7
Implementação e Resultados

Níveis de abstração são uma realidade na construção de motores (*engines*) para tratar histórias interativas [129]. Um mesmo personagem pode ser visto, em diferentes níveis de abstração, de forma totalmente diferente. No nível de geração da história (IPG), os personagens são apenas entidades lógicas, que possuem objetivos a serem cumpridos. Quando tratados no nível de interação com o usuário, são entidades fictícias que podem ser manipuladas e organizadas com certa ordem para gerar encadeamentos de eventos. No nível de representação gráfica, a abstração é substituída por uma forma física, que permite expressar graficamente seus comportamentos. O usuário pode presenciar cada ação executada e todo o nível de detalhes que o ambiente onde o personagem está inserido disponibiliza.

O nível de detalhes da implementação dos personagens (que neste caso são vistos como atores guiados [132][23]) deve ser suficiente para permitir que eles tenham uma representação "realista", saibam cumprir ordens e, ao mesmo tempo, possuam autonomia para tomar certas decisões (nível de planejamento em baixo nível de detalhes), dentro de um espectro limitado pela concepção do sistema. Estas decisões levam o personagem ao cumprimento de sua meta da maneira mais adequada do ponto de vista do usuário que está guiando a história interativa.

Como apresentado no capítulo anterior, existem vários módulos, tanto de IA como de CG, que trabalham conjuntamente para permitir que as histórias simuladas no IPG possam ser visualizadas interativamente. Este capítulo aborda aspectos de implementação do sistema proposto, ou seja, detalhes mais específicos de como cada módulo funciona e como se dá o sincronismo entre eles para garantir a coerência entre os dados processados a nível lógico com os dados apresentados por meio de recursos gráficos.

Inicialmente, apresentam-se discussões relativas à especificação do motor gráfico usado na representação das histórias. São abordados aspectos da escolha de um modelo gráfico adequado, bem como da especificação do

cenário. Do mesmo modo, apresentam-se detalhes mais específicos dos tipos de personagens utilizados, tratamento de ações individuais e em grupo, bem como mecanismos de comunicação entre personagens.

A Seção 7.4 apresenta os atributos que cenário e personagens incorporam, quando tratados a níveis de simulação e visualização. Juntamente, discutem-se estratégias de sincronismo destes atributos entre a geração e a visualização.

Na seqüência, apresentam-se detalhes da especificação da câmera virtual, que é o módulo responsável pela seleção automática do conteúdo a ser exibido ao usuário.

Na Seção 7.6, apresenta-se o ciclo de renderização das histórias ou, mais especificamente, as etapas que o motor gráfico realiza para transformar os eventos em animação gráfica. Apresenta-se também como é realizado o controle temporal durante a visualização de um evento.

Na Seção 7.7 apresentam-se as diferentes formas pelas quais o usuário pode interagir com o sistema, usando interfaces gráficas e, ao final, os resultados gráficos obtidos.

Os detalhes relativos a códigos e documentação do sistema podem ser encontrados no repositório ICAD/IGames/VISIONLAB [74].

7.1 Escolha de um Modelo Gráfico

Durante o desenvolvimento desta tese, o primeiro item levantado na especificação do motor foi o modelo gráfico a ser adotado: 2D ou 3D. Esta escolha define todos os demais aspectos referentes ao tipo de cena utilizada, tipos de personagens, ações que podem ser realizadas e, conseqüentemente, os recursos que a câmera virtual pode evidenciar.

Como a exibição da história pretende tratar de forma bastante realista a interação entre os personagens, as limitações do espaço 2D impedem a visão macroscópica do mundo de simulação, a visualização do cenário e a interação entre grupos de personagens. O uso de recursos 3D oferece maiores possibilidades, tanto referentes à visualização bem como a algoritmos, dispositivos de hardware, técnicas e modelos 3D amplamente utilizados e estudados para aplicações em jogos. Nesta escolha, não foi considerado o modelo isométrico (2.5D).

Uma vez escolhido o modelo 3D como forma de representar as histórias, buscaram-se alternativas para a sua implementação. Pode-se apontar de imediato duas abordagens possíveis: utilizar algum motor gráfico

já desenvolvido ou desenvolver um usando uma API gráfica, como OpenGL [147] ou DirectX [93]. Existe atualmente uma grande quantidade de motores criados principalmente para o desenvolvimento de jogos (como o Fly3D [52] e o 3D GameStudio [153]. Uma lista completa de motores pode ser encontrada em [50]), que aparentemente podem ser utilizados no processo de exibição das histórias. A maioria destes motores disponibiliza diversos recursos para animação de personagens, tratamento de colisão, simulação física, otimizações de renderização e scripts de IA.

Para melhor entender os critérios usados na avaliação entre a construção de um motor e a utilização de um já existente, são observados não somente os recursos disponíveis mas, principalmente, o nível de customização oferecido. 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*) usadas para prover poder computacional que permita a exibição de conteúdo iterativo em iTV. Deseja-se ter o máximo de customização sobre todos os processos envolvidos. Por estes motivos, apesar do desenvolvimento de um motor próprio ser uma tarefa demorada e complexa, a presente tese apostou nesta direção.

O motor está escrito em Linguagem C++, juntamente com a API gráfica OpenGL [147]. Sempre que possível, o motor incorpora bibliotecas de domínio público para tratar a manipulação de personagens e objetos do cenário (como no caso dos formatos MD2 e 3DS).

7.2 Estrutura do Cenário

O cenário é o local onde a história é representada graficamente. Ele serve como base para a simulação da interação entre personagens. Esta simulação, no presente trabalho, é realizada com um enfoque um tanto diferente, se comparado com o processo de geração da história pelo IPG. São considerados aspectos físicos, como distâncias, obstáculos e, conseqüentemente, aspectos do personagem, como velocidade e autonomia para determinar meios de realização das ações a eles designadas.

O cenário é definido como um ambiente aberto. O elemento de suporte aos demais componentes da cena é o terreno, onde são dispostos os objetos estáticos, como as construções, além dos objetos móveis, no caso, os personagens. Os objetos do cenário têm como principal papel servir como auxílio à navegação dos personagens em comportamentos de manobra. Para isso, todos os objetos possuem a mesma estrutura lógica baseada em *waypoints*, como apresentado na Seção 5.2.2. Os *waypoints* também são usados como auxílio ao posicionamento da câmera virtual (Seção 7.5). Os objetos são representados por modelos 3D e, para o contexto dos enredos adotado nesta tese, são usados para caracterizar casas, igrejas e castelos. Além do terreno e objetos, para tornar o ambiente mais realista, faz-se uso de uma biblioteca para a criação de um céu realista [4].

Na base Prolog, cada história a ser representada tem um elenco de personagens. Cada personagem, por sua vez, está sempre associado a locais, que podem representar residências ou objetos onde o personagem deve realizar uma ação. Como a quantidade de locais a serem representados no cenário depende da especificação da base Prolog, o terreno, associado à história, deve ter características que possam permitir uma distribuição adequada destes objetos sobre ele. Ele pode ser não-planar, desde que não possua regiões com altas irregularidades que impeçam a passagem dos personagens. Para assegurar estas restrições, a definição do terreno fica a cargo do autor da história, que também tem a função de definir a localização que cada objeto deve assumir.

A geração de terrenos não-planares 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 7.1. Para tornar a renderização mais eficiente, geralmente faz-se uso de algoritmos de LOD (*Level of Detail*) [88][58][46][89][139][140][90]. Esta é uma técnica que consiste em reduzir a complexidade da malha (número de triângulos), procurando manter, ao máximo, a geometria visual da mesma. Neste trabalho, foi utilizado um algoritmo proposto por Pozzer et al. [113].

7.2.1 Inicialização do cenário

O carregamento do cenário é realizado no início do processo de visualização, em função do contexto da história que está sendo processada. Cada contexto de história está associado a um arquivo texto que contém

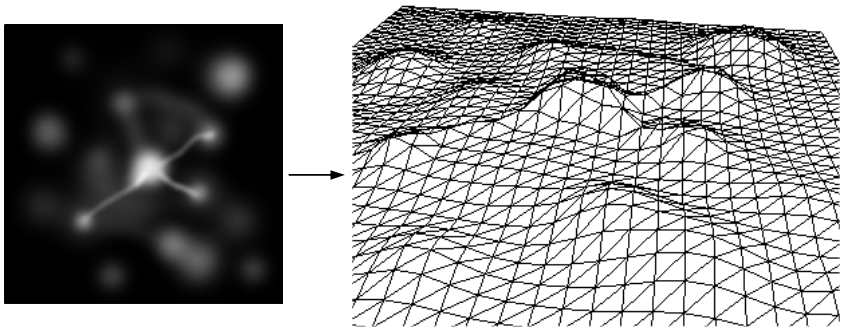

Figura 7.1: Geração de uma malha regular a partir de uma imagem.

a definição do terreno (mapa de altura e textura), dos objetos (arquivo, formato e escala) e de seus atributos (tipo, nome, posição, modelo 3D e orientação). A figura 7.2 apresenta um modelo deste arquivo.

```
//type   heightmap  texture
terrain  map.raw    textmap.jpg

//type    name            scale
modelPOZ  castle1.poz     7000
modelPOZ  castle2.poz     5000
modelPOZ  castle3.poz     5000
modelPOZ  church.poz      6000

//type       name             pos_x  pos_z  model        rotation
buildingPOZ  castle           18300  21000  castle1.poz  -30
buildingPOZ  knight_castle    24300  32700  castle1.poz   180
buildingPOZ  dragon_castle    34000  32000  castle2.poz   160
buildingPOZ  princess_castle  16300  20700  castle3.poz  -30
buildingPOZ  church           35500  19500  church.poz    160
```

Figura 7.2: Arquivo de especificação do cenário.

Esta estrutura permite que um mesmo modelo 3D possa representar mais de um elemento no cenário. O sistema faz uso de três bibliotecas para carregamento de modelos 3D: um para modelos MD2[1] [70], um para 3DS[2] [73] e um para modelos POZ[3].

Dispondo-se dos objetos 3D e do terreno, pode-se facilmente, por meio deste arquivo, criar vilarejos realistas com ruas (que podem ser representadas pela textura do terreno), igrejas, casas, comércio e fábricas. O cenário pode ter objetos (locais) que não sejam de conhecimento do IPG, porém o inverso não pode ocorrer.

[1]O formato MD2 foi criado para ser usado na modelagem das animações dos personagens do Jogo Quake II.

[2]Formato exportado pelo programa 3D Studio Max.

[3]O formato POZ foi desenvolvido pelo autor desta tese como forma de contornar limitações apresentadas pelo carregador 3DS quando está tratando modelos complexos. O arquivo 3DS é carregado pelo programa Deep Exploration [40], que exporta a malha do modelo em formato CPP, que é então convertida para o formato POZ.

7.3 Personagens

Existem dois tipos de personagens implementados no visualizador gráfico: os principais e os figurantes. Os principais estão definidos na base Prolog e são usados pelo IPG no processo de simulação da história. Os personagens figurantes existem apenas no motor gráfico e são usados para tornar a representação das histórias mais realista, uma vez que comumente existem poucos personagens principais. Até o momento, não se tem conhecimento de nenhum trabalho em *storytelling* que faça uso de personagens figurantes com este intuito.

Pela concepção do sistema integrado de simulação/visualização, como o próprio nome diz, toda a parte de geração da história deve se concentrar na simulação, sobrando para a visualização apenas a representação gráfica das ações fornecidas pelo simulador. Entretanto, é importante poder adicionar um certo nível de simulação no módulo de visualização, visto que diversos fatos interessantes podem surgir da interação física entre personagens. Isso resulta principalmente da existência dos personagens figurantes, que podem estar em contínua interação com os personagens principais da história.

A solução ideal nesta configuração é que os personagens principais possam interagir de forma um tanto realista com os figurantes, desde que o direcionamento da história não mude mediante esta forma de interação. Para que isso seja possível, trata-se a interação física de maneira normal, porém fica o personagem figurante impedido de alterar qualquer atributo de um personagem principal. Deste modo, por exemplo, um figurante jamais pode, inesperadamente, matar um personagem principal.

Os personagens figurantes podem ser usados para dois fins distintos: servir como auxílio à realização de determinadas ações que os personagens principais devem desempenhar, bem como para tornar o cenário de visualização mais povoado. Eles podem desempenhar diversas profissões. Até o momento da escrita desta tese, os guardiões são os únicos personagens figurantes implementados. Na especificação lógica da história, cada local está associado a um nível de proteção, que é um valor numérico que varia de 0 a 100. Como já mencionado, diversas ações fazem uso destes atributos como pré-condições para a sua execução. No motor gráfico, faz-se uso de figurantes, que representam guardiões, para implementar este atributo. Se a proteção de um local for 50, devem existir 5 guardiões (1 guardião equivale a uma proteção igual a 10). Assim, por exemplo, a operação `Reduce_protection` consiste em ordenar que um guardião, de um local

especificado, vá para sua casa. A operação `Attack`, por sua vez, faz com que o atacante mate dois guardiões. Com esta estratégia, fica visível para o usuário o quão protegido cada local é.

Além dos guardiões, outros tipos de figurantes poderiam ser pensados tais como:

- Vendedores - Podem ser usados para diversos fins. Como exemplos, considerando-se que os personagens tenham drives, os vendedores de alimento podem ser a forma de suprir a fome. Caso um personagem necessite de poderes para derrotar o inimigo, esse poderia ir até um mago para comprar objetos mágicos que o tornassem mais forte; e
- Trabalhadores - Podem ser usados para povoar o cenário. Cada personagem figurante deve ter um local específico no cenário para trabalhar. Desta forma, uns podem ir para as fazendas, enquanto outros, para as fábricas, por exemplo.

Além de poderem desempenhar diversas funções, os personagens figurantes, exceto os guardiões, que estão indiretamente associados à história, também podem exibir comportamentos rotineiros, como ir para o trabalho de manhã e voltar para casa ao final do expediente. Ações entre figurantes, realizando histórias paralelas, também aparenta ser um recurso adicional para tornar a representação gráfica mais realista.

7.3.1 Inicialização dos Personagens

Todos os personagens, principais e figurantes, são especificados em um arquivo, da mesma forma como ocorre com o cenário. O arquivo de personagens deve ser processado após o do cenário, visto que referencia locais que já devem estar definidos dentro do visualizador. Na Figura 7.3 é apresentado um exemplo deste arquivo e, na Tabela 7.1, o significado de cada parâmetro. Para permitir que um mesmo modelo 3D (no formato MD2) possa representar mais de um personagem no cenário, estes são definidos no início do arquivo e então referenciados nas definições dos personagens que se seguem.

```
//      model              texture             scale
model   knight.md2         knight.pcx          15  //actors
model   princess.md2       princess.pcx        15
model   dragon.md2         dragon.pcx          5
model   soldier.md2        soldier.pcx         15
model   skeleton.md2       skeleton.pcx        15

model   knightW.md2        knightW.pcx         15  //weapons
model   soldierW.md2       soldierW.bmp        5
model   skeletonW.md2      skeletonW.bmp       15

//  type name     num  home            work            model         weapon
actor P  brian     1   knight_castle   nil             knight.md2    knightW.md2
actor P  hoel      1   princess_castle nil             knight.md2    knightW.md2
actor P  marian    1   princess_castle nil             princess.md2  nil
actor P  draco     1   dragon_castle   nil             dragon.md2    nil
actor F  soldier  10   princess_castle princess_castle soldier.md2   soldierW.md2
actor F  skeleton 10   dragon_castle   dragon_castle   skeleton.md2  skeletonW.md2
```

Figura 7.3: Arquivo de especificação dos personagens.

7.3.2 Sistema de Troca de Mensagens entre Personagens

Para que um personagem execute uma ação coletiva, este deve enviar uma mensagem ao personagem com o qual deseja interagir. A mensagem é transmitida via uma chamada de método ao personagem alvo. Quando um personagem recebe uma mensagem de início de ação coletiva e a aceita, é criada uma nova ação terminal com os dados passados no corpo da mensagem, que em sua grande maioria, são iguais aos dados presentes na ação do personagem que originou a ação coletiva.

Atualmente, as mensagens têm como único propósito a convocação para uma luta a dois. A ação de luta ocorre em duas circunstâncias: quando o IPG gera uma ação `Fight` explícita, ou de forma indireta, no caso de uma ação do tipo `Attack`. No primeiro caso, o evento já contém os personagens que devem interagir. No segundo, dentro do estado `Attack` da FSM, o personagem atacante deve perguntar ao **Gerenciador de Ações** com qual personagem deve lutar. Este personagem deve ter três características: ser um figurante guardião do local onde o personagem principal deve atacar, estar próximo do personagem atacante e deve estar vivo. Na implementação corrente, a ação `Fight` nunca ocorre entre dois figurantes. Entretanto, este tipo de ação, onde os guardiões de um local atacam guardiões de outro local, já foi simulado, gerando resultados muito interessantes. O grande problema é que não se tem controle sobre o resultado deste tipo de ação, pois ambos os personagens são figurantes e também porque isso afeta a proteção dos dois locais, uma vez que vai haver baixas nos dois lados. Para representações mais realistas deste tipo de confronto, podem-se usar estratégias de simulação de confrontos [127].

Além de mensagens trocadas, também existe um mecanismo

Tabela 7.1: Descrição dos parâmetros do arquivo de personagens

Label	Descrição
Type	Tipo do personagem: Principal (P) ou Figurante (F)
Num	Número de personagens com os mesmos atributos. Para os personagens que representam guardiões, o valor deve ser 10, para permitir que o nível de proteção possa ser ajustado posteriormente conforme a especificação da base Prolog.
Home	Localização (objeto do cenário) onde o personagem reside. Na inicialização do motor, todos os personagens estão dentro de suas casas.
Work	Local do cenário onde o personagem trabalha. Somente é usado para Figurantes. Quando a visualização tem início, todos os figurantes recebem a ordem para ir trabalhar, o que faz com que os respectivos locais tenham o nível de proteção correto.
Model	Modelo MD2 usado para representar o corpo do personagem.
Weapon	Modelo MD2 usado para representar a arma do personagem.

implementado para alteração do estado dos personagens, que podem assumir três valores: `DEAD`, `FREE` e `LOCKED`. O estado `DEAD` representa que o personagem não pode mais realizar nenhuma ação. O estado `LOCKED` significa que o personagem foi raptado e que somente poderá executar novas ações quando algum outro personagem, no caso o herói da história, fizer sua libertação. Por default, todos os personagens estão em estado `FREE`, o que os permite realizar qualquer ação.

7.4 Integração dos Módulos de Geração e Visualização

Na Tabelas 7.2 e 7.3 são apresentados os atributos que são associados aos dois tipos de personagens e objetos do cenário, respectivamente. A definição destes atributos, bem como principalmente do módulo onde devem estar localizados, é resultado de um longo estudo e de várias discussões. Como apresentado nestas tabelas, são poucos os atributos presentes em ambos os módulos, o que facilita o sincronismo dos mesmos.

Para garantir que a história gerada pelo IPG seja corretamente representada no motor gráfico, os atributos que são comuns aos dois módulos devem estar sempre sincronizados. Para os atributos de personagens, isso ocorre normalmente à medida que os personagens realizam as ações. O efeito de uma ação realizada no simulador deve ter o mesmo efeito no visualizador.

Tabela 7.2: Atributos de Personagens na Visualização e no IPG ("x" indica atributo incluído)

Atributos	**Visualização**	**IPG**	**Comentários**
Localização	x	x	Na visualização, a localização representa uma posição absoluta (x,y) do personagem, enquanto no Prolog, um local (*place*).
Estado	x	x	Representa se o personagem está vivo.
Modelo 3D	x		Modelos MD2 que representam o corpo e utensílios do personagem.
Velocidade	x		Velocidade que o personagem se desloca no cenário.
Afeição		x	Parâmetro usado apenas na simulação da história, como pré-condição para o casamento.
Natureza (índole)		x	Parâmetro usado apenas na simulação da história, para impedir luta entre personagens com mesma índole (bem x mal).
Objetivos		x	No visualizador, os personagens são apenas agentes reativos que cumprem ordens.
Drives	x		*Atributo não implementado*
Emoções	x	x	*Atributo não implementado*

O atributo proteção é o único que é sincronizado automaticamente quando a aplicação é inicializada. Assim que os arquivos de definição do cenário e personagens são carregados, o visualizador realiza consultas diretamente ao IPG para saber o nível de proteção de todos os objetos definidos no arquivo. Em função deste valor, guardiões podem ser removidos da representação.

7.5 Câmera Virtual

Todos os personagens ficam continuamente executando ações fornecidas pelo **Gerenciador de Ações** em função do evento corrente.

Tabela 7.3: Atributos de Objetos do cenário na Visualização e no IPG ("x" indica atributo incluído)

Atributo	**Visualização**	**IPG**	**Comentários**
Localização	x		A localização representa a posição absoluta (x, y) do objeto no cenário.
Proteção	x	x	Único atributo comum. No Prolog é representado por um valor numérico, que é mapeado no visualizador em número de guardiões.
Modelo 3D	x		Modelos 3DS e POZ que representam a estrutura física do objeto.
Tipo		x	Possui o mesmo valor da natureza (ou índole) dos personagens que nele residem. É usado para garantir que um personagem não ataque locais dominados por personagens com índole igual à sua.

Quando não existem eventos a serem processados, ações defaults são delegadas, em função do tipo de personagem.

O papel da câmera é ficar continuamente verificando o evento corrente. Por meio deste evento, descobre-se o personagem corrente, que por sua vez, pode informar a ação específica que está desempenhando (topo da pilha - Seção 5.2).

Uma vez detectado o personagem e sua ação corrente, a câmera deve automaticamente se posicionar de modo a capturar a representação em andamento. A posição do personagem é usada como referencial para auxiliar no posicionamento da câmera, cuja localização específica é dada em função da ação. Apenas três ações têm câmera implementada, como mostrado na Tabela 7.4. Todas as demais ações, por fazerem uso de ações primitivas ou serem ações que não têm representação gráfica específica, fazem uso das demais.

A câmera deve procurar ressaltar duas características da cena: o local onde a ação está ocorrendo e os personagens que fazem parte da ação. Tomadas muito próximas (*close ups*) dos personagens não se fazem necessárias visto que estes não implementam emoções. Para melhor situar o usuário do que está ocorrendo na representação, deve-se procurar sempre contextualizar o local onde a ação vai ocorrer, pois como a câmera se baseia

Tabela 7.4: Tipos de câmeras para diferentes ações

Ação	**Implementação**	**Ações que a implementam**
`Walk`	x	
`Reduce_protection`		`Stand`
`Attack`		`Walk, Fight`
`Fight`	x	
`Kidnap`		`Walk`
`Free`		`Walk`
`Work`		`Walk, Stand`
`Kill`		`Fight`
`Marry`		`Fight`
`Stand`	x	
`Get_stronger`		`Stand`

em tomadas, com certa freqüência ocorrem cortes entre tomadas sucessivas, principalmente quando há mudança do personagem-foco.

Durante o desenvolvimento desta tese, o gerenciamento de cortes bruscos foi um dos maiores problemas enfrentados na especificação do módulo de câmera. Para reduzir o efeito negativo de cortes bruscos entre duas tomadas consecutivas, faz-se uso de uma estratégia que usa um repositório (*pool*) de três câmeras. Nesta abordagem, tem-se a câmera corrente (a que está sendo usada) e duas auxiliares que são candidatas ao cargo de câmera corrente. Cada câmera contém especificações próprias de local e orientação. A idéia de se ter várias câmeras é a possibilidade de se poder selecionar a mais adequada a cada momento. Esta escolha é dada por uma função heurística, que leva em consideração a ação antiga, a ação corrente, a posição do personagem, o peso das câmeras e a mudança de personagem-foco.

7.5.1 Criação das Câmeras

Sempre que ocorre troca de personagem ou ação, as duas câmeras que não estão sendo utilizadas são reconfiguradas para a nova situação. A mudança de ação nem sempre significa troca de personagem, porém o contrário é sempre verdadeiro. A determinação da nova posição, além da posição do personagem e ação corrente, envolve também possíveis objetos

localizados na vizinhança, uma vez que a câmera não pode ficar localizada dentro de um objeto. Exceto quando os personagens estão se locomovendo entre dois locais, eles sempre estão em um local específico. Desta forma, podem-se usar informações de *waypoints* dos objetos como forma de auxílio ao posicionamento da câmera. As três ações básicas de câmera são definidas a seguir:

Ação Walk: Esta é a única ação que envolve movimento de personagens no cenário. Quando o personagem recebe esta ação, ele deve planejar o caminho ao destino, como apresentado na Seção 5.2.2. Tendo-se o personagem e sua ação corrente, a câmera consulta a pilha do personagem e faz uma avaliação do caminho planejado. Existem dois casos possíveis, como mostrado na Figura 7.4. Se o caminho não tiver desvios, duas novas posições perpendiculares às posições inicial e final são criadas. Caso haja desvio, a posição final é dada pelo vetor normal aos dois caminhos intermediários. Para este e todos os demais tipos de câmeras, sempre que o personagem-foco se mover, a câmera corrente faz um novo enquadramento por meio de uma rotação em seu eixo vertical. Tanto para esta ação, como para as demais, caso a posição selecionada esteja dentro de um objeto do cenário, esta posição é substituída pelo *waypoint* do objeto mais próximo da posição originalmente estabelecida. Neste caso, há uma redução de seu peso, visto que a posição selecionada é apenas uma aproximação da posição originalmente estabelecida.

Ação Fight: Para a ação `Fight`, as câmeras são posicionadas de modo a capturar ambos os personagens vistos de dois pontos distintos, como mostrado na Figura 7.5. Considera-se um deslocamento tanto na direção da reta que define a linha de ação dos personagens, como na direção perpendicular a esta linha.

Ação Stand: Sempre que um personagem está parado, ele está nas proximidades de um objeto do cenário (área interna ou externa). Deste modo, faz-se uso da estrutura de *waypoints* do objeto onde ele se localiza para posicionar a câmera de modo a contextualizar o personagem e o local onde ele se encontra.

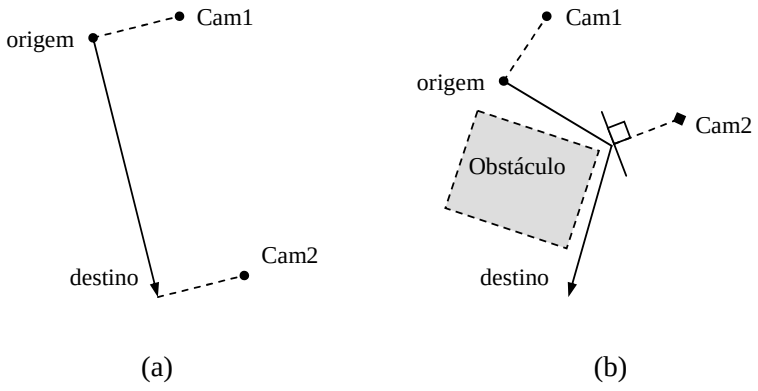

Figura 7.4: Especificação das câmeras para a ação Walk a) sem devio e b) com desvios.

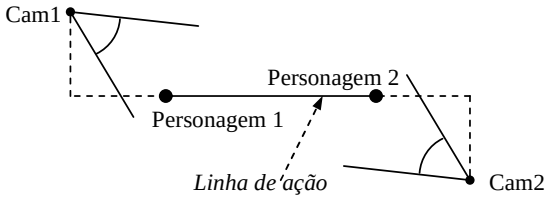

Figura 7.5: Especificação das câmeras para a ação Fight.

7.5.2 Escolha da Câmera Ativa

A cada frame, o módulo da câmera deve selecionar, entre o elenco de câmeras disponíveis, a mais adequada. Faz-se uso de funções heurísticas simplificadas para fazer a escolha das câmeras. Existem três situações claras onde pode ocorrer a troca de câmeras: quando há mudança do personagem-foco, quando há mudança da ação corrente e quando o personagem se desloca no cenário. O caso mais simples de ser tratado ocorre quando há mudança de personagem-foco, visto que a câmera deve rapidamente capturar a nova cena. Nos outros dois casos, a câmera tem maior liberdade na seleção da tomada. A câmera procura, sempre que possível, amenizar cortes bruscos quando novas situações se fazem presentes, segundo regras de cinematografia [69]. A principal regra implementada diz respeito à contextualização da nova cena, ou seja, como situar o observador do novo evento a ser apresentado.

A função heurística é selecionada em função da ação atual que está sendo desempenhada. São usadas várias regras para a seleção. Estas regras levam em conta também o peso associado às câmeras quando são criadas:

– Se o personagem mudar, a câmera deve mudar imediatamente, mesmo que haja um corte brusco e que a nova cena não seja contextualizada adequadamente;

- Se não existe um evento a ser processado, a câmera deve permanecer exibindo a última cena, caso exista;
- Em uma situação de deslocamento, caso o personagem esteja muito afastado da câmera corrente, ou se não esteja mais visível devido a oclusões de objetos do cenário, a câmera associada ao destino final/intermediário do caminho do personagem pode tornar-se ativa;
- Caso duas câmeras tenham pesos semelhantes, a preferência é sempre da câmera corrente, de modo a evitar cortes desnecessários;
- Em caso de mudança somente de ação, ocorre uma redução gradual do peso da câmera corrente com o passar do tempo, de modo que após a contextualização da nova ação pela câmera corrente, câmeras mais adequadas possam ser usadas.

7.6 Ciclo de Renderização das Cenas

O processo de visualização das cenas segue o mesmo paradigma de outras aplicações gráficas em tempo real, como no caso dos jogos. A cada laço de renderização, toda cena é processada pelo motor gráfico e, na seqüência, enviada à placa gráfica para ser desenhada na tela. O processamento do motor é dividido em várias etapas, como mostrado no seguinte algoritmo:

1. Para cada personagem:
   - Verificar término da ação corrente. Em caso afirmativo, verificar se o término da ação implica no término do evento corrente. Se isso ocorrer, solicitar ao **Gerenciador de Ações** um novo evento a ser processado. Na seqüência, delegar uma nova ação ao personagem em função do evento corrente. Se o evento corrente for nulo, associar ação default (`Stand`);
   - Invocar a FSM (*Finite State Machine*) do personagem para processar um passo de sua ação corrente. Neste processo, ações intermediárias podem ser geradas, bem como finalizadas.
2. Configurar a câmera em função da ação em execução pelo personagem vinculado ao evento corrente:
   - Se houve alteração de personagem ou ação corrente, criar duas novas câmeras;

- Chamar a função heurística para selecionar a câmera mais adequada para a situação corrente.

3. Renderizar cada elemento da cena, ou seja, objetos, terreno e céu.

4. Para cada personagem:

   - Renderizar seu modelo segundo estado atual de animação do modelo 3D.

O mapeamento dos eventos para as ações é sempre de um para um, ou seja, cada evento é convertido em uma ação terminal, que deve ser colocada na base da pilha de ações do personagem (Seção 5.2).

O corpo da ação deve incluir obrigatoriamente o seu tipo e o identificador do evento que lhe deu origem (`eventID`). Toda vez que um personagem requisita uma nova ação ao Gerenciador de Ações, esse verifica qual foi o `eventID` associado à última ação do personagem. Caso o `eventID` seja igual ao do evento corrente, significa que o evento corrente foi finalizado e que o Gerenciador de Ações deve requisitar ao Gerenciador de Enredos um novo evento. O corpo da ação também pode conter, dependendo da ação em execução, localizações estáticas (objetos), dinâmicas (referências a outros personagens - para ações coletivas) e também tempo de duração. Este mesmo elenco de parâmetros é usado em ações temporárias criadas pelos próprios personagens.

O tempo de duração de algumas ações é pré-definido (aleatório ou fixo). Por exemplo, a ação `Stand` pode assumir valores quaisquer. Já, para a ação `Fight`, o tempo é sempre fixo. Entretanto, para ações baseadas em espaço, como no caso de `Walk`, esta regra não pode ser aplicada. O tempo de duração vai depender da distância a ser percorrida e da velocidade com que o personagem se locomove. Ações de `Walk` são finalizadas quando o personagem atinge o destino. Ações `Attack` finalizam quando o atacante derrota um número estipulado de guardiões.

7.6.1 Controle temporal

A câmera virtual deve sempre buscar eventos que sejam mais importantes a cada momento, caso contrário, o usuário pode perder o interesse pela cena que está sendo apresentada. Para que esta regra se aplique, devem existir obviamente diversas ações ocorrendo em paralelo. No presente trabalho, existe no máximo uma única ação principal ocorrendo,

visto que cada evento da história é processado sequencialmente. Os personagens figurantes, quando não estão interagindo com um principal, executam somente ações de patrulha, que consistem em se locomover de um lugar para outro no local a ser guarnecido.

Cenas onde o personagem principal deve se deslocar até um determinado local para realizar uma ação específica podem se tornar monótonas caso durem por um longo período de tempo. Neste aspecto, uma questão que foi levantada ao longo desta tese, era se todas as cenas devem ser continuamente processadas ou se partes, menos interessantes, podem ser suprimidas da representação gráfica. Neste caso, a ação tem início em um local, ocorre um corte e na seqüência, a ação volta a ser exibida no local de sua finalização. Para tratar cortes de ações, mudanças significativas devem ser incorporadas principalmente à estrutura dos personagens. O maior problema refere-se à estimativa de onde o personagem deve ser reposicionado após o corte, para dar continuidade à ação. Para a câmera, basta seguir o personagem em sua nova posição. Entretanto, cortes bruscos podem tornar confuso o entendimento pelo usuário da ação que voltou a ser exibida, agora em outro local e sob outros parâmetros de visualização.

Devido a estes problemas, o presente trabalho considera a execução de cada ação de maneira contínua. Para contornar possíveis problemas desta abordagem, define-se um cenário com tamanho adequado para que ações, mesmo envolvendo o percorrimento de longos trajetos, sejam completadas em um tempo relativamente pequeno.

Já pensando na incorporação futura de drives e emoções ao modelo, bem como em interações sociais com outros personagens, a simulação contínua de cada cena permite avaliar, com mais rigor, a variação destes atributos. Entretanto, estas questões de simulações contínuas vs comprimidas devem ser alvo de trabalhos futuros, onde a representação e a manipulação da variável tempo devem ser rigorosamente pesquisadas.

7.7 Interface com o Usuário

A idéia geral do sistema é prover meios eficientes ao usuário, não apenas para guiar a ordem dos eventos a serem visualizados, mas também, e principalmente, para explorar alternativas coerentes que um dado contexto de histórias permite.

Toda a interação é realizada com o Gerenciador de Enredos, que por sua vez, interage com o IPG, para controlar a geração da história, bem como

com o motor gráfico, para controlar a visualização.

A intervenção do usuário pode ser forte ou fraca. Em uma intervenção fraca, o direcionamento da história fica mais a cargo do IPG, pois o usuário se limita a escolher alternativas de enredos parcialmente gerados pelo IPG, que sob sua visão parecem ser interessantes. Em uma intervenção forte, o usuário pode especificar a ocorrência de eventos e forçar a ocorrência de situações em determinados instantes, o que torna a história mais personalizada.

Nos dois tipos de intervenção, o usuário não tem controle direto sobre a cena, nem sobre os personagens. Além disso, a intervenção do usuário é sempre indireta, no sentido que qualquer intervenção deve ser validada pelo IPG antes de ser incorporada ao plano corrente.

7.7.1 A interface do Gerenciador de Enredos

O Gerenciador de Enredos engloba toda a interface gráfica de interação, por meio da qual o usuário pode participar na escolha dos eventos que fazem parte da história, bem como na ordem que devem ser visualizados (Figura 7.6). Em termos gerais, o papel do Gerenciador de Enredos é também servir como um canal de comunicação e conversão de dados entre o IPG e o Visualizador de Histórias.

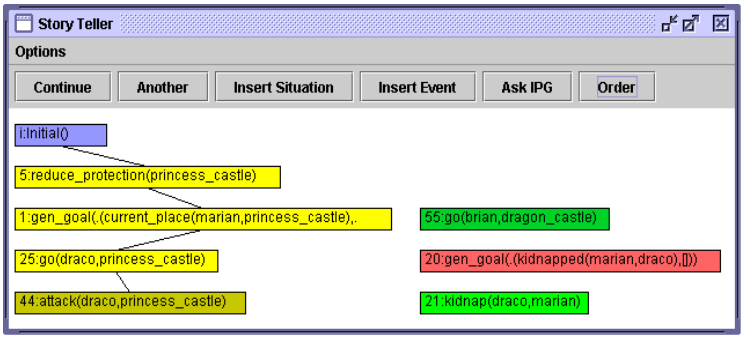

Figura 7.6: Interface de interação com o usuário onde o número à esquerda de cada evento representa o tempo do evento.

Tanto eventos como objetivos são representados por caixas retangulares (ícones) que assumem uma cor específica em função do estado corrente. Esta cor é usada para guiar o usuário no encadeamento de eventos da história. Os ícones podem assumir 7 cores distintas, conforme a Tabela 7.5.

Tabela 7.5: Esquema de cores utilizado para guiar o usuário na manipulação dos eventos

Cor	Significado
Azul	Representa apenas o ponto de partida para encadeamento dos demais eventos.
Amarelo Claro	Eventos que já estão com a ordenação total concluída. São os únicos que podem ser visualizados graficamente.
Amarelo Escuro	Evento corrente de onde o usuário pode conectar outros eventos ativos.
Verde	Representa eventos ativos que podem ser conectadas ao evento corrente, ou seja, eventos que não violam restrições temporais.
Vermelho	Eventos inativos, que não podem ser conectadas ao evento corrente pois têm restrições temporais que devem ser supridas anteriormente (oposto da cor verde).
Ciano	Evento que está sendo representado pelo motor gráfico. Quando a renderização do evento finaliza, a cor do ícone retorna ao estado anterior à renderização.
Verde escuro	Eventos inseridos pelo usuário e que ainda não foram certificados pelo IPG.

A interface de ordenação de eventos define a estrutura mais fundamental por onde o usuário interage com o sistema e é uma etapa que deve ser realizada obrigatoriamente antes de iniciar a visualização da história. Os ícones são manipuláveis no sentido de que podem ser ligados uns aos outros (definição da ordem total) para formarem encadeamentos lineares de eventos. Este encadeamento linear define o direcionamento da história. Somente pode haver a criação de um novo elo entre o evento corrente e um ativo.

Caso o usuário tente estabelecer um elo entre o evento corrente e um inativo (cor vermelha), uma mensagem de erro indicando todas as restrições temporais violadas é exibida.

7.7.2 Interação com o IPG

Quando o usuário deseja a geração de histórias sem que seja exigida interação constante e efetiva, ele pode deixar este trabalho a cargo do IPG. O

IPG disponibiliza dois meios para geração de histórias de forma automática, por meio dos comandos *Continue* e *Another* (Figura 7.6).

O comando *Continue* faz com que o IPG inicialmente infira um novo objetivo e na seqüência, por meio de algoritmos de planejamento, encontre um conjunto de eventos que levem à satisfação deste objetivo. Estes eventos são enviados ao Gerenciador de Enredos para aprovação do usuário. Caso o usuário esteja satisfeito com a solução encontrada para os objetivos inferidos, ele pode solicitar ao IPG a continuação da história (inferência de novos objetivos e subseqüente planejamento), ou definir a ordem total e requisitar a visualização da história até então gerada. Caso o usuário não esteja satisfeito com a história, pode requisitar ao IPG a geração de outra alternativa, pelo comando *Another*. Neste caso, o IPG tenta descobrir outra seqüência de eventos que levem ao cumprimento dos objetivos inferidos. O usuário pode alternar entre as operações *Continue* e *Another*, porém deve-se observar que o *Another* tem efeito somente sobre a última operação *Continue* realizada. Eventos gerados pelo IPG não podem ser removidos pelo usuário.

A todo momento, tanto o IPG, como o Gerenciador de Enredos, guardam a configuração corrente dos eventos e das ordens parciais. Quando a ação *Continue* é executada, caso o usuário não tenha feito nenhuma alteração de ordem total ou inclusão de novos eventos, o IPG é chamado sem nenhum argumento. Neste caso, um novo objetivo é inferido e a história tem prosseguimento. Caso o usuário altere somente a ordem parcial dos eventos, as novas restrições são passadas ao IPG por meio de uma lista. Estas novas restrições são levadas em conta pelo IPG, um novo objetivo é inferido e a história tem sua continuação.

Quando todos os objetivos estiverem satisfeitos, o IPG não pode mais ser usado de forma automática para gerar novos eventos, visto que, sob sua visão, a história já está finalizada. Entretanto, formas mais manuais de interação ainda podem ser usadas. A partir deste momento, o IPG age apenas como uma ferramenta de apoio lógico a autoria.

O IPG suporta dois meios adicionais para intervenção mais efetiva do usuário, de modo a permitir a geração de histórias mais personalizadas. Primeiramente, o comando *Insert Situation* permite que o usuário especifique situações que devem ocorrer em momentos específicos da história, pela inclusão de algum objetivo adicional a ser alcançado. Os detalhes específicos de como o objetivo é alcançado são deixados a cargo do IPG, que é o responsável por encontrar uma solução, se alguma existir, usando algoritmos de planejamento. Deve-se observar que, em termos de

desempenho, um plano válido computável pode não ser alcançado se os limites de busca atualmente configurados no IPG forem excedidos. Os objetivos que o usuário define têm o mesmo efeito que objetivos presentes na estrutura da história criada pelo autor.

No nível mais baixo de interação, por meio do comando *Insert Event*, o usuário pode explicitamente inserir eventos no enredo e adicionar restrições de ordem relativas a eles. Para tornar os novos eventos válidos, o usuário deve chamar o IPG pelo comando *Continue*. Neste momento, todas os eventos definidos pelo usuário, bem como possíveis ordens temporais, são submetidos ao IPG, que executa algoritmos de planejamento para verificar se eles são consistentes com o enredo corrente. Caso não sejam, o IPG pode inserir ainda outros eventos e restrições temporais de modo a garantir a consistência. Neste tipo de interação, o IPG também não faz nenhuma inferência de objetivos. Apenas certifica a validade dos eventos inseridos e as possíveis restrições de ordem também inseridas. Desta forma, o usuário pode criar histórias totalmente desvinculadas dos objetivos presentes na base de dados da história sendo manipulada.

Tanto na inserção de eventos quanto de situações, se o usuário não estiver satisfeito com uma solução apresentada pelo IPG, ele pode solicitar a geração de outras alternativas com o comando *Another*.

Quando um evento ou um objetivo é inserido pelo usuário, um novo ícone é criado dentro da interface. A principal diferença entre ícones definidos pelo usuário e pelo IPG está no fato que os inseridos pelo usuário, em seu estado inicial, não possuem restrição alguma de ordem associada, o que o permite que sejam conectados a qualquer outro ícone. Desta forma, o novo ícone pode ser inserido tanto no início da história, ao final, ou entre dois eventos quaisquer, que inclusive já podem ter uma ordenação total definida. Somente após a certificação do IPG (pela ação *Continue*), é que são definidas as restrições temporais que este novo evento deve assumir. Outra diferença reside no fato que o usuário somente pode remover ícones por ele definido, desde que ainda não incorporados (ou que foram rejeitados) pelo IPG.

Para tornar o processo de definição de situações e eventos mais transparente e fácil, o módulo de Interface disponibiliza dois diálogos por onde o usuário pode visualmente e iterativamente definí-los (Figuras 7.7 e 7.8). Eles são compostos por comboboxes que definem grande parte dos parâmetros que o usuário pode manipular. Assim que o usuário seleciona o primeiro combo, os demais são ajustados para suprir os parâmetros que o evento possui. Muitos objetivos possuem parâmetros numéricos que

devem ser preenchidos pelo usuário. Entretanto, a interface representa tais parâmetros assinalados com o símbolo "XX", como mostrado na Figura 7.8. As situações inseridas pelo usuário podem ser compostas de vários objetivos. Para isso, a interface possui o comando *Add*.

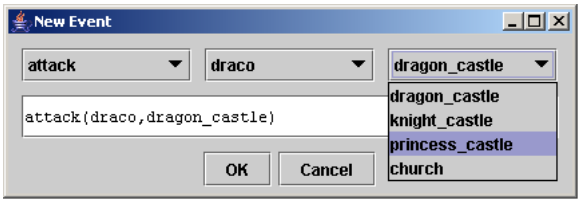

Figura 7.7: Interface para inserção de eventos.

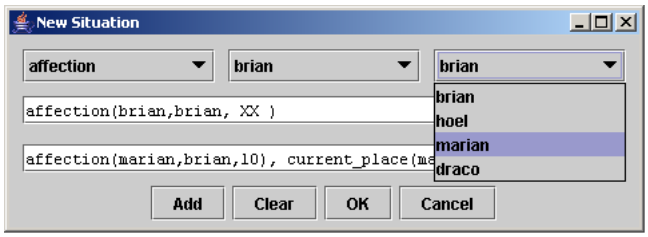

Figura 7.8: Interface para inserção de situações.

A lista de parâmetros nos comboboxes é estática. Não foram implementados ainda mecanismos para consultar o IPG, dado o contexto de histórias corrente, para saber quais são os personagens principais e os possíveis eventos e parâmetros. Estas interfaces permitem que o usuário possa também escrever seus próprios eventos e objetivos, uma vez sabendo a sintaxe utilizada pelo IPG.

7.7.3 Interface de Ícones

Clicando-se com o botão direito do mouse sobre cada ícone, é exibido um menu que disponibiliza as ações que podem ser realizadas. Estas ações incluem remoção (*Remove*), consulta (*Ask*) e renderização (*Render*) do evento. Dependendo do estado do ícone, algumas ações podem estar desabilitadas (Figura 7.9).

Ao executar a operação de consulta (*Ask*), que está sempre habilitada, pode-se consultar o IPG sobre o estado de qualquer elemento da narrativa relativo a um tempo específico T_i, por meio da lógica modal temporal do IPG [34] (Figura 7.10). Este recurso é útil para usuários avançados

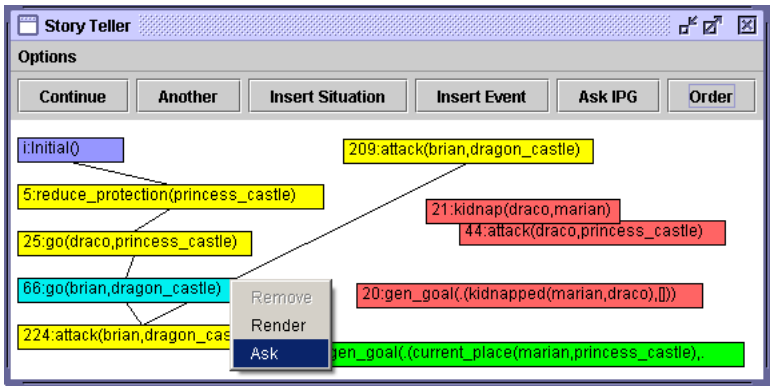

Figura 7.9: Exemplo de menu de ações sobre ícones.

descobrirem, por exemplo, porque um evento ou objetivo não é permitido, bem como para autores que desejam sintonizar os requisitos da história. Esta mesma interface pode ser ativada pelo comando *Ask IPG*, da barra geral de ferramentas. Como todo ícone está associado a um identificador numérico, que é usado como forma de definição das restrições temporais, a consulta realizada sobre os ícones já possui este argumento preenchido. Pode-se consultar por exemplo, o nível de proteção de um local no tempo 66, que para o exemplo da Figura 7.9, corresponde ao instante quando Brian vai para o castelo do Draco. Para isso, a seguinte consulta deve ser feita: `h(t(66), protection(PL,KIND,PROT))`. Neste exemplo, quase todas as variáveis são livres, o que permite, por meio do comando *Another*, desta interface, saber o nível de proteção de todos os locais no tempo 66 (Figura 7.10). A consulta `h(t(T), protection (church,KIND,PROT))` pode ser usada para saber o nível de proteção da igreja em todos os tempos já definidos pelo IPG. Esta interface deve, em trabalhos futuros, apresentar textos em linguagem natural mais amigáveis do que sentenças em sintaxe Prolog.

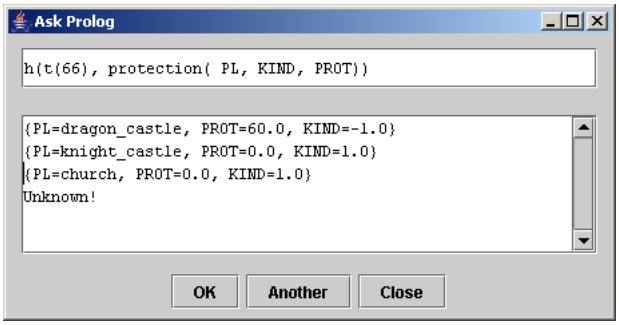

Figura 7.10: Interface para consultar o IPG.

Outra ação disponível no ícone é a renderização. Para que esta opção

esteja disponível, uma série de fatores deve ser verificada. Inicialmente, somente podem ser renderizados ícones que representem eventos. Objetivos não têm este recurso e somente estão presentes na interface como forma de auxílio ao usuário na definição da história. Para que um evento esteja ativo, ele deve ter sido gerado pelo IPG, ou já certificado pelo mesmo, no caso de ter sido definido pelo usuário. A visualização ocorre em tempo real. Atributos variáveis são alterados à medida que os eventos são dramatizados. De modo a manter as representações lógicas e gráficas compatíveis (esquema de sincronização apresentado na Seção 7.4), os valores das variáveis, antes do início da visualização de um evento, devem ser compatíveis com as pré-condições da operação, bem como os valores finais com as pós-condições.

A dramatização tem início pela seleção de um evento específico. Todos os eventos subseqüentes encadeados a partir do ponto inicial são visualizados de modo contínuo, exceto se o usuário interromper o processo. Quando um evento é selecionado para ser visualizado, o motor gráfico usa os valores correntes dos atributos pertinentes como ponto de partida para a representação. Para garantir que a cena se mantenha logicamente compatível com o enredo, não é permitido ao usuário visualizar eventos a partir de pontos aleatórios da história, que não tenham sido previamente renderizados. Um evento somente pode ser renderizado se o seu precedente já foi finalizado, exceto para o evento conectado ao nó raiz. Entretanto, se o usuário deseja visualizar eventos já renderizados anteriormente, pode-se usar qualquer evento como ponto de partida. Para que isso seja possível, quando cada evento tem sua visualização iniciada, o motor gráfico armazena, em uma base local, todos os atributos de todos os personagens e objetos da cena, tendo como chave de acesso o tempo associado a cada evento. Caso o usuário deseje visualizar novamente um evento em ordem aleatória, os dados associados a este evento são usados para atualizar o estado corrente do motor, garantindo assim a mesma representação originalmente realizada.

Como o usuário pode alternar entre simulação e visualização, novos eventos e restrições temporais podem ser inseridos tanto pelo usuário como pelo IPG. Se a visualização for reativada, ela pode iniciar apenas em eventos anteriores às modificações.

7.8 Operações de Videotape (VCR)

Uma vez definida uma ordem total e selecionado o evento inicial a ser renderizado, o motor requisita um novo evento ao Gerenciador de

Enredos assim que o corrente é finalizado. Quando não existem mais eventos conectados ao encadeamento, é enviada uma mensagem ao motor para parar a renderização.

Durante a reprodução dos eventos, o usuário dispõe de algumas operações de videotape (VCR) como *pause*, *play* e *fast forward* (FF). Estas ações têm efeito na velocidade do andamento da visualização gráfica.

A velocidade com que a história é exibida depende tanto da duração de cada evento, como do fator de animação (`FA`), que regula a velocidade com que os personagens realizam as animações, pelo controle da passagem do tempo (com efeito em ações baseadas em tempo, como `Wait` e `Fight`) e velocidade de realização das tarefas (ações baseadas em deslocamentos). A Equação 1 mostra o cálculo para determinação da velocidade com que cada personagem realiza as animações. O fator 1.0/fps assegura que, independente da taxa de quadros por segundo (FPS = *frames per second*) atingida pela aplicação, a velocidade de animação permanece constante.

```
animationSpeed = (1.0/fps)*FA
```
Eq (1)

Para uma operação de pause, o valor de `FA` assume valor zero. Isso faz congelar o estado e posição de cada personagem, que podem ser reativados com a operação *Play*, que restaura o valor de `FA` ao seu valor default. Para fazer com que a animação acelere, basta aumentar o valor de `FA`. A interface por onde o usuário controla a animação é motrada na barra de comandos inferior da Figura 7.11.

Figura 7.11: Interface para controle das operações de videotape.

7.9 Resultados e Utilização do Sistema

Para mostrar os resultados obtidos e exemplificar a utilização do sistema, apresentam-se neste capítulo alguns exemplos de histórias geradas

e cenas dramatizadas (visualizadas). Em especial, deseja-se mostrar as diferentes opções de interatividade que o usuário dispõe e os recursos que a ferramenta disponibiliza para tornar os processos de geração das histórias e visualização intuitivos e com alto poder de expressão.

O nível de diversidade e personalização das histórias geradas tem como principal elemento o nível de interação imposto pelo usuário. Como já apresentado, o sistema permite que o usuário intervenha na geração da história de maneira forte ou fraca, adequando assim as expectativas e interesses de diferentes tipos de usuários.

Para a geração das histórias, foi definida a seguinte situação inicial:

- Marian é uma princesa (no caso uma vítima em potencial), que vive em seu castelo (princess_castle);
- Draco é o único malfeitor (vilão) da história. Ele reside em seu castelo (dragon_castle);
- Brian e Hoel são os cavaleiros (heróis). Ambos residem no mesmo local (knight_castle);
- Cada local da história tem um nível de proteção (número de guardiões);
- Cada personagem tem um nível específico de força para lutar;
- Os heróis têm uma alta afeição pela princesa;
- A princesa não tem nenhuma afeição especial pelos heróis.

Partindo-se dessa situação e usando apenas o comando *Continue*, i.e. com a intervenção mais fraca possível, o processo de geração fica apenas a cargo do IPG, o que leva à geração da seguinte história:

"*A proteção do castelo da Marian é reduzida. Draco vê isso como uma boa oportunidade para raptá-la. Draco então vai para o castelo da Marian, o ataca e após rapta Marian. Como nobre cavaleiro, Brian sente-se obrigado a salvá-la. Ele vai até o castelo do Draco, ataca-o e então luta com Draco. Finalmente, Brian mata Draco e liberta Marian, que neste momento se apaixona por Brian. Motivados pela mútua afeição, Brian e Marian vão para a igreja para se casarem.*"

Esta história é mostrada na Figura 7.12, por meio do **Gerenciador de Enredos**.

Esta mesma história também pode assumir várias alternativas, com baixa interação do usuário, pelo uso dos comandos *Another* e *Continue*

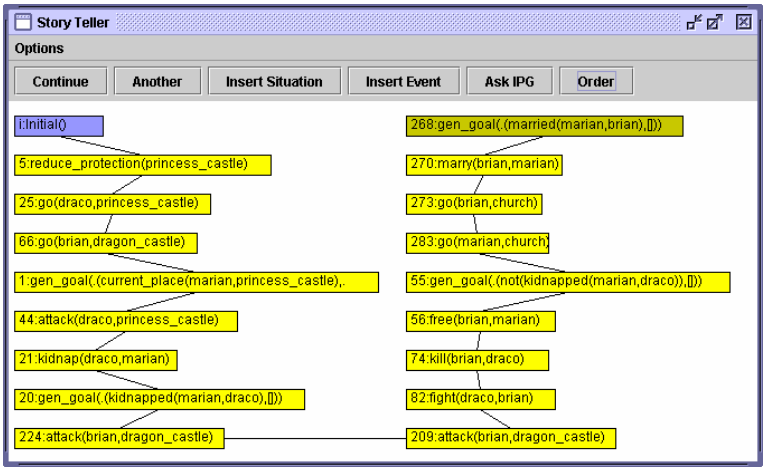

Figura 7.12: História gerada sem interferência do usuário.

combinados. O sistema, por exemplo, gera automaticamente as várias alternativas de fragilização de Marian (tais como ataques do dragão, ida a um local desprotegido) e de salvamento de Marian, onde os dois heróis cooperam de alguma forma. Na figura 7.13 é mostrada uma história onde os dois heróis deslocam-se para o castelo de Draco. Brian ataca o castelo, derrota o dragão e o mata, mas Hoel é quem liberta a princesa e casa-se com ela. A história assume este desfecho pois um dos efeitos do evento `Free` é fazer com que a afeição da vítima pelo seu libertador aumente. Esta história apresenta o exemplo clássico do falso herói.

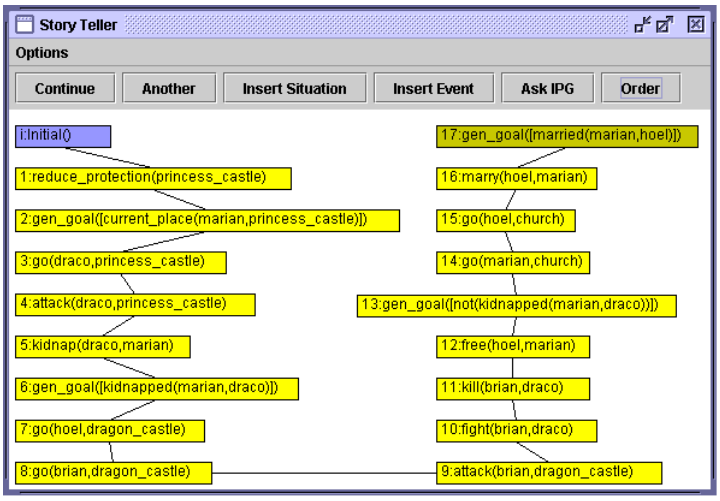

Figura 7.13: Interação pelo uso do comando *Another*.

Caso o usuário deseje histórias mais personalizadas, ele pode intervir no processo com interações fortes, pela inserção de eventos e/ou situações. Podem ser obtidas, dessa forma, histórias completamente personalizadas.

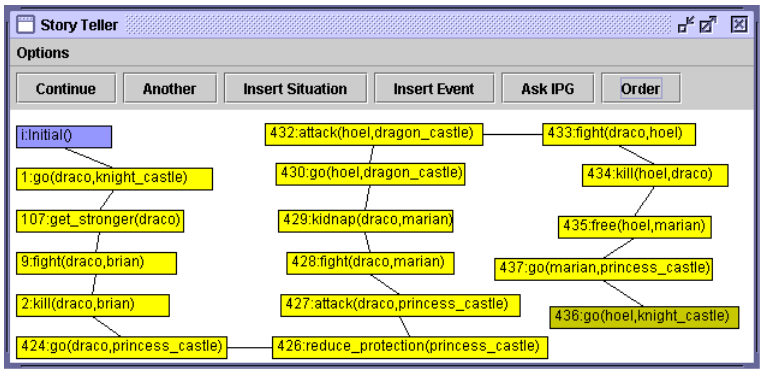

Figura 7.14: Exemplo de história com forte intervenção do usuário.

Na Figura 7.14 é apresentado um exemplo que retrata a seguinte história obtida com interações fortes:

"*Draco deseja raptar a princesa. Sabendo Draco que existe um herói chamado Brian, ele vai até o castelo de Brian, recebe poderes para tornar-se mais forte que Brian e em uma luta, mata Brian. Achando que não existem mais heróis, vai até o castelo da Marian para raptá-la. Chegando lá, tem que enfrentar as defesas do castelo e ao final, ainda sofre um forte ataque da princesa, que luta para não ser raptada. Sendo mais forte que a princesa, Draco consegue raptá-la. Entretanto, um outro herói oculto, Hoel, vai até o castelo do Draco, reduz sua proteção e o vence em uma luta. Após libertar a princesa, Hoel decide não casar-se com ela pois sabe que o verdadeiro amor da Marian era por Brian*".

Se modificada a situação inicial, podem ser geradas inúmeras variações de histórias, todas elas obedecendo à lógica definida no contexto.

Nos exemplos apresentados até este momento, a interação com a história ocorreu somente a nível de geração, pela utilização integrada do IPG com o **Gerenciador de Enredos**. Uma vez que os eventos da história estejam devidamente ordenados, o usuário pode ativar o motor gráfico para que seja realizada a dramatização destes eventos.

Para melhor contextualizar os eventos apresentados ao usuário, é escolhido um cenário que retrata histórias medievais. A estrutura geral do cenário onde os eventos são processados é apresentada na Figura 7.15.

Referente à Figura 7.15(a), o castelo da princesa é o maior e mais próximo, o dos cavaleiros é o mais à direita e o do Draco é o mais ao fundo (cor avermelhada). O usuário pode personalizar este cenário alterando o arquivo de definição do cenário, como mostrado na Seção 7.2. Para a captura destas duas figuras, fez-se uso da câmera em modo manual (controlada

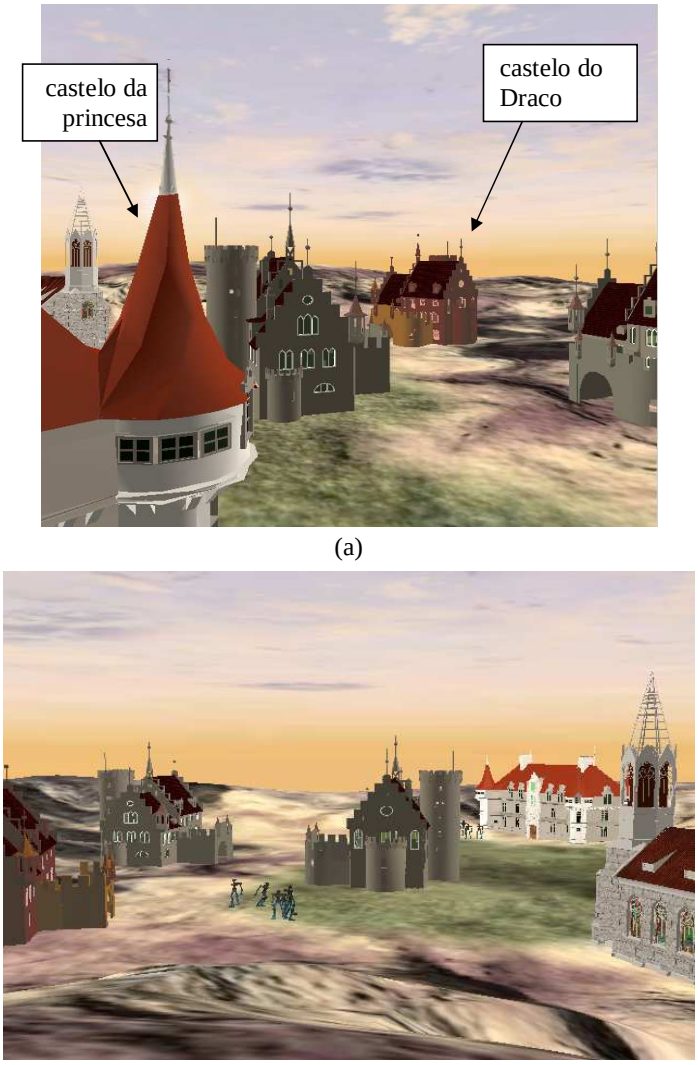

Figura 7.15: Cenário sob os pontos de vista (a) do castelo da princesa e (b) da igreja.

pelo usuário), modo este que, aliás, confere um pouco mais de caráter de videogame à dramatização.

Na Figura 7.16 são apresentados, respectivamente, os ataques de Draco sobre o castelo da princesa e de Hoel sobre o castelo de Draco. Na Figura 7.17 apresenta-se a cena onde Brian vai até a igreja para casar-se com a Marian.

No protótipo, por questões de tempo de modelagem, estão definidas animações de personagens menos elaborados do que os das Figuras 7.16 e 7.17 (com exceção do dragão). Isto, entretanto, não compromete os resultados. No repositório de documentos e aplicações do ICAD/IGames/VisionLab [74] são constantemente introduzidos melhores modelos e animações, bem como atualizações e correções do sistema.

Figura 7.16: Luta do Draco com os soldados do castelo da Marian e de Hoel com os soldados do castelo do Draco.

Figura 7.17: Casamento de Brian e Marian.

7.10 Conclusões e observações

Este capítulo provê detalhes de implementação referentes a diversos aspectos do sistema. No que se refere ao modelo gráfico, sem sombra de dúvidas, a escolha do modelo 3D trouxe benefícios em relação à qualidade do conteúdo exibido ao usuário, como observado nos resultados obtidos. Apesar de exigir maior esforço de programação, o paradigma 3D permite criar configurações de câmera mais variadas. O mesmo vale para a definição do conjunto de ações que os personagens podem desempenhar.

Os mesmos comentários podem ser aplicados à escolha do motor de visualização. Várias dificuldades foram enfrentadas, principalmente

referente a modelos 3D de personagens e objetos. Possivelmente, motores já desenvolvidos poderiam amenizar alguns dos problemas enfrentados relativo a este aspecto. Entretanto, vários outros, inerentes à estrutura de implementação do motor e às limitações no nível de personalização de rotinas e funcionalidades, poderiam ser de solução muito mais difícil. Pela decisão adotada, tem-se agora um sistema configurável e adaptável a outros tipos de histórias, bem como maior disponibilidade de ser exportado para outras plataformas de hardware, como requer a TV interativa.

A estrutura de especificação do cenário e personagens permite que o usuário do sistema possa configurá-los de maneira bastante dinâmica. Pode-se fazer alteração dos modelos 3D, sua localização e quantidade. O usuário pode também mudar o terreno e também inserir novos objetos e personagens.

Quanto à interface gráfica, pode-se observar que oferece ao usuário uma gama grande de alternativas de interação. A interação pode ser fraca ou forte, dependendo do nível de esforço que o usuário deseja empregar, bem como do tipo de história que espera obter. Por meio da interface, pode-se também controlar a visualização gráfica que, para esta implementação, é um processo separado da simulação da história.

O único módulo que não apresenta os resultados esperados é o da câmera. Apesar de incorporar recursos para tratar enquadramento e cortes de tomadas, observa-se que em situações onde ocorre a mudança de personagem, a estratégia implementada não permite que possa ser garantida uma boa contextualização da nova tomada antes que a ação tenha início, visto que a câmera não sabe de antemão quais são as próximas ações a serem realizadas. Isso resulta do fato da câmera não ser pro-ativa, a ponto de seguir as ações e não determinar quando elas devem iniciar. O problema da câmera requer uma revisão de paradigma e uma abordagem possível é discutida no Capítulo 8, na seção de trabalhos futuros.