ESCOLA
POLITÉCNICA

Desenvolvimento de um Sistema 3D para Simulação de Comportamentos de Agentes Inteligentes

Trabalho de Conclusão de Curso

Engenharia da Computação

Ronaldo Ferreira dos Anjos Filho
Orientador: Prof. Fernando Buarque de Lima Neto
Co-orientador: Prof. Veronica Teichrieb

ESCOLA
POLITÉCNICA DE
PERNAMBUCO

Ronaldo Ferreira dos Anjos Filho

Desenvolvimento de um Sistema 3D para Simulação de Comportamentos de Agentes Inteligentes

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, dezembro de 2008

A todos que estiveram sempre “ao meu lado”, me suportando e me apoiando mesmo que “de longe”

Agradecimentos

Agradeço a minha família, pois sem seu apoio, compreensão e afeto eu não poderia ter chegado até aqui.

Agradeço a meus amigos e colegas de faculdade e/ou trabalho, pois sem suas presenças, estórias/historias e “bizarrices” (que não são poucas...) eu não teria forças nem animo para continuar neste caminho até o final.

Agradeço a todos do DSC, que eu não considero apenas como departamento, mas sim como uma “segunda família”. Mesmo tendo problemas como toda família, o DSC me proporcionou aprendizado, tanto profissional/acadêmico quanto pessoal, nestes cinco anos de convívio.

Agradeço a professora Veronica, pois, mesmo eu não sendo um aluno exemplar, acreditou em meu potencial, me incentivou e me deu oportunidades para que me torna-se melhor, tanto pessoalmente quanto profissionalmente.

Agradeço ao professor Fernando Buarque, que me orientou nesse último semestre, me apontando as direções certas no lugar de me conduzir “pela mão”, reforçando assim minha capacidade de “caminhar com as próprias pernas”.

Resumindo, agradeço a todos que fizeram ou fazem parte da minha vida, sem vocês eu não teria chegado até aqui e não teria me tornado na pessoa que sou hoje.

Resumo

Efeitos gráficos de alta qualidade não representam mais o único diferencial na indústria de jogos atualmente, pois o poder de processamento dos processadores de propósito geral aliado com o das GPUs (*Graphics Process Unit*) pode proporcionar uma experiência de imersão muito superior em relação à de alguns anos atrás. Além de possuir tais efeitos os jogos devem proporcionar experiências de jogo interessantes ao jogador, como comportamento não linear dos personagens controlados pelo sistema e interação realística com o ambiente onde estão inseridos. Este Trabalho de Conclusão de Curso tem como objetivo o estudo de algumas técnicas que estão sendo utilizadas hoje para o desenvolvimento de jogos: Simulação Física (SF) e Inteligência Artificial (IA), dando mais ênfase à simulação de comportamento inteligente dos personagens controlados pelo jogo em detrimento à interação simples dos personagens com o ambiente.

Abstract

High-quality graphics effects represent no more the only difference in the game industry today. The processing power of general purpose processors together with the GPUs (Graphics Process Unit) can provide a higher immersion experience relative to what happened some years ago. In addition to graphics effects, the games should provide interesting playful experiences to the player such as nonlinear behavior of the characters controlled by the system and realistic interaction with the environment where they belong to. This work aims at studying some techniques being used today for game development: Physics Simulation and Artificial Intelligence, emphasizing the simulation of intelligent behavior of system's characters as oppesed to simple characters interaction with the environment.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

GPU – *Graphics Processing Unit*.

SF – Simulação Física

UT3 – *Unreal Tournament* 3.

FPS – *First Person Shooter*.

HST – *Hydora Sorcerer's Trainingroom Demo*.

IA – Inteligência Artificial.

SMA – Sistemas Multiagentes Inteligentes.

AD – Árvores de Decisão.

AG – Algoritmos Genéticos.

SWFU – *Star Wars: Force Unleashed*.

PPU – *Physics Processing Unit*.

MMO – *Massively Multiplayer Online.*

Capítulo 1

Introdução

Atualmente, efeitos gráficos de alta qualidade não representam mais o único diferencial na indústria de jogos. O poder de processamento dos processadores de propósito geral, aliado com o das GPUs (*Graphics Processing Units*), permite o desenvolvimento de funcionalidades tais como simulação de comportamentos físicos e animação que proporcionam uma experiência de imersão muito superior em relação à de alguns anos atrás.

Mas, a garantia de sucesso de um jogo não reside apenas em proporcionar uma experiência visual impar. Ele deve oferecer também, dentre outras características, uma interação realista do jogador com as entidades que modificam diretamente o ambiente virtual, incluindo os personagens controlados pelo sistema.

Para alcançar tal objetivo, o comportamento destas entidades não deve ser apenas reativo, ou seja, não deve ser apenas baseado em respostas pré-definidas a determinados estímulos. Estas entidades devem possuir a capacidade de observar as modificações no ambiente em que elas estão inseridas, interpretá-las, aprender com elas (*i.e*. ser inteligentes), e executar alguma(s) ação(ões) que as ajude(m) a alcançar seus respectivos objetivos.

Além do comportamento não linear das entidades do jogo, o ambiente deve oferecer características atrativas, como, por exemplo, Simulação Física Newtoniana (SF) [1] para aumentar o nível de realismo.

Atualmente, existem vários jogos no mercado que possuem as características citadas anteriormente. Por exemplo, o UT3 (*Unreal Tournament 3*) [2] é um jogo no estilo FPS (*First Person Shoter*) [3] onde o jogador interage com entidades aliadas ou inimigas possuidoras de comportamento não linear. Com isto, são capazes de executar atividades diferentes de forma paralela, como, por exemplo, atacar adversários, esquivar-se de ataques e movimentar-se na direção do alvo ao mesmo tempo. A Figura 1 mostra o combate entre o jogador e um inimigo em UT3 .

Este jogo, além de simular efeitos visuais, também faz uso de SF para construção do ambiente, sendo um dos principais elementos do jogo. O próprio cenário serve como desafio para o jogador, possuindo elementos como avalanches, construções que podem ser destruídas e até tornados, como mostrado na Figura 2.

Figura 1. Cena de combate entre o jogador e um inimigo no jogo UT3.

Figura 2. SF de um tornado no jogo UT3.

1.1 Objetivos

Neste contexto, este Trabalho de Conclusão de Curso visa obter como resultado a versão demonstrativa do jogo intitulado *Hydora: Sorcerer's Training Room*, a partir deste momento chamado de HST. Este protótipo deve oferece ao jogador um desafio não linear além de possuir elementos físicos simples, como detecção de colisão e movimentação.

O jogo desenvolvido pode ser utilizado para a simulação de diversos tipos de ambientes virtuais onde exista competição e/ou cooperação entre os diferentes indivíduos presentes nele. Assim, experimentos direcionados para a área de IA, mais especificamente SMAs (Sistemas Multiagentes Inteligentes) [4] para a simulação de sociedades [5] podem ser mais facilmente realizados.

1.2 Estrutura do Documento

O presente Trabalho de Conclusão de Curso foi dividido em seis capítulos, sendo eles:

- **Capítulo 1 – Introdução:** Capítulo contendo o texto introdutório do referente trabalho

- **Capítulo 2 – Fundamentação Teórica:** Capítulo contento a explicação de alguns conceitos importantes utilizados para o desenvolvimento deste trabalho.

- **Capítulo 3 – O Jogo *Hydora: Sorcerer's Training Room*:** Capítulo contendo detalhes do ambiente de jogo desenvolvido

- **Capítulo 4 – Simulação do Comportamento Inteligente:** Capítulo contendo maiores detalhes a respeito do modelo de inteligência utilizado para o desenvolvimento do módulo de IA de HST.

- **Capítulo 5 – Experimentos e Resultados:** Capítulo contendo explicação e resultados a respeito dos experimentos executados no ambiente de HST com o intuito de validar o modelo de inteligência explicado no Capítulo 4.

- **Capítulo 6 – Conclusão:** Capítulo contendo as impressões finais a respeito deste Trabalho de Conclusão de Curso, assim como também os próximos passos a serem seguidos para o aperfeiçoamento do ambiente virtual de HST.

Capítulo 2

Fundamentação Teórica

Para a concepção deste projeto foi necessário pesquisar na literatura quais são as técnicas de IA que estão sendo usadas atualmente para o desenvolvimento de jogos, além de avaliar quais são as ferramentas mais usadas para a SF de tempo real demandada em jogos. Nas subseções seguintes, alguns conceitos a respeito de IA e SF em jogos que foram utilizados neste trabalho serão explicados.

2.1 Inteligência Artificial em Jogos

Graças às inovações providas pela computação gráfica recentemente, os jogos de última geração proporcionam uma experiência visual impar aos jogadores. Porém, isto não garante que um jogo obtenha sucesso no mercado [6]. Por este motivo, a utilização de técnicas de IA tem se mostrado útil para melhorar a experiência de jogo dos usuários.

Muitos jogos fazem uso destas técnicas, como o UT3 (já citado anteriormente) e o *Assassin's Creed* [7], que é um jogo em terceira pessoa [8] que possui um sofisticado sistema de interação do jogador com os personagens controlados pelo jogo, inimigos ou não. Por exemplo, a Figura 3 ilustra a população da cidade (personagens controlados pelo jogo) interagindo diretamente com o jogador. Dependendo das ações tomadas pelo jogador, a população pode agir com desconfiança sobre ele, dificultando assim sua infiltração.

Figura 3. Jogador (vestido de branco) interagindo com a população no jogo *Assassin's Creed*

Todos os gêneros de jogos podem se beneficiar do uso de técnicas de IA, desde os já citados FPS e terceira pessoa até os simuladores de vôo, como Flight Simulator [9], e corrida, como Need for Speed [10], tanto na criação de ambientes virtuais realísticos como de personagens.

Um fator desfavorável ao uso em larga escala de técnicas de IA na construção de jogos é o custo computacional associado a elas. Muitas destas técnicas requerem muito processamento e/ou memória, tornando proibitivo o uso massivo destes recursos em aplicações de tempo real.

Porém, vista a crescente demanda por funcionalidades de IA nos jogos, estão sendo desenvolvidas ferramentas para auxiliar os desenvolvedores a modelar comportamentos não lineares em suas aplicações. Tais ferramentas, salvo algumas exceções, ainda não alcançaram um nível de maturidade satisfatório, diferentemente de outras ferramentas usadas para construção de jogos, como motores gráficos, de SF, de rede, entre outros.

Um aspecto importante é que muitas destas ferramentas são de domínio privado, sendo usadas apenas pelos seus criadores, não existindo descrições detalhadas destes trabalhos na literatura informando como foram desenvolvidas. Além disto, motores que se propõem a oferecer uma solução completa para o desenvolvimento de jogos dão ênfase maior aos módulos gráfico e de SF, em detrimento ao de IA.

Como conseqüência da relevância deste problema, foi criado em 2002 pela *International Game Developers Association* (IGDA) o *Artificial Intelligence Interface Standards Committee* [11], com o objetivo de definir e criar padrões para o desenvolvimento de *middlewares* de IA para construção de jogos. Este comitê é formado tanto por representantes de empresas quanto do meio acadêmico.

Nas subseções seguintes serão apresentados alguns conceitos a respeito de técnicas de IA comumente usadas na criação de jogos.

2.1.1 Árvores de Decisão

ADs (Árvores de Decisão) [12] são estruturas poderosas e úteis na resolução de problemas que envolvam tomada de decisões e que possuem muitas variáveis a serem levadas em consideração. Mais importante, ADs possuem a capacidade de adaptação necessária para fazer frente aos das dinâmicas e complexidades dos jogos atuais.

Uma AD utiliza a técnica “dividir para conquistar” na resolução de seus problemas, dividindo um macro-problema em subproblemas menores. Alguns subproblemas podem ser compostos por outros subproblemas, sendo também subdivididos para construir a árvore, até que nenhum dos subproblemas possa ser novamente dividido. Ao final, cada folha irá representar uma classe de resolução do problema – neste caso visto como uma classificação.

A utilização de ADs na resolução de problemas é simples: cada nó da árvore representa um ponto de decisão, que é o equivalente a um subproblema. Cada aresta representa uma regra de decisão que levará ou a outro nó de decisão ou a

uma das folhas. Cada percurso da raiz da árvore até a folha corresponde a uma regra completa de classificação.

Em termos formais, uma AD representa uma disjunção de conjunções de restrições nos valores dos atributos, possibilitando assim que qualquer função lógica possa ser representada por estas estruturas. A Figura 4 mostra o esquema de uma AD para esportes praticados ao ar livre.

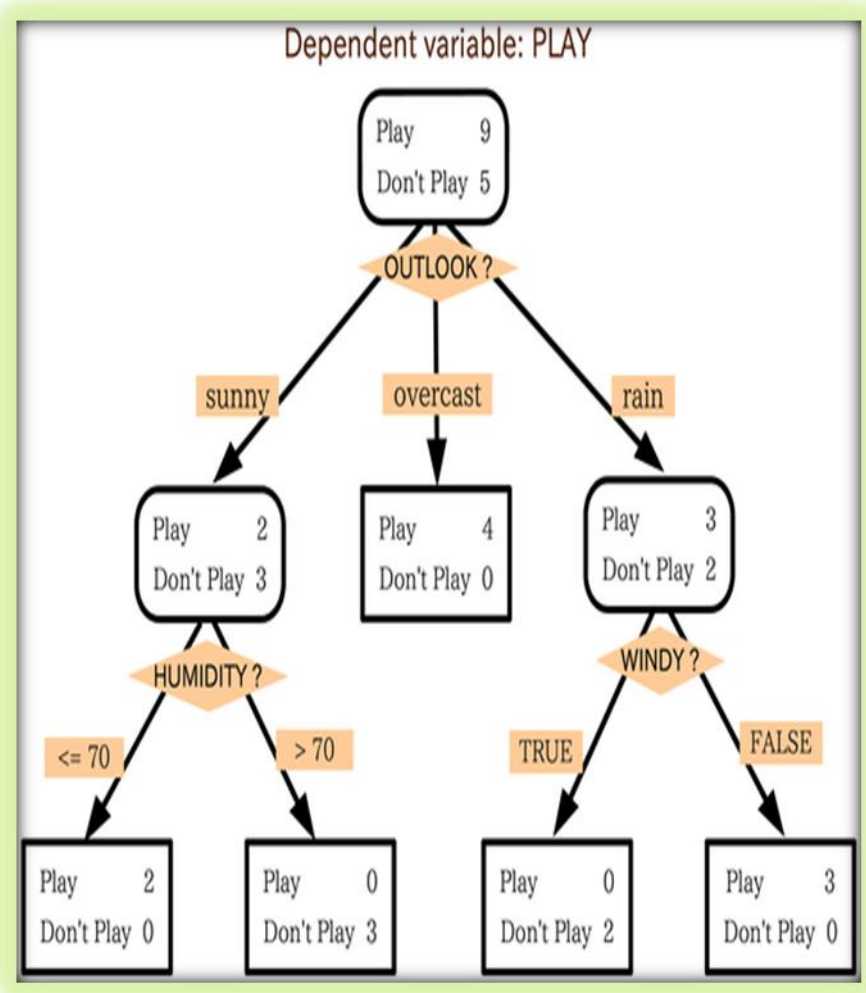

Figura 4. Exemplo de uma AD

Existem diversos algoritmos para o treinamento de ADs baseados em dados previamente conhecidos. Um deles é o ID3 [13], que usa como medida de desempenho o valor de ganho de informação [14] para selecionar os atributos que

serão usados para construção da árvore. Um exemplo de aplicação para este algoritmo é o treinamento de uma AD para execução de *Data Mining* [15].

Atualmente, diversos jogos usam esta técnica para modelar o comportamento de suas entidades. O UT3 adota a abordagem de utilizar duas ADs para o controle dos personagens, uma delas sendo para movimentação e a outra para decisões estratégicas (atirar, trocar de arma, recarregar, retornar à base, etc.).

2.1.2 Algoritmos Genéticos

AGs (Algoritmos Genéticos) [16] é uma técnica de IA inspirada em modelos biológicos para, comumente, resolução de problemas que consistam na busca de uma solução ótima, tais como problemas de roteamento.

Inspirada nos modelos de evolução propostos por Darwin [17], para a execução desta técnica é preciso antes definir exatamente qual é o “ambiente” em que os “indivíduos” serão inseridos. Após isto, é necessário mapear o problema a ser resolvido em parâmetros que irão compor as “informações gênicas” desses indivíduos. Analogamente às informações genéticas dos seres vivos, elas são chamadas de cromossomos.

Os cromossomos são compostos por várias outras unidades menores chamadas genes. Cada gene codifica uma característica específica do organismo, como, por exemplo, cor dos olhos, cabelos e pele, altura, etc. Conjuntos de genes que codificam uma mesma característica, como cabelos pretos e cabelos ruivos, são chamados de alelos. No caso de AGs, cada gene representa um aspecto do problema a ser resolvido.

Depois do mapeamento do problema, são gerados diversos conjuntos de indivíduos chamados de gerações. Cada geração possui indivíduos com informação genética derivada dos indivíduos da geração anterior, ou seja, cada nova geração é uma “evolução” da geração anterior, mas não necessariamente esta evolução significa indivíduos mais bem adaptados ao ambiente. Além disso, deve-se decidir qual(is) métrica(s) será(ão) usada(s) para avaliar se o problema foi resolvido ou não.

A execução de um AG geralmente possui cinco etapas [6] (Figura 5), sendo elas:

- **Inicializar a População:** Nesta etapa é criada a primeira população, possuindo cada indivíduo valores aleatórios em seus genes.

- **Avaliar Soluções:** Nesta etapa é avaliado se a última geração conseguiu se adaptar bem ao ambiente, ou seja, se conseguiu resolver o problema para o qual o algoritmo foi construído.

- **Escolher Reprodutores:** Nesta etapa são escolhidos os indivíduos mais bem adaptados da geração anterior. Além disto, os pares de indivíduos a reproduzir são formados.

- **Cruzamento e Mutação:** Nesta etapa são efetuadas as combinações entre os genes dos pares formados na etapa anterior com o intuito de combinar os pontos fortes destes. Além da própria combinação, é inserido em alguns casos um fator aleatório de "mutação" com o objetivo de gerar genótipos mais diversificados.

- **Gerar Nova População:** Nesta etapa são gerados os indivíduos com cromossomos formados a partir das combinações efetuadas na etapa anterior. Após esta etapa é novamente executada uma nova avaliação (novo ciclo evolucionário).

- **Encerrar Algoritmo:** Se o problema foi resolvido (*i.e.* condição de parada for atendida), a execução do algoritmo pára, caso contrário, é executada a próxima etapa.

AGs são úteis quando é difícil prever as interações entre o conjunto de parâmetros que regula o comportamento dos agentes e o mundo do jogo. No entanto, muitas vezes é necessário um grande gasto de tempo na modelagem do problema e principalmente na simulação para alcançar bons resultados.

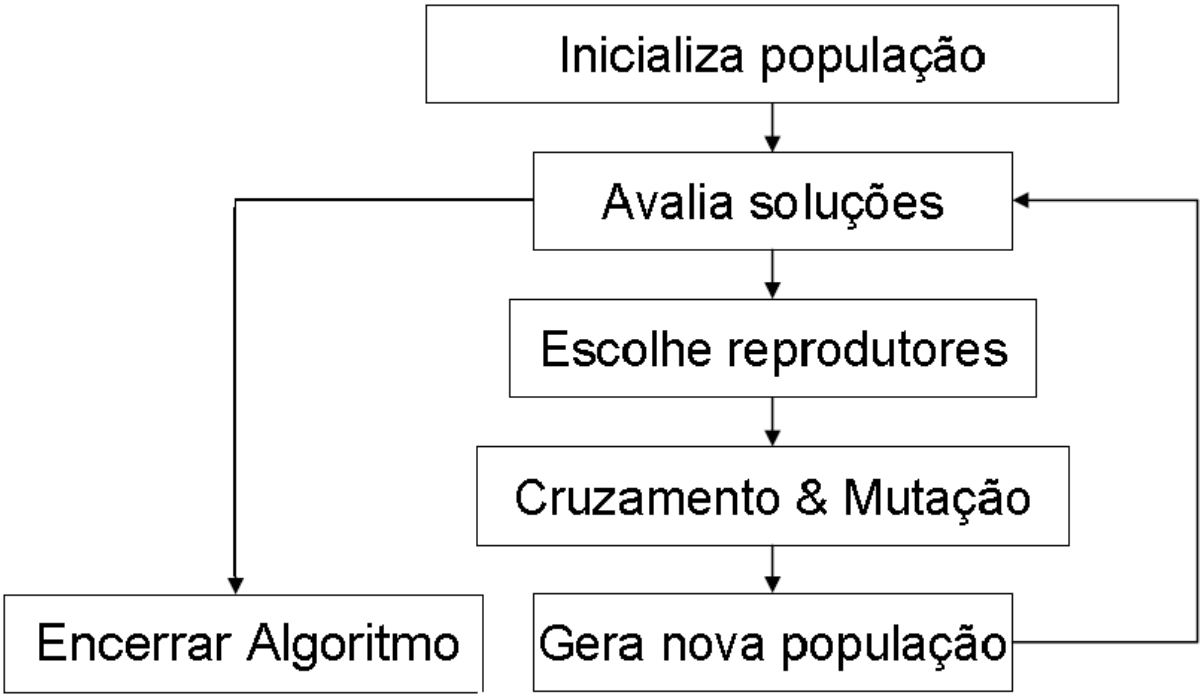

Figura 5. Etapas de execução de um AG

2.1.3 Sistemas Multiagentes Inteligentes

SMAs [4] correspondem a uma subárea da IA e realizam Computação Inteligente distribuída, através do uso de agentes independentes. O principal enfoque da técnica é a coordenação desses agentes para a realização de um determinado objetivo.

A técnica igualmente se inspira em aspectos naturais, as sociedades entomológicas (sociedades de insetos, tais como colméias, formigueiros e colônias de cupins), humanas (equipes, sociedades, até mesmo incluindo disputas, aspecto abordado neste trabalho) e agrupamentos biológicos (como por exemplo, conjuntos de anticorpos atuando contra um vírus).

Para melhor entendimento, o conceito de SMAs será aqui construído aos poucos, na seguinte ordem: Agentes → Agentes Inteligentes → Multiagentes Inteligentes.

Agentes podem ser definidos como as entidades capazes de perceber seu ambiente por meio de sensores, de dar semântica a percepção e de agir sobre o

meio por meio de atuadores, a fim de atingir seus objetivos; tudo conforme ilustrado na Figura 6. Tal definição está intimamente ligada ao conceito de cibernética [18]. Os meios pelos quais um agente pode perceber e atuar num ambiente variam de entidade para entidade. Por exemplo, um agente humano usa seus órgãos sensitivos como sensores, e mãos, pernas e outras partes do corpo como atuadores. Já um robô pode usar câmeras como sensores, lógica clássica no raciocínio e motores como atuadores.

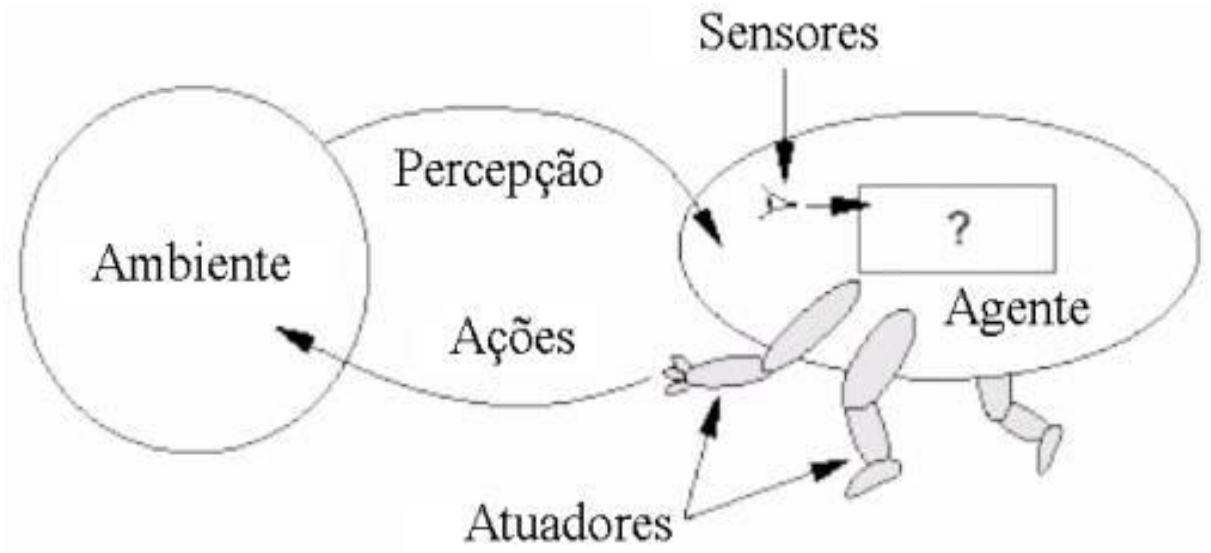

Figura 6. Modelo de um agente.

No contexto do conceito de agentes, percepção diz respeito às informações do ambiente captadas pelo agente em um dado instante de tempo. Pode-se também definir nesse escopo simplista a função de agente, sendo esta uma função que mapeia uma seqüência de percepções em uma ação. A função de agente é implementada por um programa de agente, sendo uma descrição matemática abstrata, enquanto que o programa de agente é uma implementação relacionada à arquitetura do agente.

É necessário especificar se a ação tomada por um agente é boa ou ruim. Com isso, surge a idéia de um agente racional: um agente onde as ações "causadas" por determinadas percepções são as esperadas. Para medir o sucesso de comportamento do agente utiliza-se a chamada medida de desempenho. A seqüência de ações geradas por um agente altera o estado do ambiente em que ele

está inserido. O agente tem um desempenho satisfatório quando essa seqüência de estados é a desejável. É importante salientar que cada agente tem sua própria medida de desempenho e que é melhor projetar medidas de desempenho de acordo com o resultado realmente desejado pelo ambiente ao invés de criá-las de acordo com o comportamento esperado do agente.

Desse modo, a racionalidade depende, em qualquer instante, de quatro fatores:

- A medida de desempenho que define o critério de sucesso;
- O conhecimento anterior que o agente tem do ambiente;
- As ações que o agente pode executar;
- A seqüência de percepções do agente até o momento.

Deve-se distinguir os conceitos de racionalidade e onisciência. Um agente onisciente sabe o resultado real de suas ações, diferentemente de um agente racional. Este último maximiza o desempenho esperado, enquanto que o agente "perfeito" maximiza o desempenho real. Além disso, um agente racional possui uma percepção parcial, ao invés da percepção total que o agente onisciente proporciona. Cabe ressaltar que a onisciência é impossível na prática.

Agentes inteligentes são, portanto, agentes capazes de avaliar seu desempenho, relacionado a uma função objetivo, e que mudam sua forma de agir de modo a maximizar seu desempenho.

Existem vários tipos de agentes inteligentes; dentre eles podem ser citados os seguintes:

- **Agentes Reativos**: funcionam no modelo estímulo-resposta, onde o "conhecimento" está baseado na cibernética. A inspiração dos agentes reativos advém dos modelos de organização biológica como colméias e

sociedades de cupins e formigas. Nessas sociedades um indivíduo por si só não é dotado de inteligência, mas o comportamento gerado pelo grupo provê certo conhecimento. Os agentes reativos não apresentam uma memória das ações tomadas e também não planejam suas ações futuras. Todo o conhecimento que o agente obtém de suas próprias ações e também das ações dos demais membros da sociedade é percebido através das modificações sofridas pelo ambiente. Geralmente as sociedades compostas por agentes reativos são relativamente numerosas. No contexto de jogos, este tipo de agente torna o jogo enfadonho, dada sua previsibilidade.

- **Agentes Cognitivos**: inspirados nos modelos de organização social como grupos, hierarquias e mercados. Os agentes cognitivos podem basear-se nas suas ações passadas para planejar suas ações futuras. Além disso, possuem uma representação explícita do ambiente e dos membros da comunidade e podem interagir entre si através de linguagens e protocolos de comunicação complexos. Sociedades de agentes cognitivos geralmente apresentam um número pequeno de agentes e elevada complexidade computacional. Ao contrário dos anteriores, este tipo de agente, quando incluídos num jogo, o torna desafiador e interessante.

A construção de um agente racional requer a definição do ambiente de tarefa. Os ambientes de tarefas são os “problemas” para os quais os agentes são as “soluções”. O ambiente de tarefa pode ser especificado por quatro elementos, representados pelo acrônimo PEAS (*Performance*, *Environment*, *Actuators*, *Sensors*), ou seja, desempenho, ambiente, atuadores e sensores.

Existem diversas propriedades de ambientes de tarefas, algumas delas citadas a seguir:

- **Completamente observável x Parcialmente observável:** Um ambiente de tarefa é completamente observável quando os sensores detectam todos os aspectos relevantes do ambiente, já num ambiente de tarefa parcialmente observável a não totalidade perceptiva pode ser causada por ruídos ou por deficiências na observação.

- **Determinístico x Estocástico:** Num ambiente determinístico o próximo estado é completamente determinado pelo estado atual e pela ação do agente, ao contrário de um ambiente de tarefa estocástico. Se o ambiente é determinístico, exceto pelas ações de outros agentes, tem-se um ambiente estratégico.

- **Episódico x Seqüencial:** Num ambiente episódico as ações são atômicas. Um episódio, que consiste na percepção e numa ação única, só depende do próprio episódio. Em ambientes seqüenciais a decisão atual pode afetar as decisões futuras. Estes últimos são mais complexos do ponto de vista de sua implementação.

- **Estático x Dinâmico:** Diferentemente dos ambientes estáticos, os ambientes de tarefas dinâmicos podem se alterar enquanto um agente está em execução. Se o ambiente não muda, mas a medida de desempenho é modificada, tem-se um ambiente semidinâmico.

- **Discreto x Contínuo:** O conceito de ambientes discretos ou contínuos pode se aplicar tanto ao estado do ambiente como também ao tratamento do tempo e às percepções e ações do agente. No ambiente discreto tem-se um número finito de estados distintos.

- **Agente único x Multiagente:** Diz respeito à unicidade ou não de um agente num determinado ambiente.

A base de um SMA inteligente é a comunidade inteligente. Numa comunidade inteligente, os agentes são autônomos e estão inseridos no mesmo ambiente. Tais agentes não precisam ser, necessariamente, inteligentes; na comunidade inteligente, o que importa é a inteligência na ação coletiva – que deve ser de natureza adaptativa. Vale salientar que a autonomia dos agentes pode gerar conflitos. Portanto, é absolutamente necessário que haja comunicação entre eles.

Pode-se dividir em três tipos a interação entre agentes: forte (na qual estão envolvidos a capacidade de decisão, as possibilidades e o conhecimento do outro

agente), média (em que as possibilidades e o conhecimento de um agente estão relacionados com as possibilidades e o conhecimento de outro) e fraca (quando só há relação dos modelos de agente por meio do conhecimento).

Dentre os tipos de SMAs inteligentes, pode-se citar:

- ***Blackboard System*:** Trata-se de um grupo de especialistas (Fontes de Conhecimento) que interage através de uma memória compartilhada (*blackboard*) na qual todo o ambiente pode estar representado. No momento em que um agente não se mostra capaz de realizar uma tarefa, ele a armazena no *blackboard*, para que, posteriormente, outro agente (que seja capaz) realize tal tarefa – o contrário também pode ser utilizado. A vantagem deste tipo é que não há necessidade de os agentes conhecerem uns aos outros, mas em compensação há uma excessiva troca de mensagens a fim de garantir que todos possuam o mesmo grau de percepção do ambiente.

- **SMAs Federados:** É aquele que contém agentes complexos chamados Facilitadores. Tais agentes possuem informações sobre as habilidades de outros agentes mais simples e organizam o trabalho destes. Ao receber uma requisição, o Facilitador localiza o agente mais apto a realizar tal tarefa. Sua principal vantagem é gerenciar eficientemente a comunicação entre agentes. Por outro lado, a desvantagem é que as tarefas importantes ficam centralizadas no Facilitador e, caso este funcione inadequadamente, podem acontecer falhas graves no sistema.

- **SMAs Democráticos:** Sua principal característica é o fato de todos os agentes da comunidade possuírem o mesmo nível hierárquico. Aqui, tais agentes realizam ações coletivas assincronamente, incluindo a comunicação. O SMA Democrático tem como vantagens sua modularidade e sua flexibilidade. Em compensação, cada agente da

comunidade precisa, o que não é tão simples, conhecer as habilidades dos outros (pelo menos, parcialmente).

- **SMAs Abertos:** Caracterizam-se por não possuírem uma composição fixa de comunidade; ou seja, os agentes podem entrar ou se desligar dela a qualquer momento. Portanto, dependendo da composição do sistema, pode ser que um serviço não esteja disponível naquele momento. Neste tipo de sistema, os agentes podem adaptar-se e escolher diferentes objetivos a serem alcançados. O SMA Aberto também se caracteriza por ser robusto, mas se nota uma excessiva troca de mensagens por parte de um protocolo de comunicação que garante a composição flexível da comunidade.

Com relação ao comportamento, existem dois critérios para classificar os agentes cognitivos: a alocação de tarefas (uma tarefa global envolve todos os agentes da comunidade; uma tarefa local, apenas um agente) e a habilitação (certo agente só está habilitado a executar uma tarefa se possui as aptidões necessárias para isto). Segundo tais critérios, pode-se classificar em cinco tipos os comportamentos presentes numa sociedade de agentes:

- **Coabitação:** Um agente não é obrigado a cooperar com os outros; ele só deve realizar suas tarefas locais.

- **Cooperação:** Se um agente é incapaz de realizar alguma tarefa, ele solicita aos outros agentes que o ajudem a fazê-lo.

- **Colaboração:** Se vários agentes são capazes de executar uma tarefa, deve haver um mecanismo que eleja qual agente irá realizar tal tarefa.

- **Distribuição:** Se não há um agente no sistema capaz de realizar uma tarefa, deve haver uma alocação de tarefas que faça com que os agentes trabalhem coletivamente e cumpram seu dever.

- **Competição:** Os agentes não cooperam, pois possuem objetivos mutuamente ou parcialmente exclusivos (ex.: partida de xadrez, em que apenas um pode vencer).

Embora a tecnologia de agentes tenha um papel importante no desenvolvimento de aplicações computacionais, não deve ser superestimada. Existem situações em que, mesmo que o problema apresente aspectos de computação distribuída, não deve ser usada a abordagem de multiagentes. Alguns dos problemas trazidos por essa abordagem são:

- **Nenhum controlador total do sistema:** Uma solução baseada em agentes pode não ser apropriada para os domínios em que os vínculos globais têm que ser mantidos, nos domínios onde uma resposta em tempo real deve ser garantida, ou nos domínios em que os “becos sem saída” ou os travamentos devem ser evitados;

- **Nenhuma perspectiva global:** As ações dos agentes, por definição, são determinadas pelo estado local dos agentes (ambiente parcialmente observável). Logo, os agentes tomam decisões globais não ótimas (a decisão ótima seria baseada na união do conhecimento de todos os agentes);

- **Confiança e delegação:** Para que os usuários estejam de acordo com a idéia de delegar tarefas aos agentes, devem primeiramente confiar neles. Deve-se atentar para que o agente não use excessivamente a sua autoridade com relação ao usuário ou organização, ou seja, um agente deve saber seus limites.

2.2 Simulação Física em Jogos

O desenvolvimento dos processadores de propósito geral e das GPUs foi acompanhado de perto pelos avanços relacionados à indústria de jogos eletrônicos,

esta se beneficiando mais especificamente da qualidade gráfica provida pelas GPUs de última geração.

Com o passar dos anos, a parte gráfica dos jogos evoluiu de simples *pixels* para ambientes virtuais 3D complexos, possuindo diversos efeitos de renderização, como iluminação e sombras realísticas, matérias reflexivos, dentre outros.

A indústria de jogos durante muito tempo investiu a maior parte de seus recursos no desenvolvimento da parte gráfica destes, criando ambientes com nível elevado de realismo visual. Em compensação, o comportamento do ambiente de jogo não condiz com seu realismo gráfico, este oferecendo poucas formas de interação com o jogador [6].

Algumas empresas perceberam a carência de jogos que possuíssem realismo inserido no próprio ambiente de jogo e não apenas em sua visualização. Estas empresas criaram os chamados jogos “ultra realísticos”, jogos que oferecem ambientes dinâmicos e interativos, possuindo neles outro fator de realismo: SF Newtoniana.

A SF Newtoniana em jogos proporciona aos jogadores um novo nível de interação. Anteriormente possuindo apenas propósito visual, os ambientes de jogo que possuem SF se tornaram elementos que influenciam diretamente sua mecânica, oferecendo maiores desafios aos jogadores.

Diversos jogos já utilizam SF em seus ambientes. Além de UT3 mencionado anteriormente, ainda existe *Star Wars: Force Unleashed* (SWFU) [20] (Figura 7), jogo em terceira pessoa construído usando o motor Havok [21] que, combinado com o Euphoria [22], oferece grande realismo na simulação de seres articulados.

Além do Havok, existem diversas outras ferramentas para SF, dentre elas podemos citar o NVIDA PhysX [23] que foi utilizado neste trabalho.

Anteriormente chamado de AGEIA PhysX, este motor funciona como um *middleware* de física, baseado em *software*, para a criação de ambientes físicos

dinâmicos. O PhysX suporta as principais plataformas para jogos e aplicações gráficas, tais como PlayStation3, XBOX360, PC, entre outras. Pela abrangência de plataformas, facilidade de programação e diversas *features* oferecidas, o uso deste *engine* é bastante difundido, sendo usado em diversas aplicações e jogos comerciais. Além disto, a placa AGEIA PhysX foi a única PPU (*Physics Processing Unit*) dedicada para computadores *desktop*, construída para melhorar a física executada em aplicações que usam este *engine*. Atualmente a produção destas placas foi encerrada, sendo agora oferecida aceleração física diretamente nas mais recentes GPUs da NVIDIA.

Figura 7. Simulação de personagens articulados em SWFU

Capítulo 3

O Jogo *Hydora: Sorcerer's Training Room*

O *Hydora Sorcerer's Training Room* ou HST é um protótipo de um jogo de combate em terceira pessoa desenvolvido neste Trabalho de Conclusão de Curso. Para isto, foram estudadas as principais técnicas de IA utilizadas no desenvolvimento dos jogos atuais: ADs, AGs e SMAs. Além do comportamento inteligente, o ambiente do jogo oferece simulação da física Newtoniana que, nesta primeira versão do jogo, foi aplicada para detecção de colisão e movimentação simples. O comportamento inteligente dos agentes do jogo é simulado a partir da combinação de AG, com genes mapeando características físicas e comportamentais, e AD, sendo esta última influenciada pela parte comportamental do genótipo doas agentes.

Nas seções seguintes são explicados os detalhes do conceito e da jogabilidade do HST, para entendimento do funcionamento da mecânica do seu ambiente e do comportamento de suas entidades. Além disto, é apresentada uma breve descrição da arquitetura dos módulos que o compõem.

3.1 *Game Design*

Como mencionado anteriormente, o HST é um jogo 3D em terceira pessoa onde o jogador deve controlar um personagem com certas habilidades específicas contra grupos de inimigos controlados pelo sistema. À medida que o grupo de inimigos vai sendo derrotado, novos inimigos são gerados pelo sistema para combater o jogador, até que este seja derrotado por eles.

Nas subseções seguintes são apresentadas as características gerais do HST e as representações gráficas e físicas do mundo e dos personagens do jogo. Cada tipo de personagem possui atributos e habilidades especiais, que também são descritos.

3.1.1 Características Gerais

O jogo HST possui as seguintes características:

- Ambiente tridimensional interativo (utilizando modelagem gráfica e simulação física);
- Estilo de jogo em terceira pessoa;
- Personagem/Jogador deve sobreviver em combate contra diversos inimigos controlados pelo sistema;
- Os personagens podem se movimentar em dois eixos no espaço (X e Z);
- O ambiente do jogo é limitado ao volume de um cubo de mil metros de aresta;
- A interface de entrada de comando do usuário é o teclado para movimentação e o *mouse* para mirar e executar ações;

3.1.2 O Mundo do Jogo

Neste primeiro protótipo do HST o ambiente do jogo foi simplificado para uma região invisível de formato cúbico de mil metros de aresta. Como são permitidas movimentações em dois eixos do ambiente 3D (X e Z), os personagens inseridos nele podem se movimentar por uma área de um quilômetro quadrado (1 Km^2). O ambiente é composto por um plano revestido com uma textura com padrão de rocha

representando o piso do mundo e uma textura de nuvens representando o céu (ver Figura 8).

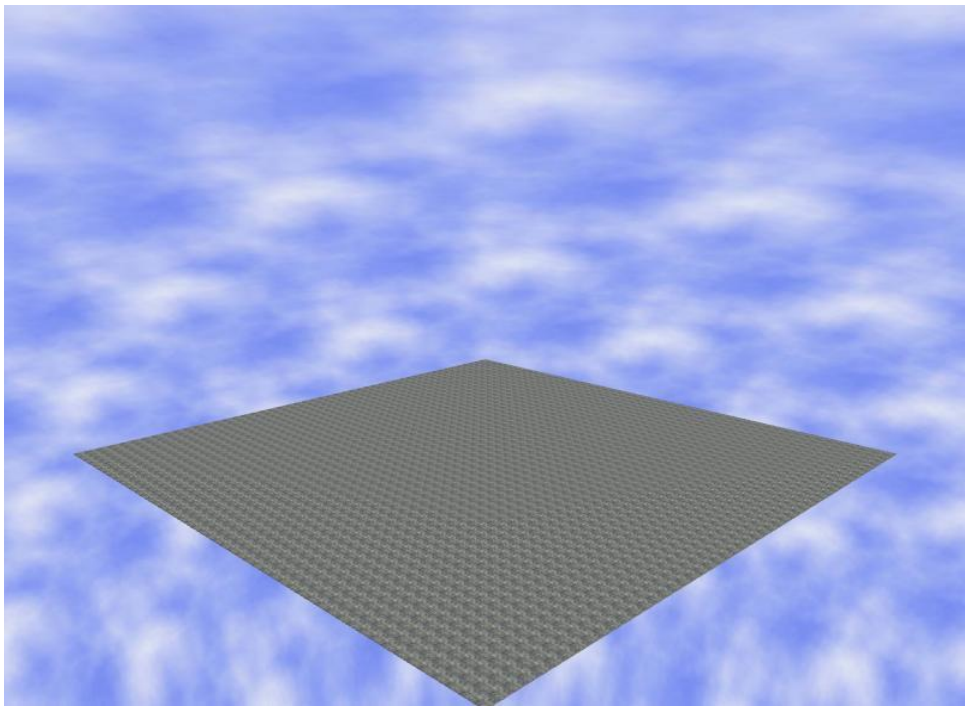

Figura 8. Ambiente 3D do HST

Cada vez que algum personagem avança além dos limites do cenário, ele é transportado de volta para o centro do ambiente, garantindo assim que todos os personagens existentes eventualmente interajam entre si. No caso de algum ataque (*e.g.* tiro de energia) avançar além dos limites, este será removido do ambiente.

3.1.3 Os Personagens do Jogo

O HST possui quatro tipos distintos de personagens: Jogador, *Fighter*, *Defender* e *Healer* (descritos na seqüência). Com exceção do Jogador, todos os demais são controlados pelo sistema. Nesta primeira versão do jogo a movimentação dos personagens não é muito realista, não possuindo animações enquanto se movimentam.

A modelagem física dos personagens também foi simplificada, se resumindo à detecção de colisão em situações de ataques executados pelos inimigos. Todos os personagens possuem a mesma modelagem física, tendo seus corpos formados por

um volume retangular de tamanho semelhante ao do corpo humano com os braços colados ao corpo (dois metros de altura e oitenta centímetros de distância entre os ombros). Cada personagem possui três sensores baseados em raios (técnica *raycast* [24]) para identificação de objetos que estão presentes em sua “linha de tiro”. A Figura 9 mostra os respectivos modelos 3D dos personagens e sua representação no ambiente de SF.

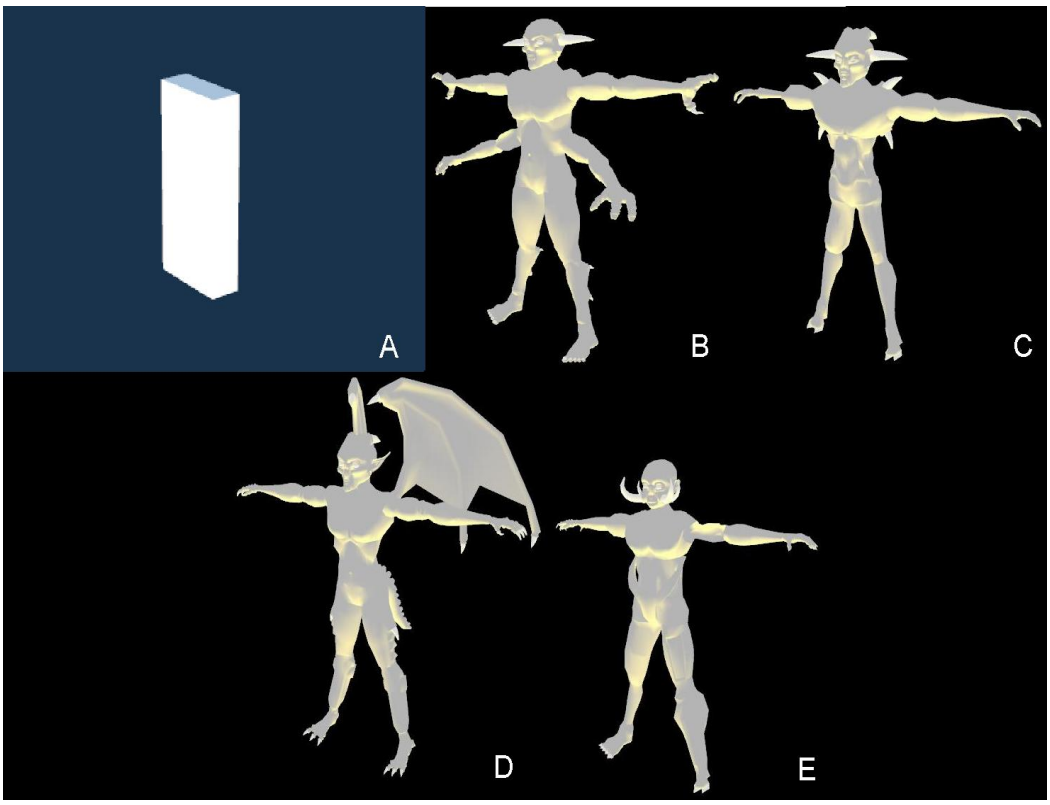

Figura 9. Modelos dos personagens do HST: (A) modelo físico; (B) *Fighter*; (C) *Healer*; (D) *Defender*; (E) Jogador.

Todos os tipos de personagens possuem cinco atributos comuns:

- **Saúde:** Quantidade de dano que um personagem pode suportar antes de morrer.

- **Energia:** Cada utilização de poder especial feita pelo personagem está associada a um custo em pontos de energia. O personagem estará impossibilitado de usar sua(s) habilidade(s) especial(is) caso os pontos de energia cheguem à zero.

- **Nível de Força:** Se refere à eficácia da utilização do poder especial. Quanto maior o valor deste atributo, maior será o dano, por exemplo, do ataque causado pelos *Fighters*.

- **Taxa de Perda de Energia:** Refere-se à eficiência com a qual o personagem utiliza sua energia. O valor deste atributo é descontado do valor de energia atual do personagem a cada vez que ele utiliza sua habilidade especial.

- **Defesa:** Nível de proteção a ataques possuído pelo personagem. Quanto maior o valor deste atributo, menor o dano sofrido pelo personagem a cada ataque.

Os quatro tipos de personagens do HST possuem as seguintes habilidades especiais (descritas com mais detalhes na próxima subseção):

- ***Fighter*:** Personagens com características ofensivas, possuindo como habilidade o disparo de bolas de energia que causam danos. O dano é calculado pela diferença entre o nível de força do personagem *Fighter* que está atacando e o nível de defesa do Jogador. Os poderes de ataque deste tipo de personagem afetam apenas o jogador, ultrapassando os outros elementos do mundo como se não estivessem presentes no ambiente.

- ***Defender*:** Personagens com características defensivas, possuindo como habilidade o poder de criar uma barreira de proteção contra ataques ao redor do alvo especificado, melhorando assim as chances de sobrevivência dele.

- ***Healer*:** Personagem com características de suporte, possuindo como habilidade o poder de curar seus alvos, assim como o poder de proteção do personagem *Defender*, melhorando as chances de sobrevivência dele.

- **Jogador:** Personagem não controlado pelo sistema, e sim pelo usuário. Como o ambiente do HST possui vários inimigos, o personagem Jogador pode utilizar todos os poderes de seus inimigos utilizam: ataque, defesa e cura.

3.1.4 Os Poderes dos Personagens

Como já foi mencionado anteriormente, cada personagem possui uma habilidade especial que o caracteriza. A utilização dos poderes pelos personagens é limitada pela sua quantidade de energia e por um intervalo de tempo fixo entre cada utilização. A seguir são descritas detalhadamente estas habilidades:

- **Poder de Ataque:** Utilizado pelos personagens *Fighters* e pelo Jogador para causar danos em seus alvos. O dano de cada ataque depende da diferença entre o nível de força do personagem atacante e o nível de defesa do personagem atacado.

- **Poder de Proteção:** Utilizado pelos personagens *Defenders* e pelo Jogador para aumentar o nível de proteção do personagem alvo. O Jogador pode utilizar este poder apenas em benefício próprio, enquanto os *Defenders* em benefício de qualquer aliado.

- **Poder de Cura:** Utilizado pelos personagens *Healers* e pelo Jogador para recuperar a saúde e energia perdidos durante o combate. Do mesmo modo que os *Defenders* e o poder de proteção, os *Healers* podem curar qualquer aliado que desejarem, enquanto o Jogador pode apenas curar a si próprio. Outra restrição com relação ao poder de cura do Jogador é que este só recupera seus pontos de energia (os pontos de vida não podem ser recuperados).

A Figura 10 mostra as representações gráficas dos três poderes disponíveis no HST.

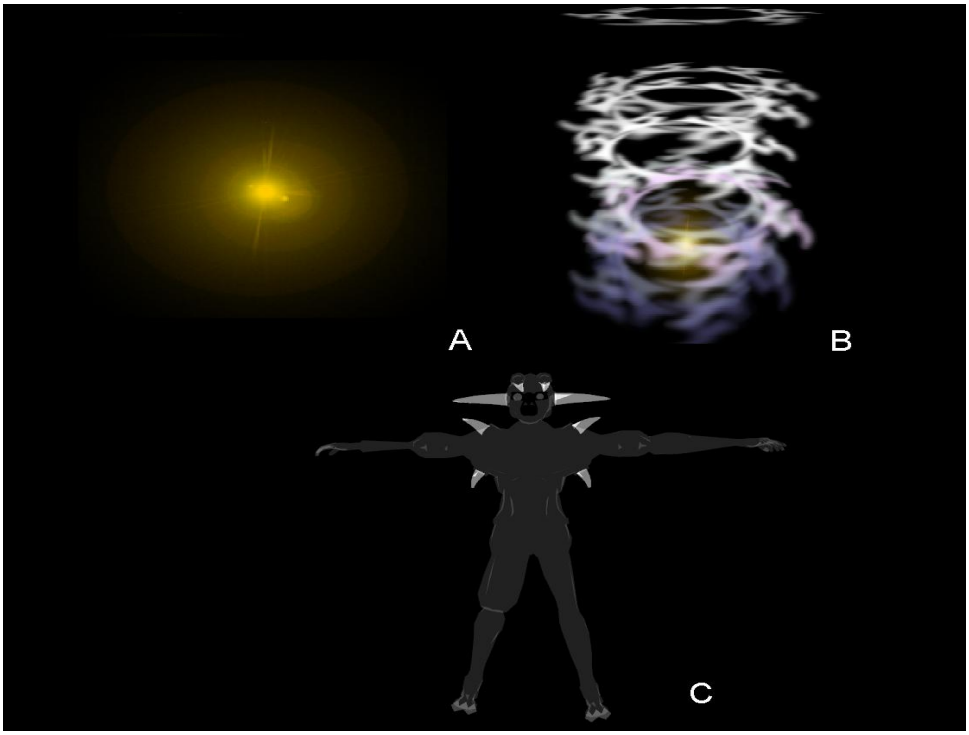

Figura 10. Tipos de poderes existentes no HST: (A) poder de ataque; (B) poder de cura; (C) poder de proteção aplicado a um *Healer*.

3.2 Arquitetura

A implementação do HST foi realizada utilizando a linguagem de programação C++ e as ferramentas OGRE (gráfico) [25] e PhysX (SF). O ambiente do HST é formado por diversos módulos distintos e independentes, com exceção do módulo chamado *Core*. Este acoplamento fraco permite que os módulos sejam alterados ou completamente substituídos mais facilmente.

Para a integração dos módulos foi utilizada a abordagem de comunicação por eventos, ou seja, a cada alteração de estado em algum módulo é disparado um evento correspondente, notificando os demais módulos interessados nessa modificação. A Figura 11 ilustra um exemplo de disparo e notificação de evento relacionado à atualização de objetos gráficos baseados na posição dos seus respectivos objetos físicos.

Figura 11. Exemplo de disparo e processamento de evento.

A seguir são listados e descritos os módulos desenvolvidos para a criação do ambiente virtual do HST, esquematizados pela Figura 12.

- **Módulo de IA:** O módulo de IA é responsável pelo controle dos inimigos do jogador, coordenando suas ações baseando-se nos estados em que o jogador se encontra e nos de seus aliados. Mais detalhes a respeito deste módulo serão descritos no próximo capítulo.

- **Módulo Gráfico:** Módulo responsável pela renderização do ambiente virtual do HST. As principais alterações de estado deste módulo são causadas pela atualização da orientação e posição das entidades presentes no ambiente, sendo executadas a cada ciclo da simulação do ambiente.

- **Módulo Físico:** Módulo responsável pela simulação do comportamento físico do ambiente do HST, oferecendo funcionalidades de detecção de colisões (útil, por exemplo, para verificação de choques durante os ataques) e movimentação.

- **Módulo *Device*:** Módulo responsável pela captura de eventos de entrada e saída disparados pelo usuário para interação com o ambiente virtual. A presente versão do jogo oferece suporte à teclado e *mouse*, mas pode ser adicionado facilmente o suporte a *joysticks*.

- **Módulo *Core*:** Módulo central do HST, responsável pela integração de todos os módulos citados anteriormente. A partir do módulo *Core* são feitas todas as configurações necessárias para simulação do ambiente virtual (número, tipo, velocidade de movimentação e comportamento das entidades existentes, dentre outras).

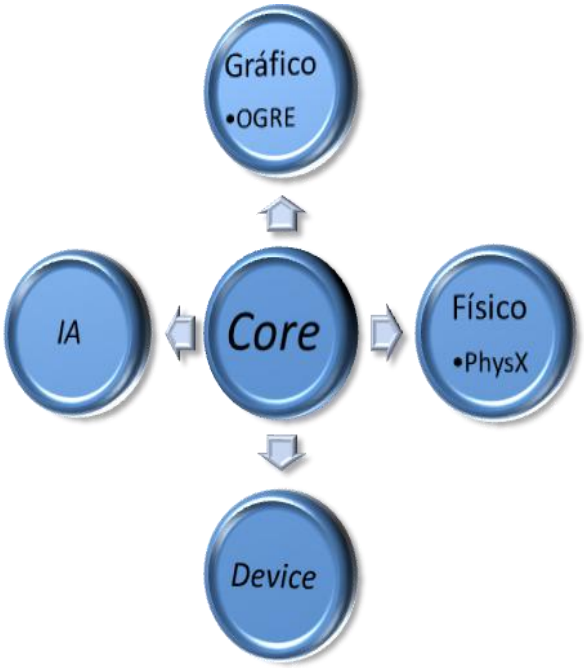

Figura 12. Esquema simplificado da arquitetura do HST.

Capítulo 4
Simulação do Comportamento Inteligente

Como já mencionado anteriormente, o HST possui alguns tipos de personagens que são controlados pelo sistema. Este controle é feito de forma descentralizada, ou seja, cada personagem tem sua própria percepção do ambiente e toma suas próprias decisões quanto a qual ação irá executar. Nas seções a seguir serão descritos detalhes do modelo de inteligência dos personagens e como procedeu a sua implementação.

4.1 Modelo Abstrato

Uma das características principais do ambiente do HST é a sua dinamicidade. À medida que os personagens controlados pelo sistema vão sendo derrotados, outros personagens mais bem adaptados ao ambiente são criados para substituí-los. O critério de seleção utilizado para definir os personagens mais bem sucedidos da última geração é simples: aqueles personagens que tiverem sobrevivido durante um certo intervalo de tempo são selecionados.

O HST foi construído na forma de um SMA Democrático com um ambiente completamente observável, estratégico, episódico, contínuo e colaborativo. A comunicação entre os agentes controlados pelo sistema se dá por memória compartilhada, sendo restrito até o momento a mensagens de pedido e confirmação de ajuda, caso o personagem esteja “ferido”. As características intrínsecas dos agentes funciona baseado em codificação genética.

Cada personagem controlado pelo sistema é definido por, além de sua classe (*Fighter*, *Healer* e *Defender*), seu genótipo e por sua estrutura de tomada de decisão, essas sendo explicadas nas subseções seguintes.

4.1.1 Genótipo

O genótipo dos personagens do HST é definido por um cromossomo composto por sete genes, sendo que três influenciam no comportamento deles e quatro definem suas características físicas (Figura 13).

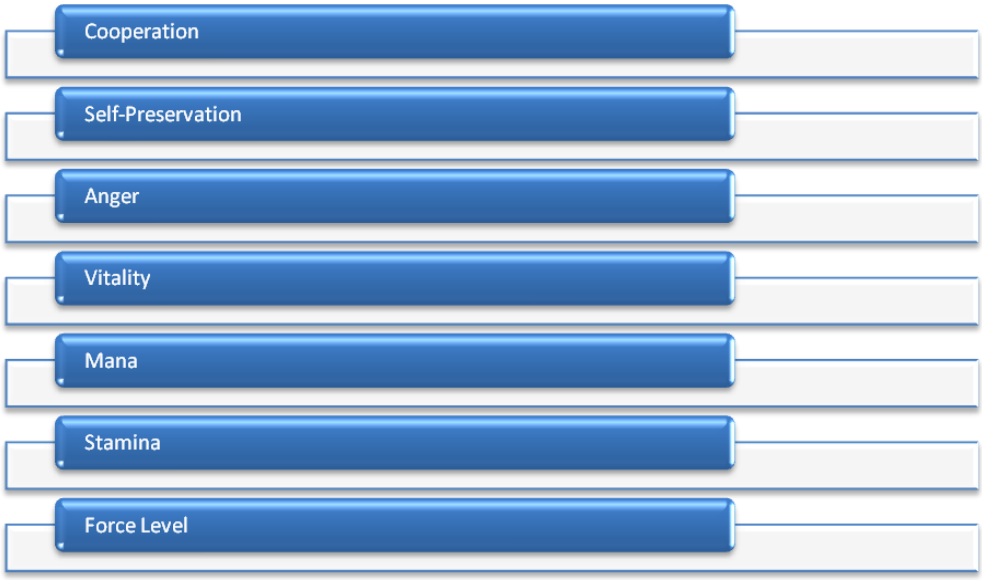

Figura 13. Genótipo dos personagens do HST

- ***Cooperation:*** Gene responsável por influenciar o "sentimento de grupo" do personagem. Quanto mais forte for esse gene, maior é a propensão do personagem em ajudar os outros que estão em dificuldade.

- ***Self-Preservation:*** Responsável por influenciar a "prudência" do personagem. Quanto mais forte for esse gene, maior será o senso de perigo do personagem.

- ***Anger:*** Gene que influencia a agressividade do personagem. Quanto mais forte for o gene, menor será o senso de perigo do personagem e maior será sua propensão em avançar contra o inimigo.

- ***Vitality:*** Gene responsável por influenciar a “saúde” do personagem. Quanto mais forte for esse gene, maior será a tolerância do personagem a receber dano. Este gene influencia diretamente a característica **saúde** do personagem.

- ***Mana:*** Responsável por influenciar o nível de energia que o personagem possui. Quanto mais forte for esse gene, mais energia o personagem possui. A característica **energia** é influenciada diretamente por esse gene.

- ***Stamina:*** Gene responsável por influenciar o “vigor” do personagem, ou seja, o quanto ele é eficiente no aproveitamento de sua energia. Esse gene influencia diretamente a característica **taxa de perda de energia**; quanto mais forte ele for, melhor o aproveitamento de energia.

- ***Force Level:*** Gene que influencia o “poder” do personagem. Quanto mais forte esse gene, maior será o nível de defesa e os efeitos dos poderes especiais do personagem. As características **defesa** e **nível de força** são diretamente influenciadas por esse gene.

A definição do genótipo de novas gerações de personagens segue as etapas descritas na seção IA em Jogos (subseção Algoritmos Genéticos) do Capítulo 2.

4.1.2 Estrutura de Decisão

Os personagens controlados pelo HST decidem quais serão suas próximas ações à medida que o estado do ambiente muda. Tal tomada de decisão ocorre em três etapas, cada uma delas sendo executada por uma AD distinta:

- **Modificação de Status:** Primeira fase na tomada de decisão dos personagens. Nesta etapa é decidido qual será o status de decisão do personagem naquele instante de tempo. São três os status possíveis: **Atacar Inimigo**, status referente a investir contra o jogador; **Ajudar Aliado**, referente a ajudar algum aliado em necessidade; e **Proteger-se**, referente a apenas se proteger. A AD executada nessa etapa é

fortemente influenciada pelo genótipo do personagem, como mostrado pelas Figura 14 e Figura 15.

- **Decisão de Movimentação:** Etapa de decisão responsável pela movimentação do personagem no ambiente 3D. As decisões possíveis são duas: **Nada**, para o personagem continuar onde está, e **Mover** para o personagem se movimentar para a posição desejada. A AD (Figura 16) de movimentação é influenciada pelo status do personagem: para o status **Atacar Inimigo**, o personagem tenta se manter de três a cinco metros do Jogador, já para o status **Proteger-se** ele tenta se manter a no mínimo trinta metros do jogador, enquanto para o status **Ajudar Aliado** ele tenta ficar no ponto médio entre seu aliado e o jogador.

- **Decisão de Ação:** Etapa referente a se o personagem irá ou não utilizar seus poderes especiais naquele instante de tempo. A AD, ilustrada pela Figura 17, responsável por essa etapa pode fornecer duas respostas: **Nada**, caso seja decidido por não executar nenhuma ação, ou **Executar Ação**, caso seja decidido executar alguma ação (atacar, curar ou proteger, dependendo do tipo do personagem).

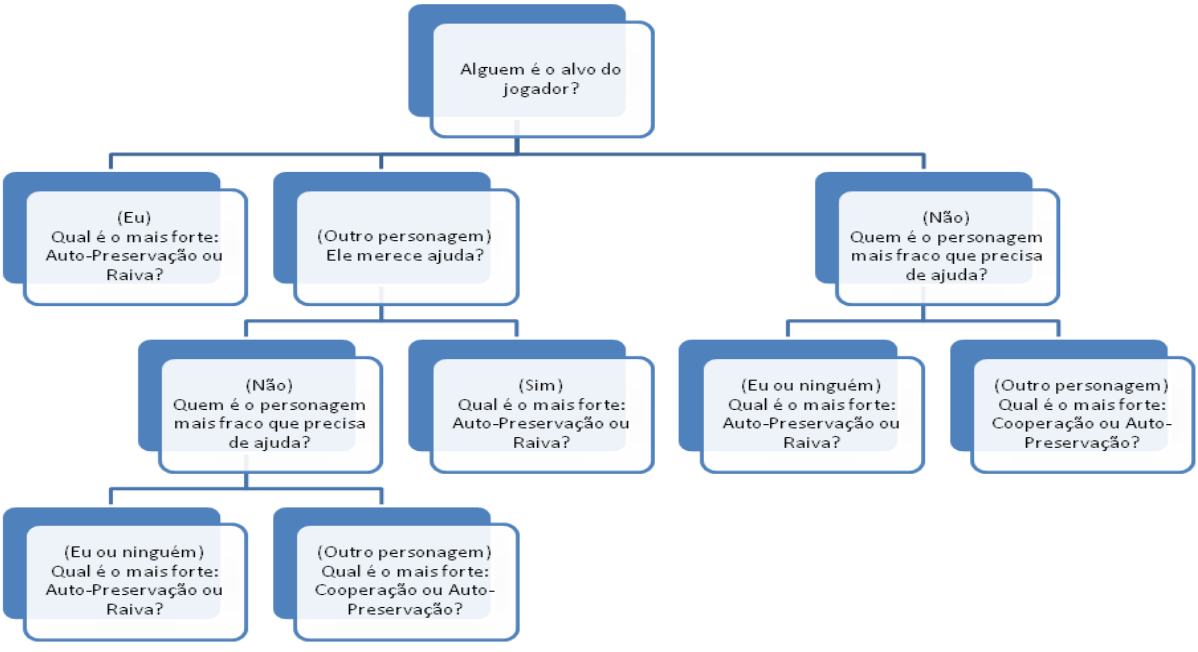

Figura 14. AD para modificação de status

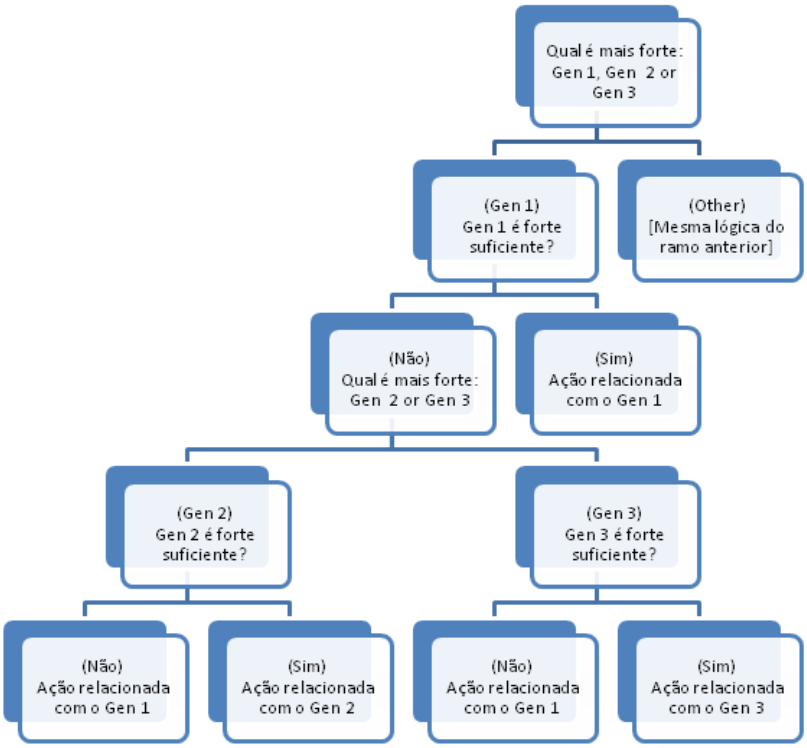

Figura 15. Um exemplo de uma AD influenciada pelo genótipo.

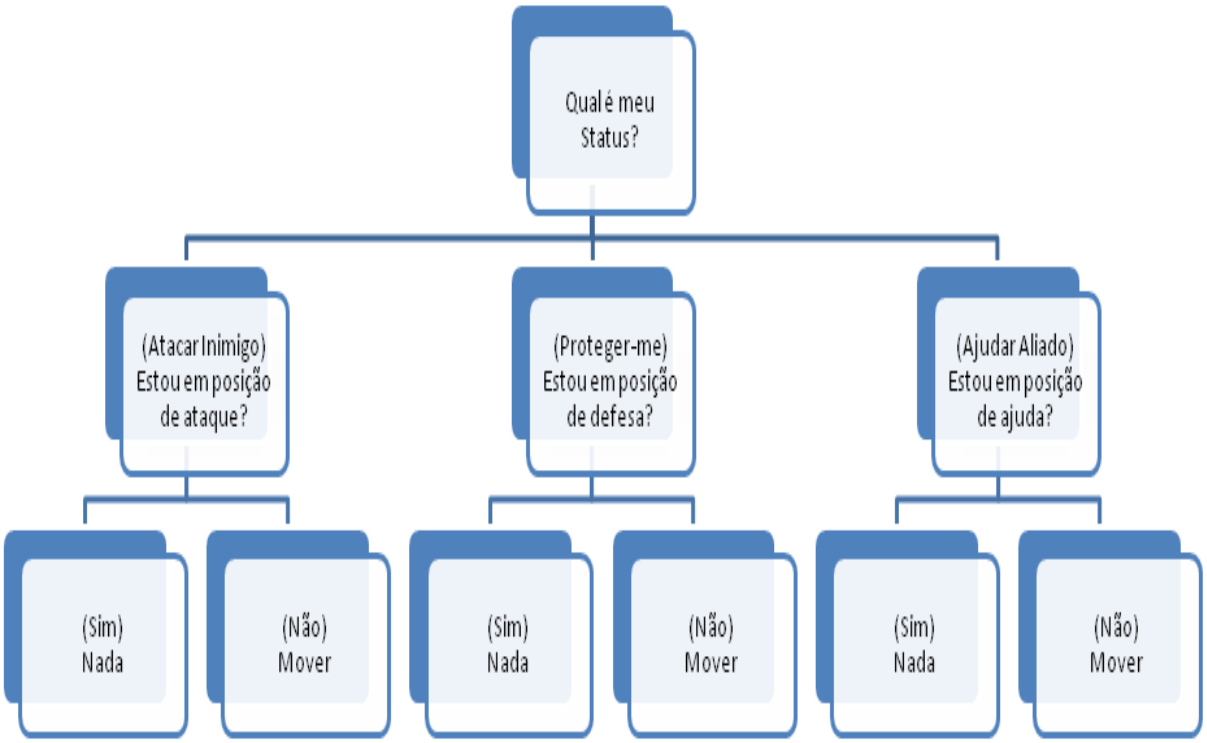

Figura 16. AD de movimentação

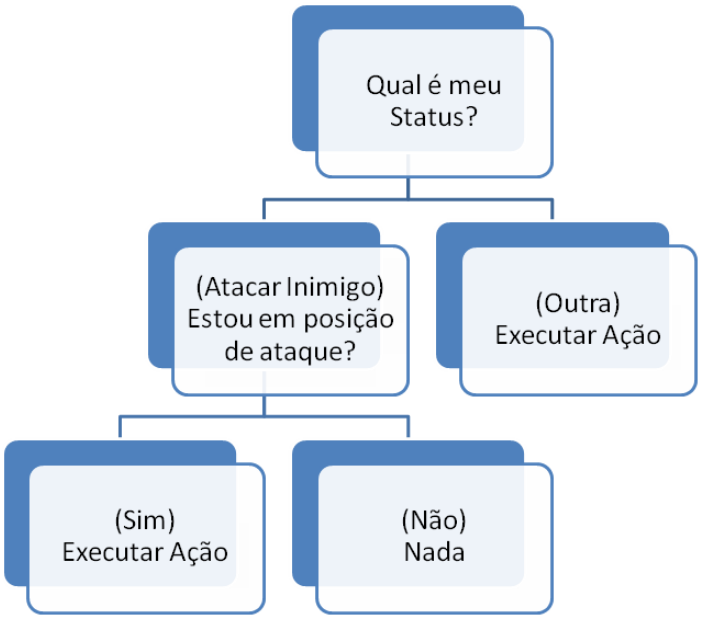

Figura 17. AD de ação

4.2 Implementação

Para construção do módulo de IA foram utilizados os padrões de projeto *Singleton* e *Factory* [26]. A Figura 18 mostra o diagrama UML do módulo de IA e as subseções seguintes explicam detalhes de cada parte do módulo.

4.2.1 Genes e Chromossome

A implementação da parte do módulo de IA relacionada ao AG é composta por uma classe, chamada `Chromossome`, e uma `enumeration` [27], chamada `Genes`. A implementação do conceito de genes foi feita de modo que cada um deles fosse representado por uma constante inteira e única. Tais constantes funcionam como "etiquetas" para referenciá-los mais facilmente no cromossomo. Como são constantes numéricas definidas em tempo de compilação, o custo de memória e processamento associados a eles possui mínimo impacto no sistema.

`Chromossome` representa a informação genética dos personagens controlados pelo sistema. Essa classe possui uma estrutura de dados do tipo

`hashmap` [27] para armazenar os genes e seus respectivos valores. Os valores que podem ser atribuídos a cada gene pertencem ao intervalo [0,1].

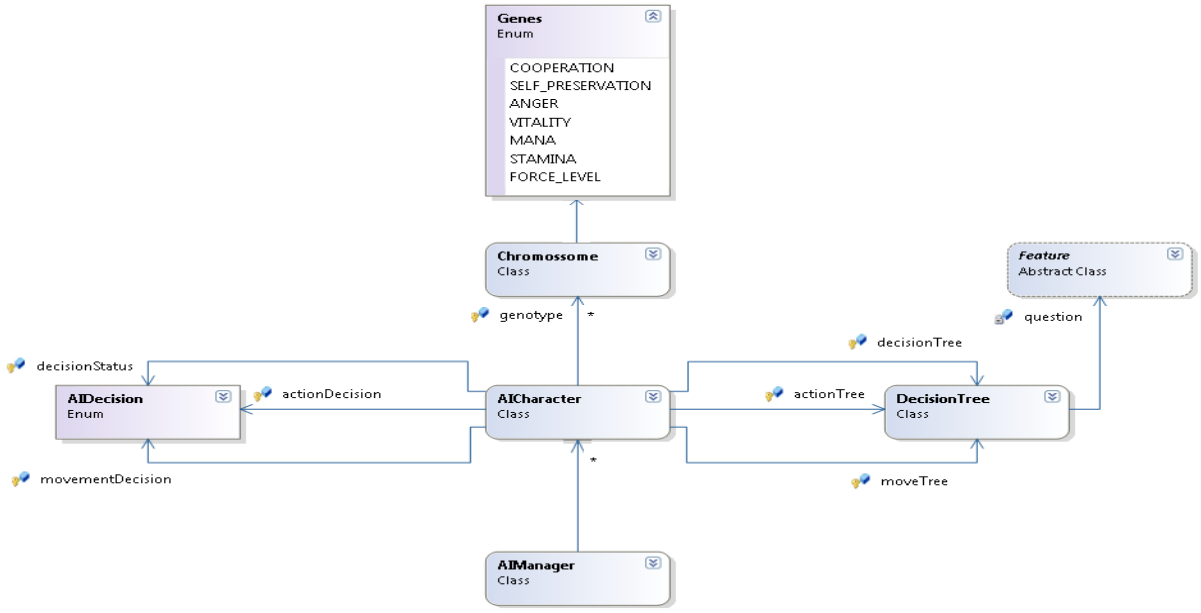

Figura 18. Diagrama UML do módulo de IA

A classe `Chromossome` possui um método para combinar a informação genética entre dois cromossomos, executando-se uma média ponderada com pesos aleatórios entre seus genes alelos, obtendo como resultado um novo cromossomo com os valores de seus genes influenciados pelos dois anteriores. Existe ainda um método que é responsável pela mutação do cromossomo, inserindo um fator aleatório em todos os seus genes.

4.2.2 AIDecision, Feature e DecisionTree

A implementação relacionada à AD possui como componentes principais duas classes, `Feature` e `DecisionTree`, e uma *enumeration*, `AIDecision`, representando as decisões que podem ser tomadas pelos personagens controlados pelo sistema, já descritas na subseção 4.1.2 deste trabalho.

`DecisionTree` representa a implementação de uma AD, enquanto `Feature` a classe-base para todas as “perguntas” feitas em cada nó de decisão da árvore. Até o momento a AD construída a partir da classe `DecisionTree` possui uma topologia estática, ou seja, a forma da árvore gerada não muda depois que ela foi construída.

4.2.3 AICharacter e AIManager

`AICharacter` e `AIManager` são as classes centrais do módulo de IA do HST. `AICharacter` representa a inteligência dos personagens controlados pelo sistema, possuindo como atributos principais:

- **genotype:** Cromossomo que define as características genéticas do personagem.
- **decisionStatus**, **actionDecision** e **movementDecision:** respectivamente, o status de decisão, ação e movimentação do personagem (subseção 4.1.2).
- **decisionTree**, **actionTree** e **moveTree:** respectivamente, as ADs para definir o status de decisão, ação e movimentação do personagem (subseção 4.1.2).

`AIManager` é o ponto central de gerenciamento do módulo de IA do HST, sendo a partir dele feita a criação da parte responsável pela inteligência dos personagens ativos, além de coordenar suas tomadas de decisão. A cada ciclo completo de simulação do ambiente do HST, essa classe faz com que os status de decisão dos personagens sejam atualizados a partir do status atual do ambiente.

A cada determinado intervalo de tempo de simulação do ambiente do HST, a classe `AIManager` verifica quais indivíduos satisfizeram o critério de seleção, faz todas as combinações possíveis entre eles criando assim novos genótipos (inserindo de forma aleatória em alguns deles um fator de mutação), para então selecionar um número predefinido de genótipos para compor a próxima geração de indivíduos a ser produzida. Cada geração, em teoria, vai ser mais adaptada ao ambiente do que a geração anterior.

Capítulo 5

Experimentos e Resultados

Para validação do comportamento inteligente implementado no sistema HST foram feitos alguns testes simples para avaliar a evolução das gerações de inimigos criadas no decorrer do jogo. A seguir serão explicados os detalhes a respeito da métrica adotada para os experimentos e os resultados obtidos a partir deles.

5.1 Métrica

O objetivo dos personagens controlados pelo sistema é único: derrotar o jogador, mesmo que para isso seja necessário seu sacrifício em benefício de seus aliados. Tal comportamento se enquadra perfeitamente na dualidade egoísmo/altruísmo [5], mapeada no genótipo dos personagens por três genes responsáveis pelo balanço de seu comportamento: *Cooperation*, *Self-Preservation* e *Anger*. A parte responsável do genótipo pelo comportamento dos personagens oferece ainda duas subdivisões dos comportamentos altruísta e egoísta: prudente e hostil.

Essas subdivisões de comportamento são refletidas pela relação de intensidade dos três genes já mencionados. Um personagem é considerado altruísta quando o gene *Cooperation* possui valor maior que os outros dois, possuindo como subcomportamento prudente aqueles que possuírem o gene *Self-Preservation* com valor maior do que o de *Anger*; caso contrário, seu subcomportamento será hostil. A definição do subcomportamento dos personagens egoístas segue a mesma lógica dos altruístas.

A métrica adotada para a validação do comportamento inteligente foi o tempo de vida médio de cada geração, medido em ciclos de simulação, e a quantidade de

indivíduos de cada um dos quatro tipos de comportamento existentes em cada uma delas. Foram executadas até o momento quatro simulações do ambiente do HST, as quais, mesmo em número reduzido, já mostraram resultados bastante interessantes.

5.2 Resultados

A seguir são mostrados os resultados obtidos a partir das quatro simulações executadas. A Tabela 1 mostra os dados gerais das simulações: o tempo (em ciclos de simulação) que o jogador permaneceu ativo no jogo, o número e o tempo de vida médio (também em ciclos) das gerações criadas na simulação. As Tabelas 2, 3, 4 e 5 mostram dados referentes ao número de indivíduos pertencentes a cada tipo de comportamento presentes em cada geração das simulações executadas.

Tabela 1. Resultados das simulações

Simulação	Tempo de Vida do Jogador	Quantidade de Gerações	Tempo Médio de Vida das Gerações
S1	1.523	1	1.523
S2	914	1	914
S3	8.051	3	2.683
S4	11028	5	2205

Tabela 2. Gerações da S1

Gerações	Egoístas		Altruístas	
	Prudentes	Hostis	Prudentes	Hostis
G1	3	3	1	1

Tabela 3. Gerações da S2

Gerações	Egoístas		Altruístas	
	Prudentes	Hostis	Prudentes	Hostis
G1	1	4	2	1

Tabela 4. Gerações da S3

Gerações	Egoístas		Altruístas	
	Prudentes	Hostis	Prudentes	Hostis
G1	2	3	3	0
G2	6	0	2	0
G3	6	0	1	1

Tabela 5. Gerações da S4

Gerações	Egoístas		Altruístas	
	Prudentes	Hostis	Prudentes	Hostis
G1	3	3	0	2
G2	8	0	0	0
G3	2	2	2	2
G4	2	0	4	2
G5	1	5	0	2

Pelos resultados obtidos nas simulações já foi possível observar que os personagens inimigos do jogador, baseados na métrica adotada, tendem a assumir um comportamento predominantemente egoísta.

Essa tendência ao egoísmo dos personagens pode ser explicada pelo fato do ambiente em que eles estão inseridos ser basicamente competitivo entre eles e o jogador. Os personagens altruístas tendem a ajudar os aliados que estão sendo atacados pelo jogador, protegendo assim seus aliados dos ataques dele, sendo então derrotados mais rapidamente. Como os personagens egoístas sobrevivem mais tempo, eles acabam passando essa característica às próximas gerações, o que se mostrou, no caso desse ambiente, uma característica vantajosa tanto para a sua sobrevivência quanto para o cumprimento do seu objetivo, que é derrotar o jogador.

Capítulo 6

Conclusão

Além da área de jogos, a própria área de IA pode se beneficiar da associação com a SF, pois essa última pode oferecer, por exemplo, os mecanismos necessários para construção de ambientes complexos para simulação de populações. Neste trabalho se obteve como resultado um ambiente virtual que simula a alteração de comportamento dos personagens inimigos ao longo do tempo. Essa alteração no comportamento é útil, por exemplo, em ambientes de jogo MMO (*Massively Multiplayer Online*), pois, apesar de os experimentos terem mostrado uma tendência ao comportamento egoísta dos personagens, ele pode vir a seguir outras configurações em cenários com vários jogadores a serem derrotados.

Os resultados de simulação mostraram que a utilização de técnicas avançadas de IA associadas a SF demonstram ser importantes no aumento da dinamicidade do ambiente de jogo, proporcionando assim melhores experiências aos jogadores.

Uma outra característica importante constatada neste trabalho foi a dinamicidade oferecida aos personagens inimigos no HST pelo uso de AGs associados a ADs. O mapeamento de características físicas e comportamentais do personagem em sua informação genética, aliado a uma estrutura de decisão influenciada por essa informação, tornou o comportamento dos personagens diversificado, variando entre o altruísmo e o egoísmo e em níveis de prudência e agressividade de forma bastante plausível.

6.1 Trabalhos Futuros

A construção de ambientes virtuais, seja para jogos ou para experimentos de IA, por exemplo, envolve muitos fatores, como renderização, SF, som, comunicação, dentre outros. Uma das dificuldades deste trabalho foi a integração entre o módulo de controle dos personagens (IA) com os demais para execução de uma simulação coesa.

O próprio ambiente virtual desenvolvido para este trabalho ainda encontra-se em estágio limitado, possuindo apenas um cenário onde os personagens estão inseridos e poucas características genéticas associadas a eles (apenas sete, três comportamentais e quatro físicas). Esse fato restringe o uso do HST na simulação comportamental de populações.

O próximo passo deste trabalho é incrementar o ambiente virtual construído para o HST, transformando-o assim num jogo completo. Para tal, será necessária a ampliação do uso de SF e a definição de novos cenários, além também do desenvolvimento de novas funcionalidades de IA, como oferecer alteração dinâmica na topologia das ADs dos personagens – algo importante a realizar. Ainda, como opção de controle deles se pode cogitar o uso de Redes Neurais [4] ou Sistemas Baseados em Regras [4]. A utilização de soluções já prontas, como bibliotecas ou motores de IA é uma possibilidade. Entretanto, o desenvolvimento de uma solução mais direcionada à simulação de comportamento de populações em jogos é uma avenida muito promissora para SMAs.

Bibliografia

[1] HALLIDAY, David; et al. **Fundamentos De Física – Mecânica**. 7. ed. Rio de Janeiro: Ltc. 2008. 376 p.

[2] Unreal Tournament 3. **Unreal Tournament 3 site**. Disponível em: <http://www.unrealtournament3.com>. Acesso em 16 de setembro de 2008.

[3] First Person Shooter (FPS), **Webopedia**. Disponível em: <http://isp.webopedia.com/TERM/F/fps.html>. Acesso em 16 de setembro de 2008.

[4] RUSSEL, Stuart; NORVING, Peter. **Artificial Intelligence: A Modern Approach.** 1. ed. New Jersey: Prentice-Hall. 1995. 946 p.

[5] PITA, Marcelo Sousa; LIMA NETO, Fernando Buarque. **Simulations of egoistic and altruistic behaviors using the vidya multiagent system platform.** Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation. London, UK.

[6] KARLSSON, BÄorje Felipe Fernandes. **Um Middleware de Inteligência Artificial para Jogos Digitais.** 2005. 110f. Dissertação de Mestrado do Programa de Pós-Graduação em Informática, PUC - Rio, Rio de Janeiro.

[7] Assassin's Creed. **Assassin's Creed Site**. Disponível em: < http://assassinscreed.uk.ubi.com/experience>. Acesso em 20 de agosto de 2008.

[8] Third Person Shooter, **Wikipedia**. Disponível em: < http://en.wikipedia.org/wiki/Third-person_shooter>. Acesso em 30 de agosto de 2008.

[9] Flight Simulator. **Flight Simulator Site**. Disponível em: < http://www.microsoft.com/brasil/games/fs2006/>. Acesso em 25 de setembro de 2008

[10] Need For Speed. **Need For Speed Site**. Disponível em: < http://www.needforspeed.com/undercover/home.action?lang=pt&region=br>. Acesso em 10 de agosto de 2008.

[11] IGDA. **IGDA Site.** Disponível em: < http://www.igda.org/ai/ >. Acesso em 20 de desembro de 2008.

[12] A. BENDER, Edward; WILLIAMSON, S Gill. **Mathematics for Algorithm and System Analysis.** 1. ed. Mineola: Dover Publications. 2005. 256 p.

[13] MITCHELL, Tom. **Machine Learning**, 1. ed. New York: McGraw-Hill. 1997. 432 p.

[14] ABRANSOM, Norman. **Information Theory and Coding**. 1. ed. New York: McGraw-Hill. 1963. 300 p.

[15] TAN, Pang-Ning; et al. **Introduction to Data Mining**. 1. ed. New York: Addison Wesley. 2005. 769 p.

[16] BODENHOFER, Ulrich. **Genetic Algorithms: Theory and Applications.** 3. ed. Linz: Fuzzy Logic Laboratorium Linz-Hagenberg. 2004. 126 p.

[17] DARWING, Charles. **On the Origin of Species**. 1. ed. London: John Murray. 1859. 247 p.

[18] ASHBY, W. Ross. **Introduction to Cybernetics**. 1. ed. Methuen: Chap. & H. 1956. 308 p.

[19] Crysis. **Crysis Site**. Disponível em: <http://www.ea.com/crysis/home.jsp>. Acesso em 01 de novembro de 2008.

[20] Star Wars: The Force Unleashed <http://www.lucasarts.com/games/theforceunleashed>. Acesso em 30 de outubro de 2008.

[21] Havok Engine. **Havok Site**. Disponível em: <http://www.havok.com>. Acesso em 01 de novembro de 2008.

[22] Euphoria Engine. **NaturalMotion Euphoria Site**. Disponível em: < http://www.naturalmotion.com/euphoria.htm>. Acesso em 25 de setembro de 2008.

[23] NVIDIA PhysX. **NVIDIA PhysX Site**. Disponível em: < http://www.nvidia.com/object/nvidia_physx.html>. Acesso em 10 de novembro de 2008.

[24] LAMOTHE, Andre. **Black Art of 3D Game Programming.** 1. ed. San Francisco: Waite Group Press. 1995. 1175 p.

[25] Ogre3D. **Ogre Site**. Disponível em: <http://www.ogre3d.org>. Acesso em 10 de agosto de 2008.

[26] FREEMAN, Elisabeth; et al. **Head First Design Patters**. 1. ed. Sebastopol: O'Reilly Media. 2004. 640 p.

[27] DEITEL, M. D. C++: Como Programar. 3. ed. Porto Alegre: Bookman. 2001. 1098 p.