Murilo Cerone do Nascimento
Tássio Naia dos Santos

Arquitetura *BDI* para agentes e *Blackboard*: um estudo de caso de técnicas de IA aplicadas a jogos

São Paulo
2010

Murilo Cerone do Nascimento
Tássio Naia dos Santos

Arquitetura *BDI* para agentes e *Blackboard*: um estudo de caso de técnicas de IA aplicadas a jogos

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Área de Concentração:
Engenharia de Computação

São Paulo
2010

Murilo Cerone do Nascimento
Tássio Naia dos Santos

Arquitetura *BDI* para agentes e *Blackboard*: um estudo de caso de técnicas de IA aplicadas a jogos

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Área de Concentração:
Engenharia de Computação

Orientadores:
Ricardo Nakamura
Roberto Cezar Bianchini

São Paulo
2010

. . . no amor e para graduar!

“O resultado de qualquer pesquisa
séria só pode ser fazer duas questões
crescerem onde antes só crescia uma.”

Thorstein Veblen
Sociólogo e economista
noruego-estadunidense
(1853 – 1929)

Resumo

Falar da motivação para uso de técnicas de IA em jogos, e do potencial que o BDI e o quadro-negro têm de agregar ao realismo da experiência dos jogos. (Citar artigos falando do emprego dessas técnicas.)

Falar do intuito do estudo de caso, do que se pretende (e como) quantificar. Mencionar as maiores escolhas e como imaginamos que elas delinearam os resultados. (E instigar o leitor à leitura do restante da tese!)

Palavras-chave: Engenharia. Engenharia da Computação. Inteligência Artificial. Arquitetura BDI. Sistema Blackboard. Jogos.

Abstract

Same text (*ibidem idem* previous section), but now in English.

Keywords: Engineering. Computer Engineering. Artificial Intelligence. BDI Architecture. Blackboard System. Games.

Lista de Figuras

Sumário

Capítulo 1

Introdução

Motivação e referências básicas do trabalho.

Quem são os orientadores, e como viemos a escolher o tema do trabalho. Falar aqui dos cursos de Computação Gráfica, Design e Programação de Games, SBGames.

Capítulo 2

Aspectos Conceituais

2.1 Inteligência Artificial

A Inteligência Artificial (IA) é uma área de pesquisa da ciência da computação e Engenharia da Computação, dedicada a buscar métodos ou dispositivos computacionais que possuam ou simulem a capacidade racional de resolver problemas, pensar ou, de forma ampla, ser inteligente.

Apenas recentemente, com o surgimento do computador moderno, é que a inteligência artificial ganhou meios e massa crítica para se estabelecer como ciência integral, com problemáticas e metodologias próprias. Desde então, seu desenvolvimento tem extrapolado os clássicos programas de xadrez ou de conversão e envolvido áreas como visão computacional, análise e síntese da voz, lógica difusa, redes neurais artificiais e muitas outras. Inicialmente a IA visava reproduzir o pensamento humano. A Inteligência Artificial abraçou a idéia de reproduzir faculdades humanas como criatividade, auto-aperfeiçoamento e uso da linguagem. Porém, o conceito de inteligência artificial é bastante difícil de se definir. Por essa razão, Inteligência Artificial foi (e continua sendo) uma noção que dispõe de múltiplas interpretações, não raro conflitantes ou circulares.

2.1.1 História

O primeiro trabalho a respeito de IA foi realizado por Warren McCulloch e Walter Pitts em 1943, em que demonstrava um modelo de neurônios artificiais, em que cada neurônio poderia estar “ligado” ou “desligado”, isso dependeria dos estados em que os neurônios vizinhos estariam. (RUSSEL & NORVIG, 2004: 18)

Donald Hebb demonstrou em 1949 uma regra de atualização simples para definir a intensidade de conexão entre neurônios, essa regra é influente ainda nos dias de hoje (RUSSEL & NORVIG, 2004: 18).

Em 1950 Alan Turing publicou o artigo *“Computing Machinery and Intelligency”*. Neste artigo ele apresentou diversos algoritmos de IA e seu famoso teste, o “Teste de Turing”. (Idem, 18).

Em 1951, no departamento de matemática de Princeton, Marvin Minsk e Dean Edmound apresentaram o primeiro computador de rede neural, o SNARC era composto por 3000 válvulas e simulava uma rede de 40 neurônios. (Op. Cit.,18)

Segundo RUSSEL & NORVIG (2004: 18-19), cinco anos após o SNARC John McCarthy realizaram um seminário em Dartmouth, reunindo todos os grandes pesquisadores do conhecimento. Nesse seminário Allen Newell e Herbert Simon do Carnegie Tech apresentaram o *Logic Theorist* o primeiro programa de raciocínio. Nesse seminário John McCarthy juntamente com os outros seminaristas percebeu que a IA devia ser um campo em separado e não uma ramificação da matemática nem mesmo estar sob o nome de pesquisa operacional ou teoria de controle, então eles definiram o nome de Inteligência Artificial e, após este seminário a IA foi definida como o campo em que tenta construir máquinas que funcionem de forma autônoma e em ambientes mutáveis.

Pesquisas sobre inteligência artificial foram intensamente custeadas na década de 1980 pela Agência de Projetos de Pesquisas Avançadas sobre Defesa (*"Defense Advanced Research Projects Agency"*), nos Estados Unidos, e pelo Projeto da Quinta Geração (*"Fifth Generation Project"*), no Japão. O trabalho subsidiado fracassou no sentido de produzir resultados imediatos, a despeito das promessas grandiosas de alguns praticantes de IA, o que levou proporcionalmente a grandes cortes de verbas de agências governamentais no final dos anos 80, e em conseqüência a um arrefecimento da atividade no setor, fase conhecida como O inverno da IA. No decorrer da década seguinte, muitos pesquisadores de IA mudaram para áreas relacionadas com metas mais modestas, tais como aprendizado de máquinas, robótica e visão computacional, muito embora pesquisas sobre IA pura continuaram em níveis reduzidos.

Os fatos acima listados marcam o nascimento e desenvolvimento da inteligência artificial, daquela época até os dias de hoje ocorreram grandes avanços neste ramo da computaçao.

Atualmente a IA está mais madura e não possui mais aquele ar de pioneirismo do inicio. O pesquisador de IA deve provar suas teorias com diversos experimentos empíricos e a demonstração dos resultados, e com o surgimento da Internet os pesquisadores foram capazes de compartilhar as evoluções de suas pesquisas e acelerar os processos de desenvolvimento.

Encorajados pela resolução dos subproblemas da IA, os pesquisadores começaram a examinar o problema do "agente como um todo". O movimento estabelecido tem como objetivo entender o funcionamento interno de agentes incorporados a ambientes reais com entradas sensoriais continuas. Um dos ambientes mais importantes para os agentes inteligentes é a Internet, diversas ferramentas utilizam a Inteligência Artificial na Internet como mecanismo de pesquisa, sistemas de recomendação e sistemas de construção de web sites. (RUSSEL & NORVIG, 2004:27-28) Na tentativa de se construir agentes percebeu-se

que subcampos que anteriormente eram isolados da IA, precisavam de uma reorganização para o melhor aproveitamento de seus resultados. Outra conseqüência foi que a IA se aproximou de outras áreas, como a teoria de controle e a economia, áreas que também lidam com agentes. (RUSSEL & NORVIG, 2004:28) No século XXI há um grande interesse em *Agentes Inteligentes* uma nova ramificação que pretende reunir todas as subáreas da IA, pois se percebeu que todas juntas poderiam criar um Agente perfeito realizando o principal objetivo da IA, entender o pensamento humano.

Entre os teóricos que estudam o que é possível fazer com a IA existe uma discussão onde se consideram duas propostas básicas: uma conhecida como “forte” e outra conhecida como “fraca”. A investigação em Inteligência Artificial Forte aborda a criação da forma de inteligência baseada em computador que consiga raciocinar e resolver problemas uma, sendo assim a forma de IA forte é classificada como autoconsciente. A Inteligência Artificial Fraca centra a sua investigação na criação de inteligência artificial que não é capaz de verdadeiramente raciocinar e resolver problemas. Uma tal máquina com esta característica de inteligência agiria como se fosse inteligente, mas não tem autoconsciência ou noção de si.

2.1.2 Áreas de aplicação

Enquanto que o progresso direcionado ao objetivo final de uma inteligência similar à humana tem sido lento, muitas derivações surgiram no processo. Exemplos notáveis incluem as linguagens LISP e Prolog, as quais foram desenvolvidas para pesquisa em IA, mas agora possuem funções não-IA. A cultura Hacker surgiu primeiramente em laboratórios de IA, em particular no MIT AI Lab, lar várias vezes de celebridades tais como McCarthy, Minsky, Seymour Papert (que desenvolveu a linguagem Logo), Terry Winograd (que abandonou IA depois de desenvolver SHRDLU). Muitos outros sistemas úteis têm sido construídos usando tecnologias que ao menos uma vez eram áreas ativas em pesquisa de IA. Alguns exemplos incluem:

- Planejamento automatizado e escalonamento: a uma centena de milhões de quilômetros da Terra, o programa Remote Agent da NASA se tornou o primeiro programa de planejamento automatizado (autônomo) de bordo a controlar o escalonamento de operações de uma nave espacial.

- Jogos: O Deep Blue da IBM se tornou o primeiro programa de computador a derrotar o campeão mundial em uma partida de xadrez, ao vencer Garry Kasparov.

- Controle autônomo: O sistema de visão de computador ALVINN foi treinado para dirigir um automóvel, mantendo-o na pista.
- Robótica: Muitos cirurgiões agora utilizam robôs assistentes em microcirurgias.
- Lógica incerta: Técnica para raciocinar dentro de incertezas, tem sido amplamento usada em sistemas de controles industriais.
- Redes Neurais: Vêm sendo usadas em uma larga variedade de tarefas, desde sistemas de detecção de intrusos a jogos de computadores.
- Aplicações utilizando Vida Artificial são utilizados na indústria de entretenimento e no desenvolvimento da Computação Gráfica.
- Sistemas baseados na idéia de agentes artificiais, denominados Sistemas Multiagentes, têm se tornado comuns para a resolução de problemas complexos.

Atualmente existem muitas aplicações práticas que envolvem conceitos de inteligência artificial. Dentre todas as áreas que, de alguma forma implementam conceitos de IA, os jogos eletrônicos tem ganhado grande destaque nos últimos anos.

2.2 Jogos digitais

Após pesados investimentos em novas tecnologias, os jogos digitais atuais possuem gráficos avançadíssimos que beiram a realidade, no entanto, pouco foi investido em seu conteúdo. Houve uma preocupação enorme por parte das empresas desenvolvedoras de jogos em acompanhar a evolução tecnológica dos últimos tempos, com isso nasceram novos consoles poderosos como o XBOX da Microsoft, o Playstation 3 da Sony e o Nintendo Wii, porém, pouco foi feito com relação ao conteúdo dos jogos. É muito fácil observar este fenômeno, basta notar que atualmente os jogos que mais são lançados são continuações de jogos antigos com gráficos mais bonitos. Os jogos que mais provam esta teoria são os RPGs (Role-Playing Game) e os jogos de aventura onde o jogador controla um personagem dentro de um universo e interage com outros jogadores ou NPCs. Apesar de toda qualidade gráfica oferecida por estes jogos, todos possuem uma lógica parecida, o usuário deverá realizar as mesmas tarefas para completar o jogo, no melhor caso existe mais de uma maneira de realizar uma mesma tarefa, no entanto, o jogo será sempre o mesmo, os NPCs sempre dirão as mesmas falas e farão as mesmas coisas não importa quantas vezes o usuário jogue. Um jogo que representa bem este fato é o jogo da Nintendo “Legend of Zelda: Ocarina of Time”, nele o jogador ontrola um personagem que vive uma

aventura para salvar uma princesa, na época em que foi lançado e até hoje o jogo ainda é um grande sucesso da Nintendo, no entanto, não importa quantas vezes seja jogado, a história é sempre a mesma, os diálogos com os NPCs são sempre os mesmos diálogos limitados, os poucos momentos em que o jogador pode "falar" com um NPC é para dizer "sim" ou "não" e em alguns casos é obrigado a escolher uma resposta pois a outra faz com que o NPC apenas repita a pergunta! Isso prejudica o jogo, pois uma vez que o usuário o termina, não existe mais a sensação do desafio a ser superado se ele jogar novamente, ou seja, a diversão associada ao jogo diminui. Talvez com um pouco de esforço e dedicação seja possível desenvolver jogos que não sejam tão simples e superficiais, onde os NPCs tenham "vontades" e possam dialogar com o usuário de forma mais parecida com a realidade humana. Outro ponto interessante seria se o NPC mudasse seu perfil toda vez que o jogador recomeçasse o jogo, tornando-o imprevisível.

2.3 Inteligência em jogos

O principal objetivo do uso de técnicas de inteligência artificial em jogos eletrônicos é a diversão. Por causa disso, o conceito de IA acabou recebendo outra interpretação por parte dos desenvolvedores de jogos, surgiu o conceito de *Game AI*.

Diferentemente da IA acadêmica que busca solucionar problemas extramamente difíceis, como imitar o reconhecimento que os humanos são capazes de realizar (reconhecimento facial e de imagens e objetos, por exemplo), ou mesmo entender e construir agentes inteligentes, a IA para jogos se preocupa com os resultados que o sistema irá gerar, e não como o sistema chega até os resultados. Isso se deve ao fato que jogos eletrônicos são negócios e os consumidores desses produtos os compram em busca de diversão, e não lhes interessa como a inteligência de um personagem no jogo foi criada, desde que ela transforme o jogo divertido e desafiador, além, claro, de tomar decisões coerentes com o contexto do jogo.

Um exemplo que deixa claro a utilidade da IA para jogos são os *shooters*. Nos jogos de tiros existem várias técnicas de IA que podem ser aplicadas, por exemplo, é possível fazer com que os NPCs se comuniquem e criem estratégias para tentar cercar um inimigo, fato que pode tornar o jogo ainda mais interessante, no entanto, também é possível utilizar técnicas de IA para fazer com que os NPCs acertem todos os disparos na cabeça de seus inimigos, fato que passa longe da realidade humana, e que poderia prejudicar a qualidade do jogo do ponto de vista de um usuário.

Ano	Descrição	IA utilizada
1962	Primeiro jogo de computador, *Spacewar*, para 2 jogadores.	Nenhuma
1972	Lançamento do jogo *Pong*, para 2 jogadores.	Nenhuma
1974	Jogadores tinham que atirar em alvos móveis em *Pursuit* e *Qwak*.	Padrões de movimento
1975	*Gun Fight* lançado, personagens com movimentos aleatórios.	Padrões de movimento
1978	*Space Invaders* contém inimigos com movimentos padronizados, mas também atiram contra o jogador.	Padrões de movimento
1980	O jogo *Pac-man* conta com movimentos padronizados dos inimigos, porém cada fantasma (inimigo) tem uma "personalidade" sobre o modo em que caça o jogador.	Padrões de movimento
1990	O primeiro jogo de estratégia em tempo real, *Herzog Wei*, é lançado. Junto, os jogadores puderam noticiar uma péssima busca de caminho.	Máquina de estados
1993	*Doom* é lançado como primeiro jogo de tiro em primeira pessoa.	Máquina de estados
1996	*BattleCruiser: 3000AD* é publicado como o primeiro jogo a utilizar redes neurais em um jogo comercial	Redes neurais
1998	*Half-Life* é lançado e analisado como a melhor IA em jogos até a época, porém, o jogo utiliza IA baseada em *scripts*.	Máquina de estados / *Script*
2001	O jogo *Black & White* é alvo da mídia a respeito de como as criaturas do jogo aprendem com as decisões feitas pelo jogador. Utiliza redes neurais, *reinforcement* e *observational learning*.	Diversos

Figura 2.1: Evolução da aplicação de IA em jogos

No começo do desenvolvimento de jogos eletrônicos, a programação de IA era mais usualmente conhecida por *programação de jogabilidade*, pois não havia nada de inteligente sobre os comportamentos exibidos pelos personagens controlados pelo computador. A figura 2.1 contém alguns exemplos de como a IA foi utilizada em jogos com o passar do tempo. **(figura IA jogos (SCHWAB, 2004) SCHWAB, Brian. AI Game Engine Programming. Hingham: Charles River Media. 2004.)**

2.4 Desenvolvimento de jogos

2.4.1 Crença, desejo e intenção — a arquitetura BDI

O modelo BDI (*Beliefs Desires Intentions*) foi originalmente proposto por Bratman como uma teoria filosófica do raciocínio prático, propondo uma análise do comportamento humano que seria baseado em crenças, desejos e intenções. Basicamente supõe-se que as ações são derivadas a partir de um processo chamado raciocínio prático. Este processo é constituído por duas etapas, na primeira, deliberação, o agente seleciona um conjunto de desejos que devem ser alcançados, de acordo com a situação atual das crenças do mesmo. Na segunda etapa ocorre a determinação de como os desejos produzidos no

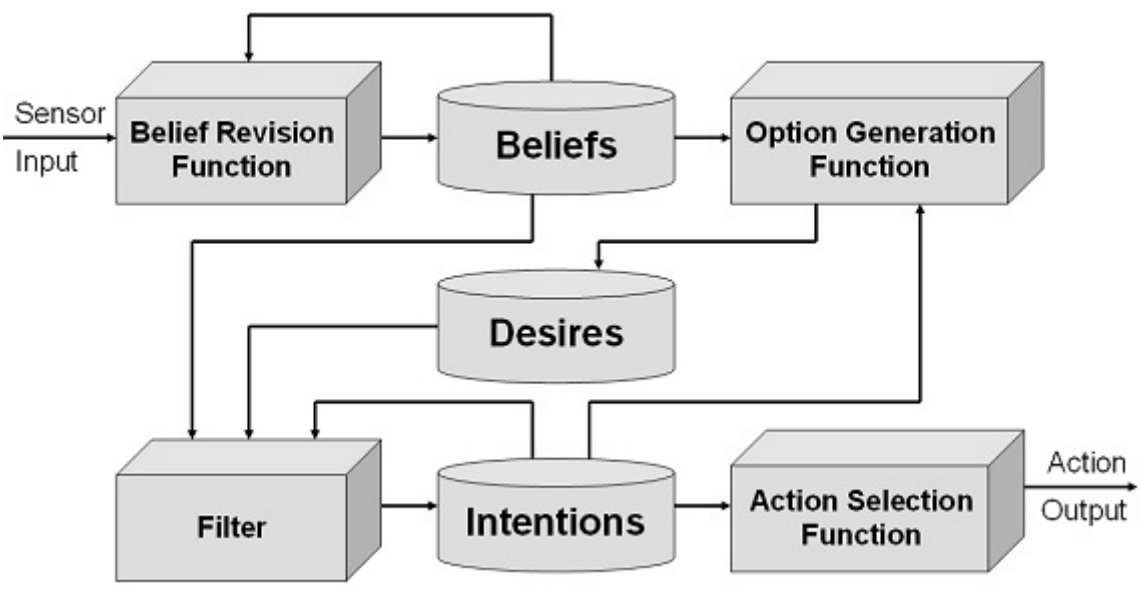

Figura 2.2: Diagrama de uma arquitetura BDI genérica [4]

passo anterior podem ser atingidos através do uso dos meios disponíveis ao agente **[4] Michael Wooldridge, Reasoning about Rational Agents. Cambridge, MA: The M. I. T. Presss 2000.**. A seguir explicaremos melhor o que são crenças, desejos e intenções.

- **Crenças (Beliefs)**: Representam as características do ambiente e são atualizadas constantemente. Podem ser vistas como a componente informativa do sistema.
- **Desejos (Desires)**: Representam os objetivos a serem alcançados. Podem ser vistos como motivações do sistema.
- **Intenções (Intentions)**: Representam o atual plano de ações escolhido.

A partir deste modelo nasceu a arquitetura BDI para agentes. Agentes BDI são sistemas localizados em um ambiente (em nosso caso o ambiente será virtual) sujeito a variações, além disso, percebem o estado deste ambiente constantemente e podem atuar sobre o mesmo para tentar alterar o estado atual. Abaixo vemos o processo de raciocínio prático de um agente BDI.

Como se pode observar na figura 2.2, existem sete elementos básicos que compõem um agente BDI, são eles **[5] Ingrid Oliveira de Nunes, Implementação do modelo e da arquitetura BDI**:

- Um conjunto de crenças (*Desires*) atuais que representam as informações que o agente tem do ambiente.

- Uma função de revisão de crenças (*Belief Revision Function*), a qual determina um novo conjunto de crenças a partir da percepção da entrada e das crenças do agente.
- Uma função de geração de opções (*Option Generation Function*), a qual determina as opções disponíveis ao agente (seus desejos), com base nas suas crenças sobre seu ambiente e nas suas intenções.
- Um conjunto de opções (*desires*) corrente que representa os possíveis planos de ações disponíveis ao agente.
- Uma função de filtro (*filter*), a qual representa o processo de deliberação do agente, que determina as intenções do agente com base nas suas crenças, desejos e intenções atuais.
- Um conjunto de intenções (*Intentions*) atual, que representa o foco atual do agente, isto é, aqueles estados que o agente está determinado a alcançar.
- Uma função de seleção de ação (*Action Selection Function*), a qual determina uma ação a ser executada com base nas suas intenções atuais.

2.4.2 Quadro-negro — blackboard

A forma mais simples de apresentar o conceito de blackboard é através de uma metáfora que propõem a seguinte situação **(Daniel D. Corkill, Blackboard Systems. Blackboard Technology Group, Inc.)**: “Imagine um grupo de cientistas reunidos em uma sala trabalhando de forma cooperativa para resolver um problema. Para chegar a uma solução é utilizado um quadro negro (*blackboard*). A resolução do problema começa quando o mesmo é escrito no quadro negro juntamente com informações iniciais. Os cientistas verificam o conteúdo do quadro e aguardam uma oportunidade para aplicar seus conhecimentos visando solucionar o problema. Quando um cientista encontra informações suficientes para fazer uma contribuição, o mesmo coloca sua contribuição no quadro negro, e com sorte, este processo ativará outro cientista e assim por diante até chegarem à solução do problema.” Da metáfora acima se pode concluir que um blackboard possui uma base de dados comum a diversos sistemas que estão tentando solucionar um problema e utilizam e atualizam as informações existentes nesta base de dados. A seguir apresentaremos algumas características desta técnica:

- **Independência de conhecimento**: No exemplo acima, supomos que cada cientista adquiriu seus conhecimentos de maneira independente dos outros, ou seja, num

blackboard cada sistema que participa da solução de um problema tem seus próprios níveis de conhecimento, independentemente dos outros sistemas.

- **Diversidade nas técnicas de solução de problemas**: Uma das grandes vantagens do blackboard é que não interessa como os sistemas que estão trabalhando na resolução do problema funcionam, ou seja, um sistema pode utilizar redes neurais enquanto outro pode utilizar simulações, que para o blackboard estas "fontes de conhecimento" são caixas pretas que fazem suas contribuições.

- **Representação da informação é flexível**: Não existe nenhuma definição de como deve ser a informação, dando liberdade a quem implementa de definir como representar os dados.

- **Linguagem comum entre os sistemas**: Ao mesmo tempo em que existe a flexibilidade na representação da informação, é necessário, por outro lado, que todos os sistemas envolvidos sejam capazes de entender tais informações.

- **Liberdade de organização dos dados**: A organização dos dados é livre, porém, deve ser feita de tal forma que, no caso da base de dados possuir muita informação, seja fácil para um sistema encontrar informações específicas de forma rápida e simples.

- **Ativações baseadas em eventos**: Os sistemas que estão trabalhando na solução do problema não se comunicam diretamente, ao invés disto, todos "observam" a base de dados comum e utilizando as informações da mesma buscam gerar novos dados que são colocados na mesma base, tais dados podem ativar outro sistema que fará a mesma coisa até que o problema seja completamente resolvido.

- **Necessidade de controle**: É preciso haver um "órgão" para controlar todos os sistemas e decidir quem deve e quem não deve poder alterar os dados do quadro negro, assim não existe o risco de mais de um sistema alterar uma mesma área de dados simultaneamente.

- **Geração de solução incremental**: É evidente que a solução de um problema acontece de passo em passo, ou seja, um sistema faz uma contribuição com novos dados, a seguir, outro sistema utiliza tais dados e faz uma nova contribuição e de ciclo em ciclo o problema tende a ser resolvido.

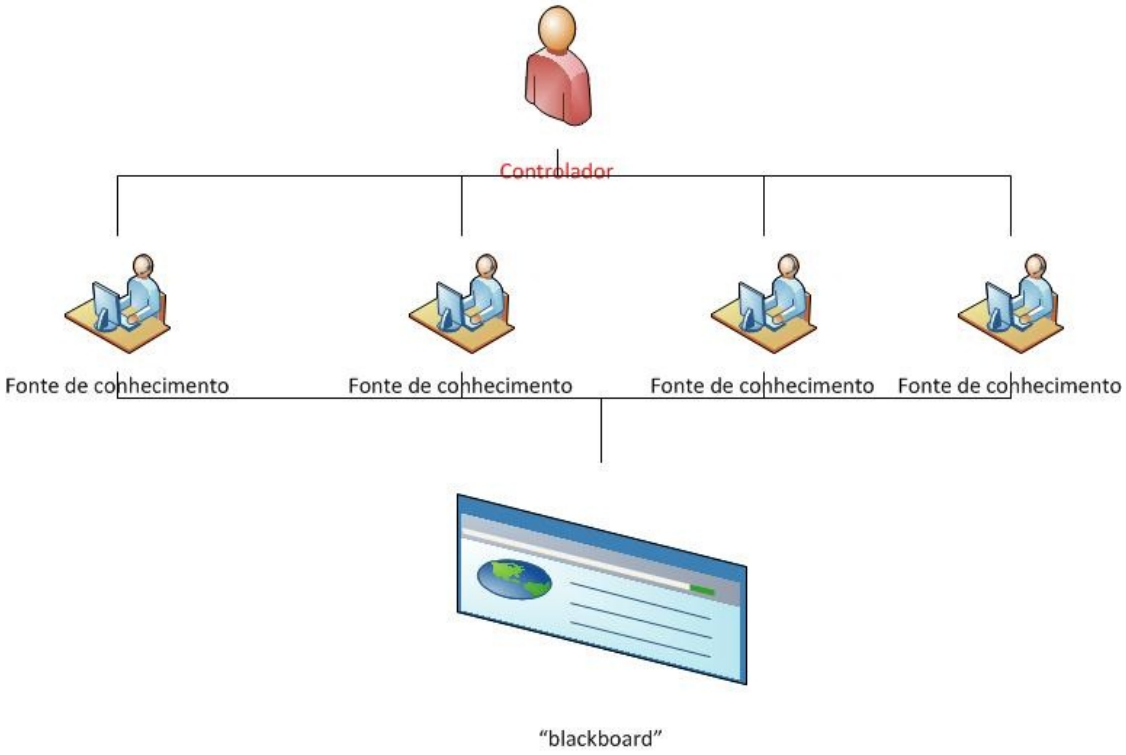

Figura 2.3: Visão geral do blackboard

Na figura 2.3 temos uma visão geral de como funciona o blackboard. Basicamente existem fontes de conhecimento (sistemas que estão trabalhando na resolução do problema) um sistema de controle que concede permissões às fontes de conhecimento para alterarem os dados do quadro negro, este por sua vez, é uma base de dados que contem informações que ficam disponíveis a todas as fontes de conhecimento.

Neste projeto utilizaremos a técnica blackboard como parte de um jogo computacional que será desenvolvido.

2.4.3 O que avaliar

Aspectos humanos

Embora o escopo deste trabalho esteja mais próximo de uma avaliação de viabilidade do que de uma análise comparativa de desempenho propriamente dita, vale ressaltar que há espaço, e mesmo a necessidade, de que se estude o impacto da aplicação das técnicas de IA (alvo deste estudo) na experiência de jogo resultante.

Assim, é importante que se tenha em mente que trabalhamos sobre a hipótese de que o campo da inteligência artificial tem a contribuir positivamente na adaptabilidade dos próprios roteiros e trama das histórias que permeiam jogos — visto que sua aplicação já

se dá efetivamente em outros contextos. A expectativa é obter um comportamento mais adaptado ao contexto e enredo em relação que se alcança hoje com máquinas de estado, sem sacrificar o controle que o designer tem.

Aspectos técnicos e de projeto

Existe um aumento na complexidade dos diálogos, em razão da necessidade de incorporar mais informações de contexto e variantes no jogo. Isso se traduz na composição de um número maior de derivações em diálogos, já que tanto o jogador quanto os NPCs passam a ter opções, e também a necessidade de projeto dos agentes em si.

Aspectos gerais

É interessante que se avalie o processo de adaptação das técnicas empregadas ao estudo feito, e se elas podem ser aplicadas a outros domínios. Em particular, o que jogos de diferentes gêneros podem ganhar com o uso da engenharia de software voltada a agentes e de técnicas como por exemplo, blackboard.

Capítulo 3

Especificação

Neste capítulo abordam-se os requisitos que o projeto deve satisfazer para que o estudo de caso seja efetivo. Foi dada atenção particular à aderência do projeto a padrões industriais de metodologia de desenvolvimento de jogos, na medida em que tal não se mostrasse incompatível com o escopo de um projeto de formatura.

Assim optou-se pelo emprego do GDD (Game Design Document — apêndice B) como meio de especificação e captura de requisitos do projeto. Da mesma forma foram tomadas decisões como a escolha das linguagens e bibliotecas empregadas, tendo em vista padrões recorrentes na indústria e portabilidade.

3.1 Decisões de projeto

3.1.1 Limitações e controle da expressão dos agentes

Apesar de variabilidade e adaptabilidade da experiência do jogo constituírem, em grande medida, o que se busca ao introduzir-se algum modelo de inteligência na lógica de jogos, é preciso ter em mente que isso não significa que se queira abrir mão do design dessa experiência. De fato, parte da complexidade da incorporação de inteligência artificial a jogos advém da necessidade de identificar que aspectos da dinâmica da inteligência devem ser coergidos, moldados ou mesmo eliminados, em prol de atributos como a jogabilidade e mesmo adaptação às limitações e anseios dos jogadores.

Nesse sentido, optou-se por um modelo — experimental e por vezes potencialmente limitante, como será visto adiante — restritivo de expressão dos agentes que controlam o diálogo com NPCs.

O modelo implementado tem por ideia central que os diálogos são interações de *estímulo e reação* do agente inteligente pelo jogador. Os estímulos estão associados ao que o jogador opta por dizer, e as reações às opções de fala do NPC.

As opções de um e outro estão registradas nos *roteiros de diálogo*, que são compostos pelos designers do jogo. Nesses roteiros, as falas de jogador e agente estão descritas, e tanto os estímulos que transmitem quanto as reações que expressam são definidos por uma *marcação de conteúdo*. Assim, uma saudação pode ser um estímulo amigável; uma afirmação pode soar suspeita; e, por outro lado, uma interjeição pode expressar contentamento ou rabugice.

Desse modo, o conjunto de condições passíveis de expressão pelos agentes fica sob controle do designer do jogo, por causa da estreita ligação entre os diálogos e os estímulos e manifestações disponíveis para os agentes. Essa é uma característica extremamente desejável, como já foi dito, por permitir introduzir a variabilidade de maneira controlada e em sintonia com a experiência que se quer produzir.

Existem algumas dificuldades inerentes a esse processo de modelagem. Está sujeito a avaliação se em projetos de maior porte a tarefa de gerência dos vários estímulos e reações não se tornaria proibitiva do uso da técnica.

3.1.2 Licença e implicações

Desde o princípio fez-se a opção por desenvolver um projeto que se adaptasse à filosofia dos softwares livres, de código aberto. Essa característica teve consequências importantes e decisivas no encaminhamento do projeto durante toda a sua evolução.

Apesar da severidade que as restrições impostas por uma licensa livre ao projeto, deve-se frisar que essa escolha se apresentou certa para o grupo. Não só é inegável o caráter inerentemente livre da produção acadêmica de universidades públicas, meio em que se insere este projeto, como é preciso que se diga que há uma janela de oportunidade na atual carência, no universo dos softwares livres, de projetos de jogos. Essa carência, note-se, vem sendo paulatinamente alvo de estudos e experimentações no mercado, fato que se observa pela progressão do número de títulos que vêm sendo portados para sistemas livres.

Há duas consequências de grande impacto prático na condução do projeto de um jogo livre, a saber a restrição no uso de software de terceiros e de conteúdo midiático que pode ser aproveitado. Como será discutido adiante (seção 5.2), essa opção reduziu a gama de arcabouços de tecnologia que puderam ser aproveitados, e impôs ao projeto a responsabilidade pela criação de parte da arte do jogo — uma carga de trabalho não desprezível dado escopo e recursos disponíveis.

3.1.3 Tecnologias empregadas

Uma das primeiras decisões que foram tomadas na definição das condições de contorno do projeto foi que a linguagem empregada para desenvolvimento do software deveria refletir algum padrão solidamente estabelecido na indústria. Optou-se por C++, uma linguagem que goza de grande popularidade na idústria com títulos conhecidos como *Doom* e *Age of Empires*.

Outra decisão de impacto foi a escolha de engines. À partir de uma primeira lista, compilada por meio de pesquisas em fóruns de discussão sobre desenvolvimento de jogos, construiu-se uma matriz de decisão em que foram eliminadas as opções que não apresentavam licensas livres, portáveis e que não possuíssem interface com C++. Por fim escolheu-se empregar o Simple DirectMedia Layer (SDL), linguagem de interface portável com dispositivos de media e periféricos[1].

O uso da ferramenta Jason, para a criação dos agentes, se deve principalmente ao fato de que o projeto nasceu motivado pela tese de doutorado [**?**] de um dos orientadores do grupo que, no desenvolvimento de seu projeto, utilizou a ferramenta. Além disso, como esta ferramenta é baseada em Java, parte do projeto foi desenvolvida utilizando essa linguagem como se pode observar na figura **??**.

3.2 Especificação dos requisitos do software

Conforme o modelo de desenvolvimento adotado pela indústria, os requisitos do software foram levantados a partir do GDD e das decisões de projeto. Vale ressaltar que no caso do desenvolvimento de jogos, o documento de consulta primário é o GDD.

É importante citar que a grande maioria dos requisitos funcionais do software são regras do jogo e estão descritos no documento de game design. Dentre os outros requisitos funcionais temos:

- O software deve fazer e administrar a comunicação entre os módulos implementados em C++ e Java.
- O diálogo entre o usuário e um NPC não pode parecer determinístico.

Os requisitos não-funcionais levantados foram os seguintes:

- A comunicação entre os módulos implementados em Java e C++ não deve demorar mais que 3 segundos.
- As tecnologias envolvidas no projeto devem ser livres.
- A interface homem-máquina deve ser simples e intuitiva
- O tempo de processamento de dados no blackboard não pode ser superior a 5 segundos.

[1] Como um adendo, citamos dois títulos conhecidos que usam SDL em pelo menos uma versão: *World of Goo* e *Doom 3*.

Capítulo 4

Metodologia

O sucesso de um projeto depende em grande medida da capacidade de adaptação dos envolvidos. Nesta seção tratamos do método de planejamento de atividades do trabalho, e de sua evolução no decorrer do projeto. Assim, em certa medida trataremos das mudanças que se operaram nas expectativas e previsões que fizemos ao longo do tempo.

4.1 Evolução

¡¡¡¡¡¡¡ .mine

¡¡¡¡¡¡¡ .mine Nesta seção discorre-se brevemente sobre as atividades desenvolvidas durante o projeto, desde sua especificação às etapas finais de sua implementação.

A ideia de desenvolvimento de um projeto no campo de jogos nasceu há mais de um ano, e diversos assuntos foram elencados como possíveis temas para o projeto de conclusão de curso. Mas foi somente em março de 2010 que a equipe se consolidou e o professor doutor Ricardo Nakamura aceitou orientar o grupo, dando início ao processo de refinamento do tema do trabalho. Foi então que entramos em contato com o professor doutor Roberto Cezar Bianchini, co-orientador do projeto, e autor da tese de doutoramento [**?**] que sugeriu o tema do presente estudo de caso.

As primeiras atividades do grupo então concentraram-se na elaboração de um primeiro cronograma, e foram selecionados assuntos a pesquisar. Nesse ponto foi iniciada a composição de um diário de bordo, em que se registrou todo o avanço do projeto.

O passo seguinte foi solidificar as condições de contorno do problema, por meio de reuniões em que se procurou unificar a visão dos envolvidos, progredindo gradualmente rumo a uma concepção amadurecida dos objetivos e métodos a serem empregados no projeto.

Após diversas reuniões, o grupo optou por utilizar uma linguagem de presença expressiva na idústria: C++ (conforme justificado na seção 3.1.3); além disso, estipulou-se que o projeto teria uma licença livre.

Com relação ao jogo, o grupo optou por desenvolver um jogo em 2D, esta escolha foi feita para simplificar o desenvolvimento, uma vez que a parte gráfica do jogo não era o foco do projeto. O grupo procurou simuladores de sistemas multi-agentes que trabalhassem

com a arquitetura BDI. Por fim, o grupo buscou listar os requisitos de engines e softwares de renderização que viria a utilizar.

O grupo então definiu que a parte gráfica do jogo seria implementada usando SDL. Além disso, o grupo decidiu trabalhar com a ferrametna Jason para o desenvolvimento de sistemas multi-agentes, fato que implicou diretamente na escolha da linguagem Java como auxiliar no desenvolvimento do projeto. Após esta decisão, nasceu a necessidade de se criar uma forma de comunicação entre as partes do jogo que foram desenvolvidas em Java e em C++, a solução escolhida foi fazer comunicação usando TCP via loop-back (foram analisadas outras soluções, porém nenhuma delas eram tão simples quanto a solução adotada). Após esta etapa a arquitetura do software começou a ficar mais transparente aos envolvidos, isto foi importante pois permitiu que o trabalho fosse dividido de forma eficiente entre o integrantes do grupo.

A seguir, o grupo apresentou aos orientadores uma proposta de um jogo que já envolvia os conceitos que foram abordados no projeto. Após algums reuniões sobre a proposta surgiram os primeiros rascunhos do documento de design do jogo.

Com a consolidação da proposta, o grupo efetuou uma primeira divisão de responsabilidades de estudos, particionando os assuntos em escopos de conhecimento: tecnológico (C++, BDI, Java, AgentSpeak, SDL, e arquiteturas de software empregadas em jogos). Ficou definido que toda a lógica do jogo e a parte gráfica seriam implementadas em C++, enquanto os agentes e o blackboard seriam desenvolvidos em AgentSpeak e Java respectivamente.

Esse estudo acompanhou o desenvolvimento do projeto até estágios avaçados, mas após alguns meses desde seu início preparou-se um cronograma mais detalhado, adiantado propositadamente de modo que as atividades previstas se encerrassem com um mês de antecedência em relação às datas oficiais de entregas e apresentações do projeto, de modo a acomodar sem transtornos eventuais imprevistos — que, adianta-se, ocorreram.

Esse cronograma destacava as atividades que então foram consideradas essenciais

- Formação da equipe
- Escolha do orientador
- Testes com as tecnologias
- Escolha das tecnologias
- Estudo das tecnologias
- Integração das tecnologias

- Game Design
- Especificação dos requisitos
- Protótipo do software
- Testes do protótipo
- Validação

A composição do GDD iniciou em fins de abril, baseando-se em previas e estudos de conceito do jogo que haviam sido armazenados no decorrer do semestre, e estendeu-se até meados de agosto. Este segundo cronograma estimava já o tempo de implementação dos componentes da arquitetura do sistema — como era concebido então — além de estimar o tempo que a equipe levaria para encontrar conteúdo artístico com licensa compatível com a do projeto e, paralelamente, modelar os agentes do sistema.

Já no segundo semestre o grupo reestruturou o cronograma criando pontos de avaliação do progresso do desenvolvimento (*milestones*), quando então ocorriam encontros com os orientadores em que se apresentava o progresso, comparavam-se as metas previstas e alcançadas, faziam-se novas estimativas para conclusão do que não se havia cumprido, e discutiam-se os problemas encontrados e vislumbrados para o futuro. As milestones estão listadas abaixo. Outras alterações de impacto na visão da equipe sobre as etapas por futuras do projeto foi o acerto de uma divisão de tarefas mais acirrada nas etapas finais, concentrando-se um e outro em áreas que bem conheciam. Desse modo, Murilo ficou responsável pela parcela do projecto associada à inteligêngia e modelagem de agentes, enquanto Tássio trabalhou com a lógica e renderização do jogo. A elaboração do protocolo de comunicação entre os agentes e a lógica do jogo foi feita em conjunto, assim como a elaboração dos diálogos.

4.1.1 Primeira milestone (30 setembro)

- pesquisa de coleções de arte (gráficos e som) que serão usadas no trabalho
- porte do projeto para o Subversion
- solução para comunicação entre processos Java e C++ do jogo
- elaboração de tabelas para modelagem de características de personagens do jogo
- mecanismos de escrita e leitura do quadro-negro (blackboard)

- roteiro de aceitação do projeto
- capítulos introdutórios e de especificação da monografia

Das atividades listadas na primeira milestone, apenas os mecanismos de escrita e leitura no blackboard e o roteiro de aceitação do projeto não respeitaram a data proposta. Isso se deveu ao fato de que ainda havia a necessidade de aumentar o nível de detalhamento nas especificações do jogo e da arquitetura do software como um todo. Por outro lado, o grupo foi capaz de adiantar algumas atividades definidas nas outras milestones, tais como a modelagem de agentes mais simples e subsistemas da lógica do jogo.

4.1.2 Segunda milestone (15 outubro)

- navegação pelos modos e telas do jogo funcional
- quadro-negro (blackboard) com um tipo de conhecimento compartilhado
- modelagem de alguns agentes mais simples
- módulos de menus e diálogos
- capítulos de sobre decisões de projeto da monografia

Por essa data o projeto já contava com testes funcionais de controle de todos os subsistemas que o jogo precisaria, e já havia sido alcançada a integração entre áudio e vídeo, documentada em pequenos demos (tópico da terceira milestone). Nesse ponto, a equipe já tomou consciência de que a complexidade de alguns tópicos havia sido superestimada, enquanto a de outros subestimada. O grupo cogitava então participar do IX Simpósio Brasileiro de Games (SBGames).

4.1.3 Terceira milestone (25 outubro)

- integração de áudio e gráficos
- terminada modelagem de todos os agentes do jogo
- quadro-negro compartilhando conhecimento de classes diversas
- testes de integração do sistema
- capítulos sobre projeto dos testes, resultados e conclusões da monografia

À terceira milestone o projeto havia atingido um estágio delicado. Boa parte do conhecimento necessário para sua conclusão já estava cominado pela equipe, mas algumas decisões importantes para a estruturação do e funcionamento do projeto estavam pendentes, necessitando de ajustes. O grupo por esta época já havia decidito participar do SBGames, como ouvinte (a princípio havia-se cogitado expor um artigo curto sobre o trabalho de conclusão de curso no evento).

4.1.4 Quarta milestone (5 novembro)

- monografia completa
- elaboração de roteiro de apresentação e material (capturas de tela, videos, etc.)

Infelizmente um dos integrantes teve que cancelar sua ida ao SBGames de última hora, em razão de seu estágio. De todo modo, a apresentação de milestones ficou suspensa durante uma semana — mesmo porque um de nossos orientadores estava participando do simpósio.

=======

4.1.5 Segunda milestone (15 outubro)

- navegação pelos modos e telas do jogo funcional
- quadro-negro (blackboard) com um tipo de conhecimento compartilhado
- modelagem de alguns agentes mais simples
- módulos de menus e diálogos
- capítulos de sobre decisões de projeto da monografia

Por essa data o projeto já contava com testes funcionais de controle de todos os subsistemas que o jogo precisaria, e já havia sido alcançada a integração entre áudio e vídeo, documentada em pequenos demos (tópico da terceira milestone). Nesse ponto, a equipe já tomou consciência de que a complexidade de alguns tópicos havia sido superestimada, enquanto a de outros subestimada. O grupo cogitava então participar do IX Simpósio Brasileiro de Games (SBGames).

4.1.6 Terceira milestone (25 outubro)

- integração de áudio e gráficos
- terminada modelagem de todos os agentes do jogo
- quadro-negro compartilhando conhecimento de classes diversas
- testes de integração do sistema
- capítulos sobre projeto dos testes, resultados e conclusões da monografia

À terceira milestone o projeto havia atingido um estágio delicado. Boa parte do conhecimento necessário para sua conclusão já estava cominado pela equipe, mas algumas decisões importantes para a estruturação do e funcionamento do projeto estavam pendentes, necessitando de ajustes. O grupo por esta época já havia decidito participar do SBGames, como ouvinte (a princípio havia-se cogitado expor um artigo curto sobre o trabalho de conclusão de curso no evento).

4.1.7 Quarta milestone (5 novembro)

- monografia completa
- elaboração de roteiro de apresentação e material (capturas de tela, videos, etc.)

Infelizmente um dos integrantes teve que cancelar sua ida ao SBGames de última hora, em razão de seu estágio. De todo modo, a apresentação de milestones ficou suspensa durante uma semana — mesmo porque um de nossos orientadores estava participando do simpósio.

¿¿¿¿¿¿¿ .r109

4.1.8 Percalços e reajustes

Um fato que prejudicou o andamento do projeto foi que ambos os integrantes do grupo começaram a estagiar. Inicialmente este fato não parecia influenciar no desempenho dos integrantes, no entanto, conforme os estágios começaram a exigir mais tempo e dedicação o andamento do desenvolvimento diminuiu. Uma parte do desenvolvimento que realmente surpreendeu o grupo foi a criação dos diálogos. O grupo sempre acreditou que a criação dos arquivos de diálogos seria uma atividade demorada, no entanto, esta atividade foi muito mais complexa do que esperado, isso se deve ao fato de que cada arquivo de diálogo

deve conter opções de falas tanto para o jogador quanto para os NPCs, sendo assim, cada diálogo do jogo, por mais simples que fosse, virou um arquivo de texto muito extenso. Um fato que atrasou um pouco o desenvolvimento do jogo foi a falta de conhecimento sobre os temas, os alunos ainda não tinham nem um contato com os conceitos de IA nem sobre programação orientada a agentes. Foram necessárias horas de estudos sobre os assuntos antes dos integrantes do grupo conseguirem iniciar o desenvolvimento.

¡¡¡¡¡¡¡ .mine

4.1.9 Considerações sobre aplicação da metodologia

É a percepção do grupo que houve uma adaptação das esperanças e estratégias de mobilização, planejamento, discussão e trabalho à medida que o projeto amadureceu, e adiquiriu-se maior confiança do que se poderia alcançar. Calibramos, aos poucos, a medida da incerteza de nossas previsões, e, embora seja inegável que há muito o que aprimorar, vislumbrou-se evolução considerável na habilidade de cada membro da equipe de lidar com imprevistos e incertezas, o que se mostrou crucial para que o término exitoso do projeto fosse alcançado.

É importante destacar o papel que a aquisição de conhecimento no domínio do projeto — e com isso referimo-nos tanto ao estudo técnico quanto ao conhecimento adquirido pelo contato com pessoas do ramo — na mudança da própria maneira e enfoque do projetar e compreender o projeto do jogo. Nesse contexto é notável o papel que desempenhou o curso da disciplina Design e Programação de Games no segundo semestre de 2010, assim como a participação no SBGames, nos quais adiquiriu-se um conhecimento que, de certo modo, não está acessível por meio das vias consagradas de estudo; um conhecimento vivo, orgânico, da esfera cultural que é parte inerente dos jogos neste tempo. Complexidade da modelagem de agentes (?) ======= ¿¿¿¿¿¿¿ .r109

4.1.10 Considerações sobre aplicação da metodologia

É a percepção do grupo que houve uma adaptação das esperanças e estratégias de mobilização, planejamento, discussão e trabalho à medida que o projeto amadureceu, e adiquiriu-se maior confiança do que se poderia alcançar. Calibramos, aos poucos, a medida da incerteza de nossas previsões, e, embora seja inegável que há muito o que aprimorar, vislumbrou-se evolução considerável na habilidade de cada membro da equipe de lidar com imprevistos e incertezas, o que se mostrou crucial para que o término exitoso do projeto fosse alcançado.

É importante destacar o papel que a aquisição de conhecimento no domínio do projeto — e com isso referimo-nos tanto ao estudo técnico quanto ao conhecimento adquirido pelo contato com pessoas do ramo — na mudança da própria maneira e enfoque do projetar e compreender o projeto do jogo. Nesse contexto é notável o papel que desempenhou o curso da disciplina Design e Programação de Games no segundo semestre de 2010, assim como a participação no SBGames, nos quais adiquiriu-se um conhecimento que, de certo modo, não está acessível por meio das vias consagradas de estudo; um conhecimento vivo, orgânico, da esfera cultural que é parte inerente dos jogos neste tempo.

Capítulo 5

Implementação

5.1 Visão global

Como as coisas funcionam. O que faz o quê (macroscopicamente).

Uma explicação sucinta da estrutura do programa, como ele inicia (forking) e como age em regime (responsabilidades java vs C++, e o protocolo).

5.2 Compromissos (*trade-offs*)

5.3 Arquitetura

5.4 Protocolo de comunicação

5.5 Estrutura de pastas

O projeto foi desenvolvido com vistas a se prestar à experimentação no design. Assim, algumas ferramentas para a escrita de *scripts* de diálogos ou pequenas cenas foram desenvolvidas.

Descrevemos a seguir a organização dos arquivos do projeto. É importante que essa estrutura seja clara e intuitiva, já que é nossa intenção que alguns desses arquivos (que configuram o comportamento do jogo) sejam alterados primariamente por designers do jogo. Assim, é preciso que haja critério na complexidade que se expõe a esses “usuários”.

5.6 Arquivos auxiliares

Uma série de arquivos de texto auxiliares são usados pelo jogo. Eles especificam

- diálogos,
- agentes,
- estados de agentes,
- tipos de estímulos a agentes,

Explicamos a seguir a função de cada um dos tipos de arquivo.

Os *diálogos* são scripts que codificam possíveis dizeres que NPCs ou o jogador podem optar por dizer. Carregam informação sobre o sentimento que expressam ou a impressão que causam. Seguem a gramática descrita no apêndice A.

Agentes são especificados em *AgentSpeak*, a linguagem interpretada pela ferramenta *Jason*. Cada arquivo contém a modelagem de um tipo de agente.

Os três últimos itens são arquivos de interface, usados para leitura pelos processos de renderização e de interpretação do sistema BDI. Optou-se por essa solução porque a comunicação se dá pelo envio de números, que, nesse caso, indicam qual a linha do arquivo de interface em que a mensagem está contida. Obviamente, essa abordagem só é possível porque as mensagens são conhecidas *a priori*.

Para comunicar o estímulo que uma determinada fala no diálogo provoca no agente, a lógica do jogo envia ao sistema BDI um inteiro indicando a linha em que está escrita a string que define o estado. Na prática, a lógica do jogo faz uma chamada para enviar o inteiro correspondente ao estímulo que se deseja comunicar, e uma busca é feita no arquivo que contém os estímulos possíveis para encontrar o número da linha correspondente. Esse número é, por fim, enviado ao sistema BDI, que fará a sua própria busca para identificar qual o estímulo recebido.

No caso de funções, o arquivo de interface deve ser gerado automaticamente pelo código Java, para que seja consultado durante a execução pela lógica do jogo.

Capítulo 6

Testes

Os testes foram concebidos de modo a englobar pequenas histórias de uso, de um lado, e funcionalidades, de outro.

6.1 Diálogos

Mecanismo de exibição de diálogo, *switches*.

6.2 Navegação pelo jogo

6.3 Carga e salvamento de progresso

6.4 Cenas

6.5 BDI em ação

6.6 Inferência pelo Blackboard

Capítulo 7

Conclusão

Sucesso do estudo de caso?

7.1 Trabalhos futuros

Próximo SBGames?

7.2 *Post Mortem*

7.2.1 *"Lessons learned"*

Referências Bibliográficas

[1] BRASIL. Centro de Pesquisas e Desenvolvimento em Telecomunicações (CPqD). **MODELO DE REFERÊNCIA**: Sistema Brasileiro de Televisão Digital Terrestre. Versão PD.30.12.36A.0002A/RT-08-AB. [Campinas]: CPqD, 13 fev. 2006. (Relatório Técnico, Cliente: FUNTTEL, OS: 40539) Disponível em ¡http://sbtvd.cpqd.com.br/¿ (seção de Divulgação). Acesso em 12 ago. 2007.

Apêndice A

Gramática dos *scripts* de diálogo

Descrevemos a seguir a gramática da linguagem de especificação de diálogos.

Apêndice B

Game Design Document