos a construção de um ambiente de simulação com foco na Inteligência Artificial nomeado JaRTS [JaRTS]: Java RTS Simulator. Apesar da linguagem de programação Java não ser o padrão no mercado de jogos, o fato da mesma ser bastante difundida no meio acadêmico da IA oferece várias vantagens dentro de objetivo desse trabalho: prover uma ferramenta para que pesquisadores possam fazer avançar a IA de Jogos RTS. Além disso, ela permite fácil uso em sala de aula, o que serve para motivar trabalhos envolvendo os dois temas: IA e Jogos. Fica fora do nosso escopo, o problema do processo de industrialização que levará a incorporação dos resultados de pesquisa gerados nos jogos comerciais.

Em seguida, no capítulo 2, são apresentadas as dificuldades encontradas na pesquisa da Inteligência Artificial em RTS. No capítulo 3, os simuladores de RTS existentes são apresentados e analisados. No capítulo seguinte, JaRTS é apresentado. Enfim, concluímos com algumas considerações.

2. IA e RTS

A pesquisa de novas técnicas, aplicações e abordagens da Inteligência Artificial na área de entretenimento eletrônico necessitam de um ambiente para simulação do jogo. Por exemplo, o estudo e desenvolvimento de um novo agente para pilotar um carro em um jogo de corrida precisam de um ambiente para simular e visualizar o comportamento do agente na pista, a interação com os demais carros, a performance do carro em comparação com um jogador humano, dentre outras características.

Esse papel de grande importância é desempenhado pelos diversos simuladores existentes. Um dos simuladores mais difundidos atualmente é o Robocode. O Robocode é uma aplicação educativa open source desenvolvida pela IBM que permite a criação de robôs (tanques de batalha) para competir com outros robôs.

A aplicação foi desenvolvida com o propósito de ajudar pessoas a aprender a programar em Java de uma forma prazerosa e acabou se tornando um valioso ambiente de simulação para a pesquisa de técnicas de Inteligência Artificial. O projeto foi muito bem aceito, dando origem a diversas competições.

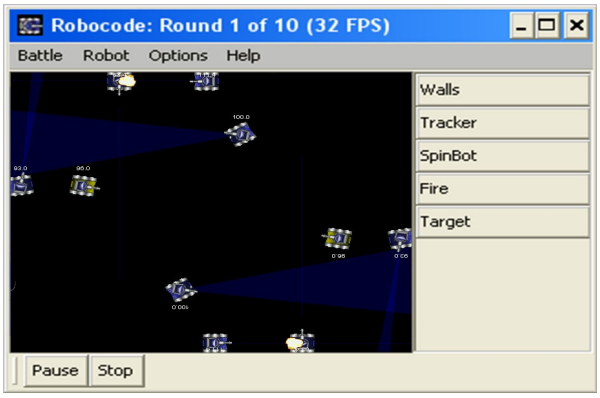

Figura 1: Simulação de batalha no Robocode

O gênero de RTS (Real-Time Strategy), foco deste trabalho, também precisa de um ambiente de simulação dos vários aspectos desse gênero tais como obtenção de recursos, combate e tática / estratégia para o estudo de novas técnicas tais como coordenação multiagente, táticas de combate e formação, tomadas de decisão multi-critério e combinação de decisões estratégicas e planejamento com decisões táticas e reatividade.

A ausência de um ambiente para simulação do gênero de RTS torna a atividade de pesquisa nessa área bastante limitada por diversos fatores:

- A própria simulação - A falta do ambiente de simulação adequado restringe e complica a atividade de pesquisa da IA em jogos. A pesquisa muitas vezes precisa parar para desenvolver um ambiente propício para a simulação, ou seja, desenvolver um jogo RTS.

- Medida de performance - A atividade de pesquisa necessita de resultados palpáveis para a realização de conclusões a cerca de determinadas soluções / estudos propostos.

- Comparação de resultados - Muitas vezes medir a performance não é suficiente e se torna necessário comparar os resultados obtidos com os resultados já existentes (estado da arte) ou com uma determinada meta a ser atingida.

3. Trabalhos Relacionados

Atualmente existem diversos simuladores e motores para desenvolvimento de jogos de estratégia em tempo real.

3.1 ORTS

O Open Real-Time Strategy (ORTS) é um ambiente de programação voltado para o estudo de problemas de Inteligência Artificial em tempo real no domínio de jogos RTS. O usuário pode definir as regras e características do jogo que pretende desenvolver por meio de scripts que descrevem todos os tipos de unidades, estruturas e suas interações. Os scripts são carregados pelo servidor e executados. A outra parte do sistema do ORTS consiste na aplicação cliente que se conecta com o servidor e gera as ações para os objetos do jogo. O servidor envia para o cliente o estado do mundo e recebe as ações de todos os objetos controlados pelo jogador, que são então executados.

Figura 2: Cliente Gráfico do ORTS

Há uma competição baseada no ORTS [Buro 2006], em parceria com a AIIDE (Artificial Intelligence and Interactive Digital Entertainment), na qual existem três categorias a serem disputadas, são elas:

- Game 1 - os agentes trabalhadores devem se coordenar para recolherem o máximo de recursos num prazo de 5 minutos.

- Game 2 - os agentes tanques devem destruir o maior número possível de bases inimigas, num tempo determinado, e ao mesmo tempo evitar que suas bases sejam destruídas.

- Game 3 - além da coleta de recursos e da batalha, o competidor tem que se preocupar com o gerenciamento de recursos, etapa fundamental para que os agentes se auto-gerenciem definindo se é necessário coletar mais recursos e ou se é preciso armazenar. Ainda há um extensivo uso de tomada de decisão para definir quando criar exércitos, atacar ou defender, entre outras ações.

3.2 Stratagus

O Stratagus é um motor de jogos de RTS. Inclui suporte ao desenvolvimento de jogos multiplayer online ou singleplayer offline. O Stratagus é

configurável e pode ser utilizado para a criação de jogos customizáveis.

3.3 Boson

Boson é um jogo RTS similar a jogos como Command & Conquer e StarCraft. A batalha acontece tanto na terra quanto no ar com um grande número de unidades atacando simultaneamente. O combate é afetado pelo vento, obstáculos naturais como árvores e tipos de terrenos variados. O Boson também dispõe de um núcleo de gráficos avançados. Unidades individuais têm seus próprios padrões de movimento, animações, múltiplas armas e pathfinding inteligente. Boson funciona tanto no modo multiplayer quanto no singleplayer contra inimigos controlados por IA.

3.4 Glest

Glest é um jogo RTS 3D livre e disponível para várias plataformas. O jogo pode ser customizado através de arquivos XML, é possível definir parâmetros básicos (life, magia, armadura, etc.). Há também um conjunto de ferramentas disponibilizadas no site do projeto. Sua versão atual inclui suporte a jogos singleplayer jogando contra jogadores controlados por IA.

3.5 Análise dos Simuladores

A análise dos simuladores considerou os seguintes aspectos: documentação, facilidade de implementação e foco no problema. Boson, Stratagus e Glest não são focadas em Inteligência Artificial o que acarreta em um esforço extra para a definição e criação do simulador. Já o ORTS se apresentou no início como o melhor candidato. Entretanto após algum tempo mostrou-se bastante complexo quanto ao uso, gerando diversos problemas desde compilação à implementação do comportamento em si. Além do problema da falta de documentação. E por fim sua instabilidade, pois a cada semana diversas alterações eram feitas pelos seus desenvolvedores, algumas delas inserindo erros de compilação no código disponibilizado.

Após a análise detalhada das alternativas supracitadas decidiu-se criar um novo simulador mais simples de ser utilizado, mais acessível e voltado para a área de Inteligência Artificial.

4. Solução

O JaRTS é um simulador de jogos RTS no qual o usuário pode desenvolver seus agentes se preocupando unicamente com o seu comportamento, uma vez que a Engine se encarrega de fazer a simulação do mundo.

Seu funcionamento se assemelha ao Robocode, no qual existe uma classe básica com os principais comportamentos dos agentes, como minerar, mover e atirar, ficando a cargo do programador estender essas classes de modo a criar comportamentos mais complexos. Como o processo para carregar os agentes no JaRTS foi inspirado no Robocode, é preciso que todos os arquivos, necessários para a implementação dos seus comportamentos, sejam salvos no mesmo diretório do simulador.

No JaRTS há três tipos de agentes básicos:

- *Worker* - simula um trabalhador cujo objetivo é coletar recursos para abastecer a base (ControlCenter).
- *Tank* - simula um tanque de guerra. É o agente "militar", responsável pelo combate.
- *ControlCenter* – serve como depósito de recursos para os *Workers*, e representam também os objetivos dos *Tanks*, seja para protegê-los (próprios) ou atacá-los (inimigos).

Estes tipos de agentes foram baseados nos existentes no ORTS, suas funções e objetivos são baseados nas categorias previamente citadas.

No JaRTS cada tipo de agente pode ser simulado de maneira isolada, ou seja, pode-se criar um tipo *MyWorker* e simular seu comportamento em um mundo em que só existirá elementos desse tipo, todos eles coletando recursos, sem interferência de inimigos, semelhante à categoria Game 1 do ORTS.

Da mesma maneira, é possível criar um tipo *MyTank* para simulá-lo num mundo em que o único objetivo é o combate, sem preocupações com coleta ou gerenciamento de recursos, igual à categoria Game 2 do ORTS. Mais ainda, se o usuário desejar ele pode implementar o comportamento de seu *Worker* e de seu *Tank*, e testá-lo num ambiente de um jogo RTS real com todos os seus desafios: coleta de recursos, combate, gerenciamento de recursos, etc.

4.1 Modelagem e Implementação

A modelagem do JaRTS busca criar uma arquitetura simples e intuitiva com o propósito de diminuir ao máximo o tempo gasto para a sua aprendizagem, fator comum ao se adotar uma ferramenta nova.

Para alcançar esse objetivo, vários fatores influenciam sobremaneira:

- Linguagem de Programação - A linguagem de programação adotada foi Java devido a sua abrangência (altamente difundida) e ao caráter multi-plataforma inerente à tecnologia permitindo o usuário focar somente no problema, a implementação dos agentes.
- Arquitetura - Para facilitar o entendimento acerca do funcionamento do JaRTS, a arquitetura e modo de simulação serão

semelhantes ao difundido Robocode. Dessa forma os usuários já habituados com o Robocode não encontrarão dificuldade para realizar simulações no JaRTS.

A seguir é apresentada a arquitetura básica do JaRTS.

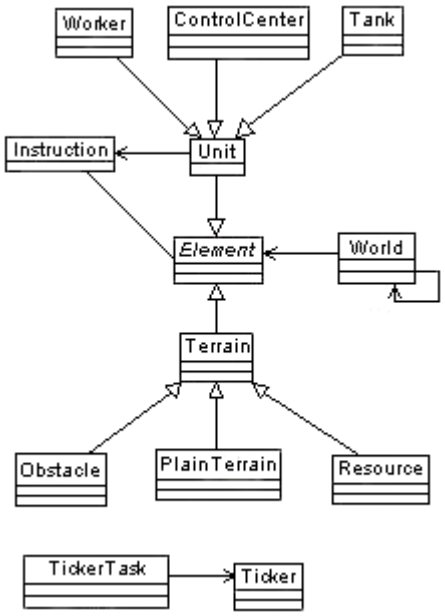

Figura 3: Diagrama de Classes do JaRTS

A classe World possui uma representação do mundo formado por *Elements*, que podem ser elementos de dois tipos:

- Unit, controlados pelo usuário, que são os Workers, Tanks e Control Centers
- Terrain, que representa os elementos do cenário, como Obstacle, que representa um tile em que o agente não pode passar, Resource, a ser coletado pelos Workers ou Plain Terrain, que é um lugar por onde o agente pode passar.

Cada *Unit* possui um objeto do tipo *Instruction*, que representa uma ação que ela deve executar naquele ciclo. Cada *Instruction* possui um indicador especificando qual a ação, e um *Element* que representa o alvo da ação, por exemplo, na instrução *Mine* de um worker é necessário configurar a *Instruction* corretamente especificando qual a *Resource* que o *Worker* vai minerar. Na instrução de *Shoot* é necessário especificar em qual *Unit* ele deve atirar. A classe *Ticker* é uma classe de controle que executa o ciclo do jogo a cada intervalo de tempo.

A implementação de um agente (comportamento) é realizada de forma semelhante à encontrada no Robocode. O usuário estende as unidades que deseja, *Worker*, *Tank* ou *ControlCenter*, e implementa os comportamentos desejados. Por exemplo, o usuário pode criar uma Classe *MyWorker* que estende da classe *Worker*. No momento de criar a simulação o usuário escolhe as classes que participarão, no caso *MyWorker* e alguma outra entidade. Após esta seleção basta dar início à simulação.

5. Conclusão

Este trabalho apresentou as dificuldades no estudo de problemas de Inteligência Artificial em jogos RTS, como falta de ferramentas para um estudo comparativo entre as diferentes abordagens. Uma análise foi realizada entre as diferentes ferramentas que permitem a implementação de técnicas de IA em jogos RTS e foi comprovado que as ferramentas existentes não suprem a necessidade atual, seja por falta de foco no problema, de documentação ou a alta complexidade.

A solução proposta consiste no desenvolvimento de um simulador semelhante ao Robocode com foco em jogos RTS. O simulador proposto possui arquitetura simples e de fácil uso, permitindo ao usuário manter o foco no problema, e posteriormente poder fazer um estudo comparativo entre as diferentes abordagens adotadas. JaRTS já está sendo usado por 65 alunos de uma turma de Agentes Inteligentes na UFPE. Porém uma avaliação mais aprofundada precisa ser feita.

Agradecimentos

Este trabalho é apoiado pelo Centro de Informática, Manifesto Game Studio, Jynx Playware e FACEPE.

Referências

SCHWAB, B. 2004. AI Game Engine Programming. *Charles River Media*, Setembro, 2004.

JARTS. [online] Disponível em: www.cin.ufpe.br/~vvf/jarts [Acessado em 28 de agosto de 2006].

BOSON. [online] Disponível em: boson.eu.org/index.php [Acessado em 28 de agosto de 2006].

GLEST. [online] Disponível em: glest.jackie3d.org [Acessado em 28 de agosto de 2006]

BURO, M., 2005. Citando referências: A Free Software RTS Game Engine [online] ORTS, Disponível em: www.cs.ualberta.ca/~mburo/orts/index.html [Acessado em 28 de agosto de 2006].

BURO, M., 2006. Citando referências: RTS Game Competition [online] ORTS, Disponível em: www.cs.ualberta.ca/~mburo/orts/AIIDE06/ [Acessado em 28 de agosto de 2006].

ROBOCODE. [online] Disponível em: http://robocode.sourceforge.net/ [Acessado em 28 de agosto de 2006]

STRATAGUS. [online] Disponível em: stratagus.sourceforge.net/ [Acessado em 28 de agosto de 2006]