Pontifícia Universidade Católica
do Rio de Janeiro

Aarão Irving Manhães Marins

Um método para simulação em massa de agentes para cenas de mar em produções para TV/Cinema

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio

Orientador: Prof. Bruno Feijó

Rio de Janeiro
Janeiro de 2014

Pontifícia Universidade Católica
do Rio de Janeiro

Aarão Irving Manhães Marins

Um método para simulação em massa de agentes para cenas de mar em produções para TV/Cinema

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Bruno Feijó
Orientador
Departamento de Informática – PUC-Rio

Prof. Mônica Maria Ferreira da Costa
Departamento de Informática – PUC-Rio

Prof. Marley Maria Bernardes Rebuzzi Vellasco
Departamento de Engenharia Elétrica – PUC-Rio

Prof. José Eugênio Leal
Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 9 de Janeiro de 2014

Aarão Irving Manhães Marins

Graduou-se em Engenharia de Computação pela Pontifícia Universidade Católica em 2010. Trabalhou como pesquisador no laboratório de visualização, TV/cinema digital, e games VisionLab/ICAD da PUC-Rio de 2009 a 2011, e desde 2011 trabalha no setor de pesquisa e desenvolvimento de efeitos visuais da Rede Globo.

Ficha Catalográfica

Marins, Aarão Irving Manhães

Um método para simulação em massa de agentes para cenas de mar em produções para TV/Cinema / Aarão Irving Manhães Marins ; orientador: Bruno Feijó. – 2014.
80f : il. (color.) ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2014.
Inclui bibliografia

1. Informática – Teses. 2. Sistema Multiagente. 3. Lógica nebulosa. 4. Efeitos especiais visuais. 5. VFX. 6. Renderização realista. I. Feijó, Bruno. II. Pontifícia Universidade

CDD: 004

Agradecimentos

À minha família e amigos por todo o apoio que me deram.

À minha noiva Fabiana Calzavara Bittencourt Lima, por estar sempre ao meu lado em todas as minhas decisões.

À equipe de Pesquisa e Desenvolvimento de Efeitos Visuais da Rede Globo, em especial a Rodrigo Marques e Pablo Bioni.

Ao meu orientador Bruno Feijó, pela grande dedicação e incentivo à pesquisa.

Ao VisionLab pelos auxílios concedidos.

À CAPES e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Resumo

Marins, Aarão Irving Manhães; Feijó, Bruno. **Um método para simulação em massa de agentes para cenas de mar em produções para TV/Cinema.** Rio de Janeiro, 2014. 80p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação de mestrado apresenta um método para simular cenas de mar para TV/Cinema com um grande número de agentes de vários tipos (embarcações, portos, pessoas, ...) para produções com elevado grau de realismo de imagem e comportamento dos agentes. Este método usa modelagem em lógica nebulosa (*fuzzy*) e programação no sistema MASSIVE de maneira a integrar os resultados com sistemas de *rendering* e composição de imagens de alta resolução. Um objetivo importante desta dissertação é criar um sistema que facilita o trabalho de modeladores e designers envolvidos no *pipeline* de produção.

Palavras-chave

Sistema Multiagente; Lógica Nebulosa; Efeitos Especiais Visuais; VFX; Renderização Realista.

Abstract

Marins, Aarão Irving Manhães; Feijó, Bruno (Advisor). **A method for massive agents simulation of sea scenes for TV / Film productions.** Rio de Janeiro, 2014. 80p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This MSc dissertation presents a method to simulate sea scenes for TV / Film with a large number of agents of various types (vessels, ports, people ...) for productions with a high degree of realism in both image and agent's behavior. This method uses modeling in *fuzzy* logic and programming in the MASSIVE system as the main approach to integrate the results with rendering systems and high-resolution images composition. An important goal of this dissertation is to create a system that facilitates the work of modelers and designers involved in the production pipeline.

Keywords

Multi-agent System; Fuzzy Logic; Visual Special Effects; VFX; Realistic Rendering.

Sumário

Lista de figuras

A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original

Albert Einstein

1.
Introdução

A simulação de multidões em ambientes virtuais ainda é um grande desafio, tanto na área de efeitos visuais para TV, cinema e jogos digitais, quanto em avaliações de engenharia, em aplicações de segurança (simulações de situação de emergência/evacuação de ambientes), em simulações de tráfego, em arquitetura e em animação (Figura 1.1). Embora as simulações em cada uma destas áreas tenham propriedades e mecanismos de interação bem particulares, todas compartilham a mesma dificuldade de atribuir um comportamento único e natural a cada novo agente, isto é, a cada instância de uma unidade da multidão. Multidões, nesta área, significam conjuntos de pessoas, veículos, animais ou quaisquer outros objetos que exibam comportamento em massa – estes objetos são genericamente chamados de agentes.

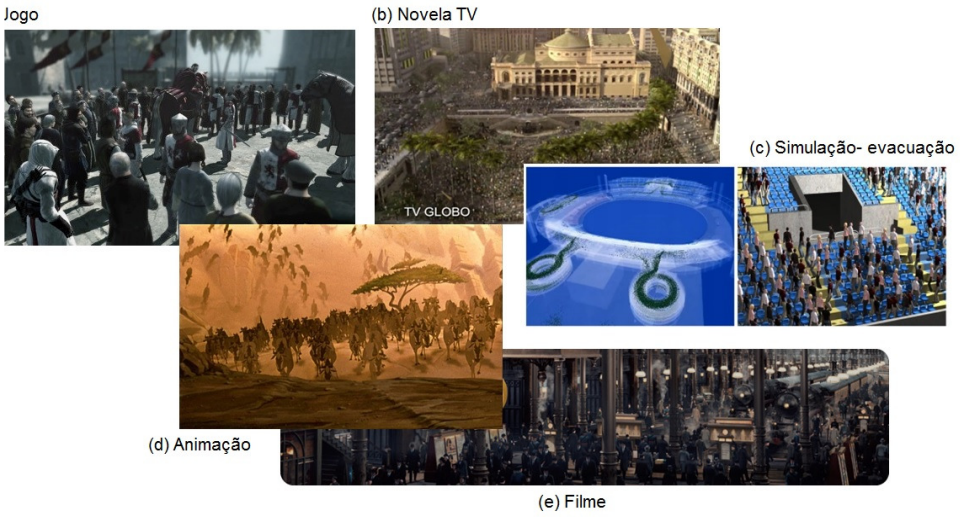


Figura 1.1: Exemplos de simulação de multidões. (a) Assassin's Creed (Ubisoft, 2007); (b) Ciranda de Pedra (TV Globo, 2008); (c) Simulação de evacuação em um estádio (Cassol et al., 2013); (d) Rei Leão (Disney, 1994); (e) Hugo (Paramount Pictures, 2011). (Copyrighted images reproduced under "fair use" policy - no other reproduction allowed)

A simulação de multidões em jogos requer mecanismos especiais de interação e deve sempre apresentar desempenho em tempo real (na prática, nunca inferior a 30 fps[1]). Simulações para engenharia, segurança/evacuação e análise/controle de tráfego usualmente requerem desempenho interativo (algo entre 5 a 15 fps) e algumas vezes requerem desempenho offline (tipicamente abaixo de 1 fps). No caso de TV e cinema, a exigência é sobre a qualidade da imagem (tipicamente produzida em 4K, i.e. 3840 x 2160 pixels, e 8K, i.e. 7680 x 4320 pixels) e o realismo das cenas (tanto em termos de movimento e comportamento, como em termos de taxa de contraste da imagem). Nesse caso de TV/cinema, o processo de simulação de multidões é complexo e não permite automação plena. Outra característica desse último caso é a necessidade de vários processamentos em mais de uma taxa de desempenho, desde verificações em tempo real com baixa qualidade de imagem até processamentos offline (que pode demorar várias horas por frame) com elevada qualidade de imagem. Na simulação de agentes em TV e cinema, ainda há a necessidade da integração entre artistas e programadores de computador, o que complica todo o processo.

O presente trabalho está focado em aplicações para efeitos visuais (VFX, termo que representa “visual effects”) em TV e cinema. Atualmente existem muitas ferramentas que podem ser acopladas a sistemas de criação de ambientes virtuais capazes de gerar multidões para TV e cinema. Porém, em geral, estas ferramentas oferecem soluções que requerem um grande esforço por parte do artista para não deixar cópias de agentes com comportamentos evidentemente iguais. O maior problema, entretanto, está na grande dificuldade que o artista tem de programar a inteligência e o comportamento dos vários agentes presentes na simulação. Principalmente em TV, este último tipo de dificuldade é um fator crítico que pode comprometer toda a produção.

[1] fps – quadros por segundo

Esta dissertação trata de uma situação ainda mais particular de agentes e multidões: a simulação de embarcações em cenas de mar e ambientes de porto (Figura 1.2). Este tipo de cena requer a simulação da superfície do mar (hora calmo, hora revolto), de tripulantes a bordo das embarcações e de pessoas no porto próximas aos pontos de ancoragem das embarcações. Ademais, as cenas de fundo costumam ser bastante elaboradas (Figura 1.2c[2]). O processo para realizar estas cenas de mar é bastante complexo e não há referências na literatura. Os softwares comerciais não vêm prontos para a realização deste tipo de cena e as empresas de cinema não revelam os processos utilizados.

(a) frota grega no filme Troy (Warner, 2004)

(b) detalhe de embarcação em Troy (Warner, 2004)

(c) detalhe de porto real (Portoroz, Eslovênia) com cena de fundo complexa

Figura 1.2: Cenas típicas de mar e porto, com embarcações (Copyrighted images reproduced under “fair use” policy - no other reproduction allowed).

Esta dissertação propõe um método para simulação em massa de agentes para cenas de mar e de porto com embarcações, com a qualidade de imagem e movimento típica de TV e cinema, e que facilita o trabalho dos artistas, modeladores e animadores. Neste trabalho foi escolhido o software MASSIVE[3] como sendo o ambiente central de simulação de multidões. Em torno do MASSIVE foram desenvolvidos módulos de suporte ao tipo de simulação em questão. Os dois principais

[2] Imagem extraída de site privado: www.diplomatic-corporate-services.si/services/tourist-information/slovenia-seaside

[3] www.massivesoftware.com

módulos, especialmente desenvolvidos para este suporte, foram um simulador de oceano que gera a superfície do mar em várias condições e um módulo de especificação das principais variáveis com modificadores nebulosos (*fuzzy*), também denominados de *hedges* (Zadeh, 1972). Com estes *hedges*, o artista pode facilmente determinar que um determinado grupo de navios deve se mover “mais ou menos rápido” ou “extremamente rápido”.

O software MASSIVE, além de funcionar como uma ferramenta para gerar multidões, possui uma estrutura que permite a manipulação de *rig*[4], pele (*skin*), animações e materiais e a renderização da imagem final no viewport (Figura 1.3). Porém, a sua principal característica é a existência de um "cérebro"[5] que, apesar de ter a mesma estrutura para um determinado tipo de "agente", funciona de maneira independente em cada uma de suas instâncias. Este cérebro funciona baseado em Lógica Nebulosa (*Fuzzy Logic*). Este software é especialmente voltado para a área de efeitos visuais da indústria de cinema e televisão.

[4] *Rig* é o termo usado pelos animadores para se referir à montagem feita para facilitar o controle de um personagem. O *rig* consiste de cadeias de ossos articulados entre si que formam o esqueleto do personagem, de scripts e de qualquer atributo de controle da animação do esqueleto.

[5] O termo “cérebro” é usado pelo software MASSIVE, apesar de academicamente falando não ser adequado. Este termo é mantido ao longo do presente texto para ficar coerente com o software.

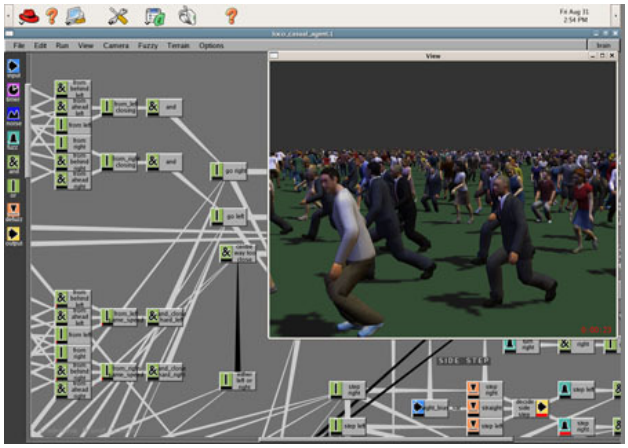


Figura 1.3: Interface de manipulação do "cérebro" do agente e o viewport com o resultado visual final do software MASSIVE. (Copyrighted images reproduced under "fair use" policy - no other reproduction allowed).

A interação entre os diferentes tipos de agentes é um desafio adicional, tendo em vista que o funcionamento de cada instância de um desses tipos envolvidos na simulação é sempre única e que cada tipo tem uma organização de cérebro distinta.

O presente trabalho apresenta todos os procedimentos necessários para a simulação de cenas de mar, isto é: desde a geração dos arquivos de entrada do sistema até a obtenção do vídeo final em alta definição. A dissertação está organizada como se segue. O capítulo 2 discute os trabalhos, processos e padrões relacionados ao tema da dissertação. O capítulo 3 descreve a lógica *fuzzy* e apresenta a definição dos *hedges* utilizados na simulação. O capítulo 4 apresenta o método proposto, na forma de um fluxo de trabalho (*workflow*) descrito como um diagrama de atividades UML. A simulação do oceano usada na simulação é detalhada no capítulo 5. O capítulo 6 conclui a dissertação apresentando os resultados obtidos com a implementação da cena protótipo (usando o *workflow* proposto), discutindo os principais resultados e sugerindo algumas direções para trabalhos futuros.

2.
Trabalhos, Processos e Padrões Relacionados

2.1. Trabalhos Relacionados

Existem várias lacunas na literatura com relação ao tipo de simulação tratado no presente trabalho. Em primeiro lugar, não há referências a métodos ou procedimentos que tratem de todo o processo necessário para obtenção de imagens realistas e de alta definição de uma cena de simulação de agentes virtuais. Em segundo lugar, não há referências a simulação em massa de agentes para cenas de mar. Esta seção apresenta trabalhos de âmbito mais geral.

Dois dos trabalhos mais citados na área de simulação de multidões são referências pioneiras de animação procedimental: Reynolds (1987) e Reynolds (1999). O autor destes trabalhos fez um estudo empírico com membros de bandos (de aves e peixes) que se movem em relação ao movimento de seus companheiros. Com comportamentos simples, Craig Reynolds foi capaz de recriar um bando de “agentes” autônomos tanto para filmes (Reynolds 1987) como para jogos (Reynolds, 1999).

Uma outra referência sobre agentes autônomos para animação comportamental é a proposta de Costa e Feijó (1996). Apesar deste trabalho não lidar com multidões, trata-se de trabalho pioneiro no uso de entidades autônomas, com comportamentos e emoções, que estão na base de simulações com agentes.

Na linha de autômatos celulares, pode-se citar os seguintes trabalhos com simulação de multidões: Joselli et al. (2009), que propõe uma arquitetura de grid em GPU especialmente desenvolvida para

simulações em tempo real; e Chenney (2004) que propõe uma técnica baseada em campos de velocidade.

Alinhada com a indústria de jogos, há a discussão proposta por Pottinger (1999), que apresenta uma implementação de movimento coordenado de grupos. No seu trabalho apresentam-se diversos conceitos importantes para controles de grupos guiados por inteligência artificial para jogos de estratégia, assim como são classificados diferentes tipos de grupos, formações e tratamento de obstáculos.

Outro trabalho relevante na área foi a separação dos agentes em modelos hierárquicos, procurando simular multidões humanas em tempo real (Musse et al., 2001). Estes autores definem uma hierarquia em termos de multidões, grupos e indivíduos.

Forças sociais também são propostas para a simulação de multidões (Cordeiro et al., 2005) (Gayle et al. 2009).

Como referência mais geral na área de simulação de multidões, há o excelente livro escrito por Thalmann e Musse (2013).

Há poucos trabalhos acadêmicos retratando os aspectos da programação do "cérebro" dos agentes utilizando lógica *fuzzy* no software MASSIVE. Amit (2007) implementa uma simulação de um grupo de pessoas jogando a "dança das cadeiras". Alguns trabalhos foram pesquisados para a auxiliar na implementação das *hedges* em conjunto com a lógica *fuzzy* dos agentes (Esteva et al ., 2013; Huang et al ., 1995). Neste trabalho, o autor demonstra muitas maneiras de lidar com diferentes situações, enquanto se está produzindo um agente e suas interações no MASSIVE.

Uma referência interessante, apesar de ser mais uma nota técnica do que um artigo, é o trabalho de Mononen (2010). Um dos objetivos deste trabalho é deixar o movimento dos agentes o mais flexível possível. Para isto, Mononen (2010) demonstra uma iteração de navegação, que

cobre todas as etapas (desde encontrar o caminho até o movimento de um objeto na cena). Apesar de ser uma ideia muito simples, este trabalho pode ser útil para a movimentação de pessoas em embarcações (Figura 2.1).

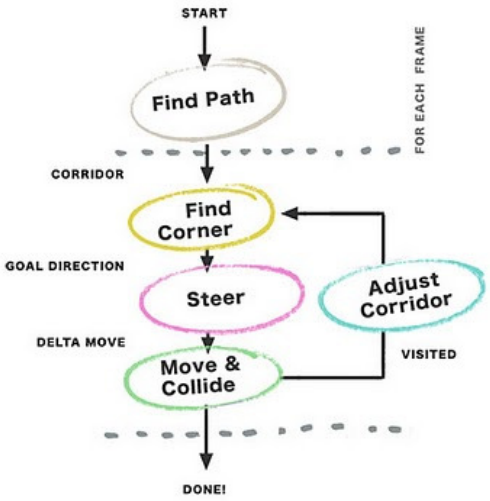


Figura 2.1: Iteração de navegação (*navigation loop*) proposta por Mononen (2010).

Outra referência muito interessante que também se enquadra como um relatório técnico é o texto de Mars (2011), que situa em que estado estão os atuais estudos e utilização das técnicas da literatura para navegação de agentes.

Entre as ferramentas disponíveis atualmente no mercado, destacam-se as seguintes: Basefount Miarmy[6] e Golaem Crowd[7], ambos plugins para o Autodesk Maya. Estas ferramentas são capazes de realizar muito bem as cenas de multidões nas seguintes condições: quando não há complexidade nas interações entre os agentes; e quando os agentes se encaixam nas limitações impostas pelo software em que estão sendo executados. A maior parte destas ferramentas funciona com ações pré-definidas, não dando possibilidade de combinar diferentes animações por

[6] http://www.basefount.com/miarmy

[7] http://www.golaem.com/

meio de um controlador de agente programável, “cérebro”, como é o caso do Massive Software.

Seguindo a mesma linha, porém ainda em fase de testes, existe o Project Geppetto[8], um plugin da própria Autodesk (para o Autodesk 3ds Max), que consiste em uma pesquisa buscando uma abordagem que torne mais simples (para o artista) adicionar multidões em cenas virtuais, com grande nível de detalhamento de animações. Sendo assim, suas principais metas são gerar animações convincentes de maneira fácil, inserir características populacionais/demográficas e culturais e criar um *framework* eficiente para adicionar dezenas de milhares de personagens em cenas virtuais. Entretanto, esta solução só atende a simulações de agentes humanos, não sendo uma alternativa ao presente trabalho pois não possui e nem possibilita a implementação de agentes tipo embarcação como os do presente trabalho.

2.2. Processos e Padrões Relacionados

Como este trabalho é focado no desenvolvimento de técnicas para produções de TV e Cinema, torna-se necessária a apresentação dos processos e padrões relativos a formatos e resoluções de imagens, arquivos de interoperabilidade e produção artística. O restante deste capítulo é dedicado a estes tópicos relacionados à presente dissertação.

2.2.1. Formatos e Resoluções

2.2.1.1. 4K

4K é a forma simplificada de se referir à primeira resolução do padrão conhecido como UHD – Ultra High Definition, que é disponibilizada em dois formatos. O primeiro, voltado para o mercado consumidor, é a de 3840 x 2160 pixels, principalmente por ser uma resolução múltipla do HD (High Definition) – 1920 x 1080 pixels com fator 2 para largura e altura. O segundo formato, com 4096 x 2160 pixels, é destinado à produção de

[8] http://labs.autodesk.com/utilities/geppetto/

conteúdo para cinema digital. A definição do 4K foi feita pelo consórcio DCI – Digital Cinema Initiatives[9].

Este padrão é muito utilizado para armazenamentos de mídia digital. Porém películas e sistemas óticos ainda são vastamente utilizados, uma vez que a resolução em película é basicamente dependente do grão fotossensível e não é dependente de equipamentos eletrônicos para decodificar o formato – evitando, assim, problemas de reprodução por falta de equipamento. Outros formatos de resolução são praticados, porém não há uma regulamentação formal pela SMPTE (Society of Motion Picture & Television Engineers – www.smpte.org) do padrão para transmissão. No Brasil, a primeira câmera utilizada foi a F65 da Sony, durante o Carnaval de 2012.

2.2.1.2. 8K

Semelhante ao 4K, o 8K é a forma simplificada de se referir à segunda resolução do padrão conhecido como UHD – Ultra High Definition, disponibilizada no formato de 7680x4320 pixels para o mercado consumidor, como televisores. Esta resolução é múltipla do HD com fator 4, tanto para altura quanto largura. O 8K também é a forma utilizada no padrão *fulldome* para planetários, com 8192 x 8192 pixels. A NHK, empresa japonesa de *broadcast*, foi a primeira a desenvolver uma câmera proprietária 8K junto com a JVC e com *encoders* da Panasonic. Esta câmera foi utilizada, em caráter de teste, no Carnaval 2013 pela Rede Globo e, de maneira mais definitiva, na aquisição de imagens da Copa das Confederações 2013.

A Figura 2.2 apresenta a relação de proporção entre os vários formatos apresentados neste capítulo.

[9] DCI é uma *joint venture* dos maiores estúdios de cinema (Disney, MGM, Fox, …), criada em 2002, para estabelecer uma arquitetura aberta para sistemas de cinema digital: http://www.dcimovies.com.

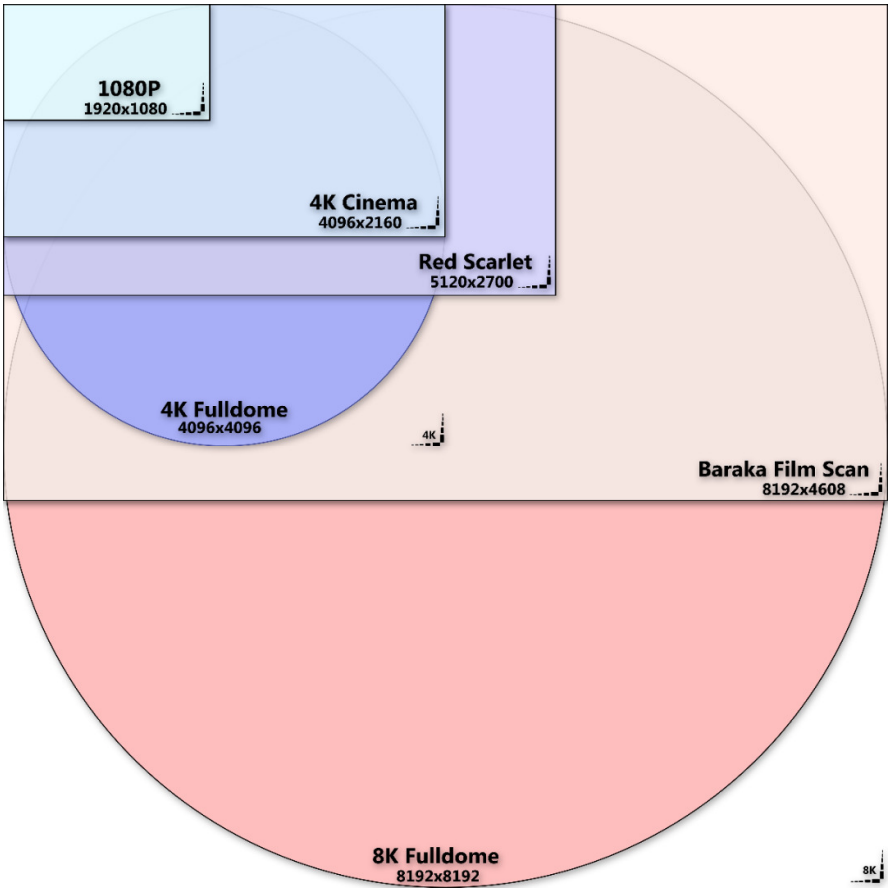


Figura 2.2: Relação de proporção dos formatos. Imagem extraída de http://upload.wikimedia.org/wikipedia/commons/7/79/8KFulldomeResolutionChart-HALFRES.png. . (Copyrighted images reproduced under “fair use” policy).

2.2.2. Arquivos de armazenamento de imagem

Além da necessidade de armazenar resoluções cada vez maiores, a outra preocupação é conseguir armazenar cada vez mais informações de cores. Há vários formatos disponíveis para isto, tais como os antigos Targa (.TGA), os .DPX (evolução do Cineon), os .HDR e os .EXR (Lucas Digital Ltd, 2006). Para entender a necessidade deles, é preciso entender alguns aspectos importantes, que são apresentados a seguir. A forma mais conhecida de se armazenar imagens é através da codificação das três cores primárias, vermelho (R), verde (G) e azul (B) – Conhecido RGB para cada pixel da imagem. Dessa forma, cada canal possui um determinado número de bits para quantizar a intensidade do seu sinal. Os formatos mais comuns possuem 8 bits por canal, permitindo uma

combinação que gera 1.6 milhões de cores. Porém, em uma análise isolada por canal, este formato retorna apenas 256 valores para quantizar tonalidades. Estes arquivos são conhecidos como LDR (*Low Dynamic Range*).

O olho humano é capaz de operar em até 10 ordens de grandeza de intensidade luminosa e suportar 5 ordens em simultâneo, em uma mesma situação luminosa (essas ordens definem os diferentes níveis de intensidade das fontes luminosas visualizados). Em contrapartida, uma imagem de 8 bits só consegue representar 2 ordens de magnitude. Sendo assim, o grande desafio é ter uma forma de captar e armazenar não apenas a informação de cor, mas também a luminosidade do pixel.

A capacidade de lidar com ordens luminosas maiores é chamada de *Dynamic Range* (*Range* Dinâmico ou Latitude de uma imagem), que é uma razão que representa a relação entre os valores máximo e mínimo de uma medição física. Esta razão pode se referir a vários espaços, tais como: espaço de cena (relação entre a parte mais escura e de maior intensidade luminosa); espaço de câmera (relação entre saturação do sensor e nível exatamente superior ao ruído de baixa intensidade); espaço de display (relação entre a intensidade de emissão máxima e a mínima do componente do display). As imagens com elevado range dinâmico são conhecidas por imagens HDR (*High Dynamic Range*). Os principais arquivos na indústria que lidam com estas imagens de grandes latitudes e resoluções usam os formatos HDR e EXR para armazenar as imagens.

2.2.2.1 HDR

O formato HDR (.hdr), originalmente conhecido como formato de imagem Radiance (.pic), foi introduzido em 1998 como parte do software Radiance para renderização e simulação de iluminação. Entretanto, uma primeira descrição deste formato (conhecida por RGBE) foi proposto por Greg Ward em 1991, um dos autores do Radiance, em artigo histórico no Graphics Gems II entitulado “Real Pixels” (Ward, 1991). Este formato

recebeu novo foco de atenção com os trabalhos de Paul Debevec sobre fotografia HDR e iluminação baseada em imagem (Debevec e Malik, 1997; Debevec, 1998). Na realidade, uma imagem HDR pode ser armazenada em uma variedade de formatos, tais como o inicial RGBE, o 32-bit LogLuv (Larson, 1998) incluído no padrão TIFF e o EXR proposto pela ILM (também denominado OpenEXR). Este último formato é descrito com mais detalhes na próxima seção. Uma análise mais profunda sobre *encoding* de imagens HDR pode ser encontrado no artigo de Greg Ward na Web: www.anyhere.com/gward/hdrenc. Deve-se, portanto, fazer distinção entre HDR se referindo ao formato popularizado pelo software Radiance e HDR significando a imagem com elevado *range* dinâmico.

Um dos usos de imagens HDR com grande significado para a área de efeitos especiais em TV e cinema é o processo de IBL – *Image Based Lighting*. IBL é o processo de iluminar cenas e objetos (reais ou sintéticos) com imagens de luz do mundo real (Debevec, 2002). IBL consiste em recriar uma iluminação real captada em um arquivo HDR em uma cena virtual. A Imagem HDR, ao armazenar informações de luminosidade, é capaz de ser usada como fonte de luz. Neste processo, captura-se uma imagem 360, mapeia-se uma esfera com esta imagem e cria-se um ambiente de luz virtual.

Os formatos para imagens HDR possuem *encoding* com ponto flutuante de 32 bits ou 16 Bits, que praticamente são suficientes para cobrir o *range* tonal das cenas reais. Desta forma, cada pixel possui 96 bits de informação, o que na prática tende ao infinito quando o instrumento de avaliação é o olho humano. A Imagem HDR armazena valores lineares, é proporcional à quantidade de luz medida pela câmera e utiliza o espaço de cena para as medições.

2.2.2.2 Formato EXR

O Formato EXR foi desenvolvido pela Industrial Light & Magic (ILM) e é também denominado formato OpenEXR. A principal característica em relação ao formato HDR é que o EXR foi um formato criado com propósito

de atender a necessidade da indústria de efeitos visuais. Com isto tornou-se rapidamente mais robusto e, por ser uma plataforma aberta, o formato pode ser vastamente utilizado pelos softwares de mercado. Outra grande vantagem é a forma como ele armazena os dados, podendo chegar a trinta f-stops, que representam os níveis de abertura da câmera observadora.

O formato EXR utiliza ponto flutuante de 32 ou 16 bits. No caso mais comum de uso, que é o formato de 16 bits (chamado de “half”), a ILM utiliza 1 bit para o sinal, 5 bits para o expoente e 10 bits para a mantissa. Outra grande vantagem do EXR é que ele pode armazenar um número arbitrário de canais e atributos. A Versão 2.0 do EXR suporta também informação de profundidade para processos de *Deep Compositing* (um *workflow* de composição, desenvolvido pela Weta Digital, que desacopla a renderização de diferentes elementos em cena). Já o formato HDR armazena 8 bits para cada canal de cor e mais 8 bits para o canal de transparência. A Figura 2.3 ilustra a distribuição de bits dos formatos HDR e EXR.

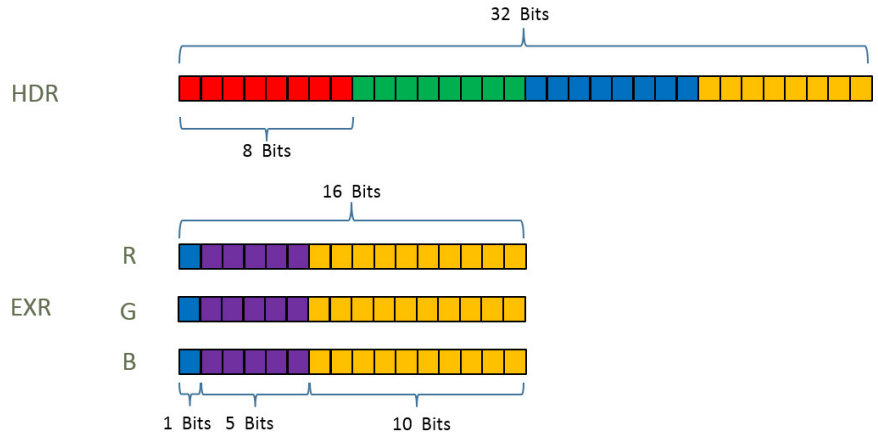


Figura 2.3: Distribuição de bits dos arquivos HDR e EXR.

2.2.3. Arquivos de interoperabilidade entre pacotes 3D

Nesta seção são apresentados os formatos mais usados para facilitar a interoperabilidade entre vários softwares da área de efeitos especiais:

FBX, RIB e Alembic. A presente dissertação usa apenas os dois primeiros.

2.2.3.1. FBX

FBX é um dos mais antigos formatos para armazenagem de conteúdo 3D, com a finalidade de facilitar a interoperabilidade entre os pacotes de sistema 3D disponíveis no mercado. Cada um desses pacotes possui seu próprio arquivo nativo de trabalho e, muitas vezes, faz-se necessário o uso da malha gerada por um pacote em outro de diferente fabricante, muitas vezes sendo um fabricante concorrente. A solução encontrada pelo mercado foi a adoção do formato FBX. Originalmente desenvolvido pela Kaydara e atualmente formato proprietário da Autodesk, o FBX (abreviação de FilmBoX) era a forma principal de migração de malhas entre softwares com o foco em soluções de captura de movimento (*motion capture*).

2.2.3.2. RIB

RIB é o arquivo padrão descritor de cena 3D proprietário do Renderman da PIXAR. O arquivo RIB basicamente é uma sequência de linhas de comandos que criam uma cena nos padrões para que possa ser renderizada[10] pelo Renderman (www.renderman.com). Dentro do RIB, é possível definir Modelos 3D, Animações, *Shaders*, iluminação. Geralmente há um arquivo RIB por frame de animação.

2.2.3.3. Alembic

Alembic é um novo formato (desenvolvido pela Sony e pela Lucasfilm e anunciado no SIGGRAPH de 2011), que é atualmente o mais utilizado pela indústria para a função de interoperabilidade de softwares (http://www.alembic.io). Por ser uma iniciativa *open source*, o Alembic foi muito bem aceito e surgiu da necessidade de um formato que funcionasse sem muitos problemas. O FBX, embora bem conhecido e estável, sempre

[10] Rederizar significa gerar sequências de imagens que representam a visualização de uma camera posicionada e possivelmente animada de uma cena virtual.

foi um formato complicado e que não cumpria o papel de interoperabilidade de forma eficiente. A principal forma de utilização do Alembic é como *bake*[11] de cena para renderização.

2.2.4. Processo de Produção Artística

A criação de qualquer cena fundamentada em efeitos visuais, passa por três etapas principais. *Storyboard*, Conceituação gráfica dos elementos do *storyboard* (*Concept*) e a Execução. As Figuras 2.4 a 2.7, a seguir, ilustram estas etapas. No workflow do capítulo 4, os produtos destas etapas são condições para a execução de várias atividades.

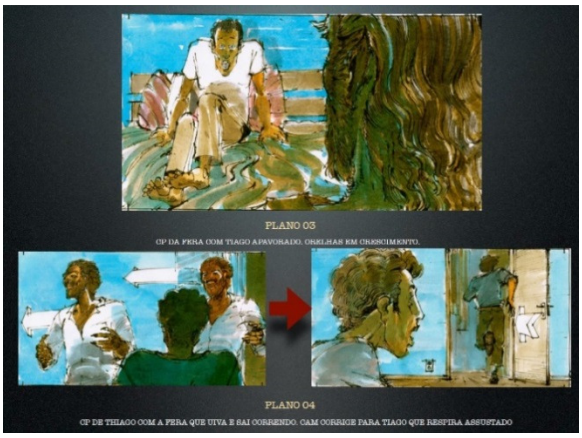


Figura 2.4: Storyboard de uma cena da novela Saramandaia da Rede Globo. Cortesia do grupo Visual Effects R&D da TV Globo (Copyrighted images reproduced under “fair use” policy - no other reproduction allowed)

Figura 2.5: Concept 2D do personagem principal da cena da Figura 2.4. Cortesia do grupo Visual Effects R&D da TV Globo (Copyrighted images reproduced under “fair use” policy - no other reproduction allowed)

[11] Esta operação de renderização também é conhecida como “Texture Baking” (que significa “assar” várias texturas e mapas de luz num objeto, tornando-os parte do objeto).

Figura 2.6: Concept 3D do personagem conceituado na Figura 2.5. Cortesia do grupo Visual Effects R&D da TV Globo (Copyrighted images reproduced under “fair use” policy - no other reproduction allowed)

Figura 2.7: Frame da cena final gerada a partir do storyboard e dos concepts. Cortesia do grupo Visual Effects R&D da TV Globo (Copyrighted images reproduced under “fair use” policy - no other reproduction allowed)

2.2.5. Fotogrametria

Fotogrametria é o processo usado para se ter uma rápida aquisição de modelos 3D. Utilizando uma máquina fotográfica, é possível tirar uma sequência de fotos de uma pessoa ou ambiente e ter uma rápida prototipagem 3D. Esse protótipo geralmente é usado como referência em cenas para o software Massive ou outras aplicações 3D. No caso da presente dissertação, esta técnica é recomendada para a geração de background de terrenos.

A Figura 2.8 ilustra o uso da fotogrametria para gerar um modelo 3D (i.e. uma malha 3D de muitos triângulos) de um ator real. Usam-se, no mínimo, cerca de 12 tomadas ao redor do ator ou cena real para se obter um bom resultado. Uma outra forma de capturar a face (e até mesmo partes mais completas) de um ator real é através da tecnologia de Light Stage desenvolvida pelo USC ITC por Paul Debevec (http://gl.ict.usc.edu/LightStages). Neste caso, obtém-se milhões de pontos com informações de iluminação extremamente ricas. Esta forma é, entretanto, extremamente custosa.

Fotogrametria também pode ser usada para gerar sequências de vídeo (tipicamente um fundo com movimento). Neste caso, torna-se necessária a sincronização dos frames gerados.

Para o caso simples de cenas estáticas e de fundo (onde se é mais tolerante com a baixa qualidade das malhas), recomenda-se o uso do Autodesk 123D Catch. Para outras cenas mais detalhadas recomenda-se o Agisoft Photoscan Professional. Este último é capaz de gerar ortofotos georeferenciados de alta resolução com uma acurácia de até 5 cm.

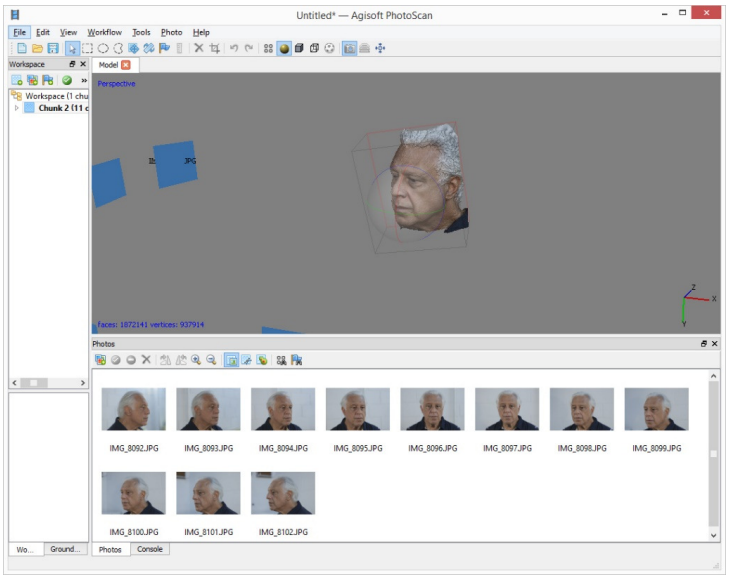


Figura 2.8: Resultado de um processo de fotogrametria. Cortesia do grupo Visual Effects R&D da TV Globo (Copyrighted images reproduced under “fair use” policy - no other reproduction allowed)

3.
Lógica *Fuzzy* e *Hedges*

Em meados da década de 60, Lotfi Zadeh fez uma contribuição seminal para o conceito de incerteza ao propor conjuntos nebulosos (*fuzzy sets*)[12]. Nestes conjuntos, onde os limites não são precisos, a pertinência de um elemento não é uma questão de verdadeiro/falso mas uma questão de grau (Zadeh, 1965). Quando esta ideia evolui para proposições, inferências e raciocínio impreciso tem-se a lógica *fuzzy*. Nesta dissertação não se pretende discutir as possíveis interpretações desta lógica. Neste particular, adota-se a interpretação da lógica *fuzzy* no seu sentido mais amplo, conforme sugerido pelo próprio Zadeh (1994), onde se buscam soluções rápidas e simples, sem entrar em questões profundas de lógica matemática.

A lógica *fuzzy* possibilita o desenvolvimento de sistemas que configuram decisões humanas, nos quais a lógica clássica se mostra insuficiente, inadequada ou ineficiente. Em von Altrock (1997) pode-se encontrar uma introdução à lógica *fuzzy* com uma abordagem prática, onde essa lógica procura se aproximar da maneira como o ser humano relaciona dados e se expressa coloquialmente para gerar uma resposta aproximada ao problema.

Kaehler (1998), por sua vez, define a lógica *fuzzy* como "uma maneira simples de chegar a uma conclusão definitiva, baseado em informações de entrada que são vagas, ambíguas, imprecisas, ruidosas ou faltantes".

[12] Nesta dissertação adota-se o termo *fuzzy* ao invés de "nebuloso/a" por ser o termo em inglês o usado pela comunidade profissional de computação gráfica e VFX.

Através de inúmeros artigos e aplicações práticas, a lógica *fuzzy* mostrou ser um bom método para auxiliar a manipulação de dados, bem como provou ser uma escolha excelente para muitas aplicações de controle de sistemas.

A lógica *fuzzy* provê métodos bem eficientes para sistemas de solução de problemas do tipo “se X e Y então Z”, baseando-se em regras e no conceito de pertinência. De uma maneira simplista, pode-se dizer que o seu modelo é baseado em dados empíricos, que podem ser atribuídos mais à experiência subjetiva e imprecisa do operador do que ao seu conhecimento técnico preciso. Por exemplo, ao invés de tratar a temperatura com termos como: “T<100C”, ou “25<TEMP<50”, utilizam-se termos como: “SE (processo está muito frio) E (processo está resfriando) ENTÃO (adicionar calor ao processo)”. Esses termos são imprecisos, porém muito descritivos em relação ao que está realmente acontecendo. Considerando que uma pessoa está no chuveiro, se a temperatura do chuveiro estiver muito quente, a pessoa rapidamente é capaz de ajustar para uma temperatura confortável.

O comportamento apresentado pela lógica *fuzzy* tem grandes semelhanças com a forma humana de processar as informações, não sendo booleana (no sentido de verdadeiro/falso), mas trazendo consigo inferências e aproximações. Esta característica faz com que a Lógica *Fuzzy* seja amplamente utilizada em sistemas de Inteligência Computacional onde se busca sempre esta proximidade do comportamento humano.

A lógica *fuzzy* tem se estendido e se combinado com outras técnicas de inteligência computacional com sucesso, tais como sistemas neuro-*fuzzy* (Vellasco et al, 2008) e sistemas *fuzzy* genéticos (Koshiyama et al., 2013).

O restante deste capítulo apresenta os conceitos básicos da lógica *fuzzy*, seguido da utilização desta lógica para a implementação do

cérebro Massive do agente de embarcação implementado neste trabalho. O agente humano não possui alterações significantes em relação ao cérebro do agente humano que acompanha o Massive Software, e, portanto, seus detalhes podem ser vistos no capítulo 4.

3.1. Conjuntos Fuzzy

A lógica *fuzzy* se baseia na teoria dos conjuntos *fuzzy* introduzida por Zadeh (1965). Um conjunto *fuzzy* representa a intensidade de uma característica em um grupo de objetos, que podem ser números, pessoas, embarcações, etc. Ao contrário dos conjuntos da teoria clássica dos conjuntos (onde um elemento só pode pertencer ou não a um conjunto), os elementos de um conjunto *fuzzy* podem participar parcialmente de um conjunto, apresentando assim um grau de pertinência relacionado à característica que o representa.

A característica de um conjunto *fuzzy* pode ter níveis de incerteza e imprecisão. A representação do grau de pertinência que representa esses níveis são valores que vão desde a representação da ausência da característica de um elemento do conjunto (0) até a total incidência dessa característica em outro elemento (1), sendo os valores intermediários pertencentes ao intervalo [0,1].

3.1.1. Operações

Assim como na teoria clássica dos conjuntos, na teoria dos conjuntos *fuzzy* existem operações típicas, tais como união, interseção e complemento (Klir et al ., 1995).

Na teoria clássica dos conjuntos, a interseção entre dois conjuntos contém aqueles elementos que são comuns a ambos. Na teoria dos conjuntos *fuzzy*, entretanto, o elemento pode pertencer parcialmente aos dois conjuntos, ainda que não pertença completamente a nenhum deles. Assim, quando é considerada a interseção desses conjuntos, não se pode dizer que um elemento possa pertencer mais ao conjunto da interseção do que a qualquer um dos conjuntos originais. De acordo com isto, o

operador *fuzzy* é usado para gerar a interseção entre dois conjuntos *fuzzy* *A* e *B* definidos em *X* pode ser dada por:

$$\mu_{A \wedge B}(X) = \min(\mu_A(x), \mu_B(x)) \text{ para todo } x \in X$$

$$= \mu_A(x) \wedge \mu_B(x)$$

$$= \mu_A(x) \cap \mu_B(x)$$

O símbolo ^ (chamado "E" lógico) é usado para representar o operador "min" que simplesmente toma o mínimo entre os valores em consideração. Existem outras formas de determinar a interseção entre conjuntos *fuzzy*, como por exemplo, o produto.

Outra forma de combinar conjuntos *fuzzy* é através de sua união. A união de dois conjuntos é compreendida como sendo o conjunto dos elementos que pertencem a pelo menos um deles (ou a ambos). Os elementos do conjunto união não podem possuir valor de pertinência menor do que o que possuía em qualquer um dos conjuntos originais. Uma das formas que a lógica *fuzzy* usa para obter a união entre dois conjuntos pode ser a seguinte:

$$\mu_{A \vee B}(x) = \max(\mu_A(x), \mu_B(x)) \text{ para todo } x \in X$$

$$= \mu_A(x) \vee \mu_B(x)$$

$$= \mu_A(x) \cup \mu_B(x)$$

O símbolo $\vee$ (chamado "ou" lógico) é usado na lógica *fuzzy* para representar a operação "max", que toma o valor máximo dentre os valores em consideração. Assim como na interseção, a união também possui outras formas de ser representada nos conjuntos *fuzzy*. Um outro exemplo para representação da união é a soma.

Essas operações são a base da construção das regras de inferência, que compõem relações entre os dados de entrada e os de

saída, essas relações podem ser representadas por implicações (se, então).

3.1.2. Variáveis Linguísticas

Uma variável linguística é uma variável cujos valores são nomes de conjuntos *fuzzy*. Estes valores são descritos por intermédio de conjuntos *fuzzy*. Generalizando, os valores de uma variável linguística podem ser sentenças em uma linguagem especificada, construídas a partir de termos primários (alto, baixo, pequeno, médio, grande, zero, por exemplo), de conectivos lógicos (negação "NÃO", conectivos "E" e "OU", conectivos mascarados, como "mas", "porém"), de modificadores (como "muito", "pouco", "levemente", "extremamente") e de delimitadores (como parênteses).

As variáveis linguísticas têm, pois, a função de fornecer uma maneira sistemática para uma caracterização aproximada de fenômenos complexos ou mal definidos. Através da utilização de variáveis cujos valores são nomes de conjuntos *fuzzy*, a simplificação e o melhor entendimento do problema é conseguido com maior sucesso que na lógica tradicional.

3.1.3. Hedges

As variáveis linguísticas podem utilizar qualificadores a fim de alterar a intensidade com que um certo elemento faz parte de um determinado conjunto. Qualificadores são termos que atuam na modelagem de sistemas *fuzzy* da mesma forma que advérbios e adjetivos atuam em um sentença. Com isso, qualificadores modificam a forma de um conjunto *fuzzy*. Estes qualificadores são conhecidos como modificadores linguísticos, operadores semânticos ou *hedges*.

Os *hedges* possuem diversas classes como se segue. Intensificadores: muito, extremamente; Diluidores: pouco, mais ou menos; Complemento: não; Aproximadores: em torno, aproximadamente, quase; entre outras. Neste trabalho, foram aplicados dois desses tipos de

hedges: intensificadores e diluidores. A Figuras 3.1 mostra exemplos destes tipos de *hedge*.

$$\mu_{(x)_{intens}} = \mu_{(x)}^{n} \qquad \mu_{(x)_{diluid}} = \mu_{(x)}^{n}$$

Figura 3.1: Exemplos de *hedges* intensificadores e diluidores. Imagens retiradas das notas de aula do Curso de Inteligencia Computacional Aplicada do Departamento de Engenharia Elétrica da PUC-Rio pelo site: http://www2.ica.ele.puc-rio.br/Downloads/31/ICA-cursop4-Hedges1.pdf. (Copyrighted images reproduced under "fair use" policy - no other reproduction allowed).

Esses operadores intensificadores e diluidores possuem o mesmo suporte que o conjunto *fuzzy* original, ou seja, o mesmo valor no domínio para μ(x) = 0 e μ(x) = 1.

3.2. Sistema de Inferência Fuzzy

Com base na teoria dos conjuntos *fuzzy*, é possível construir o Sistema de Inferência *Fuzzy* mostrado na Figura 3.2, adaptada de Cox (1994), onde estão indicadas as etapas deste sistema.

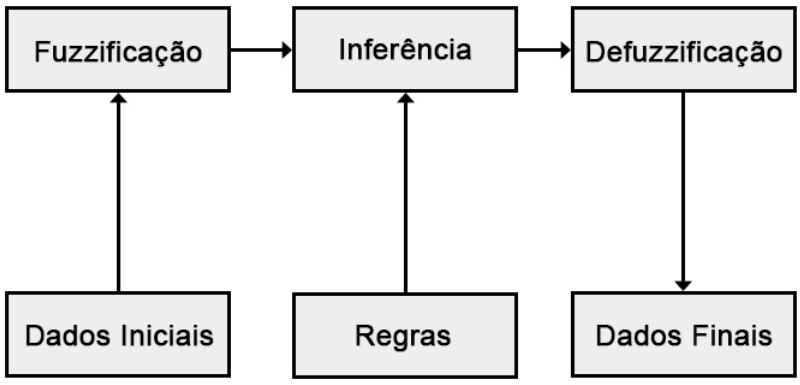


Figura 3.2: Sistema de Inferência fuzzy. Fonte: Cox (1994)

Os sistemas de inferências fuzzy apresentam uma arquitetura onde há um fuzzificador, as regras relevantes, o mecanismo de inferência, o defuzzificador e os conjuntos de dados iniciais e finais (entrada e saída).

3.2.1. Fuzzificação

Nesta primeira etapa do Sistema de Inferência *fuzzy*, o problema é analisado e os dados de entrada são transformados em variáveis linguísticas. Neste momento, é de extrema importância que todos os dados de imprecisão e incerteza sejam considerados e transformados em variáveis linguísticas. Após esta transformação, as funções de pertinência também são determinadas.

É recomendável que especialistas da área do problema em questão sejam consultados durante a atribuição de valores relacionados aos graus de pertinência para cada uma das variáveis em estudo.

3.2.2. Inferência

Uma vez que os dados passam pela etapa de fuzzificação, nesta segunda etapa tem-se o momento em que são aplicadas as regras ou proposições através da associação das variáveis linguísticas já criadas.

A finalidade desta fase é relacionar as possíveis variáveis entre si, através de regras pré-estabelecidas, cumprindo-se assim os objetivos do algoritmo.

Conforme Cox (1994), as proposições são geradas do relacionamento entre as variáveis linguísticas do modelo e as operações entre conjuntos *fuzzy* conforme foi visto na seção 3.1.1 deste capítulo. Essas regras resultantes das associações podem ser condicionais ou não condicionais.

Segundo von Altrock (1996), esta fase pode ser dividida em dois componentes, visualizados na Figura 3.3 e denominados agregação e composição. O primeiro diz respeito à chamada parcela “Se” das regras

que irão reger o processo de inferência, e o segundo, refere-se à parcela "Então" do conjunto de regras chamadas Se-Então.

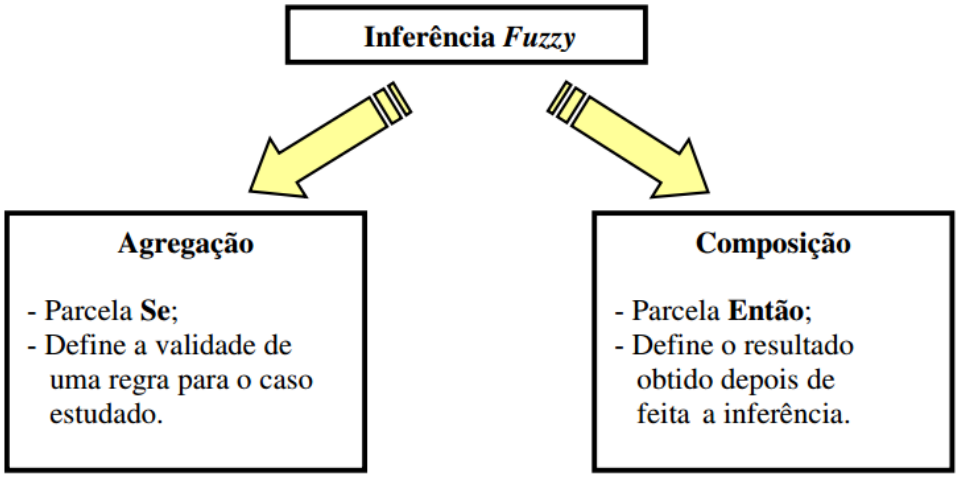


Figura 3.3: Inferência Fuzzy. Fonte: Jané (2004).

Tais componentes compõem o chamado processo de inferência *fuzzy*, controlando as relações entre variáveis linguísticas através de seus respectivos operadores lógicos. Os agentes implementados neste trabalho permitem a configuração do método de inferência a ser usado em cada caso, como pode ser visto no capítulo 4, seção 4.3.2.

3.2.3. Defuzzificação

É a última etapa do sistema de Inferência *fuzzy*, divergindo alguns pesquisadores quanto ao seu conceito. Segundo Cox (1994), a defuzzificação é a etapa em que os valores *fuzzy* são convertidos em números reais. Von Altrock (1996), por sua vez, a define como a tradução do resultado linguístico do processo de inferência *fuzzy*, em um valor numérico.

O problema é como determinar esse número. Existem diversos métodos para determinar tal número e a escolha do método deve ser feita com muito cuidado, pois ajudará a determinar, significativamente, a acurácia e a velocidade do sistema *fuzzy*.

Silveira (2002) cita que dois dos métodos mais utilizados atualmente são o centro de área (COA) e a média de máximo (MOM). Assim como na inferência, o presente trabalho permite a configuração de como será realizada a defuzzificação em cada caso, como será visto no capítulo 4, seção 4.4.2.

3.3. Implementação do Agente Embarcação

Neste trabalho, a implementação dos agentes virtuais de embarcação foi feita usando uma linguagem visual baseada em nós, disponível no software MASSIVE, que é baseada na lógica *fuzzy* para definir características individuais para cada instância desses agentes. Todos os módulos implementados para construção deste agente estão representados na Figura 3.4.

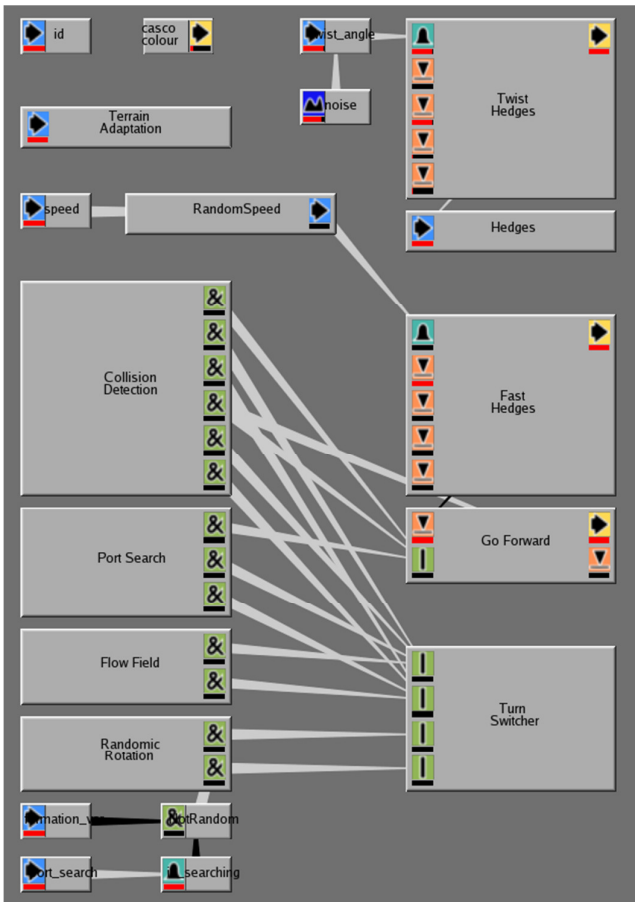


Figura 3.4: Módulos do agente embarcação usando lógica fuzzy. As caixas cinza-claro na imagem são todos os módulos implementados para este agente.

A introdução sobre a linguagem visual baseada em nós para definição dos cérebros dos agentes e o detalhamento dos principais módulos implementados estão apresentados no restante deste capítulo.

3.3.1. Linguagem Visual Baseada em Nós

O ambiente de programação visual disponível para definição dos cérebros dos agentes neste trabalho possui oito tipos de nós, que possibilitam a implementação de todas as funcionalidades que os agentes executam neste trabalho.

O nó "input" é responsável por entrar com os dados numéricos no ambiente de edição do cérebro do agente. Essa entrada de dados pode ser tanto a partir das variáveis de entrada do agente como pode ser a combinação de outros valores obtidos no cérebro a partir de expressões que podem ser usadas neste nó.

O nó "timer" gera um valor que varia de acordo com o tempo, que pode ser usado como entrada em algumas situações específicas. Este temporizador pode ser cíclico ou infinito.

O nó "noise" gera um valor aleatório entre zero e um, que pode ser alterado de acordo com uma taxa ao longo do tempo.

O nó "fuzz" é responsável pela fuzzificação citada na seção 3.2.1, ou seja, é responsável pela definição de uma função de pertinência relacionada a uma variável linguística para avaliar algum valor de entrada.

Os nós "or" e "and" realizam, respectivamente, as operações de união e interseção especificadas na seção 3.1. Esses nós também podem ser usados para realizar operações matemáticas de soma ("or") e produto ("and"), definindo a etapa de inferência vista na seção 3.2.2.

O nó "defuzz" é responsável por definir os valores reais para cada conjunto fuzzy criado a partir das regras, conforme a etapa de

"defuzificação" descrita na seção 3.2.3, ou seja, associa um valor de saída a uma variável linguística para um nó de saída.

Finalmente o nó "output" é responsável por decidir e executar um método de defuzzificação que atua sobre um conjunto de nós "defuzz". Este método pode ser escolhido dentre três opções: Average (média ponderada de todos os nós "defuzz"), Max (o valor total do nó "defuzz" com maior grau de pertinência conectado ao nó "output") e Blend (o valor do nó "defuzz" com maior grau de pertinência conectado ao nó "output" multiplicado por esse grau).

A representação gráfica dos nós presentes no framework para edição do cérebro dos agentes está apresentada na Figura 3.5.

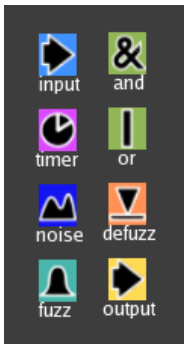


Figura 3.5: Nós da linguagem baseada em lógica fuzzy.

3.3.2. Adaptação ao Terreno

Os agentes tipo embarcação, implementados neste trabalho, possuem um módulo de adaptação das embarcações ao terreno em que elas navegam (oceano simulado), como pode ser visto na Figura 3.6.

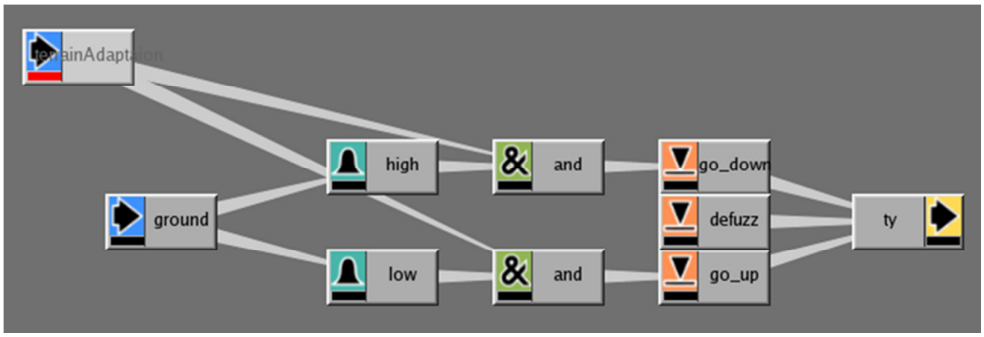


Figura 3.6: Módulo de adaptação ao terreno dos agentes embarcação.

A adaptação do terreno é feita a partir de uma informação de entrada chamada “ground”, que representa o valor da distância entre um agente e o terreno abaixo dele (essa informação é gerada pelo MASSIVE para cada instância de agente). A Figura 3.7 mostra as curvas que representam os nós “fuzz” deste módulo.

Figura 3.7: Curvas que definem os nós “fuzz” high e low da Figura 3.6. A curva que representa o nó low está para o lado esquerdo do zero (representado pela barra azul em pé), já a curva que representa o nó high é a que está ao lado direito do zero.

A adaptação só ocorre caso a variável de adaptação do terreno esteja ativada na instância do agente. Caso esteja, existem dois nós *fuzzy* representando as faixas de valores acima e abaixo do terreno (nós fuzzy “high” e “low” respectivamente), dependendo qual esteja ativa, o defuzzificador respectivo é ativado (“go_down” para quando o agente está acima do terreno e “go_up” para quando está abaixo), então, o valor de translação vertical (ty) é atualizado com o valor real com unidade que foram definidas para deslocamento para cima e para baixo. Estes passos fazem com que o agente esteja sempre acompanhando o nível do terreno em que foram simulados (no caso, oceano). Finalmente o nó “defuzz” que

está sobrando funciona como *else* no nó de saída e seu valor é zero, portanto se nenhuma regra estiver ativa, ele garante que o valor de saída é zero.

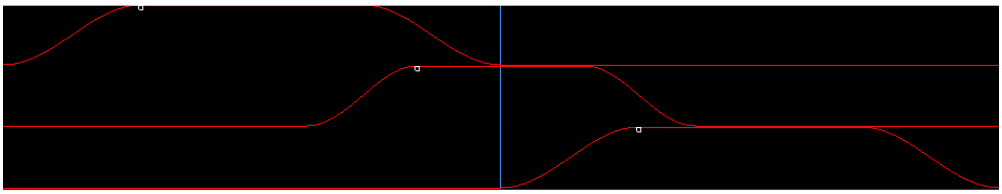
3.3.3. Detecção de Colisão dos Agentes

Ambas as detecções de colisões e a busca por portos usam a detecção de localização dos agentes baseada em modelos de frequência de som disponíveis na linguagem de programação visual do cérebro dos agentes.

Um valor de frequência é atribuído a cada agente (sound.f)[13], sendo possível definir uma frequência geral para cada tipo de agente (como são os agentes humanos e embarcações) ou para cada instância de um agente, definir diferentes frequências (caso dos agentes tipo porto).

Outras informações de entrada para a detecção de localização são a distância entre o agente emissor e o agente receptor da frequência (sound.d), assim como a coordenada polar horizontal do emissor em relação ao receptor (sound.x).

São criados então os conjuntos de verificação, onde são definidas as curvas para cada uma das informações de entrada, que são a faixa de frequências a serem reconhecidas, a distância máxima de busca e em qual direção procurar. Na Figura 3.8 é possível observar curvas que representam a coordenada polar horizontal dos agentes embarcação (sound.x).



[13] No software Massive, agentes podem emitir campos esféricos de som que podem ser captados por outros agentes. Estes sons podem ser especificados em termos de frequência e amplitude. Baseado no som que recebe, o agente pode estimar a posição de um outro agente.

Figura 3.8: Curvas que representam os nós "fuzz": *left*, *front* e *right*, respectivamente. A barra central azul representa o ângulo zero horizontal (posição central na frente da embarcação).

A resposta de um conjunto de verificação normalmente está atribuída a um nó de decisão, que vai ativar uma decisão de acordo com a situação descrita por esse conjunto. Por exemplo, se um agente porto foi encontrado muito perto e à esquerda de um agente embarcação, então este agente embarcação tem a sua decisão de virar para direita ativada. A Figura 3.9 mostra o módulo de detecção de colisão do agente embarcação com os agentes tipo porto.

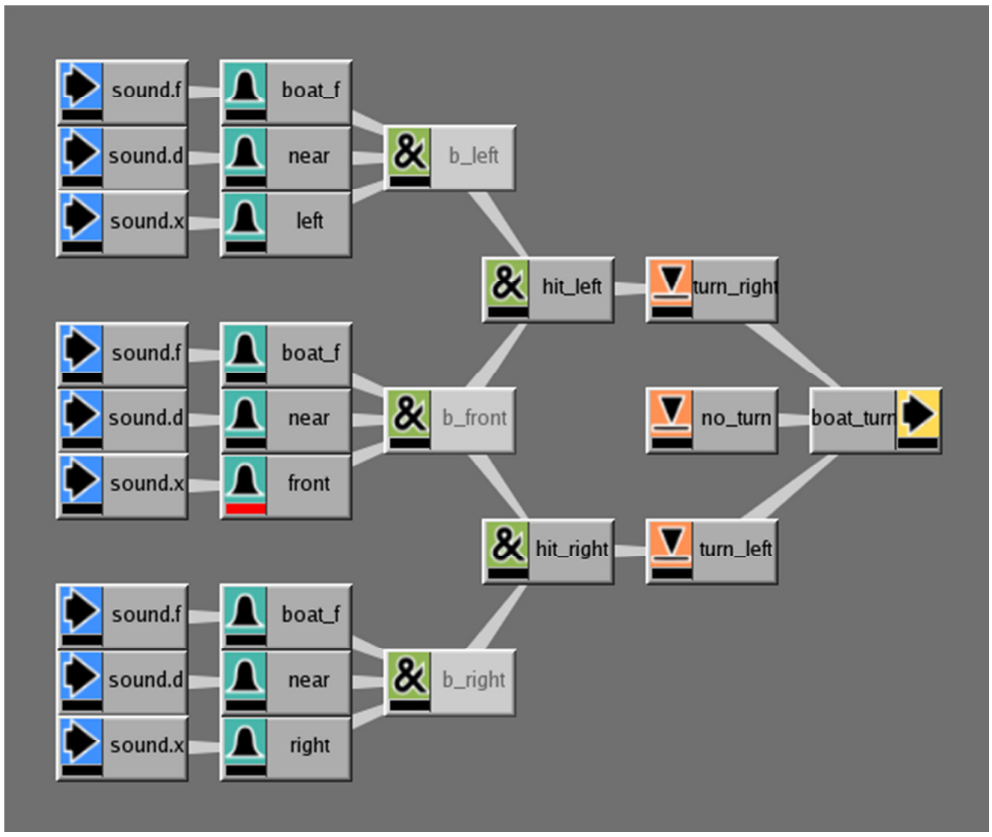


Figura 3.9: Módulo de detecção de colisão do agente embarcação.

Os módulos de detecção de colisões estão sempre ativos nos agentes embarcação, para evitar que ocorra sobreposição de agentes no espaço durante a simulação. As curvas da Figura 3.8 mostram o comportamento dos nós "fuzz*" left, front, right*, respectivamente, que estão

na Figura 3.9. Ainda nesta figura, é possível ver que todas as regras geradas verificam a frequência de uma embarcação (boat_f) e a proximidade da mesma (near), e considerando os três conjuntos fuzzy (esses dois mais o que representa a direção, aplica-se a interseção sobre todos estes, formando três conjuntos (b_left, b_front e b_right) que indicam se existe outra embarcação por perto, à esquerda, à frente e à direita, respectivamente. Outra operação é feita fazendo a interseção do b_front com o b_left (indicando que é necessário ativar a curva para direita) e a interseção do b_front com o b_right (que indica a ativação da curva para esquerda). O valor no_turn é zero e serve como else para o nó "output" que define o quanto a embarcação tem que virar. Os valores reais de turn_right e turn_left são 5º e -5º respectivamente. A defuzzificação foi feita usando a média entre os nós "defuzz". Todos os nós de interseção usaram o mínimo para inferência dos conjuntos.

Já no módulo de busca por portos, existe um nó de entrada para habilitar ou não a definição de um porto de destino para o agente embarcação ("port_search"). Este porto de destino é definido através de outro nó de entrada que define a frequência que deve ser procurada ("port_id"). Como cada instância de agente porto deve possuir um identificador único, os agentes embarcações que buscam por portos sempre terão um único porto como destino.

3.3.4. *Hedges* de Rapidez e Turbulência

Os controladores mais importantes para a simulação das embarcações implementadas neste trabalho são os de controle de velocidade e balanço dos navios. Para estes, foram implementados dois módulos que usam *hedges* (como vistos na seção 3.1.3) para definir a intensidade das características de rapidez e turbulência.

Os modificadores (*hedges*) implementados para ambas as situações foram os intensificadores "muito" e "extremamente" e o diluidor "mais ou menos". Estes *hedges* são escolhidos através da variável correspondente ao módulo ("fastness_type" para o módulo de rapidez e

"turbulence_type" para o de turbulência). Essa variável define se algum modificador estará ativo, e, em caso afirmativo, qual será este. A Figura 3.10 mostra a organização interna do módulo que implementa os *hedges* para a característica de rapidez.

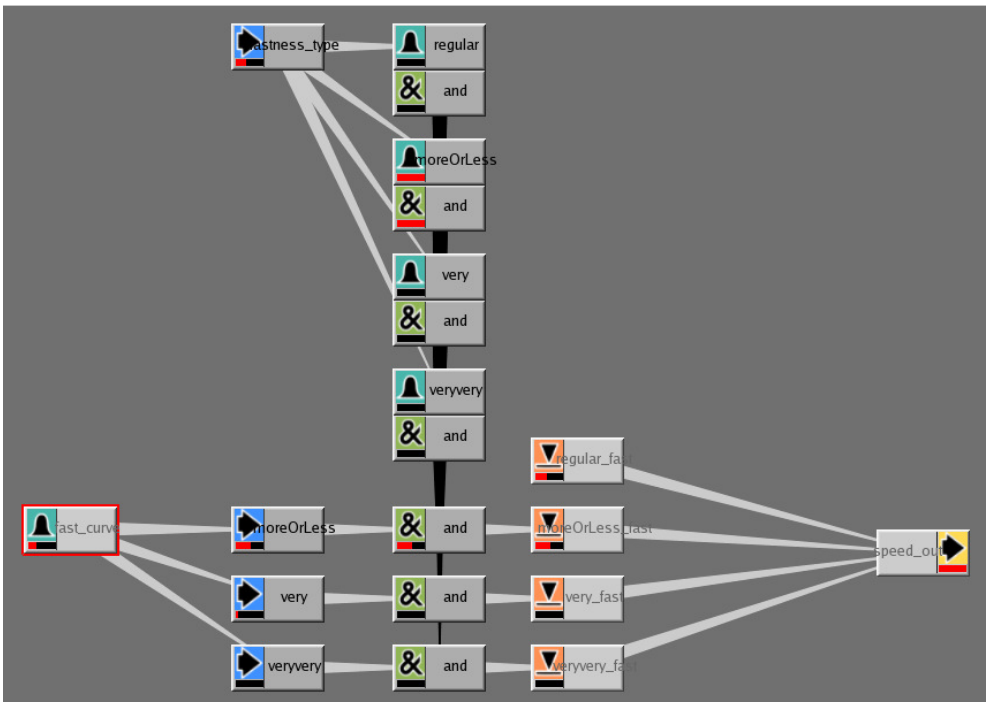


Figura 3.10: Módulo que implementa os *hedges* para controlar a rapidez do agente.

Assumindo que um valor de velocidade está sendo usado para alimentar o conjunto *fuzzy* "rápido" (representada pelo nó *fuzzy* "fast_curve" na Figura 3.8), foram usados nós de entrada que transformam o resultado da função de pertinência de acordo com o *hedge* escolhido. A Tabela 1, a seguir, relaciona os *hedges* com as funções de pertinência após a aplicação de cada modificador (μ_A) em um conjunto *fuzzy* original (μ_B). Por exemplo, se uma entrada no nó fuzzy "fast_curve" obtém a saída com valor 0.4 na função de pertinência, o nó "moreOrLess" terá valor 0,632 (que é igual $(0,4)^{1/2}$). Deve-se considerar como representação dessas curvas, algo como foi mostrado nos exemplos da Figura 3.1 da seção 3.1.3, tanto para os intensificadores ("very" e "very very") quanto para o diluidor ("moreOrLess").

Tabela relacionando as funções de pertinência aos seus respectivos *hedges*.

Hedge	**Função de pertinência**
mais ou menos	$\mu_A(x) = [\mu_B(x)]^{1/2}$
muito	$\mu_A(x) = [\mu_B(x)]^2$
extremamente	$\mu_A(x) = [\mu_B(x)]^4$

Usando os *hedges* implementados, é possível determinar, de forma simples, que um grupo de instâncias dos agentes embarcação, por exemplo, devem se mover “mais ou menos rápido” ou “extremamente rápido”, ou devem estar “muito turbulentos” ou “mais ou menos turbulentos” (para simular um mar revolto).

4.
Workflow

De uma forma geral, o processo de criação de cenas virtuais usando o método proposto para simulação em massa de agentes para cenas de mar, pode ser dividido em cinco etapas principais: modelagem tridimensional, simulação de oceano, criação e simulação dos agentes, renderização e composição. A Figura 4.1 apresenta uma versão simplificada e sequencial do *workflow*. Na realidade, o *workflow* deve ter várias atividades concorrentes e interdependências mais elaboradas.

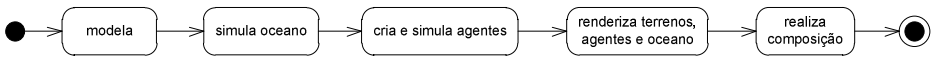


Figura 4.1: Workflow simplificado com as principais etapas da simulação

A partir de conceitos de cena recebidos por uma equipe de produção virtual (equipe responsável por construir e reproduzir todos os elementos de cenas que necessitem ser gerados virtualmente), a fase de modelagem tridimensional pode ser iniciada, podendo sofrer ajustes ao longo da produção. Nesta etapa também são produzidos alguns elementos para auxiliar na etapa de modelagem do *workflow*[14] proposto.

Para obter todos os elementos da modelagem tridimensional necessários para a etapa de customização dos agentes, há a necessidade de integração com um sistema de geração de ambientes virtuais. Neste trabalho, propõe-se o uso do Autodesk Maya para realizar esta integração. Esta integração também foi usada para gerar o mar

[14] No presente trabalho, ficou mantida a palavra em inglês "workflow" por ser um termo muito usado pela comunidade profissional no Brasil.

utilizado nas simulações, que nesta dissertação é uma extensão da simulação de oceanos proposta por Silva (2010).

O desenvolvimento das partes de customização[15] dos agentes e simulação deste *workflow* tem como base o MASSIVE Software, que possui um modo de programação visual com ferramentas capazes de manipular a inteligência artificial dos agentes usando lógica *fuzzy*. Neste software foram gerados os cérebros dos agentes que definem todos os tipos de interação das possíveis simulações que podem ser obtidas seguindo este *workflow*.

Quando todas as simulações necessárias para a cena estão concluídas, as respectivas sequências de imagens são geradas a partir dos dados de cada uma dessas simulações. Nesta etapa, que é chamada de renderização, indica-se o uso do renderizador Pixar Renderman[16], que, nesta etapa, define todos os materiais associados aos objetos tridimensionais criados no fim da etapa de modelagem tridimensional.

Finalmente, na etapa de composição, todas as sequências de imagens obtidas na etapa de renderização são compostas usando o The Foundry Nuke[17]. O resultado desta composição é a sequência final de imagens que será enviada para os outros setores de uma produção de TV/Cinema, completando o processo do *workflow* proposto.

Este capítulo procura detalhar cada uma das etapas citadas acima, apresentando um *workflow* mais completo. Diagramas de atividades UML são utilizados para descrever o *workflow*.

[15] Novamente, nesta dissertação, aceitam-se determinados anglicismos por não haver traduções fiéis ao significado. O termo “customizar” significa modificar e/ou adaptar algo de acordo com preferências individuais ou particulares. O seu significado vai além da idéia de “adaptar ao gosto de alguém”; e o termo “personalizar” não é adequado.

[16] www.renderman.pixar.com

[17] http://www.thefoundry.co.uk/products/nuke-product-family

4.1. Notações Usadas nos Diagramas de Atividades

Na presente dissertação, adotam-se os diagramas de atividade UML 2.4 (OMG 2011) como sendo a base para a descrição do *workflow*. Esta escolha é adequada para representar os fluxos de ação presentes no método. A seguir, são apresentadas as notações utilizadas pelo *workflow*, nas quais foram introduzidas pequenas extensões para caracterizar alguns conceitos de produção. Os conceitos originais do padrão UML 2.4 estão indicados em negrito. Estas notações podem ser acompanhadas pelo diagrama da Figura 4.3, sem a necessidade de se entender o significado de cada atividade no presente momento do texto. Na presente dissertação, os diagramas são apresentados com textos em inglês, pela mesma razão que se documentam códigos de computador em inglês: padronização de documentação internacional.

Atividades do nosso método são consideradas unidades fundamentais, cuja execução representa uma transformação ou um processamento, ou seja: são ações. Nesta dissertação, mapeia-se uma atividade a um nó de ação (**ActionNode**) representado por um retângulo com cantos arredondados. Na nossa extensão da notação UML, um nó de ação sempre tem o nome do programa, ferramenta ou software utilizado, que é introduzido através da nomenclatura de esteriótipo UML (**stereotype**). O esteriótipo UML é indicado por marcas de citação de ângulo duplo $\ll \cdots \gg$, também denominadas de *guillemets*. Nesta dissertação, toma-se a liberdade de incluir, na descrição da ação, os nomes gerais dos objetos que serão gerados pela ação (indicados por termos em itálico). Uma outra extensão é a indicação de mais de um objeto do mesmo tipo pelo sufixo (1...n). Por exemplo, no início da Figura 4.3, há um nó de ação com as seguintes linhas: $\ll Maya \gg$, model terrains, *terrain_global*, *terrain_simplified*, *walls (1...n)*. Isto significa que a ação de modelar terrenos é realizada no software Maya e gerará um objeto do tipo terreno global, outro do tipo terreno simplificado e vários objetos do tipo paredes.

Produtos e documentos são representados por nós de objeto (**ObjectNode**) que, no *workflow* proposto, são arquivos de computador. Estes nós são indicados por retângulos com sombras e com o esteriótipo $\ll Files \gg$. Quando é mais de um arquivo referente a um mesmo tipo de objeto, acrescenta-se o número 1 (ou _1) ao nome do objeto seguido do sufixo (1...n). Por exemplo, boat1.obj (1...n) refere-se a vários arquivos do tipo boat: boat1.obj, boat2.obj, ...

Ações são conectadas por arestas de atividade (**ActivityEdges**) direcionadas (i.e. com setas). Uma aresta de atividade admite, opcionalmente, um guarda (**guard**) que denota uma condição que deve ser satisfeita ou produto que deve estar disponível para que o fluxo continue através da aresta. Um guarda é sempre apresentado entre cochetes. Por exemplo, [character concepts] são conceitos artísticos sobre todos os personagens da simulação com suas várias roupas.

Os círculos negros têm a conotação usual de pontos de início (**ActivityInitialNode**), totalmente negro, e fim do processo (**ActivityFinalNode**), dois círculos concêntricos.

Atividades (i.e. ações) distribuídas podem ser agrupadas em partições de atividade (**ActivityPartitions**), também conhecidas por **swimlanes**. Uma partição é indicada por um grande retângulo envolvendo várias atividades e com um cabeçalho na forma de um pequeno retângulo. Na Figura 4.3 a primeira partição é a de “Modeling”, que engloba todas as atividades de modelagem. Uma partição está associada a um grupo de profissionais com o mesmo tipo de especialidade.

Ramificações em atividades são especificadas por nós de decisão (**DecisionNodes**) que podem ter uma ou mais arestas de atividade entrando e saindo. Todas as arestas de saída devem ter guardas (**guards**) especificando condições de ramificação. Os nós de decisão são indicados por pequenos losangos.

Atividades em paralelo, simultâneas, são especificadas com nós de bifurcação (**ForkNodes**) e de junção (**JoinNodes**) que são indicados por barras negras.

Notas podem ser associadas a atividades ou arestas de ação através de um símbolo de nota que se liga ao elemento em questão através de uma linha pontilhada. A nota é indicada por um retângulo com uma dobra no canto superior direito. Notas são usadas como itens de esclarecimento, detalhamento ou de especificação.

4.2. Modelo do Workflow

4.2.1. Modelo Macro do Workflow

A Figura 4.2 apresenta um modelo macro do *workflow*, onde logo após a modelagem 3D de pessoas, barcos e terrenos, materiais são atribuídos (*shaders*). Logo após a atribuição de materiais, a atividade de renderização de *frames* de terreno pode ser concluída (*render terrain frames*) em paralelo com as atividades de criar, simular e renderizar[18] agentes. A simulação do oceano é condição para simular os agentes. As três renderizações (terreno, agentes e oceano) convergem em um nó de junção para haver a composição final.

Quando o *workflow* proposto se refere a terrenos, está implícita a existência de apenas dois terrenos principais: terreno global e terreno simplificado. No terreno global estão modelados todos os elementos (a região que será preenchida pelo oceano, as praias, os portos, as edificações, as ruas, os morros e os vales). No terreno simplificado há apenas as partes onde existirão movimentos de agentes (normalmente uma parte do oceano e um ou dois portos), ficando o restante do terreno representado por uma malha simplificada.

[18] renderizar – gerar imagem com qualidade superior à de visualização, baseado em técnicas que simulam o comportamento de cameras, luzes, materiais e todos os outros requisites que compõem uma cena.

Uma observação importante é que uma cena (*scene*) é composta por várias tomadas (*takes*). Por sua vez, cada tomada tem uma câmera e um conjunto de luzes. Desta maneira, na Figura 4.2 estão salientadas as condições de simular o oceano para cada tomada e simular os agentes para cada tomada. Um exemplo de cena seria a navegação de uma esquadra de embarcações com duas tomadas (uma tomada geral de longe, como na Figura 1.2a, e um close de embarcação, como na Figura 1.2b). Para cada uma destas tomadas há uma câmera e um conjunto de luzes apropriados. Neste exemplo, as tomadas visualizam apenas parte do terreno global (a parte do oceano sem terra firme). Esta questão de terrenos está apresentada com detalhes na seção 4.3.3.

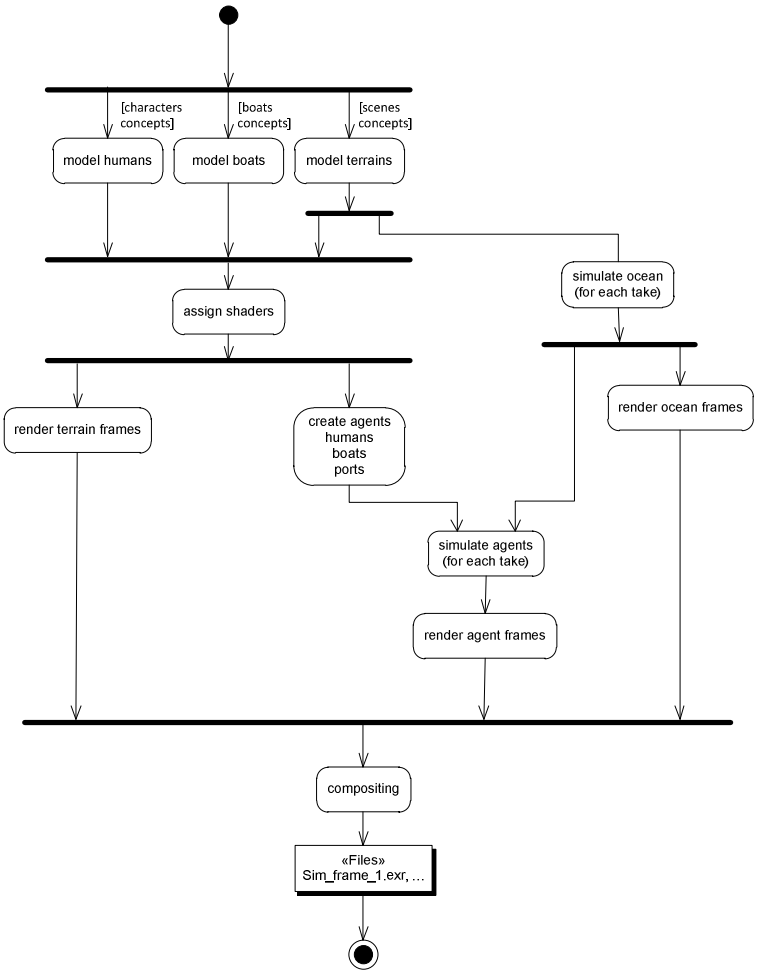


Figura 4.2: Modelo macro do *workflow*

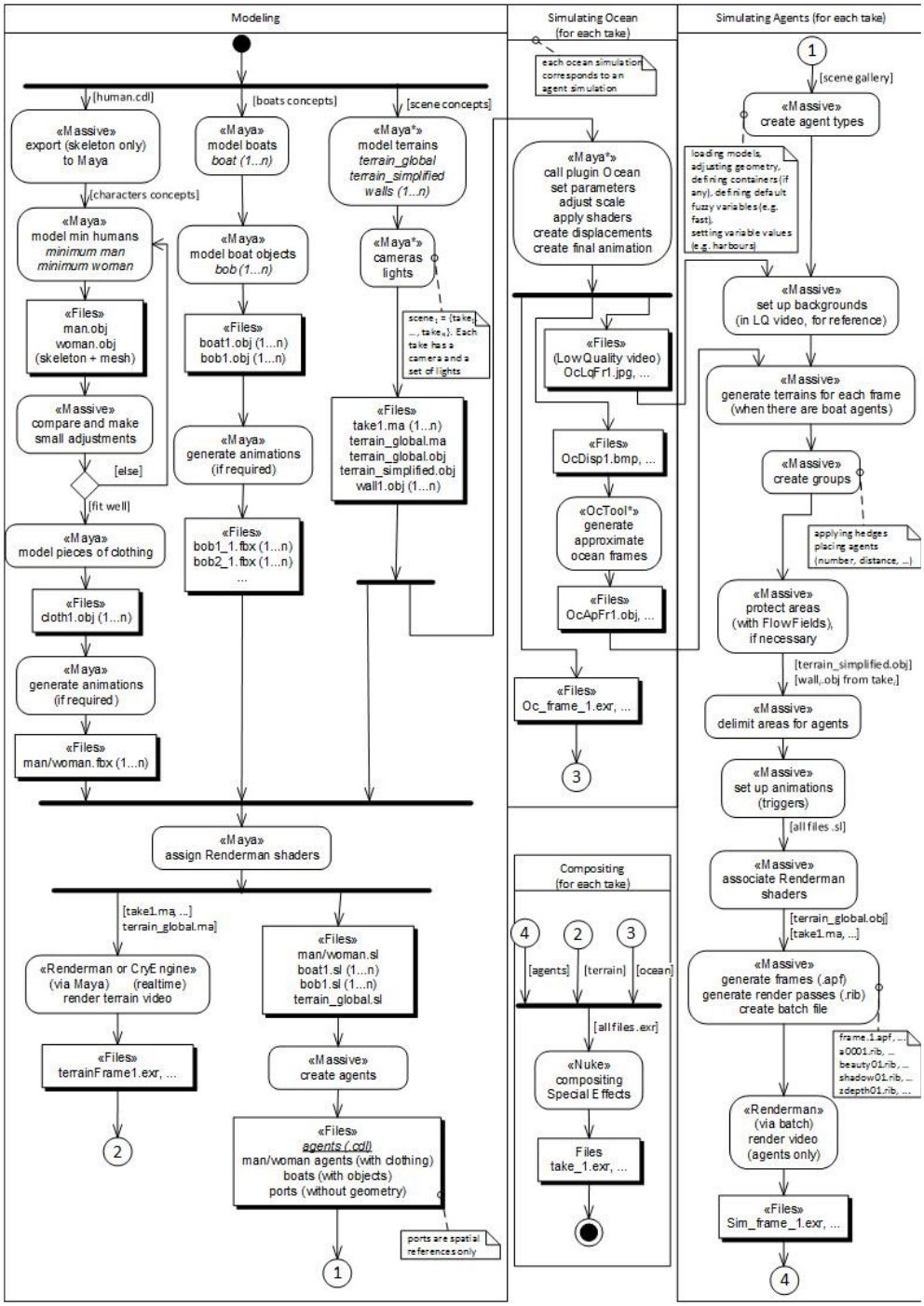


Figura 4.3: *Workflow* detalhado

4.2.2. Modelo Detalhado do Workflow

A Figura 4.3 apresenta o modelo detalhado do *workflow*, com quatro partições: Modelagem, Simulação de Oceano, Simulação de Agentes e Composição. A menos da primeira partição, as 3 restantes são realizadas para cada tomada (*take*). Este modelo é descrito no restante do presente capítulo. Este capítulo procura detalhar cada uma das etapas citadas.

4.3. Modelagem Tridimensional

Para começar a etapa de modelagem tridimensional, há necessidade de um material de referência (*concepts*) para garantir que a cena esteja adequada com a necessidade da produção envolvida. Esse material deve ser composto por desenhos, fotos e vídeos do ambiente, pessoas e qualquer outro tipo de agentes ou objetos que façam parte da cena em produção.

Este trabalho está considerando, como modelagem tridimensional, a geração de todos os insumos necessários para abastecer as etapas seguintes do *workflow*. Estes insumos são: as malhas tridimensionais, o mapeamento para texturização dessas, as texturas, a configuração dos *shaders*, e, nos casos que houver necessidade, a geração de animações extras para incrementar as animações básicas disponíveis nos agentes disponibilizados.

4.3.1. Modelos dos Agentes Humanos

A etapa de modelagem dos agentes humanos requer a disposição inicial do esqueleto do agente que está em fase de modelagem. A Figura 4.4 mostra a disposição inicial do esqueleto do agente humano usado.

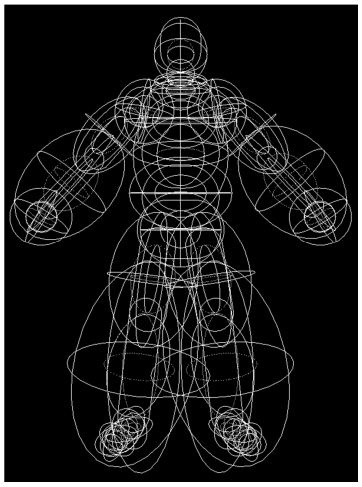


Figura 4.4: Disposição inicial do esqueleto do agente humano.

A partir dessa disposição inicial de esqueleto, o artista sabe em que posição deverá criar cada uma das malhas tridimensionais necessárias de acordo com os *concepts*, que em geral, no caso de agentes, são malhas que representam: a pele do corpo do humano (com variações de gênero, faixa etária, raça, entre outros), diversas roupas, calçados e acessórios.

A geração desses diferentes elementos requer uma integração com MASSIVE para garantir que estes modelos estão funcionando de acordo com a disposição espacial do esqueleto, para verificar se as proporções da malha não excedem a região de cobertura do “skinning” (processo interno do MASSIVE para animar a malha virtual de acordo com os movimentos do esqueleto). É indicado que sejam produzidos diversos tipos de texturas para cada elemento, assim garantindo maior individualidade na aparência dos agentes humanos a serem customizados.

No caso de uma cena precisar de animações específicas, o esqueleto da disposição inicial deve ser usado para gerar as animações, e com este, cada animação extra gerada no Autodesk Maya deve ser salva em um arquivo “.fbx”. Estes arquivos com as animações serão usados na etapa de customização dos agentes humanos.

4.3.2. Modelos dos Agentes Embarcações

A modelagem das embarcações só depende da conceituação dos barcos necessários para a cena. O trabalho de ajuste de um agente embarcação é feito na etapa de customização dos agentes (seção 4.3.4).

Além do modelos básico do navio, outros objetos opcionais podem ser modelados para serem usados na customização desse agente, como velas, mastros, cercas, etc. Esses objetos ajudam a aumentar a variação das instâncias dos agentes embarcações na cena. Elaborar um grande número de texturas para cada tipo de embarcação e seus objetos também colaboram para aumentar essa variação.

O modelo base do agente embarcação (casco) não possui animação, pois seus movimentos são definidos pela inteligência artificial contida no "cérebro" implementado no agente embarcação de acordo com o mar gerado pela simulação de oceano (que é detalhada no capítulo 5).

Os objetos opcionais deste agente, no entanto, podem ser animados. Neste caso, é necessário que o esqueleto que representa o agente embarcação possua articulações que sirvam para o controle dessas animações. É necessário também que este esqueleto esteja disponível na etapa de modelagem das embarcações para que as animações sejam geradas no Maya, assim como acontece no agente humano.

4.3.3. Modelo Tridimensional do Terreno

A malha que define o terreno (ambiente virtual da cena) deve ser gerada com todos os elementos que sejam necessários virtualmente na cena. Ou seja, analisando os conceitos de cena, deve-se decidir quais elementos serão modelados. Dependendo da situação, a malha do terreno pode ser gerada com o auxílio do recurso de técnicas de fotogrametria, explicado na seção 2.2.

Além da modelagem da malha principal do terreno, é necessário extrair uma malha simplificada para a etapa de simulação. Outra necessidade desta etapa é a geração de *walls*, que são os limites que definem a área onde os agentes poderão circular no terreno, estas são representadas por malhas em forma de paredes que cercam essas regiões. Essas paredes permanecem invisíveis para as etapas seguintes, exceto para a simulação dos agentes humanos em terra.

Finalmente, com os modelos de terreno disponíveis, a câmera e luzes de cada *take* são geradas de acordo com os enquadramentos dados nas imagens dos conceitos de cena.

4.3.4. Customização dos Agentes

Nesta etapa são associados aos agentes, os insumos produzidos para estes na etapa de modelagem tridimensional. Além disso, é nesta etapa que são definidas as combinações das diversas malhas e texturas possíveis, por exemplo, nos agente humanos são definidas todas as combinações de vestuário disponíveis em cima de modelos mínimos de homem e mulher (as malhas referentes às regiões de pele, podem ser as mesmas em todas as instâncias de cada gênero).

Para o workflow propõe-se, para o agente humano, a utilização de uma versão modificada do “Mayhem Agent” do MASSIVE Software, por possuir um esqueleto e diversas animações básicas pré-definidas. Como o foco deste trabalho não é o desenvolvimento das animações, essa escolha visa facilitar a operação para geração de um novo agente com grande variedade de animações, sem a necessidade de gerá-las. Este agente já possui todas as ligações necessárias para o cérebro, bastando apenas ativar o que for necessário de acordo com o manual do agente humano.

A customização dos agentes tipo embarcação deve se basear em um agente padrão que foi desenvolvido para o workflow proposto nesta dissertação. É necessário que se faça um redimensionamento da

representação do casco (esqueleto) para o tamanho da malha que foi modelada para a embarcação que está sendo criada.

Nesta etapa ainda são inseridos, no processo, os agentes de porto, que serão usados na etapa de simulação como pontos de destino das embarcações. Estes agentes são posicionados nos pontos de destino das embarcações, de acordo com as conceituações de cena. A representação visual desses agentes é simbólica, ou seja, eles não aparecem na imagem final obtida após a simulação dos agentes, pois a renderização do terreno virtual já possui as imagens com os portos.

4.4. Simulação

As simulações de oceano e dos agentes são realizadas a partir dos insumos criados na etapa de modelagem tridimensional (seção 4.3). A simulação dos agentes embarcação depende da malha animada do oceano que é gerada na simulação do oceano. Assim nota-se que há uma dependência entre as simulações, facilitando o entendimento da ordem em que estas são detalhadas nesta seção.

4.4.1. Simulação de oceano

A simulação do oceano é feita por um plug-in para o Autodesk Maya, que foi desenvolvido para este trabalho baseado na técnica apresentada por Silva (2010). Esta simulação está descrita no capítulo 5.

Usando este plug-in, deve-se configurar os parâmetros da simulação e ajustar a escala do oceano gerado em relação ao modelo tridimensional do terreno. Nesta etapa, caso necessário, deve-se aplicar o *shader* de oceano para finalmente gerar os arquivos de saída do Maya nesta etapa, que são: a sequência de mapas de altura do oceano para cada frame e as renderizações em baixa e alta qualidade da simulação do oceano.

A renderização em baixa qualidade é usada como referência na simulação dos agentes (seção 4.4.2) e a renderização em alta qualidade é usada na etapa de renderização do oceano (seção 4.5).

Usando a ferramenta OcTool, implementada em parceria com Silva (2010) para este trabalho, são geradas as malhas virtuais que representam o oceano a cada *frame*. Essas malhas também são usadas na simulação dos agentes embarcação (seção 4.4.2). Mais detalhes sobre o funcionamento e implementação desta ferramenta estão descritos na seção 5.3.

4.4.2. Simulação dos agentes

A etapa de simulação dos agentes começa na configuração das variáveis destes (a descrição de como usar essas variáveis está nos manuais dos respectivos agentes). A definição dos valores dessas variáveis que controlam o tipo de comportamento que cada agente terá quando o agente for simulado. Cada configuração de variáveis salvas geram um tipo de agente diferente.

Quando há simulação de agentes tipo embarcação, deve-se usar a saída em baixa qualidade gerada na simulação de oceano (seção 4.4.1), para verificar se a simulação do movimento das embarcações está acompanhando a animação da simulação de oceano. Nesta etapa deve-se usar, como terreno de simulação, a sequência de malhas de oceano (geradas pelo OcTool), que dão a movimentação do oceano ao longo do tempo, como foi dito na seção anterior.

A partir de cada tipo de agente, é possível criar grupos com agentes do mesmo tipo que possuam somente algumas variações básicas (como a manipulação dos *hedges* de rapidez e de balanço nos agentes tipo embarcação, conforme a seção 3.3.4). Nesta etapa devem ser configurados também todas as regras *fuzzy dos agentes embarcação*, ou seja, definir como serão realizadas as operações de inferência para

cada nó “and” e “or” do cérebro do agente com opções para: interseção (mínimo ou produto) e para união (máximo ou soma).

Para realizar a simulação dos agentes, é necessário que os mesmos sejam posicionados no cenário virtual (terreno ou oceano). Além disso, devem ser definidas as zonas de navegação dos agentes tipo embarcação (isolando as regiões onde os barcos não podem navegar). A definição dessas zonas de navegação é feita usando *Flow Field* na malha do oceano. *Flow Field* é uma ferramenta do MASSIVE Software para desenhar campos de direção no terreno de simulação dos agentes.

Para a simulação dos agentes humanos em terra, são usadas os *walls*, que são as áreas de delimitação geradas na etapa de modelagem tridimensional do terreno (seção 4.3.3).

Caso haja alguma animação extra, esta deve ser importada para a galeria de ações do agente. É necessário também configurar os controles de disparo dessas animações usando o modo de edição de movimentos do software MASSIVE.

Após a definição de todos os passes de “render”, a etapa de simulação dos agentes gera as sequências de arquivos “.apf” de simulação e as de arquivos “.rib” de renderização. Ambas as sequências de arquivos são necessárias para a geração da sequência de imagens finais dos agentes na etapa de renderização (seção 4.5).

4.5. Renderização e Composição

A etapa de renderização do *workflow* proposto é composta de três fases isoladas: renderização de oceano, renderização do terreno (ambiente) e renderização dos agentes. Todas essas fases geram sequências de imagens com seus respectivos passos de vídeo (como take_xy_beauty_###.exr, take_xy_shadow_###.exr, etc...) que poderão ser usados na etapa de composição.

A renderização do terreno é realizada independente da etapa de simulação. Já a renderização do oceano e dos agentes dependem de suas respectivas simulações concluídas. Todos as três renderizações devem ser feitas usando Renderman, sendo que a renderização dos agentes é feita a partir de um arquivo batch gerado pelo software MASSIVE.

Finalmente, a etapa de composição completa o processo do *workflow* proposto, usando o The Foundry Nuke - um software de composição de imagens com ênfase em composição de cenas tridimensionais. O Nuke compõe, a partir das sequências de imagens geradas nas três etapas de renderização, uma sequência de imagens que é o produto final. Este produto normalmente ainda é enviado para outros setores de uma produção de TV/Cinema antes de serem exibidos, para outras tarefas de finalização, tais como sonorização e colorização. A Figura 4.5 mostra o ambiente de composição de uma cena usando The Foundry Nuke.

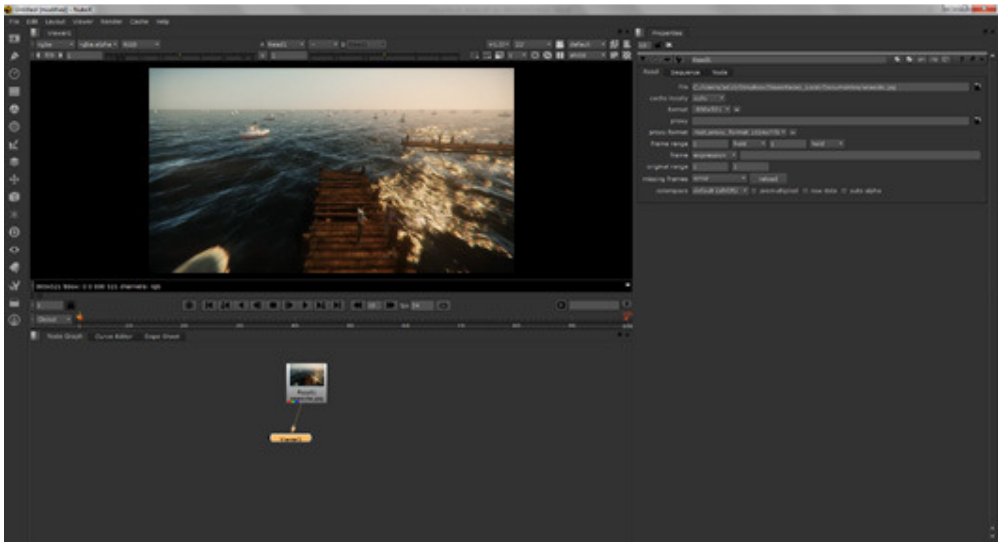


Figura 4.5: Ambiente de composição do Nuke.

5.
Simulação de Oceano

A simulação de oceano em tempo real proposta por Silva (2010) foi a base para a etapa de simulação de oceano desta dissertação. O trabalho de Silva (2010), que visa aplicações de tempo real (como jogos), teve de ser adaptado para ser aplicado a cenas para produções de TV e Cinema.

A simulação das ondas do oceano é um tema muito estudado em computação gráfica, subdividindo-se em duas vertentes principais: a simulação física, que tem como pilar as equações de Navier-Stokes (Sissom e Pitts, 2001), e a simulação que foca na reprodução do aspecto da superfície do mar.

Um modelo bastante realístico pôde ser obtido usando a equação de Navier-Stokes, a mais detalhada de todos os modelos de fluídos. Devido a esta característica e algumas simplificações dos termos, a equação de Navier-Stokes tem sido amplamente utilizada na computação gráfica para simulação do movimento da água (Chen e Lobo, 1995; Foster e Metaxas, 2000).

Contudo, um problema desse método é que ele trabalha em espaços locais, impedindo, com os recursos computacionais atuais, o seu uso em um ambiente de larga escala, como o oceano aberto, em taxas interativas. Mas essa restrição de desempenho em tempo real não é o foco da presente dissertação. Isso porque o presente estudo está concentrado na outra vertente, voltada para a reprodução do aspecto da superfície do mar, na qual se faz uma análise empírica da formação das ondas. Neste grupo de técnicas, incluem-se as simulações realizadas no espaço do tempo e no espaço da frequência.

As primeiras são comuns em quase todas as áreas de pesquisa, já que a análise do movimento ou a alteração de estado no decorrer do tempo são questões usuais e intuitivas. As simulações físicas de aceleração e velocidade são exemplos clássicos do uso do espaço do tempo. Dessa forma, pensar-se em analisar o movimento das ondas no tempo é a forma mais direta de simulação.

5.1. Simulação de oceano no espaço da frequência

Uma forma de aumentar o realismo e reproduzir mais fidedignamente o comportamento do oceano é inserir mais ondas harmônicas no oceano, fazendo os ajustes necessários. Porém, o aumento do número de harmônicas a serem simuladas causa uma significativa perda de desempenho, dificultando a sua utilização – o que, entretanto, não causa impacto no tipo de simulação previsto nesta dissertação.

As simulações no espaço da frequência, por sua vez, além de não causarem perda significativa de desempenho, são mais simples para a especificação das ondas harmônicas, utilizando-se a Transformada de Fourier e sua inversa.

A transformada de Fourier (Nilson & Riedel, 2001) é um caso especial da transformada de Laplace (Boyce & DiPrima, 2001) bilateral, na qual a parte real da frequência complexa é nula. Dessa forma, ela analisa fenômenos em regime (não transiente).

A interpretação física da mesma pode ser encarada como o limite da série de Fourier, que consiste em uma série que representa uma função em termos de componentes harmônicas. Ser o limite da série nesse caso indica que essa transformada leva qualquer função no domínio do tempo para o domínio da frequência.

Não obstante, há três funcionalidades importantes dessa transformada que justificam o seu uso no problema em análise. A

transformada é inversível, ou seja, pode-se especificar várias frequências no domínio da frequência e retornar o resultado no domínio do tempo. Além disso, pode-se utilizar a técnica de transformada rápida de Fourier (FFT) para aumentar a eficiência da técnica. Por fim, como a convolução de funções no espaço da frequência é muito mais simples que no espaço do tempo, pode-se aplicar filtros sobre as frequências para melhorar análises e coibir problemas de ruídos.

Um exemplo da última funcionalidade citada pode ser entendido quando se deseja analisar somente uma faixa de harmônicas, sendo assim, pode-se especificar um filtro passa-faixa, o qual será convoluido com as harmônicas e irá reduzir a amplitude das harmônicas fora da faixa especificada, isto sem mudar os parâmetros das mesmas.

Apesar de intuitiva a especificação de cada frequência para a computação da FFT, é extremamente difícil a definição das mesmas a fim de se obter um resultado realístico. Além disso, a grande quantidade de harmônicas no oceano é um fator impeditivo.

Sendo assim, pode-se utilizar modelos estatísticos combinados com observações experimentais. Nesses, a amplitude das ondas é considerada uma variável aleatória da posição e tempo [h(x,t)].

Esses modelos estatísticos são elaborados para terem a capacidade de se decompor em séries de senos e cossenos, o que além das várias propriedades matemáticas associadas, permite a utilização facilitada da FFT, que permite uma rápida avaliação das mesmas.

Dessa forma, a representação da amplitude da onda h(x,t) na posição x = (x,z) pode ser expressa como a soma das senóides complexas no tempo da seguinte forma:

$$h(x,t) = \sum_{k} \tilde{h}(k,t) e^{jk \cdot x}$$

Equação 1: Altura definida em termos da frequência

Onde t é o tempo, k é um vetor bidimensional, chamado de número de onda, sendo $k = (k_x, k_z), k_x = \frac{2\pi n}{L_x}, k_z = \frac{2\pi m}{L_z}$, e m, n são números inteiros no intervalo $n \in \left[-\frac{N}{2}, \frac{N}{2}\right)$ e $m \in \left[-\frac{M}{2}, \frac{M}{2}\right)$, L_x e L_z é o tamanho da região de simulação e N, M são o número de amostras. Esse processo gera um mapa de altura (Figura 5.1) nos pontos discretos $x = \left(\frac{nL_x}{N}, \frac{mL_z}{M}\right)$, os demais podem ser gerados, nesse caso, por interpolação. A função $\tilde{h}(k,t)$ descreve a estrutura da superfície que é discutida a seguir. Pesquisas oceanográficas demonstram que essa representação é adequada para ondas em mar aberto com vento (Tessendorf, 1999).

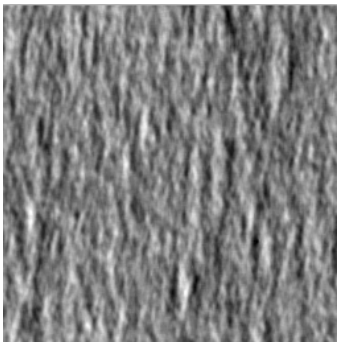


Figura 5.1: Mapa de Altura gerado pela IFFT

Para a geração dos efeitos visuais de iluminação, é necessária a especificação do vetor Normal da superfície. Em muitas aplicações de computação gráfica, dado um mapa de altura, uma forma de se gerar a normal é através da variação de Δx e Δz ao redor de um ponto do mapa, sendo muito eficiente em termos de memória. No entanto, essa abordagem oferece uma aproximação muito ruim para as variações de alta frequência presentes no mapa. Dessa forma, pode-se realizar o cálculo exato com o uso do gradiente do ponto e mais FFTs, sendo:

$$N(x,t) = \nabla h(x,t) = \sum_k jk\,\tilde{h}(k,t)e^{jk\cdot x}$$

Equação 2: Normal definida em termos da frequência

Conforme suscitado, L_x e L_z definem a região de simulação, e devido à definição de k, a função é periódica fora do intervalo, repetindo-se indefinidamente. Isso pode ser um problema caso essa repetição se torne visível para o usuário. Para evitar o problema aconselha-se utilizar grades de tamanhos grandes, que no caso de simulação em tempo real fica no intervalo de 128 a 512 por dimensão. No caso de simulações mais detalhadas ou para pós-produção (não tempo real como este trabalho), utiliza-se dimensões de 1024 a 4096.

Tensendorf (1999) relata que no filme Titanic e Waterworld foram utilizadas grades de 2048 x 2048. Além disso, relata que a utilização de número de ponto flutuante simples deve ser cautelosa para evitar ruídos indesejáveis. Sendo assim, caso seja necessária uma simulação com muitos detalhes, o uso de dupla precisão é necessário.

Mais detalhes sobre as FFT usadas neste trabalho, tais como o detalhamento do seu modelo estatístico e de gerações de ondas, podem ser vistos no trabalho de Silva (2010).

5.2. Plug-in para Maya

Um plug-in para o Autodesk Maya foi desenvolvido de acordo com as especificações do trabalho de Silva (2010), descrito na seção 5.1. O plugin utiliza o SDK em C++ do Maya e é capaz de produzir os mapas de altura e os mapas de normais usando a CPU com *multitread*.

A transformação dos planos do espaço da frequência para o espaço do tempo utilizam a biblioteca FFTW para números de ponto flutuante.

Para a realização desse processo de transferência de domínios, utiliza-se o diagrama da figura 5.2:

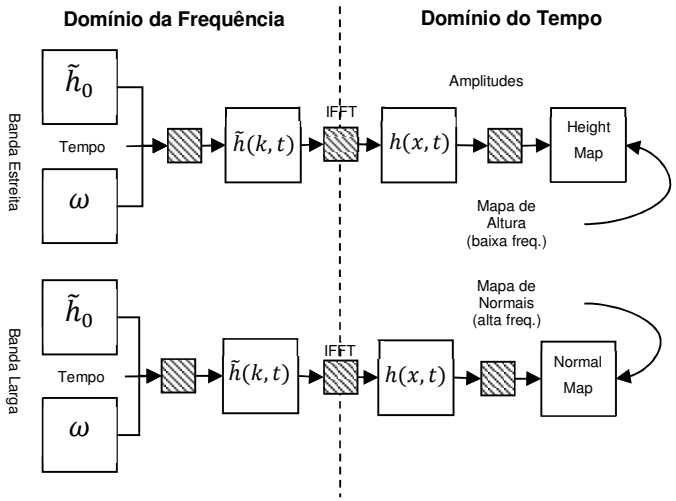


Figura 5.2: Simulação das Ondas com IFFT.

Assim, uma vez definidos os parâmetros de simulação, o sistema computa os planos no domínio da frequência (Equações 1 e 2) e aplica a transformada inversa para um determinado tempo, que pode ser animado (Figura 5.3), gerando os mapas no espaço temporal.

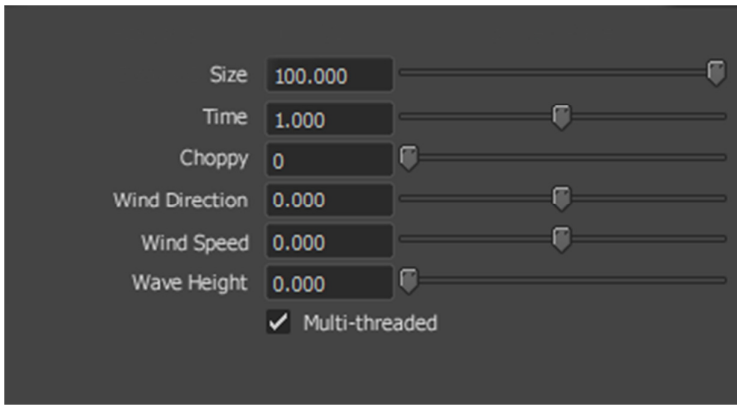


Figura 5.3: Parâmetros disponíveis do Plug-in para Maya desenvolvido na presente dissertação

Para facilitar o uso, foi criado uma ferramenta externa para gerar a sequência de imagens com as informações de deslocamento do oceano. Desta maneira, pode-se utilizar o sistema mesmo que não se tenha o Autodesk Maya.

A chamada da linha de comando para o help da ferramenta pode ser observada na Figura 5.4.

```
wavebuilder [options]

Options:
  -size [size]: Change the grid size. ex: -size 512
  -t0 [t0]: Change first time to write. ex: -t0 5.2
  -tf [tf]: Change last time to write. ex: -tf 10.2
  -step [step]: Change last time to write. ex: -step 0.1
  -choppy [choppy]: Change the wave choppyness. ex: -choppy 0.54
  -height [height]: Change the wave height constant. ex: -height 0.00000075
  -wind [wind]: Change the wind speed. ex: -wind 10
  -disp_prefix [disp_prefix]: Change the displacement output file prefix.
  -norm_prefix [norm_prefix]: Change the normal output file prefix.
  -dir [angle]: Change the wind direction. ex: -dir 45.0
  -tga : Save output in tga format
  -h : Show this help
```

Figura 5.4: Ferramenta de Geração de Imagens.

5.3. Ferramenta OcTool

Essa ferramenta usa a sequência dos mapas de altura gerados pelo plug-in para Maya para gerar a sequência de malhas de oceanos que representam o estado do oceano a cada frame da simulação. Esta sequência é usada na simulação dos agentes deste trabalho (como visto na seção 4.4.2). A ferramenta funciona como ponte de comunicação de dados entre as duas simulações.

O algoritmo dessa ferramenta foi implementado em C# e usa uma estrutura de dados simples, chamada “TexturePositionVertex”, que é usada para armazenar as coordenadas de textura (u, v) e sua posição espacial relacionada (x, y, z).

A função principal, responsável pela análise dos parâmetros de entrada, faz a chamada da função SaveTerrain(), que realiza todas as etapas para a geração de malha tridimensional do oceano a partir da entrada de um mapa de altura de oceano. Os demais parâmetros dessa função são os modificadores de altura (coordenada Y), escala (relacionada ao plano XZ) e o do fator *choppyness*, que insere um deslocamento no plano XZ nos vértices da malha gerada.

As informações de altura dos vértices são lidas a partir do canal de cor verde das imagens. Já as informações para o cálculo do

deslocamento usando o fator *choppyness* citado, são dadas de acordo com os outros canais de cor da imagem de entrada (vermelho e azul).

Uma vez definidas as posições de todos os vértices, o algoritmo faz a geração dos triângulos a partir destes, criando assim a malha que é exportada como um arquivo ".obj" na etapa final deste algoritmo.

Pode-se observar a chamada dessa ferramenta na figura 5.5.

```
Usage: ocTool [options] [file]
   Ex: ocTool -c 3.0 terrain.tga

   Options:
       -a [val]: Use height scale factor. val is the amount of offset.
          height = height * val.
       -s [val]: Use horizontal scale factor. val is the amount of scale (defaul
t 1 m).
          (x,z) = (x,z) * val.
       -c [val]: Use choppyness info. val is the amount of offset.
          (RGB) -> (x+R*val,G,z+B*val)
```

Figura 5.5: Ferramenta de Geração de Malhas.

6.
Resultados

Este capítulo apresenta os principais resultados desta dissertação. Para demonstrar os resultados obtidos, a seção 6.1 apresenta uma cena protótipo que segue o *workflow* proposto no capítulo 4.

6.1. Cena Protótipo

A cena protótipo, onde o *workflow* proposto no capítulo 4 foi implementado, consiste em simular uma cena de mar perto da costa, na qual se observam um grande grupo de embarcações, algumas que se aproximam do porto e outras que no mar navegam em diferentes rotas. Nesta simulação, ainda estão presentes agentes humanos, que estão distribuídos dentro dos agentes tipo embarcação e na região do porto.

Existem algumas limitações na construção deste protótipo. As principais limitações estão relacionadas com a produção artística, uma vez que, para produções de alta qualidade, normalmente é necessária a existência de equipes com diversos profissionais especialistas. Estes profissionais são responsáveis pelo desenvolvimento de modelos, texturas, animações e *shaders* de todos os elementos que compõem a cena (conforme ilustra a seção 2.2.4). Por isso, o protótipo implementado foi baseado em fotos de cenários reais, usados como alternativa à criação dos cenários virtuais descritos no *workflow*. A opção de obtenção de cenários através de fotogrametria (conforme descrito na seção 2.25) foi descartada pela mesma razão (i.e. a falta de uma equipe de modeladores para concluir as malhas e texturas). Na Figura 6.1 pode-se ver um *frame* de um plano geral da cena criada.

Figura 6.1: Frame de um plano geral da cena protótipo, com embarcações em alto mar.

As outras restrições dizem respeito às opções dos softwares de modelagem e renderização. Estas últimas restrições foram estabelecidas em função das limitações do equipamento usado (um notebook com 2 núcleos, Intel i7, 8Gb de RAM e HD de 500 Gb). Neste caso, apesar do *workflow* estabelecer o uso do Renderman e de arquivos .EXR, foram utilizados o renderizador Velocity (do próprio software Massive) e arquivos .TIF para gerar as imagens HDR. Neste particular, ao invés de se ter um único arquivo .EXR por frame de animação dos agentes (como está no final do swinlane Simulating agents da Fig. 4.3), têm-se 3 conjuntos de arquivos tipo *shader*:

beauty_frame_1.tif, beauty_frame_2.tif, ...
shadow_frame_1.tif, shadow_frame_2.tif, ...
zdepth_frame_1.tif, zdepth_frame_2.tif, ...

Uma outra simplificação é quanto ao número e tipo de áreas protegidas com *flow fields*, visto que o terreno global adotado é pequeno e apenas um único porto foi considerado. Por fim, não foram geradas animações de objetos secundários (tais como bandeiras e veículos) para não sobrecarregar o processo.

Foram geradas duas tomadas de vídeo (duas câmeras virtuais distintas), onde cada uma delas teve uma animação diferente e retratando momentos específicos de simulação. Na primeira tomada exibe-se um plano geral com uma câmera rápida e então na segunda tomada com uma câmera mais lenta, é possível ver pessoas andando pelo píer, enquanto é possível observar a movimentação das embarcações da simulação.

Ambas as composições usaram as simulações do Massive com as pessoas e embarcações conforme o workflow proposto, porém, para gerar as imagens do oceano e porto para este protótipo, ao invés de usar o Maya como proposto no workflow, a CryEngine[19] 3 foi usada, já que possui ótima qualidade visual com um custo de tempo ótimo para renderizar as sequências de imagens. Na Figura 6.2 é possível ver um frame da outra tomada da cena protótipo.

Figura 6.2: Frame da tomada dois da cena protótipo.

[19] Uma das melhores engines para jogos virtuais tridimensionais. A CryEngine possui uma solução para renderizar cenas (realtime) internamente.

7.
Conclusões e Trabalhos Futuros

Neste capítulo são apresentadas as conclusões e outras considerações desta dissertação, além das propostas de continuação para este trabalho.

7.1. Conclusões

A principal contribuição deste trabalho foi a definição de um método para a geração de cenas de mar e de porto com embarcações, através da simulação em massa de agentes e com o grau de qualidade de imagem e de movimento que TV e cinema digitais requerem. Este tipo de conhecimento e este tipo de procedimento não estão disponíveis na literatura nem são divulgados pelos grandes estúdios internacionais de cinema. Ademais, mesmo nestes grandes estúdios, as cenas de mar e porto não são automatizadas no mesmo grau que o método proposto nesta dissertação permite alcançar[20]. O método proposto consegue este grau de automatização e de qualidade porque disponibiliza três módulos de suporte que não estão disponíveis no mercado: (1) um simulador de oceano que gera a superfície do mar em várias condições; (2) um conjunto de agentes customizados para cenas de mar e porto; e (3) um módulo de especificação de variáveis com modificadores *fuzzy* (*hedges*) que facilita o trabalho dos artistas e diretores de cena.

Como não foram encontrados trabalhos na literatura que, como este, abordassem uma apresentação de todo o processo necessário para gerar uma cena de simulação em massa para cenas de mar (desde a sua criação, até o resultado final) é possível afirmar que o presente trabalho pode ser de grande utilidade para a indústria de entretenimento digital.

[20] Os grandes estúdios costumam chegar aos resultados finais por "força bruta", contratando batalhões de finalizadores (usando softwares de composição) em países com baixo custo de mão de obra (notadamente na Índia).

Avaliando a complexidade de gerar cenas de grandes quantidades de embarcações, vale observar que este processo simplifica muito o processo de geração das cenas, principalmente quando existem muitas cenas a serem realizadas usando o workflow implementado. Isso pois apesar da geração de todos os insumos necessários para configuração dos agentes (principalmente sua customização) consumir um tempo considerável, quando estão finalizados, é possível realizar uma maior quantidade de cenas em um tempo menor, uma vez que a inteligência implementada no cérebro dos agentes gera automaticamente todas as animações de trajetória das embarcações bem como sua diversificação visual e de movimento (de acordo com o movimento do mar).

Considerando que a inteligência no "cérebro" dos agentes utiliza a variação de hedges, pode-se dizer que a modificação de uma cena usando este workflow se daria muito mais rápida que em um workflow de cenas tridimensionais tradicional, onde os comportamentos de todos os agentes teriam de ser reanimados praticamente do início.

Para começar a seguir o workflow proposto neste trabalho, o tempo para organizar todo o material necessário pode chegar a ser mais que o dobro do tempo que seria necessário para fazer uma cena tridimensional com a mesma complexidade da forma tradicional. A vantagem deste workflow está na facilidade de recriar novas cenas, onde é gerado a cada simulação, novos comportamento para todos os agentes, o que em uma produção com muita demanda de cenas, pode diminuir muito o tempo de execução das cenas, uma vez também que todos os insumos já estão pronto para serem usados. Essa facilidade de criar novas cenas pode dar um ganho de até dez vezes mais rapidez para entrega de cenas.

O resultado visual das imagens do protótipo realizado seguindo o workflow proposto foi bem satisfatório. E considerando que os agentes embarcações utilizados foram totalmente desenvolvidos para este projeto, é possível considerar que esse modelo de agente poderá ser aproveitado em outros trabalhos.

Todos os arquivos e instruções deste trabalho poderão ser encontrados em www.icad.puc-rio.br/~MassiveBoatSim. Caso sejam necessárias credenciais para acessar este endereço eletrônico, basta entrar em contato com o autor ou orientador deste trabalho para obter acesso.

7.2. Trabalhos Futuros

Como um trabalho ainda a ser desenvolvido, pode-se considerar a transformação do modelo de *workflow* proposto neste trabalho para que possa ser usado em aplicações em tempo real, tais como jogos ou simuladores. Nesses casos deve-se estudar o uso de técnicas que possam usar aceleração por placas gráficas.

Uma outra opção de trabalho futuro é adaptar o *workflow* para outras situações que possuam simulações semelhantes às encontradas na presente dissertação. Desta maneira, pode-se estabelecer uma linha de pesquisa que busque por famílias de *workflows* mais genéricos e que se apliquem a vários tipos de cena. Estes *workflows* devem continuar tendo os objetivos de encurtar o tempo de produção e diminuir os custos das cenas, ao mesmo tempo que garantem uma qualidade superior de imagem e movimento.

Referências Bibliográficas

Amit, L. (2007). *Multi-agent Systems in Massive*, NCCA Bournemouth University

Boyce, W. E. e DiPrima, R. C. (2001). *Elementary Differential Equations and Boundary Value Problems*. 7th ed. John Wiley & Sons, Inc.

Chen, J. e Lobo, N. (1995). Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. In Graphical models and Image Processing., 1995.

Chenney, S. (2004). "Flow tiles". In: *SCA'04 Proceedings of 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p. 233-242.

Cordeiro, O., Blum, C., Braun, A., Musse, S. R. e Cavalheiro, G. G. H. (2005). "Concurrency on Social Forces Simulation Used in Crowd Modelling. In: First International Workshop on Virtual Crowds", Lausanne. *Proceedings of International Workshop on Crowd Simulation*. Lausanne: EPFL, vol 1, p. 117-126.

Costa, M. e Feijo, B. (1996). *Agents With Emotions In Behavioral Animation. Comput. & Graphics*, Great Britain, v. 20, n.3, p. 377-384.

Cox, E. (1994). *The fuzzy systems handbook: a practitioner's guide to building, using, and maintaining fuzzy systems*. New York: AP Professional, 1994.

Cruz, A. P. e Bedregal, B. R. C. (2007). *Fundamentos de Sistemas Fuzzy Baseados em Dados Intervalares à Luz de uma Teoria de Representação Fuzzy Intervalar*. In: VI Congress of Logic Applied to Technology, 2007, Santos-SP. Proceedings of VI Congress of Logic Applied to Technology, 2007. p. 1-8.

Debevec, P. E. e Malik, J. (1997). "Recovering high dynamic range radiance maps from photographs", in *SIGGRAPH 97 Conference Proceedings*, August 1997, p. 369-378

Debevec, P. E. (1998). "Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with illumination and hygh dynamic range photography". In *SIGGRAPH 98 Conference Proceedings*, p. 189-198.

Debevec, P. (2002). "Image-based lighting". *IEEE Computer Graphics and Applications*, March/April 2002, p. 26-34.

Esteva, F., Godo, L. e Noguera, C. (2013). "A logical approach to fuzzy truth hedges", *Information Sciences: an International Journal*, Elsevier Science Inc., New York, NY, v. 232, p. 366-385.

Foster, N. e Metaxas, D. (2000). Modeling water for computer animation. *Commun ACM*, 43(7), p. 60-67.

Gayle, R., Moss, W., Lin, M. C. e Manocha, D. (2009). "Multi-robot coordination using generalized social potential fields". In Proc. *IEEE Conf. Robotics and Automation 2009*.

Gomide, F. A. C., Gudwin, R. R., e Tanscheit R. (1995). "Conceitos fundamentais da teoria de conjuntos fuzzy, lógica fuzzy e aplicações". *Sixth International Fuzzy Systems Association World Congress/ Tutorials - IFSA95*, p. 01-38., July 1995.

Huang, C. Y., Chen, C. Y., e Liu, B. D. (1995). “Current-mode linguistic hedge circuit for adaptive fuzzy logic controllers”, *Electronics Letters 31*, no. 17, 1517–1518.

Jané, D. de A. (2004). *Uma Introdução ao Estudo da Lógica Fuzzy*, Hórus - Revista de Humanidades e Ciências Sociais Aplicadas, Ourinhos/SP, Nº 02, 2004.

Joselli, M., Passos, E. B., Zamith, M., Clua, E., Montenegro, A. e Feijo, B. (2009). “A Neighborhood Grid Data Structure for Massive 3D Crowd Simulation on GPU”. In: Proc. of *SBGAMES 2009 - VIII Brazilian Symposium on Games and Digital Entertainment*, IEEE, p. 121-131.

Kaehler, S. (1998). “Fuzzy Logic Tutorial”. *Encoder The Newsletter of the Seattle Robotics Society*. (http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html).

Klir, G. J. e Folger, T. A. (1988). *Fuzzy Sets, Uncertainty and Information*, Prentice Hall.

Klir, G. J. e Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall.

Koshiyama, A. S., Escovedo, T., Dias, D. M., Vellasco, M. M. B. R., Tanscheit, R. (2013). GPF-CLASS: A Genetic Fuzzy model for classification. In: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, Cancun. 2013 IEEE Congress on Evolutionary Computation, p. 3275-3282.

Larson, G. W. (1998). The LogLuv Encoding for Full Gamut, High Dynamic Range Images. Journal of Graphics Tools archive, vol. 3, n. 1, p. 15-31.

Lucas Digital Ltd, (2006). Technical Introduction to OpenEXR. Available from: http://www.openexr.com/TechnicalIntroduction.pdf (Acessado em 20 de Maio, 2013).

Mars, C. (2011). "Group Navigation State of Art". Disponível em: https://github.com/cloderic/Group-navigation-state-of-the-art-report/blob/master/Group_navigation_state-of-the-art_report.pdf

Mononen, M. (2010). "Navigation Loop." Paris Game/AI Conference 2010. Disponível em: http://digestingduck.blogspot.com/2010/07/my-paris-game-ai-conference.html (Acessado em 26 de Junho, 2012).

Musse, S. R. e Daniel T. (2001). "Hierarchical Model for Real Time Simulation of Virtual Human Crowds." Transactions on Visualization and Computer Graphics 7(2):152-164.

Nilson, J. W. e Riedel, S. A. (2001). *Eletric Circuits*. 6th ed. Prentice-Hall.

OMG, (2011). UML Superstructure Specification, v2.4, Available from: http://www.omg.org/spec/UML/2.4/Superstructure/Beta2/PDF (Accessado em 2 de Dezembro de 2011).

Pottinger, D. (1999). "Implementing Coordinated Movement." Gamasutra. Acessado em 20 de Junho, 2012 (http://www.gamasutra.com/view/feature/3314/implementing_coordinated_movement.php?print=1).

Reynolds, C. W. (1987). "Flocks, herds and schools: A distributed behavioral model." Pp. 25-34 in 1987 International Conference and Exhibition on Computer Graphics and Interactive Techniques, vol. 21.

Reynolds, C. W. (1999). "Steering behaviors for autonomous characters." Game Developers Conference 1999.

Silva, R. M. A. (2010). Simulação e visualização de Oceano em tempo real utilizando a GPU. Dissertação (Mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2010.

Silveira, M. M. M. T. (2002). Teoria fuzzy intervalar: Uma proposta de integração da matemática intervalar à teoria fuzzy. Master's thesis, Universidade Federal do Rio Grande do Norte.

Sissom, L. E. e Pitts, D. R. (2001). Fênomenos de Transporte. 1st ed. Rio de Janeiro: LTC.

Tessendorf, J. (1999). Simulating Ocean Water. In SIGGRAPH'99 Course Notes, 1999.

Thalmann, D. e Musse, S. R. (2013). Crowd Simulation. 2nd Ed. Londres: Springer.

Vellasco, M. M. B. R., Pacheco, M. A. C., Figueiredo, K. T., Souza, F. J. (2008). "Hierarchical Neuro-Fuzzy Systems - Part I." In: Juan R. Rabuñal, Julián Dorado, and Alejandro Pazoz (Org.), *Encyclopedia of Artificial Intelligence, Information Science Reference*.

von Altrock, C. (1997). *Fuzzy Logic e NeuroFuzzy Applications in Business and Finance.* New Jersey: Prentice Hall PTR, 1997.

Ward, G. (1991). "Real Pixels", *Graphics Gems II,* James Arvo (ed.), Academic Press, p. 80-83.

Zadeh, L. A. (1965). "Fuzzy Sets". Information and Control, V. 8: 338-353.

Zadeh, L. A. (1972). *A fuzzy-set-theoretic interpretation of linguistic hedges*, Journal of Cybernetics 2:3, 4-34.

Zadeh, L. A. (1994). *Soft computing and fuzzy logic*, IEEE Software, Vol 11, Issue 6, Nov 1994, p. 48-56.