Representação Gráfica de Histórias Interativas

Cesar Tadeu Pozzer
Departamento de Informática
e-mail: pozzer@inf.puc-rio.br

Marcelo Dreux
Departamento de Engenharia Mecânica
e-mail: dreux@mec.puc-rio.br

Bruno Feijó
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225
Rio de Janeiro, RJ, 22453-900, Brasil
e-mail: bruno@inf.puc-rio.br

PUC-RioInf.MCC40/03 Outubro, 2003

Abstract: This work presents a general architecture to represent dynamic interactive stories which encapsulates the generation, execution and visualization of those stories, given a special attention to the visualization process, by means of Computer Graphics techniques. It has been used cinematographic techniques to capture the essence of the scenes, which are composed by a virtual 3D environment. Characters, implemented as reactive agents, interact among each other and with the scene to accomplish the plot of the narrative.

Keywords: Computer Graphics, virtual camera, characters, animation, real-time rendering.

Resumo: Neste trabalho é apresentada uma arquitetura para a representação de histórias interativas dinâmicas, que encapsula tanto a geração, execução e visualização, dando uma atenção especial para o processo de visualização, por meio de técnicas de Computação Gráfica. Faz-se uso de técnicas cinematográficas para capturar a essência das cenas, que são compostas de um ambiente virtual 3D, onde personagens, implementados por meio de agentes reativos, interagem com outros e com a cena para realização das ações na história.

Palavras-chave: Computação Gráfica, câmera, personagens, animação, renderização em tempo real.

This work is being sponsored by CNPq.

1 Introdução

Com o avanço do hardware de placas gráficas e de técnicas específicas, simulações que levavam horas, em baixa qualidade, hoje podem ser realizadas em tempo real, com um nível de realismo muito elevado. Técnicas de renderização em tempo real estão se tornando muito freqüentes nas pesquisas acadêmicas, bem como a nível de entretenimento, que é um foco muito promissor e grande consumidor de tecnologias de hardware e software.

Estes avanços podem ser observados em jogos de computador que, a cada momento, causam espanto e surpresas pelo nível de detalhes e realismo apresentado, tanto no que se refere a detalhes gráficos, como simulações físicas, efeitos especiais, inteligência artificial, dentre outros.

Atualmente, pesquisas e desenvolvimento em jogos de computador estão chamando atenção não somente da indústria e investidores, mas também de universidades e centros de pesquisa. A área de jogos, por fazer uso de quase toda infra-estrutura de Informática, como Redes, Computação Gráfica, Inteligência Artificial, Sistemas Distribuídos, linguagens de script, otimização, dentre outras, está passando a ser vista como uma área de pesquisa séria, visto o surgimento de congressos científicos na área, bem como cursos de graduação, especialização e pós-graduação.

As pesquisas nesta área concentram-se tanto no desenvolvimento de novas técnicas de CG, jogos, bem como na definição de novos paradigmas de enredos e formas de entretenimento.

A criação de um jogo envolve o trabalho conjunto de programadores, analistas, artistas, projetistas de níveis, game designers, sonoplastas, testadores, dentre outros [CUN 01]. Por isso, o sucesso de um jogo digital não está vinculado somente à qualidade gráfica, mas também à originalidade. A quebra de paradigmas e conteúdos é um fator muito importante para o sucesso do produto. Isso pode ser observado em jogos como Doom, The Sims, Ultima Online, SimCity, que foram inovadores, fugindo dos estilos tradicionais existentes.

Neste trabalho, pretende-se desenvolver e discutir técnicas de Computação Gráfica para um novo paradigma de aplicação que se assemelha em alguns pontos com jogos de computador: a exibição automática de histórias interativas. O objetivo final é a construção de uma ferramenta integrada que permita a geração, execução, visualização e exibição de histórias, cujo contexto e enredo possam ser guiados pelo usuário.

Para construir essa ferramenta, é necessário desenvolver pesquisas em três linhas distintas, mas fortemente relacionadas, como mostrado na Figura 1. A linha de TV Interativa corresponde ao estudo de conteúdos interativos e técnicas relacionadas de exibição e interação. A linha de Agentes e IA trata de geração de personagens autônomos. Por fim, a linha de Computação Gráfica é necessária para transformar as representações simbólicas das histórias em informação gráfica que possa ser apresentada como um conteúdo interativo.

O elemento central da arquitetura concentra-se na geração da história, que é realizada por um processo de simulação, fazendo uso de algoritmos de planejamento e de Constraint Logic Programming. Este processo gera um conjunto de eventos, baseados em operações com pré e pós-condições que estabelecem fatos que devem ser válidos antes e após os eventos, respectivamente. As ações descritas pelas operações devem ser

realizadas pelos personagens, obedecendo a uma ordem cronológica parcialmente definida pelo planejador.

Figura 1: Linhas de pesquisa para a construção de um sistema de geração e visualização de histórias interativas. A região sombreada representa o foco deste trabalho.

Para tornar este conjunto de operações possível de ser representado como um conteúdo interativo em iTV (Interactive TV) [DRI 00, FUR 96], de imediato reconhece-se a necessidade de estudos em outras duas linhas de pesquisa: Computação Gráfica e Inteligência Artificial. A IA se faz necessária para assegurar que cada personagem, aqui representado como um agente de software, seja capaz de expressar um comportamento inteligente e realista, por meio de animações gráficas condizentes com as ações sendo realizadas, frente ao telespectador (usuário). A IA exerce um papel fundamental no comportamento do personagem, pois é responsável pela determinação das animações que cada um deve realizar para o cumprimento das tarefas. Tendo-se as descrições de posição e animação de cada personagem, faz-se uso de técnicas de Computação Gráfica para visualizar estas informações, por meio de algoritmos de renderização e de técnicas de modelagem de personagens, que sejam suficientemente realistas para o propósito em questão.

Essa informação gráfica, resultado da história sendo simulada, é o conteúdo final apresentado ao usuário, por meio de recursos de iTV. Cada usuário poderá interagir com o sistema na alteração de atributos dos personagens e história, e até mesmo definir o rumo da mesma.

Este trabalho está organizado em 5 seções. Na Seção 2 são apresentados conceitos sobre geração e técnicas de execução de histórias interativas. A execução é realizada pela associação de agentes às operações que definem a história. Na Seção 3 são discutidos diversos temas relacionados com a exibição e representação gráfica das histórias, incluindo criação e manipulação do cenário, personagens e câmera. Na Seção 4 são apresentados alguns resultados na Seção 5 conclusões finais.

2 Geração e Execução de Histórias

Para tornar possível a exibição de histórias, toda uma infra-estrutura de geração e execução deve estar definida. A primeira parte consiste na geração de um enredo de um certo gênero, realizado por um processo de *simulação*, que tem como ponto de partida a descrição da situação inicial dos personagens, o modelo de comportamento atribuído a eles (especificado em termos de objetivos a serem perseguidos em situações previstas), e as alternativas que cada personagem tem para atingir seus objetivos, que são especificadas em termos das operações (ou padrões típicos) que caracterizam o gênero [CIA 99, FUR 99, CIA 02]. O resultado do processo de simulação é um conjunto de operações, em formato texto, associadas a personagens, com pré e pós-condições, bem como possíveis valores de atributos de cada personagem (Figura 2).

```
0: Init()
1: Absence_of_younger_people (princess)
2: Kidnapping_of_a_person (princess, dragon)
3: Call_for_help (hero, tsar)
4: Departure_of_seeker_hero (hero)
5: Fight_in_an_open_field (hero, dragon)
6: Out_of_earth_receipt (hero)
7: Victory_in_open_battle (hero, dragon)
8: Return (hero)
9: Reward (hero)
```

Figura 2: Exemplo de um enredo criado pelo simulador

A segunda etapa consiste em transformar esta representação textual em ações, individuais ou em grupo, realizadas por personagens virtuais, que sejam capazes de transmitir ao usuário a sensação de estar vivenciando uma história real. Estes personagens são implementados por meio de agentes [WOO 99b, NAR 03], que devem ser autônomos ao nível de encontrar meios de realizar ações a eles fornecidos em um ambiente virtual 3D, composto por um cenário e outros agentes, de forma que o fluxo da história seja corretamente representado.

Finalmente, na terceira etapa é realizada a visualização da cena, por meio de recursos de Computação Gráfica. Cada uma destas etapas é realizada por módulos separados, que interligados por canais de dados, conseguem transformar descrições textuais em animações gráficas realistas. A estrutura deste sistema integrado é apresentada na Figura 3.

O IPG (*Interactive Plot Generator*) [CIA 99, CIA 02] é o módulo responsável pela geração dos enredos. Estes enredos, quando simulados, são passados para o módulo CEH (módulo de Controle de Execução da História), que é o responsável pela distribuição das operações aos respectivos personagens. Além de delegar operações, também realiza um acompanhamento do desempenho dos mesmos no cumprimento destas operações. Existe uma contínua comunicação bidirecional entre o módulo de controle com cada personagem, que informam, a cada passo, sobre o estado do cumprimento da última operação fornecida. Quando uma operação for finalizada, uma nova é delegada. Uma operação é finalizada quando o agente obtém sucesso ou não no

seu cumprimento. A escolha da operação a ser delegada depende do agente e do andamento do fluxo da história (*timeline*). Sob uma visão de agentes, o CEH pode ser visto como o módulo de controle centralizado do SMA (Sistema Multiagente)[JEN 96, JUC 01, SHE 98]. Os personagens são agentes reativos, implementado por meio de uma máquina de estados finita (FSM).

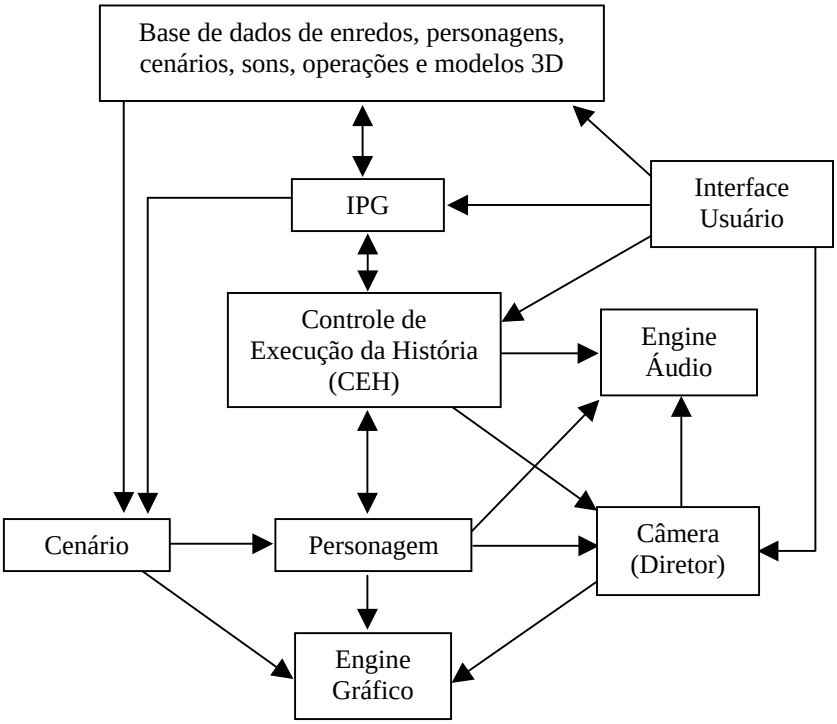

Figura 3: Estrutura de módulos de controle e execução de operações.

Existe um conjunto fixo de micro-operações que cada personagem pode realizar (Tabela 1). A escolha destas micro-operações foi baseada na simplicidade e possibilidade da representação gráfica inerente à operação.

O resultado do processo de execução da história são informações geométricas que representam a estrutura pontual dos objetos dinâmicos (ex: personagens) controlados por técnicas de IA, em um dado momento. Essas informações, juntamente com informações estáticas provenientes do cenário, devem ser enviadas para o módulo gráfico para serem representados na tela.

3 Visualização Gráfica de Histórias

No processo de geração de enredos [POZ 03a], não existe uma descrição do ambiente de representação da história. Cada personagem é tratado apenas de forma simbólica no processo de inferência de objetivos. O mesmo vale para as operações que descrevem as ações a serem realizadas.

O primeiro passo para tornar estas operações mais concretas foi dado com a associação de agentes aos personagens [POZ 03b]. Estes agentes interagem com um

ambiente virtual 3D e com outros agentes, em um espaço geométrico, e não mais simbólico, podendo assim criar uma representação menos abstrata, que pode mais facilmente ser transformada em imagens e animações gráficas. Em outras palavras, os agentes dão vida a cada operação e a cada personagem.

Tabela 1: Conjunto de micro-operações

Micro-operação	**Descrição**	**Especializações**
Deslocar	Fazer o deslocamento de um personagem de um local para outro	Voar Caminhar Rastejar Nadar Teletransportar
Lutar	Duelo entre dois ou mais personagens	Espada Arma de fogo Fogo (dragão) Magia
Esperar	Representação de ação inexistente ou momento de descanso	Permanecer parado Dormir Descansar
Pegar/soltar	Capturar ou adquirir objetos ou personagens	Rapto personagens Roubo objetos Aquisição de poderes
Pedir/dar	Requisição de algo por meio de diálogo	Informação (fala/escuta) Poderes Objetos Ajuda
Observar	Captação dos dados do ambiente próximos ao personagem	Procurar Ver Ouvir
Trabalhar	Ato de trabalhar	Doméstico Agrícola

O grau de detalhes neste universo geométrico, bem como no simbólico, pode assumir níveis intratáveis de complexidade. Quanto maior for, melhores (mais realistas) podem ser os resultados obtidos a nível de simulação, execução e visualização dos enredos. Sob o aspecto de visualização das histórias, apontam-se três questões que devem ser tratadas para garantir um nível de realismo razoável:

- **O que existe para visualizar (Ambiente de Representação):** O universo geométrico de representação das cenas pode ser muito variado. Além de um cenário 3D, aqui representado com um terreno e um céu, podem também existir objetos como árvores, casas, além das representações geométricas de cada personagem da história;

- **O que deve ser visualizado (Câmera Virtual):** Dado um cenário dinâmico e personagens agindo neste ambiente, deve-se determinar qual cena deverá ser exibida a cada momento. Esta escolha é realizada por uma câmera virtual, que leva em consideração diversos fatores, como número de personagens, importância das ações, dentre outros;
- **Como deve ser visualizado (Computação Gráfica):** Uma vez determinado o que exibir, deve-se prover recursos de Computação Gráfica para transformar as representações geométricas em imagens com certo grau de realismo.

3.1 Definição do Ambiente de Representação

O ambiente de representação (cenário) é o local onde a história vai ser representada e visualizada. Ele serve também como base para simulação da interação entre personagens. Esta simulação, aqui é realizada com um enfoque um tanto diferente comparado com processo de geração da história pelo IPG. São considerados aspectos físicos, como distâncias, obstáculos e, conseqüentemente, aspectos de necessidades básicas e propriedades físicas de personagens, como velocidade, força, capacidade de percepção de dados do ambiente e autonomia para determinar meios de realização das operações a eles fornecidos.

A criação do cenário deve ser realizada no início do processo de visualização, em função do contexto e gênero do enredo, fazendo uso de primitivas armazenadas na base de dados, como mostrado no diagrama da Figura 3. Pretende-se também permitir que o usuário possa interferir neste processo.

Uma maneira prática de definição de um cenário é pelo uso de grafos de cena [MOL 02]. Os grafos de cena são estruturas destinadas a facilitar a distribuição, manipulação e visualização de entidades geométricas que possuem uma certa hierarquia. A informação está distribuída em nós, que podem tanto conter primitivas como operações geométricas, como rotações, translações, etc. Quando uma transformação é aplicada a um nó deste grafo, todos os nós filhos também sofrem o efeito desta transformação.

Neste trabalho, o grafo de cena assume uma configuração como mostrada na Figura 4, e é composto basicamente de um céu, um terreno, objetos e personagens.

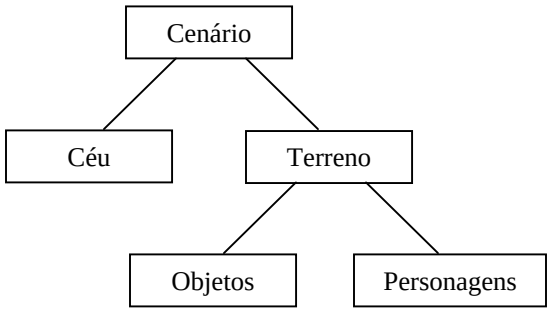

Figura 4: Grafo de cena do módulo de visualização

3.1.1 Terreno

O terreno é o elemento base da cena, pois sob ele está disperso o conjunto de personagens e objetos, como moradias e árvores. A idéia é que este terreno possa ser criado de forma dinâmica segundo intenção do usuário e aspectos gerais da história. Sua dimensão e relevo, por exemplo, podem ser dependentes do número de habitações, que por sua vez é proporcional ao número de personagens da história.

A geração de terrenos não planares e irregulares é uma tarefa relativamente simples, pois consiste em gerar uma malha de triângulos, a partir de um mapa de alturas (*height field*), que pode ser oriundo de uma função randômica ou de uma imagem, como mostrado na Figura 5. Uma abordagem detalhada de técnicas de geração e manipulação de terrenos pode ser obtida em [SNO 03].

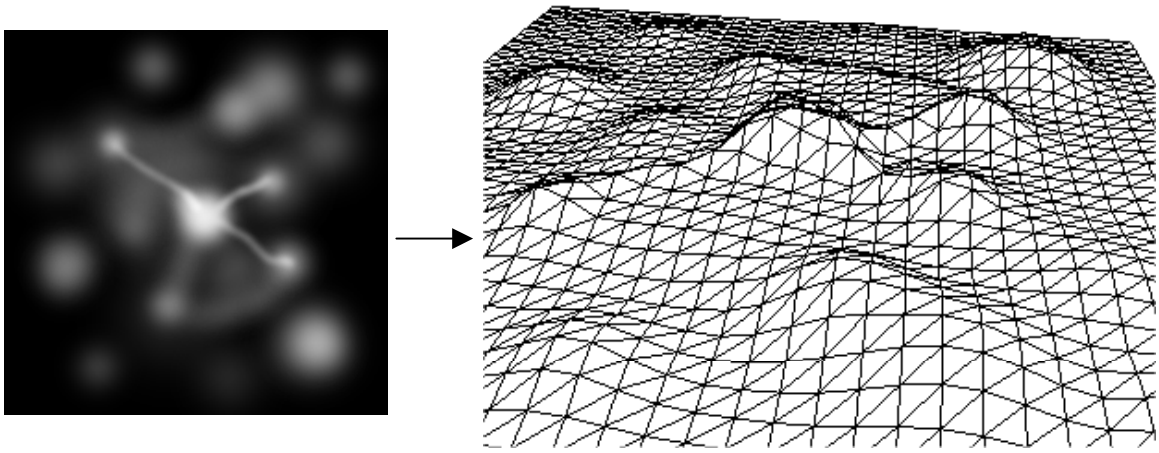

Figura 5: Geração de um terreno irregular com malha regular a partir de uma imagem

Existem diversas técnicas para aceleração de rendering de terrenos. A mais comum é o uso de LOD (*Level of Detail*) [GAR 97, GAR 98, EBE 01, MOL 02, ULR 02, ULR 00, TOL 00], que consiste em reduzir a complexidade da malha (número de triângulos), procurando ao máximo manter a geometria “visual” da mesma.

Existem diversas métricas para determinação das regiões das malhas a serem simplificadas. Referente à métrica de câmera, sabe-se que à medida que objetos se distanciam, devido à projeção em perspectiva, tornam-se menores e conseqüentemente menos visíveis, podendo assim terem menor resolução geométrica. Pela métrica da geometria, regiões mais planas podem ser representadas com menos polígonos que regiões com mais detalhes e com maior irregularidade. O resultado destas métricas pode ser observado na Figura 6.

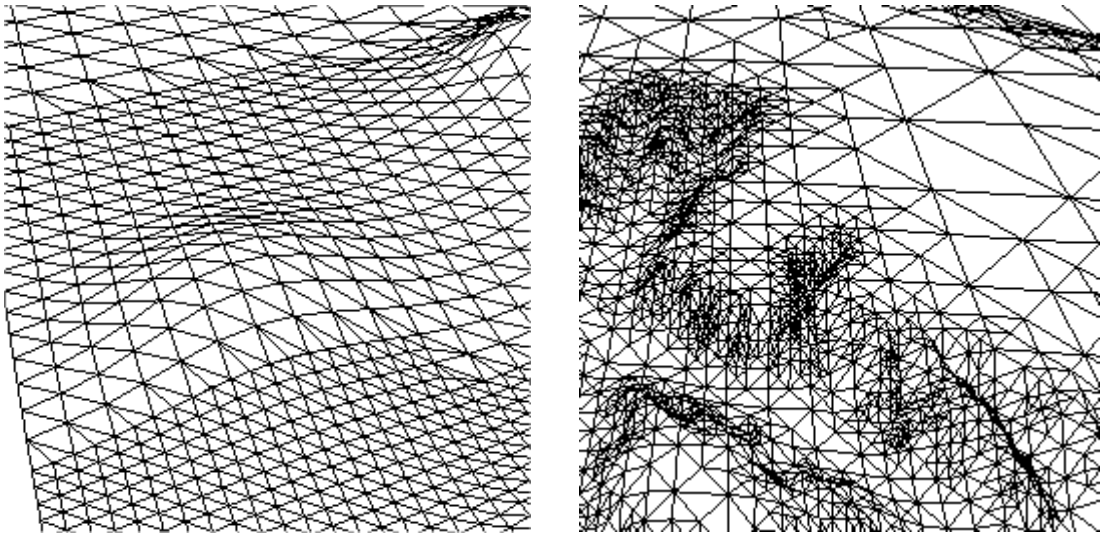

Figura 6: Exemplos de diferentes métricas de LOD: Distância e irregularidade da superfície.

3.1.2 Objetos da cena

Para que um personagem possa se deslocar no cenário sem colidir com objetos e outros personagens, deve continuamente adquirir informações do terreno (coordenada y associada a sua posição corrente) e de objetos presentes em seu campo de visão, como mostrado na Figura 7.

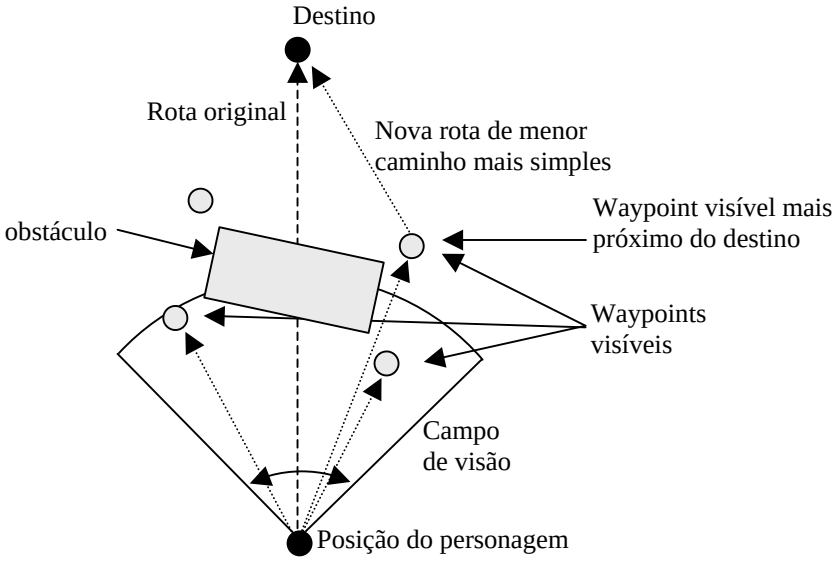

Figura 7: Campo de visão dos personagens e uso de waypoints como estratégia de desvio de obstáculos fixos.

Para o tratamento de colisão, cada objeto está envolto por um polígono convexo definido por quatro waypoints. Estes waypoints são determinados automaticamente no momento da criação da cena em função das dimensões dos objetos. Waypoints são comumente usados em jogos para facilitar a locomoção de NPCs (*Non-Player Characters*) em cenários dinâmicos. Eles podem ser vistos como nós em um grafo, que representa todos os caminhos que o NPC pode navegar no ambiente [LID 02].

Quando um objeto estiver dentro do campo de visão do personagem, caso for detectado que haverá colisão com a rota do personagem, encontra-se o waypoint mais próximo do destino que seja visível pelo personagem. Esta posição é então adicionada ao novo caminho do personagem, para composição da nova rota, como mostrado na Figura 7. Esta estratégia de colisão pode falhar caso existam interseções entre as áreas convexas definidas pelos waypoints de cada objeto.

À medida que o personagem se desloca na cena, existem no máximo, a cada instante, duas posições a serem atingidas no caminho: o destino final e um possível desvio. Assim que o personagem atinge um waypoint, ele certamente deve mudar sua direção para atingir o destino final. Para garantir suavidade de curvas e não violar leis da física, o trajeto é definido segundo um raio de curvatura variável. [PIN 02] apresenta um algoritmo para determinação da rota a ser executada em função do raio de curvatura, centro de rotação, posição, orientação e destino (Figura 8). Tomando-se o centro de rotação com o raio de curvatura, obtém-se um círculo, que é a região que não pode ser alcançada pelo personagem fazendo-se uso deste algoritmo de suavização de curvas.

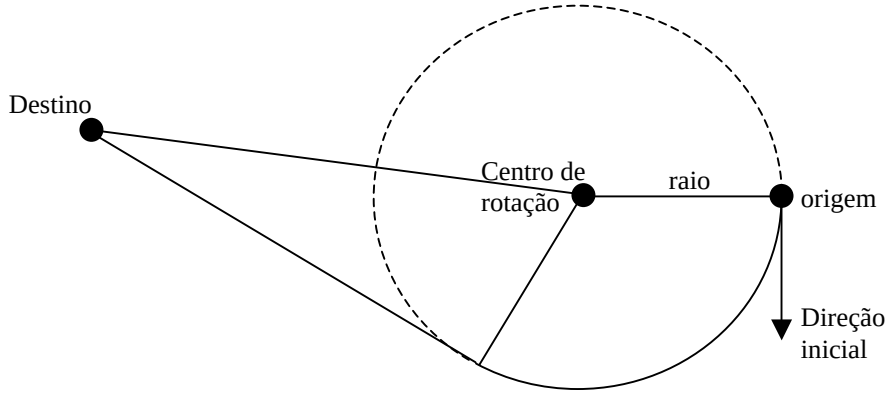

Figura 8: Estratégia para suavização de curvas [PIN 02].

3.1.3 Personagens

Os únicos elementos animados do cenário são os personagens, e por isso devem disponibilizar estruturas que possam tratar animações. Para definir com maior clareza suas estruturas, deve-se analisar a que se destinam estes personagens, bem como quais atributos e comportamentos cada um deve exibir.

Dois conjuntos de características podem ser observados de imediato: animações a nível de representação de micro-operações e representação de atributos cognitivos, como emoções e expressões faciais.

Como apresentado na Tabela 1, definiu-se um conjunto de micro-operações que possam mapear diversos eventos das histórias. Como este conjunto é fixo, pode-se facilmente associar animações pré-definidas a cada operação.

Além de animações comportamentais, também pretende-se incorporar nos personagens o tratamento de emoções. Neste momento não se está preocupado na forma como as emoções são geradas, mas sim na forma como são expressas ao mundo externo do agente. As expressões faciais podem ser vistas como resultado do estado emotivo do personagem [BRE 98, VEL 97].

Existem várias estratégias para animação de modelos 3D em tempo real. A solução mais simples é pela definição de uma única malha na descrição da estrutura do personagem. Não existem estruturas internas para distinguir partes do corpo, como cabeça, braços, etc. Para permitir animações, são definidos conjuntos de malhas, que retratam posições específicas de cada quadro da animação (*keyframe*). Pela interpolação linear entre estes quadros, pode-se obter animações contínuas. Como toda a animação é definida em pré-processamento por um sistema de modelagem, não é possível realizar animações adicionais em tempo de execução. Cada keyframe é descrito pela especificação da posição de cada vértice da malha. Isso torna a base de dados maior quando comparada com estratégias de representação que utilizam estruturas hierárquicas. A grande vantagem deste método é a sua simplicidade e baixo custo computacional na renderização e interpolação entre keyframes.

Uma estratégia mais aprimorada e, conseqüentemente, de processamento mais elevado faz uso de estruturas hierárquicas, semelhantes a um grafo de cena. Nesta abordagem, cada parte do modelo é armazenada e manipulada separadamente nos processos de animação e renderização. Cada elemento da estrutura é ligado, de forma hierárquica, com os demais nós por um pivô. No topo da hierarquia existe um nó raiz.

Devido à forma como a estrutura do modelo é armazenada, esta abordagem permite que animações sejam criadas dinamicamente, por meio de cinemática inversa. Da mesma forma como na representação que faz uso de malhas, pode-se armazenar animações pré-definidas com o uso de keyframes, com a diferença que neste caso armazena-se somente a posição de cada elemento da estrutura, ao invés de todos os vértices da malha. Além de desempenho reduzido, esta técnica pode produzir falhas na superfície do modelo em locais onde partes do modelo se conectam.

A técnica de animação por esqueleto [WAT 92] tira vantagens das duas técnicas apresentadas anteriormente. Ela facilita o processo de animação de personagens articulados, pois apenas os ossos, representados por matrizes de transformação, precisam ser manipulados. Cada elemento que define a malha (pele) do personagem é associado a um ou mais ossos, o que possibilita a representação de superfícies contínuas, evitando o colapso da malha nas proximidades das juntas quando estas sofrem rotações.

Estas três técnicas se mostram suficientes para representação do comportamento do personagem, pois com um número significativamente pequeno de elementos ou vértices, pode-se expressar uma grande quantidade de movimentos e comportamentos.

Para a representação de expressões faciais que possam transmitir realismo de imagem e movimento, geralmente utilizam-se estruturas que tratam tanto a configuração dos ossos [Parke 96], bem como dos músculos presentes na face [PER 97]. Os músculos faciais podem tanto ser usados para representação das expressões bem como da fala do personagem [LUC 02].

A anatomia facial possui uma estrutura muito complexa e flexível. Por isso, implementações computacionais geralmente fazem uso de estruturas simplificadas, que possam produzir as animações faciais que se deseja obter com um nível de complexidade tratável. Geralmente utilizam-se malhas poligonais irregulares para a representação das expressões. Para expressar emoções, deve-se fazer uso da cabeça, boca, olhos, lábios, dentes e língua, orelhas, dentre outros [Lucena 02].

Existem diversas técnicas para fazer a animação de faces definidas por um modelo geométrico [Parke 96]: interpolação, baseada em performance, parametrização direta, baseada em pseudomúsculos e baseada em músculos. Dentre estas técnicas, a mais simples e, provavelmente, a mais usada é a interpolação, que faz uso de keyframes.

São seis as expressões faciais mais comuns: raiva, tristeza, alegria, medo, repulsa e surpresa [EKM 92, IZA 91], como mostradas na Figura 9. Elas possuem traços característicos, principalmente nos olhos e boca, e em locais onde há formação de rugas.

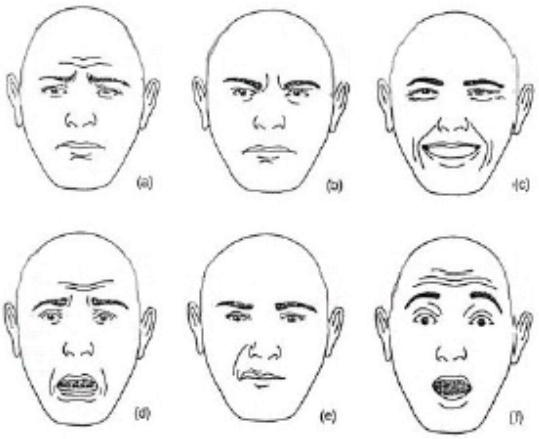

Figura 9: Exemplos das expressões faciais universais [PAR 96].

Para permitir a manipulação conjunta de animações comportamentais e faciais simultâneas, evitando a explosão combinatória de animações pré-definidas, e com um nível de suavidade entre cada transição, cada grupo de animações deve ser implementado separadamente, com respectivos keyframes. Para que isso seja possível, a técnica de representação e animação do modelo deve permitir que partes ou ossos específicos do modelo sejam tratados separadamente, o que não pode ser obtido utilizando uma estratégia que faz uso de uma única malha por keyframe para representação completa do personagem.

Entretanto, para se obter resultados intermediários, com animações comportamentais reais, e ao mesmo tempo exibindo diferentes tipos estáticos de animações faciais, pode-se usar uma combinação da técnica de malha única com uso de texturas. Nesta abordagem, além da estrutura geral do modelo, deve-se criar um conjunto de texturas, para cada personagem individualmente, que possam representar as respectivas expressões faciais. Como cada expressão é representada por um único keyframe, não se pode realizar transições suaves entre duas expressões.

Esta abordagem se mostra atraente visto que já se disponibiliza em um protótipo atual um modelo de animação que faz uso de malhas, mas que não implementa nem técnicas nem estrutura poligonal que permitam a manipulação de expressões faciais.

Desta forma, pode-se continuar usufruindo as vantagens do modelo de malha única com baixo número de polígonos, e ao mesmo tempo poder exibir algum tipo de expressão facial.

Atualmente, utiliza-se o modelo MD2, um formato de definição de arquivo, desenvolvido para o jogo Quake 2 [Henry 02], que disponibiliza um conjunto constante de keyframes e animações. Cada animação é mapeada em um intervalo predeterminado de keyframes do arquivo. Todos os keyframes do arquivo devem possuir o mesmo número de vértices.

3.2 Estipulação da Cena a Ser Exibida

No processo de execução das histórias, o módulo CEH tem como principal função fazer a distribuição de ações a cada personagem, segundo o fluxo da história sendo representada. O número de personagens de uma história pode ser muito variado e, com isso, diversas ações podem ocorrer em paralelo.

Para determinação da melhor tomada a ser exibida a cada momento, pretende-se usar regras de cinematografia. Elas foram desenvolvidas e aprimoradas para serem usadas em configurações específicas de cena e número de personagens, de forma que a tomada escolhida possa melhor transmitir ao usuário a essência da cena.

Existe uma clara diferença na forma como estas regras são aplicadas a filmes, em comparação ao trabalho sendo proposto. Em filmes, diretores tem de antemão uma descrição exata de quais cenas irão ocorrer, o que permite elaborar estratégias de filmagem específicas. Caso o resultado não produza o efeito desejado, pode-se também fazer pós-processamento ou até mesmo fazer a regravação tantas vezes quanto forem necessárias.

De forma oposta, neste trabalho tem-se a representação das cenas ocorrendo em tempo real, em um cenário aleatório, e por isso não se dispõem de recursos de pós-processamento, bem como alteração de parâmetros para corrigir erros de tomadas. O conhecimento que se disponibiliza das ações futuras dos personagens é muito abstrato. Somente da interação real (temporal) dos personagens com a cena e com outros personagens é que se pode determinar parâmetros mais detalhados de tomadas, como posição e orientação de câmera. Outro diferencial em relação ao cinema concentra-se no paralelismo de ações ocorrendo simultaneamente na representação da história.

Uma boa analogia dos problemas enfrentados neste trabalho em comparação com técnicas televisivas pode ser observado na captura de cenas de eventos esportivos ocorrendo em tempo real, nos quais não se tem conhecimento de ações futuras, e nem a capacidade de pós-edição [HE 96]. Além disso, além da ação principal, diversas outras estão ocorrendo paralelamente.

3.2.1 Princípios de Cinematografia

Segundo [HE 96], um filme pode ser visto como uma hierarquia, onde no nível mais alto estão as cenas, sendo cada uma delas composta por uma ou mais tomadas (*shots*). É considerada uma tomada o intervalo de tempo que a câmera grava continuamente. Em geral, as tomadas não duram mais do que poucos segundos.

Para a composição do filme, as tomadas não podem ser consideradas separadamente. Cenas específicas são obtidas pela combinação de uma seqüência de tomadas. Para definição da ordem destas tomadas, existem fórmulas específicas que melhor capturam a essência da cena. Uma longa discussão destas fórmulas, com situações práticas de uso pode ser encontrada em [ARI 76].

No processo de posicionamento da câmera, podem-se considerar os movimentos, relacionamentos e emoções dos personagens, bem como a construção do ambiente (configuração física da cena), luz do ambiente, dentre outros [TOM 00].

A distribuição dos personagens na cena tem um papel muito importante na definição da linha de ação. Esta linha é um vetor imaginário que pode tanto conectar dois personagens, ser apontada na direção de movimentação de um personagem ou na direção que estiver olhando [HE 96]. Esta linha divide a cena em duas áreas distintas, como mostrado na Figura 10. Diversas configurações de câmera podem ser obtidas relativas a esta linha: internal, external e apex.

Relativo a esta regra existe uma heurística associada, também conhecida como "*180-degree rule*" [HAW 03], que estabelece que tomadas de uma certa cena não podem cruzar esta linha, ou seja, se a primeira tomada for feita em um lado desta linha, todas as subseqüentes, referentes à mesma cena, devem ser realizadas no mesmo lado. Com isso, a direção da cena na tela e espaço são preservados [HAW 03].

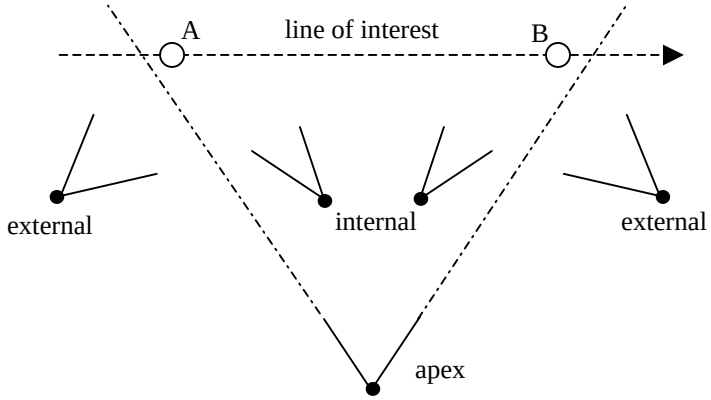

Figura 10: Especificação do posicionamento da câmera em relação à linha de interesse. [HE 96]

Diversas outras heurísticas e restrições referentes a tomadas sucessivas na criação das cenas são propostas [HE 96, CHR 96]:

- Devem-se evitar cortes bruscos. Toda vez que houver um corte entre tomadas consecutivas, deve-se deixar clara a alteração ou no tamanho, visão ou no número de personagem presentes na cena;
- Defina a cena antes de realizar tomadas próximas (*extreme closeup*). Se ocorrer um novo evento na cena, a nova situação precisa ser restabelecida antes de realizar novas tomadas próximas;
- O personagem deve iniciar movimentos, e a câmera deve seguí-lo. O contrário deve ocorrer quando o personagem estiver próximo de parar;
- Cenas que ilustram movimentos devem ser quebradas em pelo menos duas tomadas;
- As cenas devem procurar alternar entre diferentes personagens, localizações e duração.

Outro aspecto de posicionamento refere-se à distância da câmera em relação ao personagem. Esta distância tem influência na determinação da área visível do personagem em relação à tela. Pode-se definir 7 posições [HAW 03], que variam deste *full shot*, que captura toda a imagem do personagem, até *extreme closeup*, que captura somente a cabeça.

Cenas onde existam mais de dois personagens podem exigir configurações especiais, de modo a garantir que todos sejam visíveis. Para isso, caso a câmera não possa ser corretamente posicionada, deve-se alterar a posição de certos personagens.

3.2.2 Trabalhos Relacionados

Diversos trabalhos foram desenvolvidos com o intuito de criar sistemas que procuram determinar o melhor posicionamento da câmera em cenas interativas 3D [GLE 92, PHI 92]. Para situações de tomadas individuais, diversas estratégias e restrições de tomadas podem ser encontradas em Drucker et al [DRU 92, DRU 95]. Aspectos matemáticos para definição de parâmetros de câmera, dada a geometria da cena e posição do personagem desejado, podem ser vistos em [BLI 88, HAW 03, DRU 94].

Esta seção é destinada ao estudo mais detalhado de dois trabalhos que procuram criar técnicas automáticas para posicionamento e escolha de cenas em sistemas em tempo real.

[Tomlinson 2000]

Descreve um sistema automático de cinematografia para ser usado em ambientes virtuais interativos. Este sistema é responsável pelo controle de uma câmera e luzes em um mundo virtual 3D habitado por personagens autônomos e controlados pelo usuário, de forma que o conteúdo emocional possa ser evidenciado.

A câmera é implementada por um agente chamado CameraCreature. Ele encapsula sensores para extrair informações sobre os estados emocionais e motivacionais, além das ações sendo realizadas pelos agentes do ambiente. Além dos sensores, esta câmera autônoma também possui emoções, motivações e ações.

O modelo de emoção da câmera é usado na escolha da tomada a ser realizada e parâmetros relacionados. Cada estado emocional causa um efeito visual característico. O estado emocional da câmera é calculado por uma função que leva em consideração o seu temperamento, a taxa de variação da emoção no tempo e fatores internos e externos que afetam o estado emocional. Uma vez que o estado emocional dos agentes da cena tem influência no estado emocional da câmera, os estilos de tomadas refletem o estado emocional dos personagens. Para ressaltar este estado emocional do grupo, são usados tanto estratégias de posicionamento de câmeras como recursos de iluminação.

[Li-Wei He 1996]

Neste trabalho, é descrito o *Virtual Cinematographer* (VC), uma arquitetura que automaticamente gera especificações de câmera para captura de eventos em ambientes virtuais 3D, em tempo real, a partir de informações e eventos passados pela aplicação. A estrutura geral é apresentada na Figura 11.

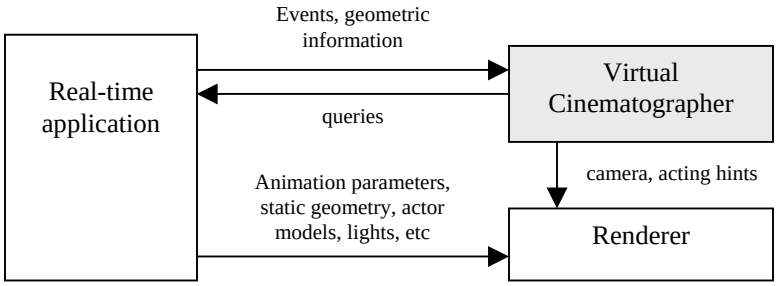

Figura 11: Arquitetura de uso do Virtual Cinematographer [He 96]

O conhecimento necessário para determinação das seqüências de tomadas para cada cena, bem como as condições necessárias fazer a transição de tomadas, são encapsulados em *idioms*. Eles são implementados por máquinas de estados finitas hierárquicas (HFSM), onde cada estado está associado com um módulo de câmera, como mostrado na Figura 12. Cada idiom tem a capacidade de capturar um tipo de situação particular, como o diálogo entre dois personagens ou o deslocamento de um personagem de um lugar para outro.

Cada módulo de câmera implementa seu posicionamento específico, sempre levando em consideração a linha de interesse da cena a ser exibida. Além disso, para melhorar as tomadas, podem também reposicionar personagens. Alguns exemplos de módulos incluem:

- apex(personagem1, personagem2): Este módulo considera dois personagens e posiciona a câmera de modo que cada um fique centrado em lados opostos da tela (Figura 10). A distância da câmera é dada em função da distância entre os dois personagens. Para tomadas mais próximas, do tipo *closeapex*, pode-se aproximar os personagens para que possam ser enquadrados pela câmera;

- track, pan e follow(personagem1, personagem2): Na Figura 13 são apresentados três exemplos de cenas onde personagem1 está se locomovendo. Nestas situações, a câmera sofre mudanças de posição e orientação de forma que o personagem sendo seguido permaneça próximo ao centro da tela. O módulo *track* posiciona a câmera em uma posição perpendicular à direção do personagem e se movimenta juntamente com ele, mantendo a mesma orientação. No módulo *pan*, a câmera permanece adiante do personagem e muda de orientação à medida que o mesmo se desloca. O módulo *follow* combina as duas transformações. Inicia com a mudança de orientação, seguida de um deslocamento.

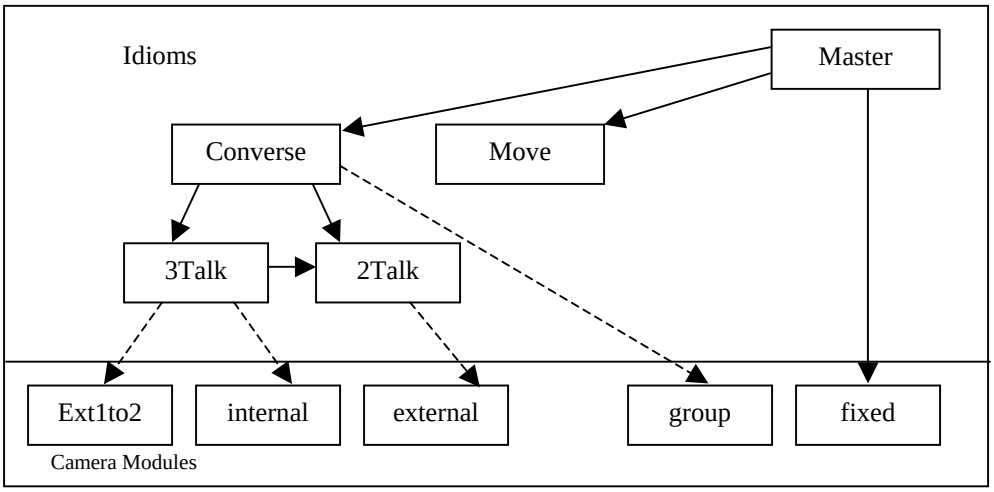

Figura 12: Associação entre idioms e módulos de câmera

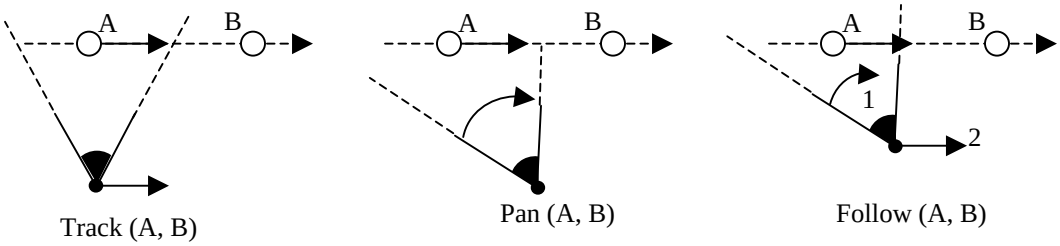

Figura 13: Exemplos de câmeras em movimento

3.2.3 Aspectos Práticos de Implementação

Um diagrama detalhado proposto para o módulo de câmera é mostrado na Figura 14. Depois do módulo de geração da história, o módulo de câmera é sem dúvida um dos mais importantes de todo o sistema, pois age como um elemento centralizador e controlador de tudo que será exibido ao usuário.

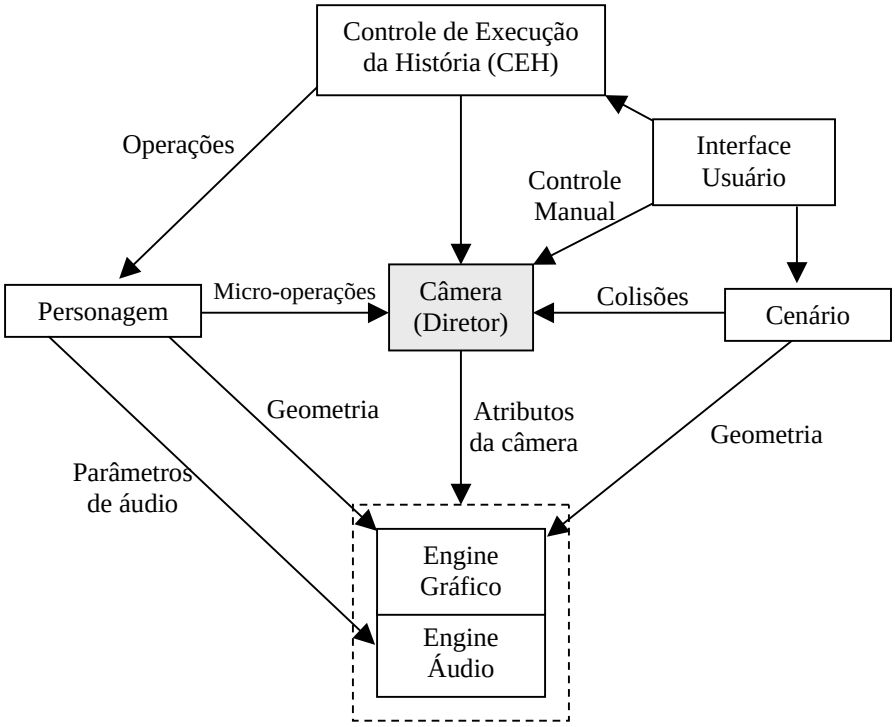

Figura 14: Diagrama detalhado do módulo de câmera e relacionados

Esta estrutura tem uma forte influência do trabalho de [HE 96]. Entretanto existem vários aspectos que diferenciam os dois modelos. Um deles está no tipo de cena a ser visualizada. Trabalhos relacionados definem modelos para serem usados no monitoramento de ações de um personagem específico, que geralmente é o que está sob o controle de cada usuário, como acontece em [HE 96]. Neste sistema sendo proposto, a câmera deve decidir, entre um elenco de personagens que compõem a história sendo visualizada, qual deve ser exibido e de que forma. Estes personagens podem estar realizando ações individuais ou grupais, onde o número de participantes pode ser muito variado.

Por existirem diversos personagens, diversas ações estão ocorrendo simultaneamente, e por isso a câmera deve saber decidir o que é mais importante a ser exibido. Esta escolha é fundamentada basicamente no grau de importância das operações sendo realizadas por personagens individuais. Para isso, como mostrado na Figura 13, existe um canal de comunicação entre a câmera com cada personagem, por onde são transmitidas as micro-operações sendo realizadas. Cada micro-operação está associada a um peso, que determina, em teoria, o que pode ser considerado de maior importância. Por exemplo, cenas de luta terão maior peso que cenas de locomoção.

Uma vez que, a cada momento (interação), tem-se um elenco de micro-operações candidatas, cabe à câmera decidir qual será exibida. Para isso, valem também as regras cinematográficas apresentadas nas Seções 3.2.1 e 3.2.2.

Para a especificação completa de parâmetros da câmera, deve-se também levar em consideração o cenário, que por ser dinâmico, pode assumir diferentes configurações. Tanto em cenas estáticas como em movimento, a câmera deve saber se posicionar de forma que não fique oclusa por possíveis objetos ou pela própria irregularidade do

terreno, e nem colida com os mesmos. Para isso, a câmera deve continuamente se comunicar com o cenário e planejar estratégias de movimentação que não violem regras cinematográficas e que consigam exibir a cena focando aspectos mais relevantes.

Além destes recursos automáticos, a câmera pode ser controlada pelo usuário de duas maneiras:

- **Navegação livre:** Neste modo de operação, o usuário pode "sobrevoar a cena" e fazer observações de quaisquer lugares sob qualquer ângulo. Mesmo nesta forma de utilização, possíveis colisões com o cenário devem ser tratadas;
- **Fixada a um personagem:** usada para acompanhar as tarefas de um determinado personagem, sob o foco de 1ª ou 3ª pessoa. Nesta configuração, a câmera deve prover recursos semelhantes aos propostos no trabalho de [HE 96].

A câmera vai ter também um papel muito importante na definição do áudio. Como mostrado na Figura 14, cada personagem tem como output tanto a geometria como o áudio associado com as micro-operações sendo executadas. Pela combinação dos áudios, juntamente com parâmetros da câmera, como posição e deslocamento, pode-se extrair com muito realismo a configuração 3D das fontes de som da cena nas proximidades do observador.

O módulo de áudio será implementado com o uso da biblioteca OpenAL (*Open Audio Library*) [OPENAL 03], que é uma API com interface para hardwares de áudio, multi-plataforma, que manipula áudio em um espaço 3D simulado. Possui extensões para tratamento de atenuação, além de efeitos Doppler, e efeitos de ambiente, como reflexão, transmissão, obstrução, dentre outros.

3.3. Recursos Gráficos para Representação

3.3.1 Modelo de representação dos dados

Como apresentado nas seções anteriores, faz-se uso de elementos 3D para a representação da cena, incluindo terrenos, personagens e objetos. Apesar do modelo 3D ser mais completo que um 2D, este também foi cogitado como uma forma de representação das histórias, porém foi rapidamente descartado.

Como a exibição da história pretende tratar de forma bastante realista a interação entre os personagens, as limitações do espaço 2D impediriam uma visão macroscópica do mundo de simulação, bem como da visualização do cenário e interação entre grupos de personagens. O uso de recursos 3D oferece maiores possibilidades, tanto referentes à visualização, bem como em algoritmos, dispositivos de hardware, técnicas e modelos 3D amplamente utilizados e estudados para aplicações em jogos e engines gráficos.

3.3.2 Estrutura Geral do Engine Gráfico

O engine gráfico realiza a última etapa do processo de visualização das histórias. Ele não realiza nenhum processamento inteligente, como ocorre com os personagens e câmera. Em suma, ele deve pré-processar e enviar para a placa gráfica os dados provenientes do cenário, personagens e câmera, como mostrado na Figura 15.

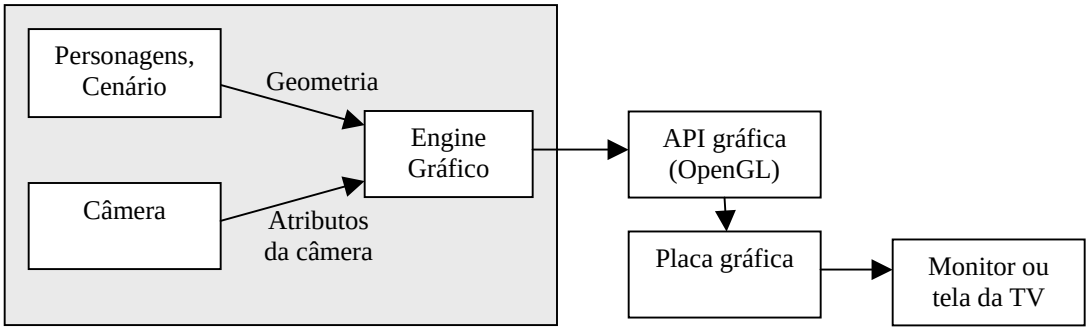

Figura 15: Pipeline de renderização do engine gráfico

Nesta etapa, o engine realiza principalmente um trabalho de seleção de informação com maior probabilidade de visibilidade, pelo uso de técnicas de cerceamento e *culling*, em função do campo de visão do observador (*view frustum*). Quanto menor for a quantidade de informação enviada para a placa gráfica, melhor será o desempenho global do sistema. Porém, deve-se sempre observar o balanceamento entre o gasto computacional realizado pelo engine com o gasto da placa gráfica, para tentar reduzir a formação de gargalos. Uma descrição mais detalhada desta análise pode ser vista em [MOL 02].

As técnicas de culling podem tanto ser aplicadas sobre terrenos como para qualquer outro objeto. Trabalhando de forma integrada com o view frustum, e com técnicas de modelagem do cenário, como grafo de cena, BSP, quadtrees, octrees, portais, *bounding volumes*, dentre outros, pode-se determinar com rapidez regiões visíveis da cena.

Uma vez que se tenha em mãos a informação visível, esta é enviada para a placa gráfica, via uma API gráfica OpenGL [WOO 99a] ou DirectX [DEM 02]. Dentro da placa gráfica, um pipeline de geometria, determinação de visibilidade, iluminação, texturização e rasterização é realizado, como mostrado na Figura 16 [MOL 02]. Além destes processos, atualmente as placas mais modernas disponibilizam pipelines programáveis, que permitem o desenvolvimento de efeitos que antes não eram possíveis, em tempo real [FER 03].

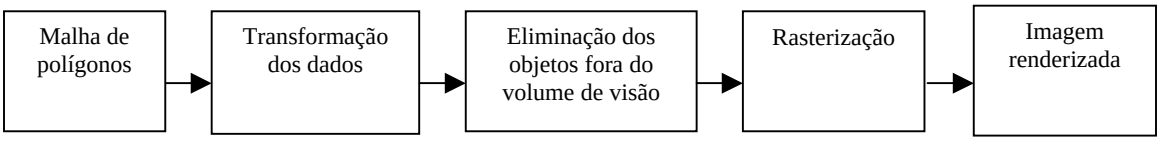

Figura 16: Pipeline de Renderização

3.3.3 Escolha de uma Estratégia de Visualização

Podem-se apontar duas abordagens claras para representação de enredos: utilizar algum engine gráfico já desenvolvido, ou desenvolver um usando uma API gráfica, como OpenGL ou DirectX. Existe atualmente uma grande quantidade de engines criados principalmente para o desenvolvimento de jogos, como o Fly3D [FLY 03], 3D GameStudio [GAM 03], dentre outros, que aparentemente podem ser utilizados no processo de exibição das histórias. A maioria destes engines implementa diversos algoritmos de animação de personagens, colisão, simulação física, otimizações de rendering, dentre outros recursos.

Para melhor entender os critérios de avaliação entre a construção de um engine e a utilização de um já construído, devem-se observar não somente os aspectos gráficos, mas também alguns outros relacionados com exibição, manipulação das histórias, comunicação com o módulo de geração das histórias, e também o contexto de aplicação que se pretende desenvolver.

Geralmente, engines de jogos concentram sua eficiência no fato de usarem algoritmos específicos e otimizados para determinadas situações, o que faz com que existam engines dedicados a jogos de aeronaves, em primeira pessoa, etc. Muitas otimizações são resultado de pré-processamento de dados (geralmente da cena), antes do início da exibição. Este pré-processamento impede que os dados possam ser alterados durante a exibição.

Ao se trabalhar com histórias interativas, deseja-se criar cenários dinâmicos, para que o usuário possa usufruir de histórias que definitivamente sejam personalizadas. A solução adotada deve fornecer recursos que permitam a total manipulação de atributos de personagens e do cenário, permitir fácil expansão e adaptação a qualquer tipo de história que um sistema de representação gráfico possa exibir, além de garantir exibição em tempo real de cenas, que podem ser complexas e dinâmicas. Além disso, o sistema deve ser facilmente adaptado a diferentes arquiteturas de hardware (*set-top boxes*) [DRI 00] usadas para prover poder computacional que permita a exibição de conteúdo iterativo em iTV.

Apesar do desenvolvimento de um engine ser uma tarefa demorada e complexa, está neste momento desenvolvendo-se no ICAD/IGAMES um conjunto de bibliotecas para manipulação gráfica (VLTools – VisionLab Tools), escritas em C++, que venham um dia a se tornar suficientes para o desenvolvimento dos mais variados gêneros de jogos. Estas bibliotecas procuram ser modulares e independentes, permitindo a total adaptabilidade a diferentes cenários de aplicações, além de disponibilizarem todo o código fonte.

Sendo desenvolvidas em níveis hierárquicos, camadas mais inferiores têm total acesso e controle sobre os dados, enquanto níveis mais elevados, já se assemelhando muito a micro-engines, realizam tarefas mais globais, e que fazem uso de diversos módulos da biblioteca. Nestas bibliotecas, faz-se uso de diversos algoritmos de domínio público, muitos deles desenvolvidos para determinados engines.

Este conjunto de bibliotecas está sendo usado no desenvolvimento do engine de visualização, que está servindo também como um campo de provas para verificar funcionalidades e definição de requisitos básicos para tornar estas bibliotecas genéricas,

funcionais e eficientes. Elas procuram abstrair a API gráfica, e poderão tanto fazer uso de OpenGL, como DirectX.

4 Resultados Parciais Obtidos

Uma primeira versão do engine de visualização já está implementada em linguagem C++, fazendo uso da API gráfica OpenGL. O principal objetivo do desenvolvimento de uma versão de testes foi levantar os principais requisitos e problemas a serem enfrentados no desenvolvimento de uma aplicação genérica que pudesse dar suporte a exibição automática de enredos.

Nesta primeira versão, objetivou-se fazer a simples representação de uma cena onde um vilão (dragão), rapta sua vítima. Os personagens (agentes) possuem implementados apenas os recursos mínimos necessários à realização de operações e micro-operações necessárias a esta representação, bem como alguns recursos para permitir a interação com o cenário que dá suporte a representação da cena.

Como este protótipo ainda não possui recursos dinâmicos para geração, armazenamento e manipulação de operações, descreveu-se no módulo CEH um script estático que define, a cada momento, quais ações cada personagem deve realizar.

Na Figura 17, é apresentada uma cena do rapto, onde um dragão captura um personagem de uma história fictícia. Esta cena é caracterizada pela perseguição da vítima, captura e transporte até o cativeiro.

Figura 17: Cena representando o rapto

Cenas de luta também já foram simuladas com o engine (Figura 18). Nelas pode-se melhor compreender e definir formas de comunicação entre agentes. Serviram também para melhor compreender os efeitos que a alteração de atributos dos personagens têm sobre o fluxo da história.

Figura 18: Cena de luta entre três personagens

5 Conclusões

Neste trabalho apresentou-se uma grande variedade de tecnologias de Computação Gráfica necessárias para tornar possível a execução e visualização de histórias interativas. Partindo-se do nível mais elementar, a definição de cada entidade que representa a história, chegou-se a uma estrutura que define todo o ferramental necessário à exibição de animações gráficas que possam representar cada uma das operações que definem a história sendo apresentada.

Pela definição de módulos e diagramas apresentados neste trabalho, tem-se uma estrutura robusta e dinâmica que se imagina ser suficiente para permitir a visualização de

um grupo muito variado de histórias, com um nível de interatividade e qualidade gráfica muito elevados.

Nesta estrutura, a câmera tem um papel fundamental, pois é a responsável pela determinação das tomadas a serem realizadas. Para isso, mantém comunicação constante com o cenário e com os personagens para adquirir informação necessária na tomada de decisões.

Ainda encontram-se em estudos estratégias para representação dos personagens, visto que podem assumir níveis muito elevados de complexidade tanto na geometria como nas características que podem representar. Diversos algoritmos de detecção de colisão, movimentação e interação entre grupos de personagens ainda devem ser discutidos.

Experimentos mais reais e detalhados vão poder definir com clareza quais são realmente os requisitos mínimos e essenciais para tornar a etapa de representação gráfica das histórias não somente um meio de exibição, mas também de simulação, que tem também como objetivos engrandecer a experiência interativa para com o usuário.

Agradecimentos

Os autores agradecem o suporte financeiro do CNPq e da FINEP através das bolsas de doutorado e de pesquisa e através do Projeto VisionLab.

Referências Bibliográficas

[AME 00] Amerson, D., Kime, S. Real-time cinematic camera control for interactive narratives. American Association for Artificial Intellingence, pp. 1-4, 2000.

[ARI 76] Arijon, D. Grammar of the film language. Communication Arts Books, Hastings House, Publishers, New York, 1976.

[BLI 88] Blinn, James. Where I am? what am I looking at? IEEE Computer Graphics and Applications, pp. 76-81, 1988.

[BRE 98] Brezeal, C. A motivational system for regulating human-robot interaction. Proceedings of the Fifteenth National Conference on Artificial Intelligence, Madison, pp. 54-61, 1998.

[CHR 96] Christianson, D., Anderson, S., He, L., Salesin, D., Weld, D., Cohen, M. Declarative camera control for automatic cinematography. In Proceedings of the AAAI-96, August 1996.

[CIA 99] Ciarlini, A. Geração interativa de enredos. Tese de Doutorado, Departamento de Informática, PUC-Rio, 1999.

[CIA 02] Ciarlini, A., Feijó, B., Furtado, A. An integrated tool for modelling, generating and exhibiting narratives. AIS'2002 AI, Simulation and Planning in High Autonomy Systems, Lisboa, Portugal, pp. 150-154, April, 2002.

[CUN 01] Cunha, L. S., Giraffa, L. M. M. Um estudo sobre o uso de agentes em jogos computadorizados interativos. Technical Report Series, number 017, October 2001, PUCRS

[DEM 02] Dempski, K. Real-time rendering tricks and techniques in directx. Premier Press, 821p, 2002.

[DRI 00] Driscoll, G. The essential guide to digital set-top boxes and interactive TV. Ed. Prentice Hall, 2000.

[DRU 92] Drucker, S., Galyean, T, Zeltzer, D. Cinema: a system for procedural camera movements. In David Zeltzer, editor, Computer Graphics (1992 Symposium on Interactive 3D Graphics), Vol. 25, pp. 67–70, March 1992.

[DRU 94] Drucker, S.M. Intelligent camera control for graphical environments. PhD Thesis, MIT Media Lab, 1994.

[DRU 95] Drucker, S. M., Zeltzer, D. Camdroid: a system for implementing intelligent camera control. In Michael Zyda, editor, Computer Graphics (1995 Symposium on Interactive 3D Graphics), Vol. 28, pp. 139–144, April 1995.

[EBE 01] Eberly, D. 3d game engine design, a practical approach to real-time computer graphics. Ed. Morgan Kaufmann, 2001.

[FER 03] Fernando, R., Kilgard, M. J. The cg tutorial: the definitive guide to programmable real-time graphics, Addison-Wesley, 384 p., 2003.

[FLY 03] Fly3D Homepage. (March 2003). Disponível em: http://www.fly3d.com.br/

[FUR 96] Furht, B. Interactive television systems. Proceedings of the 1996 ACM symposium on Applied Computing, pp. 7-11, February 1996.

[FUR 99] Furtado, A., Ciarlini, A. Operational characterization of genre in literary and real-life domains. In Proc. of the ER'99 Conceptual Modeling Conference, Paris, France, November 1999.

[FUR 02] Furtado, A. L., Ciarlini, A. E. M. Cognitive and affective motivation in conceptual modelling. Revista Colombiana de Computación, Vol 3 (2), December 2002. Disponível em: http://www.unab.edu.co/editorialunab/revistas/rcc/rev32.htm

[GAM 03] 3D Game Studio. (June 2003) Disponível em: http://www.conitec.net/a4info.htm

[GAR 97] Garland, M., Heckbert, P. S. Surface simplification using quadric error metrics. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 209-216, August 1997.

[GAR 98] Garland, M., Heckbert, P. S. Simplifying surfaces with color and texture using quadric error metrics. IEEE visualization 98, pp. 263-269, July 1998.

[GLE 92] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In Edwin E. Catmull, editor, Computer Graphics (SIGGRAPH'92 Proceedings), Vol. 26, pp. 331–340, July 1992.

[HAW 03a] Hawkins, B. Creating an event-driven cinematic camera, part one. January 8, 2003. Disponível em: http://www.gamasutra.com/features/20030108/hawkins_01.htm.

[HAW 03b] Hawkins, B. Creating an event-driven cinematic camera, part two. January 8, 2003. Disponível em: http://www.gamasutra.com/features/20030110/hawkins_01.htm.

[HE 96] He, L., Cohen, M. F., Salesin, D. H. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (Proceedings of SIGGRAPH), pp. 217-224, August 1996.

[HEN 02] Henry, D. The quake II's md2 file format. December 2002. Disponível em: http://tfc.duke.free.fr/us/tutorials/models/md2.htm

[JEN 96] Jennings, N. R. Coordination techniques for distributed artificial intelligence. In: O'HARE, G.M.P.; Jennings, N.R. (Eds.). Foundations of distributed artificial intelligence. New York: John Wiley & Sons. pp.187-210, 1996.

[JUC 01] Juchem, M., Bastos, R. M. Engenharia de sistemas multiagentes: uma investigação sobre o estado da arte. Technical Report Series, number 014, April 2001, PUCRS.

[KEN 02] Kennedy, K., Mercer, R. E. Planning animation cinematography and shot structure to communicate theme and mood. Proceedings of the 2nd international symposium on Smart graphics, pp. 1-8, June 2002.

[LID 02] Lidén, Lars. Strategic and tactical reasoning with waypoints. In: AI Game Programming Wisdon. Ed. Charles River Media, 2002.

[LUC 02] Lucena, P. S. Expressive talking heads: um estudo de fala e expressão facial em personagens virtuais. Dissertação de Mestrado, PUC-Rio, 2002.

[MOL 02] Moller, T., Haines, E. Real-time rendering, A.K.Peters Ltd, Natick, MA, 900 p., 2nd edition, July 2002.

[NAR 03] Nareyek, A. AI center homepage. (April 2003) Disponível em. http://www.ai-center.com/home/alex/publications.html

[OPE 03] Open audio library. (July 2003). Disponível em: http://www.openal.org/

[PAR 96] Parke, F. I., Waters, K. Computer facial animation. A K Peters Ltd, 384p, September 1996.

[PER 97] Perlin, K. Responsive face. Technical report, Media Research Lab, New York University, 1997. Disponível em: http://mrl.nyu.edu/~perlin/demox/Face.html

[PHI 92] Phillips, C. B., Badler, N. I., Granieri, J. Automatic viewing control for 3D direct manipulation. In David Zeltzer, editor, Computer Graphics (1992 Symposium on Interactive 3D Graphics), Vol. 25, pp. 71–74, March 1992.

[PIN 02] Pinter, M. Realistic turning between waypoints. In: AI Game Programming Wisdon, Ed. Charles River Media, 2002.

[POZ 03a] Pozzer, C. T., Feijó, B., Ciarlini, A. Proposição de um novo paradigma de conteúdo para TV interativa. Série Monografias em Ciência da Computação (MCC38/03), DI/PUC-Rio, Rio de Janeiro, 2003.

[POZ 03b] Pozzer, C. T., Furtado, A., Ciarlini, A. Agentes e emoções em histórias interativas. Série Monografias em Ciência da Computação (MCC39/03), DI/PUC-Rio, Rio de Janeiro, 2003.

[SHE 98] Shehory, O. Architectural properties of multiagent systems. Pittsburgh, PA: [s.n.], 1998.

[SNO 03] Snook, G. Real-time 3D terrain engines using C++ and directx 9. Charles River Media, 374 p., June 2003.

[TOL 00] Toledo, R. P. R. Quadlod, uma estrutura para a visualização interativa de terrenos. Dissertação de Mestrado, PUC-Rio, Abril de 2000.

[TOM 00] Tomlinson, B., Blumberg, B. Nain, D. Expressive autonomous cinematography for interactive virtual environments. Proceedings of the fourth international conference on Autonomous agents, pp. 317-324, June 2000.

[ULR 00] Ulrich T. Continuous lod terrain meshing using adaptative Quadtrees. (February 2000). Disponível em: http://www.gamasutra.com/features/20000228/ulrich_pfv.htm

[ULR 02] Ulrich T. Rendering massive terrains using chunked level of detail control. (April 2002). Disponível em: http://tulrich.com/geekstuff/chunklod.html#downloads

[VEL 97] Velázquez, J. D. Modeling emotions and other motivations in synthetic agents. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, pp. 10-15, Menlo Park, Calif.: AAAI Press, 1997.

[WAT 92] Watt, A., Watt, M. Advanced animation and rendering techniques. Addison-Wesley, Wokingham, England, 1992.

[WOO 99a] Woo, M., Neider, J., Davis, T., Shreiner, D. OpenGL programming guide, third edition. Ed. Addison Wesley, 730p. 1999.

[WOO 99b] Wooldridge, M. Intelligent agents. In: WEISS, G. (Ed.) Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.