UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

IMPLEMENTAÇÃO DE UM PERSONAGEM COM ESQUELETO PARA DRAMATIZAÇÃO DE STORYTELLING COM AUXÍLIO À CÂMERA E AO DIRETOR

TRABALHO DE GRADUAÇÃO

Henrique Moraes Ramos

Santa Maria, Rio Grande do Sul, Brasil
2008

IMPLEMENTAÇÃO DE UM PERSONAGEM COM ESQUELETO PARA DRAMATIZAÇÃO DE STORYTELLING COM AUXÍLIO À CAMERA E AO DIRETOR

por

Henrique Moraes Ramos

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Cesar Tadeu Pozzer

Trabalho de Graduação N. 275

Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

IMPLEMENTAÇÃO DE UM PERSONAGEM COM ESQUELETO PARA DRAMATIZAÇÃO DE STORYTELLING COM AUXÍLIO À CAMERA E AO DIRETOR

elaborado por
Henrique Moraes Ramos

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Prof. Dr. Cesar Tadeu Pozzer
(Presidente/Orientador)

Prof. Dr. Benhur de Oliveira Stein

Profa. Dr. Marcos Cordeiro d'Ornellas

Santa Maria, 16 de dezembro de 2008.

“O bravo não é quem não sente medo, mas quem vence esse medo.”

- Nelson Mandela

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me dado a vida, por ter colocado desafios em meu caminho para que eu pudesse superá-los, por ter feito meu destino cruzar com o de pessoas que sempre me ajudaram a crescer e a amadurecer de uma forma ou de outra.

Agradeço aos meus pais Rosalino e Eliana pelo auxílio financeiro e moral, por sempre apoiarem minhas decisões e por transformarem um garotinho no ser humano que hoje sou. À Suelen minha irmã, pelas conversas infindáveis que me fortaleciam e não me deixavam esmorecer frente aos problemas e dificuldades.

Agradeço à minha namorada Gisele pela compreensão, pelas horas que ficou ao meu lado em frente ao computador olhando trechos de códigos somente para não me deixar solitário, pelo carinho e pelo apoio.

Agradeço ao meu orientador Cesar Pozzer pela paciência e pelas boas idéias que me ajudaram a fazer um bom trabalho.

Agradeço aos meus colegas pelas dicas valiosas, pelo conhecimento que juntos construímos e compartilhamos. Ao amigo de todas as horas Jesse que sempre teve uma palavra de compreensão e amizade.

Aos mestres que nos ensinaram e nos instigaram a extrairmos de nós o nosso melhor, o meu sincero agradecimento.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

IMPLEMENTAÇÃO DE UM PERSONAGEM COM ESQUELETO PARA DRAMATIZAÇÃO DE STORYTELLING COM AUXÍLIO À CÂMERA E AO DIRETOR

Autor: Henrique Moraes Ramos
Orientador: Prof. Dr. Cesar Tadeu Pozzer
Local e data da defesa: Santa Maria, 16 de dezembro de 2008.

Este trabalho tem por objetivo o desenvolvimento de um personagem tridimensional com suporte a animações, para ser usado em um ambiente de *storytelling*. Esse personagem deve poder passar informações geométricas ao diretor virtual (simulado por meio de técnicas de Inteligência Artificial), que o auxiliem a encontrar o melhor posicionamento para uma determinada tomada de cena. Este personagem interativo deve possuir a capacidade de encontrar pontos de posicionamento da câmera que permitam fazer tomadas pré-definidas como: visão em primeira pessoa, terceira pessoa, tomadas sobre o ombro, *close* da face. Pode também fornecer o ponto para onde a câmera deve voltar-se: para onde o personagem está se deslocando, para onde está olhando ou para uma parte específica do corpo.

A linguagem de programação utilizada para a implementação será C++ juntamente com a biblioteca gráfica OpenGL. Os resultados deste trabalho poderão auxiliar o desenvolvimento de pesquisas na área de dramatização de histórias dinâmicas, tanto para jogos como para conteúdo de TV digital.

Palavras-chave: Personagem tridimensional, OpenGL, esqueleto, câmera sintética, *storytelling*, Computação Gráfica.

ABSTRACT

Trabalho de Graduação
Computer Science
Universidade Federal de Santa Maria

IMPLEMENTATION OF A CHARACTER WITH BONES FOR STORYTELLING DRAMATIZATION TO SUPPORT CAMERA AND THE DIRECTOR

Author: Henrique Moraes Ramos
Advisor: Prof. Dr. Cesar Tadeu Pozzer
Place and date of Presentation: Santa Maria, December, 16, 2008.

This work aims at the development of a three-dimensional character with support for animations, to be used in an environment of storytelling. This character must be able to pass geometrical information to virtual director (simulated by means of techniques of Artificial Intelligence), that helps him to find the best positioning for a given take within the scene. This interactive character must have the ability to find points to place the camera to make some predefined takes as: vision in the first person, third person, takes of kind over the shoulder, and closes on the face. It can also provide information to target the camera: where the character is moving to, where it is looking at or for a specific part of the body.

The programming language used for the implementation is C++ with the OpenGL graphics library. The results of this study could assist in the development of research in the area of dynamic dramatization of stories, both for games and content for digital TV.

Keywords: three-dimensional character, OpenGL, skeleton, synthetic camera, storytelling, Computer Graphic.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O mercado de entretenimento digital tem movimentado bilhões de dólares por ano no mundo todo. Jogos fazem uso de diversas tecnologias computacionais como Computação Gráfica, Inteligência Artificial, redes, interfaces, etc, e a cada vez mais estão fazendo uso de enredos para contextualizar o ambiente do jogo, para torná-lo mais interessante e interativo. Tanto em nível de *hardware* como em nível de inteligência de *software*, temos hoje em dia a possibilidade de obtermos uma história criada de forma dinâmica, sendo que esta pode ser contada apenas pelo computador ou pode ser criada com interação humana para o caso de um jogo ou de aplicações para a TV digital.

Para atingir esse objetivo faz-se uso de ambientes de *storytelling* (POZZER, 2005) (CAVAZZA, 2002)(MATEAS 1999), que são responsáveis por colocar lógica à narrativa deixando assim o conteúdo mais interessante aos olhos do espectador.

Utilizando a computação gráfica pode-se então dramatizar a narrativa para que a mesma não seja apenas uma informação textual na tela do computador e possa ser apresentada de forma mais agradável ao espectador. Para isso, torna-se necessária uma interface responsável por converter os dados obtidos nos geradores de histórias, como exemplo (CIARLINI, 1999), em imagens que contenham um poder dramático maior. Esta interface pode ser considerada um diretor virtual, responsável por definir a melhor estrutura da apresentação de cenas, em um ambiente 3D.

Este diretor precisará obrigatoriamente estar em contato com os elementos desta dramatização, sendo estes basicamente três: a câmera, o ambiente e o ator. A câmera é responsável pelas tomadas das cenas, posicionando-se corretamente, respeitando técnicas de cinematografia (AZEVEDO, 2007; CHRISTIANSON, 1996) para poder exibir um conteúdo relevante. A Figura 1 apresenta uma dessas técnicas cinematográficas, que dada a interação entre dois personagens (X e Y) a câmera não deve fazer tomadas que ultrapassem uma linha que passas por ambos (linha de interesse), permitindo fazer tomadas externas (a, g) em que ambos os personagens aparecem em cena, mas apenas um deles de frente. Assim como tomadas paralelas ou internas (b, c, e, f) em que apenas um

personagem aparece na cena. Note que as tomadas padrões não devem ultrapassar a linha de interesse.

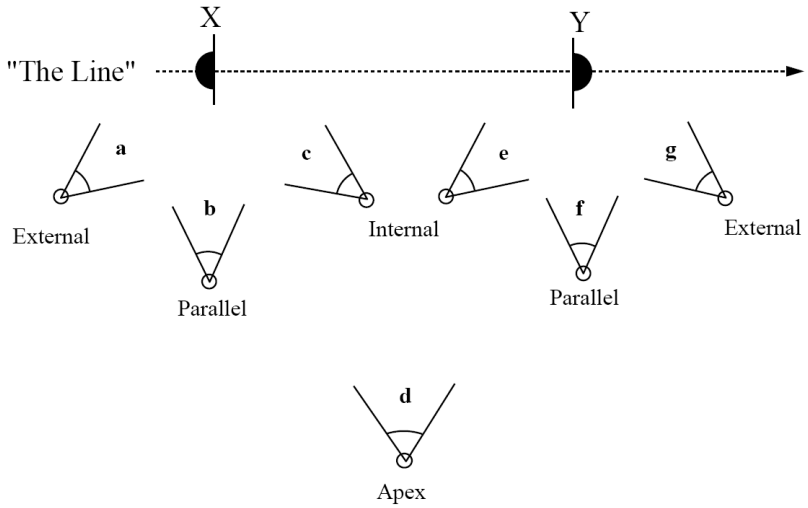

Figura 1 - Técnica cinematográfica para posicionamento da câmera
Fonte: CHRISTIANSON (1996, p. 2).

O ambiente é o local com o qual o diretor deve interagir para fazer as tomadas de câmera e de posicionamento dos atores, isto é, o ambiente é o cenário da dramatização. O diretor deve conhecê-lo para que possa fazer o seu trabalho.

O ator é o responsável pela dramatização da cena e deve obedecer ao diretor e interagir com o ambiente. Cabe aos atores realizar ações de acordo com a história sendo apresentada. Os atores são representados por avatares 3D animados e geralmente são implementados baseados em conceitos de agentes de softwares (SILVA, 2004). Neste caso podem ser vistos como entidades “autônomas” que executam ações em um ambiente 3D, onde podem existir outros agentes com os quais podem ocorrer interações.

Tendo presente este conceito de programas contadores de história e um desenrolar dinâmico de um enredo, seja ele em um jogo, aplicativo comercial ou conteúdo para TV digital, faz-se necessário o uso de personagens que sejam capazes de executar qualquer tipo de animação e integrar-se de forma satisfatória com o sistema diretor. Essas ações genéricas têm o objetivo de enriquecer a história possibilitando ao personagem tridimensional executar qualquer tipo de ação e ter atrelado a ele os parâmetros para melhor visualização possível do personagem e do evento que está sendo representado por meio de recurso gráficos.

Os formatos *Open Source* de personagens disponíveis no mercado não possibilitam auxiliar a câmera a fazer tomadas específicas. O formato MD2

(HENRY'S, 2004), por exemplo, além de possuir uma quantidade limitada de animações disponíveis, não disponibiliza recursos no auxilio ao posicionamento de câmera baseado em pontos como ombro ou cabeça o que dificulta os cálculos para um posicionamento coerente da câmera. Já o formato MD3 (UCLA, 2006), que foi a geração seguinte de formatos de personagens também não dispõe de esqueleto que poderia auxiliar o cálculo de posicionamento da câmera. A sua inovação é referente à separação das animações da cabeça, torso e pernas, possibilitando agora que o personagem, por exemplo, ao mesmo tempo em que corre, consiga atirar. Contudo, ainda não possui pontos de controle da animação durante a execução do programa. Dessa forma o posicionamento da câmera para tomadas específicas é dificultado, pois para isso precisa-se antes descobrir o posicionamento dos pontos de controle dos ossos através de cálculos de aproximação para depois fazer os cálculos para o correto posicionamento da câmera.

Existem vários formatos proprietários que têm auxílio de uma estrutura de esqueleto para executar animações e até mesmo testes de colisão (EUPHORIA), mas mesmos estes formatos mais complexos não possuem um mecanismo de assistência à câmera.

Para resolver esse problema, propõe-se nesse trabalho a criação de um personagem que possua características para auxiliar a câmera no melhor posicionamento possível além de poder executar qualquer tipo de animação previamente preparada que seja passada para ele.

1.1 Objetivos do trabalho

O objetivo deste trabalho de graduação é encontrar uma boa solução de modelagem para este ator, fazendo que além de obedecer ao diretor, possa se comunicar com o mesmo, informando-lhe possíveis pontos de tomadas de cenas, ou seja, indicar ao diretor (e por conseqüência à câmera) qual a melhor posição para colocar a câmera e fazer uma tomada específica: close da face, visão em primeira pessoa, visão sobre o ombro, tomada frontal, etc.

A justificativa desse trabalho se dá pela necessidade de existência de um personagem que possa receber/executar qualquer animação que esteja presente na história, desvinculando a animação do modelo computacional e vinculando-o ao esqueleto que será o mesmo para todos os personagens. Ao se fazer isso

consegue-se dar liberdade para que o personagem consiga executar qualquer animação atribuída à um esqueleto genérico. Além disso, pode auxiliar a câmera a fazer tomadas específicas de si mesmo, já que a câmera não possui conhecimento da geometria dos personagens e não se pode garantir que os personagens tenham sempre a mesma geometria, sendo que os personagens podem ser humanos, dragões, seres alados, etc.

Este trabalho está organizado em cinco capítulos. No segundo capítulo, apresenta-se a revisão bibliográfica sobre *storytelling*, o diretor para a dramatização, animação de personagens 3D e XML (TINY XML). O terceiro capítulo descreve o projeto de implementação deste trabalho, detalhando o personagem, o esqueleto utilizado para concepção de movimentos, detalhes sobre a movimentação da cabeça e o auxílio que o personagem dará ao posicionamento da câmera. No quarto capítulo, são apresentados detalhes da implementação, como as estruturas usadas, os métodos aplicados e os cálculos utilizados para solucionar os problemas encontrados. Por fim, o quinto capítulo apresenta as conclusões acerca do trabalho.

2 REVISÃO DE LITERATURA

2.1 Storytelling

Com o auxílio de técnicas de Inteligência Artificial e Computação Gráfica pretende-se dar ao computador a capacidade de contar uma história coerente utilizando-se de lógicas construtivas que são implementadas nos ambientes de *storytelling*. Seus usos são variados, podendo ser empregados em conjunto com a interação humana, possibilitando assim um nível maior de entretenimento em jogos e desta forma permitir que um jogo possa ser jogado mais de uma vez de formas distintas, resultando em finais/enredos diferentes. Também pode ser aplicado às mídias da televisão digital de forma a gerar um conteúdo específico para o seu público alvo.

De acordo com Pozzer (2005), as pesquisas em *storytelling* seguem basicamente em três ramos distintos:

1) Geração: Responsável por definir os aspectos gerais da história.

2) Interação: Definições de que forma o usuário interferirá no desenrolar da narração.

3) Exibição: Cuida da representação gráfica da história em um cenário virtual 3D.

Os sistemas de *storytelling* atuam sobre um conjunto de dados que definem o escopo da história, formas de narrativas, gênero, enfim, fornecem informações necessárias para a geração automática de uma narrativa delimitada em algum tema.

A geração desse fluxo da história pode ser feita por abordagens diferentes e essas diferenças alteram drasticamente a forma como se desenrola a história. As principais abordagens são:

Abordagem orientada a enredo (*plot-based*): Apresenta uma interação com operações em alto nível. Nessa abordagem o autor define previamente os principais pontos da história como, por exemplo, início, meio e fim. O usuário pode apenas mudar a forma de como a história é contada, não podendo alterar o enredo completamente. Geralmente faz-se uso de uma estrutura de nós, como um grafo, onde cada nó representa um conteúdo da história, podendo o usuário percorrê-lo de várias formas, contando-a de diferentes maneiras (MATEAS, 2002) (POZZER,

2005). A Figura 2 exibe um ambiente de s*torytelling* orientado a enredo, cada retângulo representa um evento da história e o encadeamento de eventos define o fluxo da história.

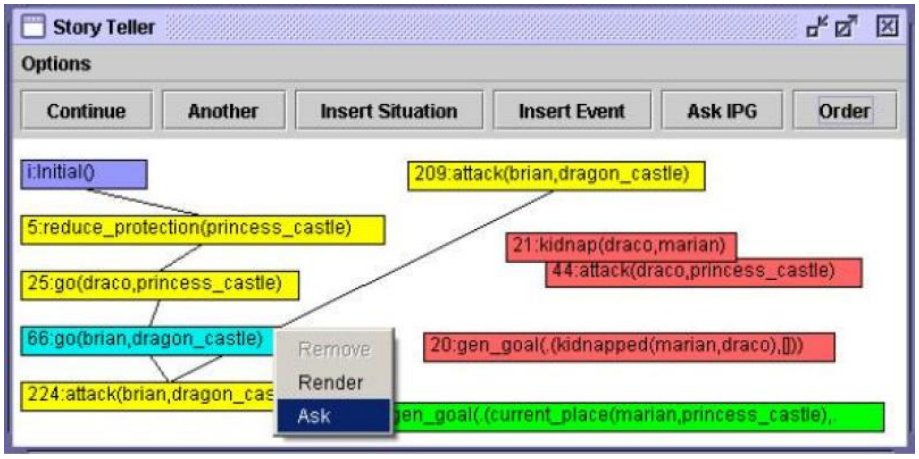

Figura 2 - Interface do gerenciador de enredos Logtell
Fonte: POZZER(2005, p. 116).

Abordagem orientada a personagens (*character-based*): Apresenta uma interação em mais baixo nível, permitindo alterações em atributos que podem mudar drasticamente o enredo da história. Envolve a interação entre os personagens (agentes) e o usuário. É também chamada de “narrativa emergente”. Essa abordagem permite uma liberdade muito maior por parte do espectador, o que faz aumentar a interatividade. Esta característica também pode ser negativa, pois dessa forma, o enredo pode convergir para situações impossíveis ou que não sejam logicamente coerentes vindo a frustrar o usuário (MATEAS, 2002)(POZZER, 2005). A Figura 3 apresenta cenas de interação com um casal em um ambiente *storytelling* orientado a personagens.

Figura 3 - Ambiente storytelling orientado a personagens
Fonte: MATEAS(2002, p. 9).

Existem vários programas contadores de história, entre os quais vale ressaltar o Logtell, que é um sistema de geração, interação e visualização de histórias interativas proposto por Pozzer (2005). Este sistema é orientado a enredo (*plot-based*), isto é, a história segue um rumo previamente definido. Pode-se determinar objetivos a serem executados pelos personagens, sendo que entre um objetivo e outro várias ações podem ocorrer para que o personagem consiga executá-los de forma coerente. Na Figura 2, pode-se observar a forma como o enredo é definido através de ações e objetivos no Logtell.

Existem três grandes módulos que compõem o sistema Logtell: O IPG (*Interactive Plot Generator*), a Interface do Gerenciador de Enredos e o Módulo de Visualização Gráfica, que são implementados em três linguagens distintas: Prolog, Java e C++, respectivamente. A Figura 4 apresenta um esquema representativo da interação dos módulos do Logtell. O IPG é o responsável pela geração da história. Faz uso de lógica para garantir a coerência das histórias. O usuário guia o rumo da história pela interface do gerenciador de enredos e ativa o módulo gráfico para fazer a dramatização da história até então definida. Na Figura 5 é ressaltado o módulo gráfico do Logtell.

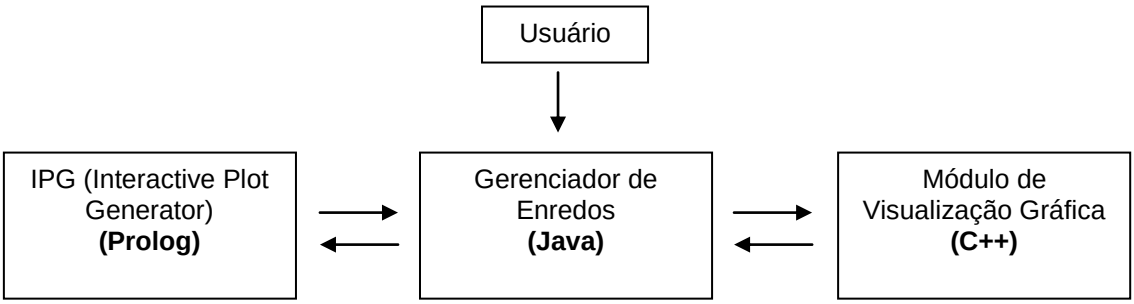

Figura 4 - Interação entre os módulos do Logtell
Fonte: POZZER (2005, p. 17).

No estágio atual do desenvolvimento desse tipo de dramatização, a câmera consegue fazer tomadas distantes, pois conhece a posição dos personagens, mas não possui informações mais detalhadas do mesmo para poder se posicionar de forma a fazer tomadas específicas do personagem, como tomadas de close da face, tomadas sobre o ombro, etc.

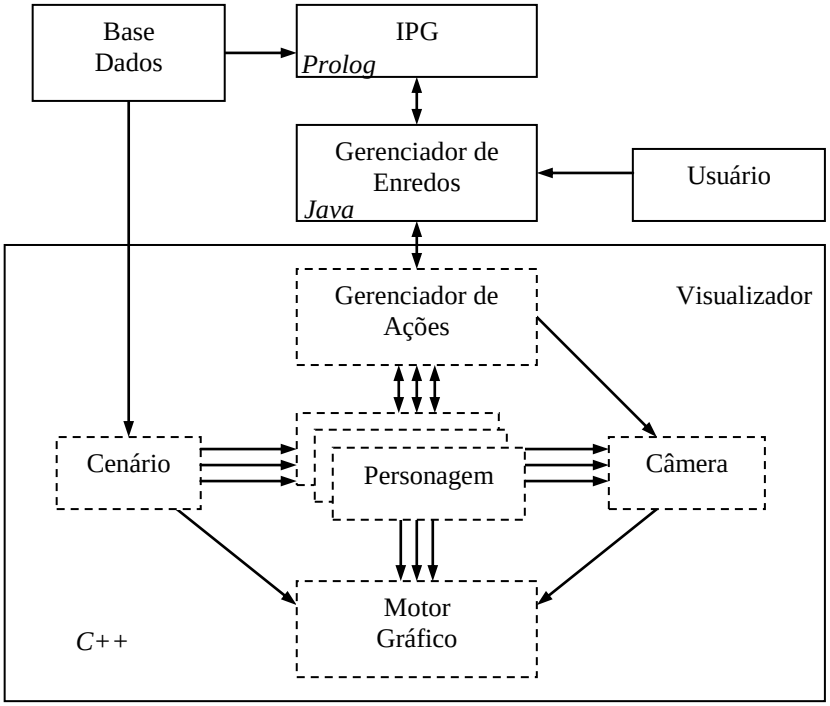

Figura 5 - Apresentação detalhada do modo de visualização do Logtell.
Fonte: POZZER (2005, p. 89).

2.2 Diretor para Dramatização de *Storytelling*

Atualmente está sendo desenvolvido um sistema diretor para dramatização de *Storytelling* (AZEVEDO, 2007, 2008). Este sistema diretor é a motivação do trabalho atual, pois o mesmo necessita de um personagem com as características de auxílio à câmera e direção. A idéia geral é transformar o módulo de câmera sintética presente no sistema Logtell de um papel passivo, em um papel ativo (diretor) na definição das ações a serem realizadas.

O sistema diretor proposto baseia-se em técnicas cinematográficas (AZEVEDO, 2007)(HE, 1996), utilizando-se de uma câmera que desempenha o papel de um diretor em um filme, tendo a capacidade de definir parâmetros que ressaltam os fatos que estão ocorrendo, além de possuir autonomia para definição do início e duração das ações, posicionamento e movimentação.

2.3 Animação de Personagens 3D

A animação de personagens tridimensionais é uma área alvo de muitas pesquisas, sempre focadas na melhor forma de gerar animações realistas e que imitem o movimento do corpo humano.

Os melhores resultados foram encontrados ao se desenvolver estruturas de esqueletos que sofrem transformações geométricas simulando o movimento do corpo humano. Essas transformações são aplicadas à malha (a forma do corpo) que envolve o esqueleto e juntamente com a técnica de manipulação temporal de quadros-chaves formam um bom editor e manipulador de animações 3D que facilitam a criação e a visualização das ações que serão realizadas pelos personagens (LASSETER, 1994).

Outra forma bastante realista para animação de personagens 3D é a utilização de captura de movimento. A captura de movimento se baseia na idéia de "gravar" a movimentação do corpo humano e aplicá-la a um personagem tridimensional que possua um esqueleto (já que as transformações são informações aplicadas à esse esqueleto). A captura pode ser feita através de câmeras ou de sensores colocados sobre o corpo de uma pessoa. As informações obtidas podem variar desde simples posição do corpo no espaço até movimentos mais detalhados dos músculos e da face (STURMAN, 1994).

Ao se trabalhar com animação de personagens tridimensionais é preciso ter idéia da importância da expressão corporal para a dramatização de uma história, pois através da movimentação de um personagem consegue-se inferir vários atributos psicológicos ao mesmo, o que faz agregar expressividade à cena. Para tanto o personagem deve ser projetado de modo a suprir essa necessidade.

2.3.1 Animação Baseada em Keyframes

A técnica de *keyframes* baseia-se na utilização de quadros-chaves, isto é, um conjunto de informações importantes relativas ao objeto que se está animando (MALVEZZE, 2004). A Figura 6 mostra seis quadros-chaves, onde é representada a a animação de uma lâmpada que salta. Fica fácil perceber esse movimento visualmente. Computacionamente a implementação é simples, supondo que basta

fazer uma interpolação da posição dos três vértices do modelo que compõe os quadros-chaves, juntamente com as rotações da base do mesmo.

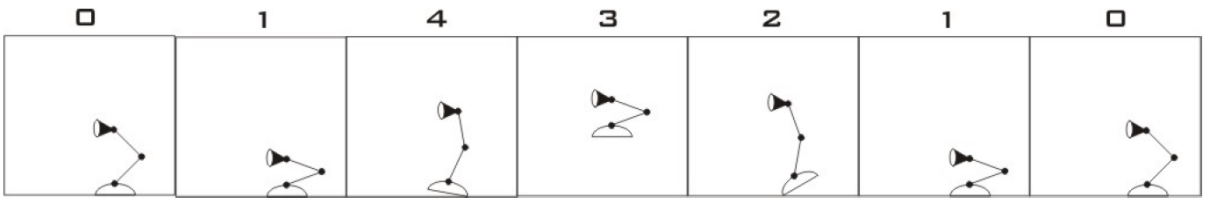

Figura 6 - Exemplo de keyframes
Fonte: MALVEZZE (2004, p. 46).

A aplicação da técnica de quadros-chaves se sai muito melhor ao se ter o controle da quantidade de quadros que o computador consegue mostrar na tela em um determinado intervalo de tempo. Esse parâmetro chamado de FPS (*frames* por segundo) é uma medida de velocidade de renderização. Ele define a quantidade de frames (quadros) que o computador consegue mostrar na tela em um segundo.

O FPS varia de computador para computador, pois sofre variação desde a velocidade do processador até a quantidade de processos que podem estar executando ao mesmo tempo. Por isso deve ser calculado a cada renderização da imagem na tela. Seu cálculo deve ser baseado no tempo que o computador leva para renderizar uma cena, ao calcularmos: $\frac{1}{tempo\ de\ renderização\ de\ uma\ cena}$ obtemos a quantidade de frames que o computador consegue exibir em um segundo (FPS).

2.4 Formato XML

O XML é um formato utilizado para organização de informações que adota a estruturação dos dados em forma de árvore. É um formato eficaz para a manipulação de um conjunto relativamente pequeno de elementos, não substituindo um banco de dados, porém sendo muito útil para organizar pequenas porções de informações (WIKIPEDIA, 2007).

Para ser utilizado com a linguagem C++ é aconselhável o uso de alguma biblioteca para a integração como a Tiny XML(TINY XML). Essa biblioteca de código aberto oferece uma gama de instruções de fácil compreensão e facilita muito a utilização de arquivos XML com a linguagem C++ (só possui suporte ao C++ por ser orientada à objetos).

Esta biblioteca oferece acesso aos nós através de *parsers* que apontam para o arquivo, nós e sub-nós do documento XML, sendo assim possível recuperar dados do XML simplesmente passando-lhe o nome do atributo do qual se deseja obter o valor.

Neste trabalho o XML é utilizado para importar os dados referentes à malha do personagem e os dados que compõem um ciclo de animação do personagem.

3 MODELAGEM E PROJETO DO PERSONAGEM

Este capítulo aborda os métodos de modelagem e de implementação da arquitetura dos personagens 3D propostos neste trabalho, bem como alguns detalhes de sua concepção.

3.1 Personagem

O modelo proposto de personagem possui sua implementação dividida em basicamente duas partes: cabeça e corpo. Estas partes podem atuar de forma dependente ou independente entre si, recebendo ações do diretor e respondendo a ele. A Figura 7 representa a interação entre os módulos que fazem parte do sistema de dramatização de *storytelling*.

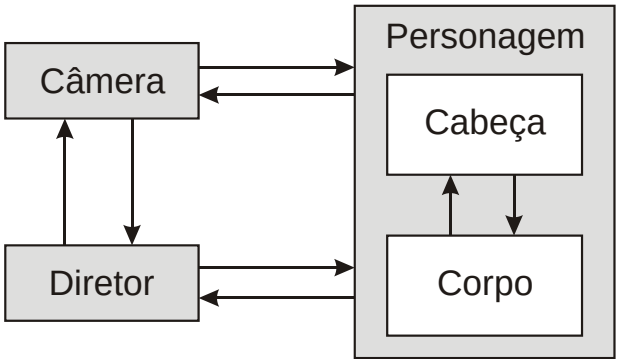

Figura 7 - Esquema representativo da interação entre os módulos.

Note que o diretor passa informações de posicionamento à câmera para uma tomada em que o ator não seja o ponto principal. A câmera por sua vez pode informar ao diretor seus parâmetros atuais, assim como informar ao personagem qual tomada deseja fazer. O personagem pode informar para a câmera o posicionamento pedido. O diretor pode dar ordens diretas ao personagem (ordem de posicionamento na cena) e o personagem pode informar ao diretor o seu posicionamento nesta.

A parte do corpo do personagem compreende o esqueleto utilizado para a sua movimentação e a malha que dá forma ao personagem. Este, além do corpo e da cabeça, possui informações de posicionamento e de controle de animação (detalhes que serão vistos nas seções seguintes).

Tanto o diretor quanto a câmera podem enviar dados ao personagem, posicionando-o no mundo, indicando-lhe a animação que deve executar ou informando-lhe qual a tomada de câmera pretende fazer. O diretor e a câmera também podem receber dados do personagem, como o posicionamento para a câmera (onde está e para onde deve filmar) e a posição do personagem no mundo.

Este trabalho concentra-se na manipulação do corpo do personagem e dos *waypoints* (pontos de controle que definem o posicionamento da câmera) para tomadas específicas, dando auxílio para a integração das expressões faciais, tais como a posição e a direção da face do personagem (SANGOI, 2008).

Os *waypoints* que definirão o posicionamento da câmera são calculados dinamicamente apenas quando solicitados, isto é, somente quando uma tomada específica for exigida. Estes cálculos são feitos levando em consideração os pontos do esqueleto e poderão auxiliar o posicionamento da câmera para as tomadas necessárias.

3.2 Esqueleto

O esqueleto, por ser responsável pela movimentação do personagem torna-se a estrutura mais importante para a implementação desse projeto. Para tanto será utilizada a idéia de *bones* que são na verdade hierarquias de objetos para o posicionamento dos vértices que representam os pontos de controle do esqueleto (GOMES e VELHO, 1998). Esta estrutura é amplamente utilizada em vários editores 3D.

A estrutura de dados possui o formato de árvore, em que a raiz é o ponto que representa o centro do personagem, isto é, o personagem é desenhado a partir do seu centro (âmago) em direção às extremidades (pés, mãos e cabeça). Essa abordagem foi adotada por ser a mais natural para o movimento do corpo humano, já que o movimento do ombro (nó pai) afeta o movimento da mão (nó filho), mas o contrário não acontece. A exposição dessa estrutura pode ser vista na Figura 8, que mostra a árvore que define o esqueleto. Vale ressaltar que o nome do *bone* é dado ao nó que está em sua extremidade, ou seja, para rotacionar um determinado *bone*, precisa-se movimentar o ponto que corresponde à extremidade do mesmo.

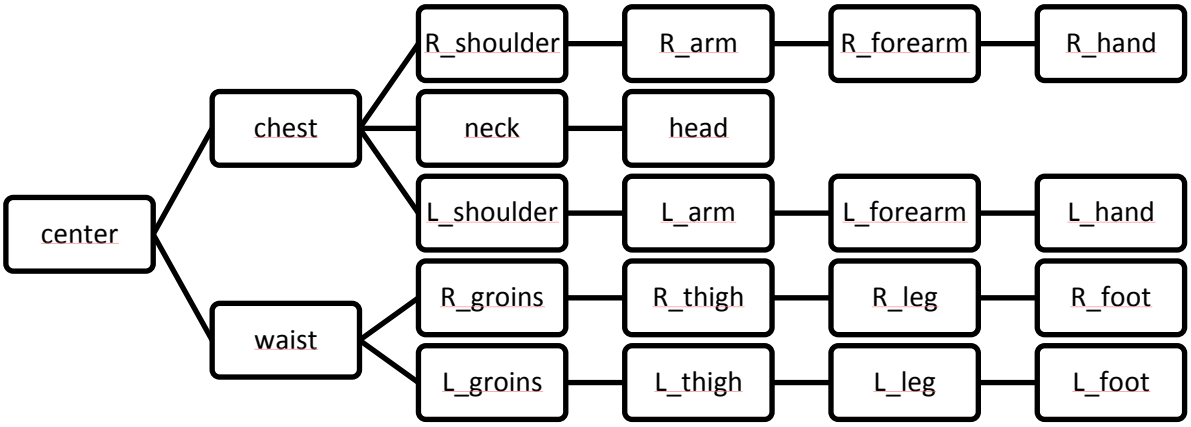

Figura 8 - Estrutura de dados em forma de árvore que define os bones do esqueleto proposto.

A Figura 9, por sua vez, exibe a mesma estrutura posicionando os *bones* no corpo do personagem.

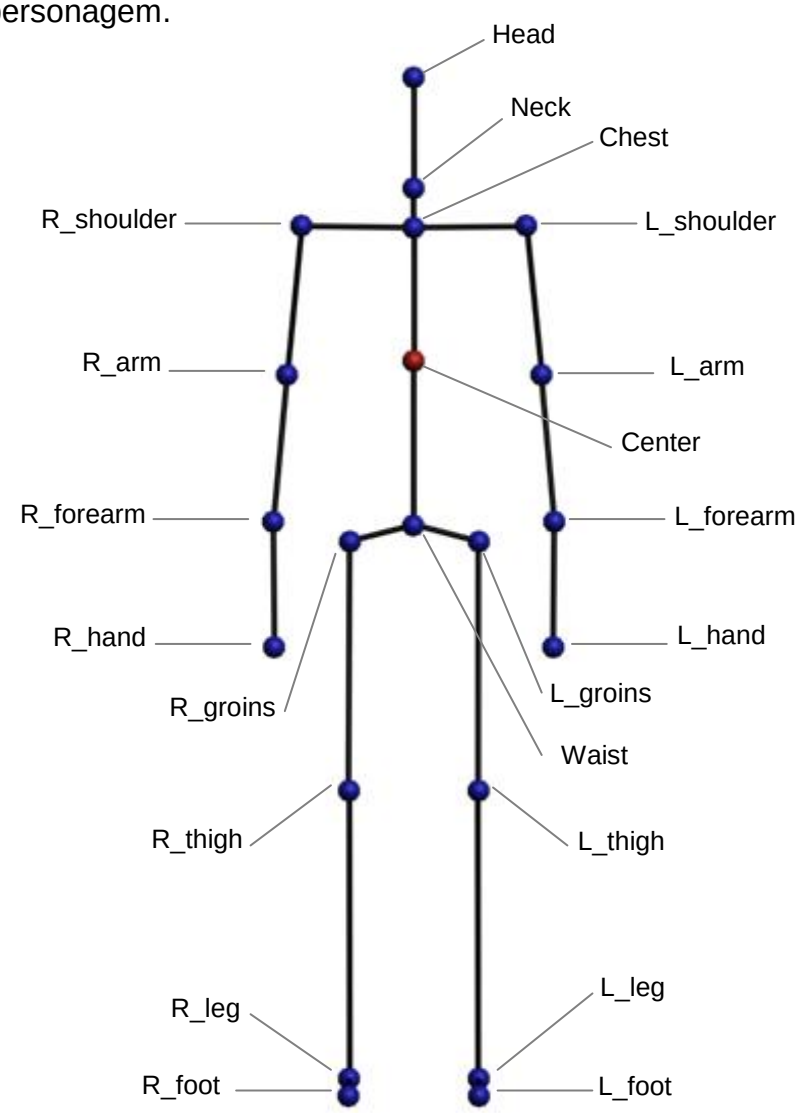

Figura 9 - Estrutura do esqueleto

Os *bones* são criados com um posicionamento inicial (posicionamento padrão para quando todas as rotações forem nulas) que pode ser visualizado na Figura 10. A imagem foi gerada no 3D Studio Max, que foi o software utilizado para gerar as coordenadas originais dos *bones*, já que tanto a imagem da Figura 9 quanto a imagem da Figura 10 servem somente para visualização de uma estrutura de dados abstrata.

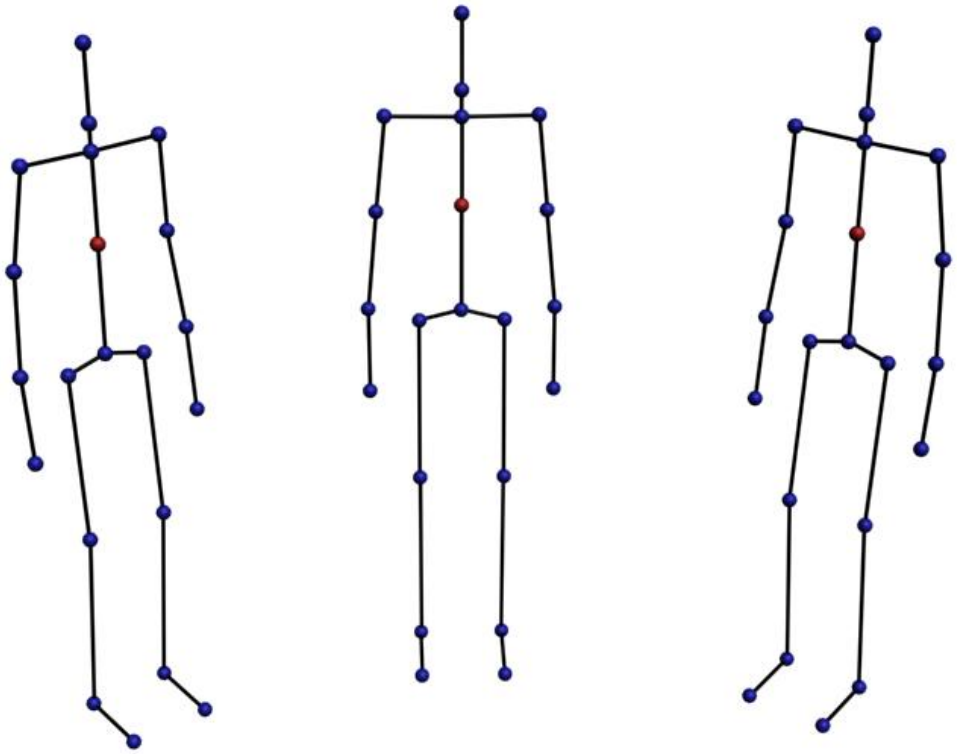

Figura 10 - Posicionamento original dos bones

Dessa forma, o movimento humano pode ser descrito através de rotações sobre os *bones* e a estrutura de árvore permite que uma rotação aplicada sobre um *bone* seja propagada para todos os seus *bones* filhos.

A classe *bone* deve apresentar basicamente os seguintes dados:

• Posição relativa ao *bone* pai;

• Posição global (sua posição em relação ao mundo);

• *BoneId* (identificador do *bone*);

• Rotação em X, rotação em Y e rotação em Z;

• Vetor de 3 posições para *bones* filhos.

Vale ressaltar que para essa implementação, em que são desconsiderados os dedos do personagem, o *bone* com o maior número de filhos é

o *bone* do peito (identificado por CHEST) que tem como filhos os dois ombros e o pescoço e assim sucessivamente como visto na Figura 8 e na Figura 9. A implementação desse esqueleto é sugerida por GOMES (1997), sendo representado por uma árvore em que cada nó é um *bone*.

As rotações são aplicadas aos nós filhos através de uma pilha responsável por guardar as rotações que serão aplicadas, de forma a inserir uma nova posição a cada sub-nó que é acessado. A cada rotação de um nó (ao se descer na árvore) insere-se no topo da pilha o valor do antigo topo somado com o valor da rotação atual e assim sucessivamente, até que se comece a voltar à raiz da árvore quando se remove o topo da pilha e onde se encontra, então, o valor da rotação desse nó pai. Essa estrutura de pilha já é oferecida pelo próprio OpenGL, como pode ser visualizado no trecho de código apresentado na Figura 11 que implementa a operação de exibição da imagem:

```
void render(){
  int i=0;
  //o topo da pilha é duplicado
  glPushMatrix();
  //as rotações são aplicadas ao topo da pilha
  glRotatef(rotateX,1,0,0);
  glRotatef(rotateY,0,1,0);
  glRotatef(rotateZ,0,0,1);
  glTranslated(relativePosition.getx(), relativePosition.gety(),
  relativePosition.getz());
//a forma é renderizada com as aplicações que estão no topo da pilha
  if((haveMesh == sim)){
    shape->render();
  }
//chamada recursiva aos filhos, o topo da pilha continua com o mesmo valor
  for(i=0;i<3;i++){
    if(child[i] != NULL){
      child[i]->render();
    }
  }
  //após retornar da execução dos filhos o topo da pilha é removido
  glPopMatrix();
}
```

Figura 11 - Trecho de código que exibe o personagem

A importância de se ter um identificador se dá pela necessidade de aplicar uma rotação sobre um determinado *bone*, fazendo do identificador uma peça fundamental. Para essa implementação foram utilizadas as identificações conforme apresentada na Figura 8 e na Figura 9.

3.3 Animação

Para gerar uma animação, o personagem deve carregar um conjunto de informações baseadas em *keyframes* que definem as rotações de cada um dos ossos e o tempo de movimentação entre um *keyframe* e outro. Com essas informações, mais a taxa de FPS (frames por segundo) da aplicação, consegue-se saber a rotação que deve ser aplicada em cada quadro, simplesmente utilizando a fórmula:

$$rotação\ por\ frame = \frac{rotação\ da\ animação}{tempo\ da\ animação\ em\ segundos * FPS}$$

Vale ressaltar que essa técnica de cálculo linear não acarreta erros de deformação de malha como ocorre em alguns modelos que trabalham apenas com o posicionamento dos pontos de controle.

A Figura 12 representa uma animação com deformação de malha, onde a movimentação do braço do personagem entre dois *keyframes* (primeiro *keyframe* com o braço estendido e segundo *keyframe* com o braço dobrado) e a aplicação de interpolação linear de um ponto ao outro causa uma deformação na malha entre os quadros intermediários, visualizado através do encurtamento do antebraço.

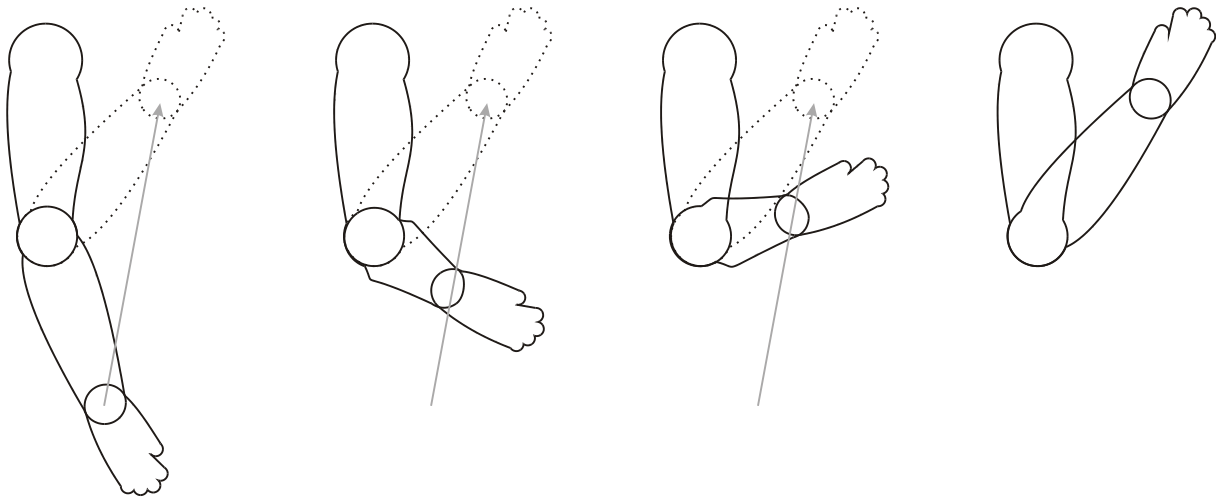

Figura 12 - Animação por posicionamento de pontos de controle, com deformação de malha

A Figura 13 demonstra a mesma animação, desta vez aplicando-se a técnica de rotação proposta, onde percebe-se que não há deformação na malha.

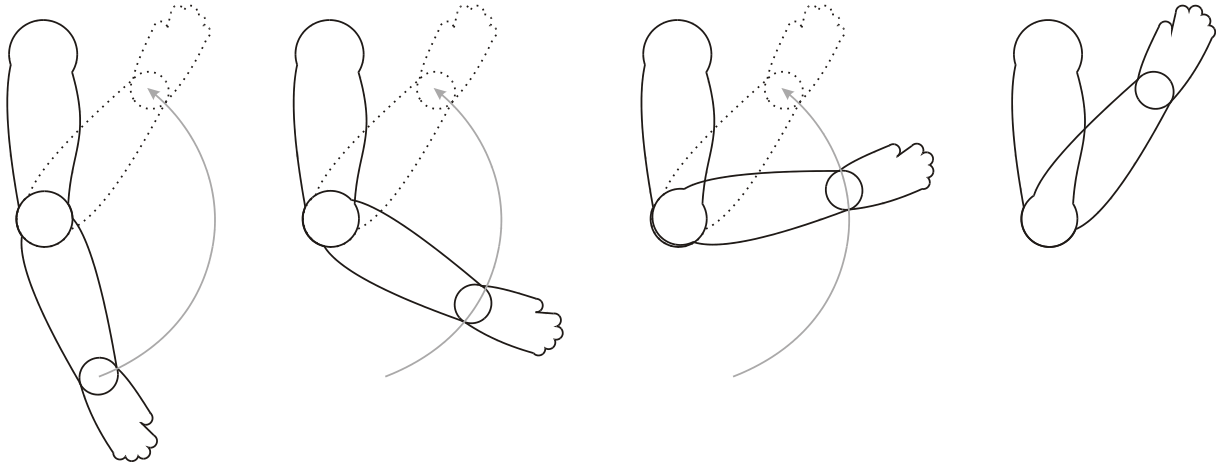

Figura 13 - Animação por rotação de pontos de controle, sem deformação de malha.

3.4 Malha do Personagem

Para a implementação proposta, a malha do personagem é desmembrada para as partes correspondentes aos *bones* do esqueleto. Quase todos os *bones* tem uma malha ligada a ele, sendo que apenas o *bone* raiz (*center*) e os *bones* das virilhas não possuem malha.

A Figura 14 mostra as formas que foram utilizadas para a construção do personagem (à esquerda, separadas para melhor visualização e à direita, agregadas de modo a montar o personagem). A malha foi gerada no 3D Studio Max e exportada para o formato XML, fazendo automaticamente as devidas conversões de eixos e gerando os vetores normais por vértices (padrão do OpenGL).

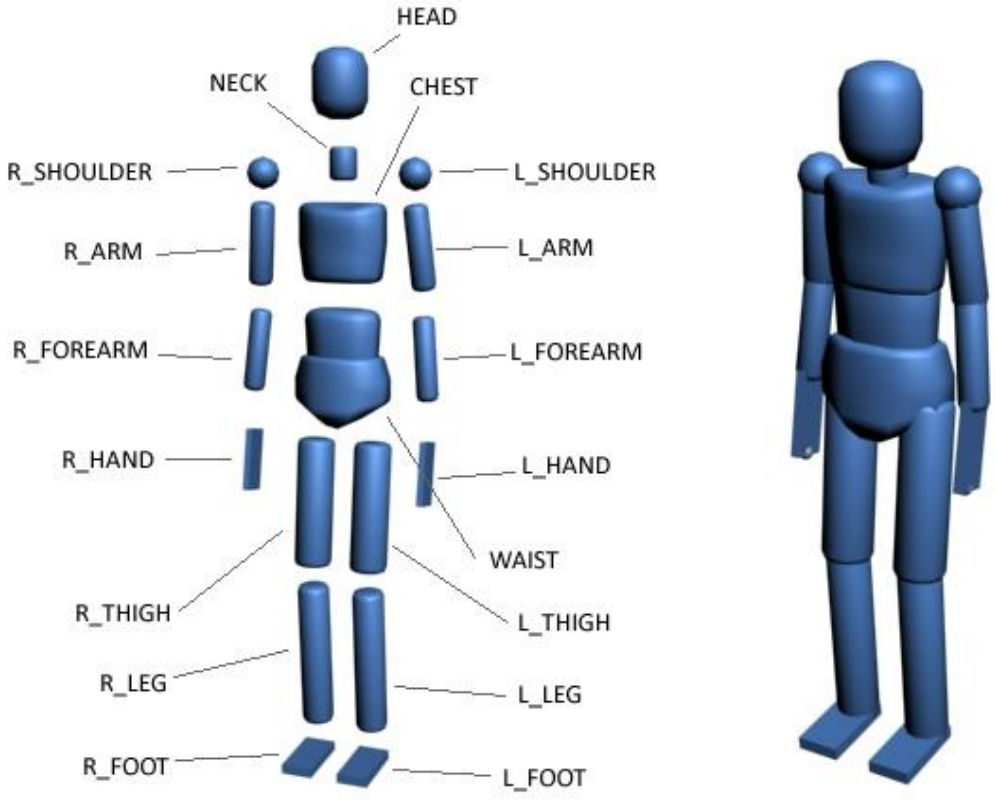

Figura 14 - Malha genérica que cobre o personagem

O padrão do formato XML exportado pelo 3D Studio Max é descrito pelo código apresentado na Figura 15. Nele verifica-se um conjunto de faces da geometria que é definida pelos índices dos vértices que são apresentados posteriormente. Esses vértices além de sua posição tridimensional levam consigo a informação de um vetor normal e de coordenadas de textura, que na implementação proposta é ignorada, por não ser utilizada.

```
<?xml version="1.0" ?>
<mesh>
  <submeshes>
    <submesh material="DefaultMaterial" usesharedvertices="false"
    use32bitindexes="false">
     <faces count="12">
      <face v1="0" v2="2" v3="3" />
      <face v1="3" v2="1" v3="0" />
      <face v1="4" v2="5" v3="7" />
      ...
     </faces>
     <geometry vertexcount="8">
      <vertexbuffer positions="true" normals="true"
      colours_diffuse="false" texture_coords="1"
      texture_coords_dimensions_0="2">
```

```
            <vertex>
              <position x="-6.132783" y="-1.664001" z="1.823757" />
              <normal x="0.000000" y="-2.000000" z="-0.000000" />
              <texcoord u="0.000000" v="1.000000" />
            </vertex>
            <vertex>
              <position x="5.867217" y="-1.664001" z="1.823757" />
              <normal x="0.000000" y="-1.000000" z="-0.000000" />
              <texcoord u="1.000000" v="1.000000" />
            </vertex>
            <vertex>
            ...
            </vertex>
          </vertexbuffer>
        </geometry>
      </submesh>
    </submeshes>
    <submeshnames>
      <submeshname name="r_foot" index="0" />
    </submeshnames>
</mesh>
```

Figura 15 - Exemplo de XML exportado pelo 3D Studio Max

A malha utilizada nessa implementação é uma malha genérica, usada apenas para visualização e demonstração das funcionalidades do programa, o qual permite a aplicação de qualquer outra malha/forma de modo que se possam ter personagens diferentes na dramatização da história.

3.5 Dependência/Independência do Movimento da Cabeça

O movimento da cabeça em uma animação é algo que exige uma atenção diferenciada, especialmente nesse projeto em que as animações se dão de forma dinâmica. A cabeça pode ter sua animação atrelada à animação do corpo ou pode ser independente, gerando assim alguns possíveis problemas relativos à realidade da movimentação do corpo humano que devem ser solucionados. Para tanto, faz-se duas subdivisões:

- Movimento da cabeça sugerido pela animação do corpo: para esse caso o *script* da animação possui uma pré-definição do movimento da cabeça, mas não obriga o diretor a usá-la. Por exemplo, a animação de subir uma escada: normalmente uma pessoa sobe uma escada olhando para cima, com a cabeça inclinada para trás, mas nada impede que por uma necessidade de dramatização o diretor interfira nessa animação e peça para o personagem executar a animação de subir a escada olhando para o lado ou para baixo.

- Movimento da cabeça ligada obrigatoriamente à animação do corpo: a animação do personagem quando gerada deve trazer consigo uma *flag* que indica a obrigatoriedade de ser respeitada a animação da cabeça ligada ao corpo. Por exemplo, em uma animação de uma dança em que o personagem pode dar um giro, a sua cabeça não pode ficar olhando sempre para o mesmo ponto ou causará a impressão de o pescoço ter girado 360°, o que é impossível.

3.6 Auxílio ao Posicionamento da Câmera

O personagem deve auxiliar a câmera a fazer tomadas pré-definidas, ajudando no posicionamento e no comportamento da mesma.

Um sistema de câmera é composto por um ponto que define a posição da câmera e dois vetores: um que define a direção para onde a câmera está virada e outro que define a direção para onde a câmera está inclinada (vetor *up*). Essas informações podem ser visualizadas na Figura 16.

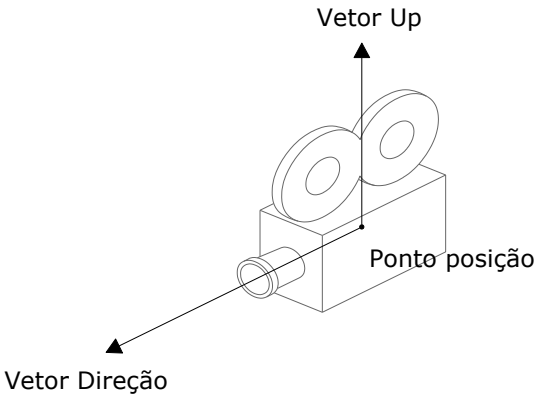

Figura 16 - Componentes de uma câmera

Esse auxílio do personagem se dá quanto à posição e a direção da câmera, deixando sua inclinação a cargo do diretor.

Tem-se cinco posicionamentos possíveis para a câmera: *over the shoulder* (tomada feita sobre o ombro, tanto esquerdo quanto direito, um pouco atrás do personagem), *third person* (tomada em terceira pessoa, onde a câmera se posiciona atrás do personagem, acima da cabeça), *first person* (primeira pessoa, onde a câmera se posiciona de forma que dê a impressão de ser a visão do próprio personagem) e *face close* (*close* do rosto, onde a câmera posiciona-se um pouco a

frente da face do personagem, sendo que este posicionamento já configura automaticamente o direcionamento para um ponto posicionado no centro da face do personagem, usando também para mostrar/realçar as expressões faciais).

4 DESENVOLVIMENTO E IMPLEMENTAÇÃO

O personagem da implementação proposta é composto basicamente por um esqueleto e alguns parâmetros como direção, posição, um vetor que guarda as animações que poderão ser executadas pelo esqueleto e o tipo de tomada que está sendo feita.

Como o diretor também tem liberdade para definir para onde o personagem deve olhar, faz-se necessário guardar a informação deste ponto, assim como uma variável do tipo *boolean* que informa se o personagem está ou não olhando para essa direção.

A seguir descreve-se detalhes da implementação de vários aspectos deste trabalho.

4.1 O Esqueleto do Personagem

O esqueleto é a principal estrutura do personagem, já que é o responsável por guarda as informações de rotações dos ossos e exibir essas estruturas na tela. É uma estrutura em formato de árvore, portanto, todos os métodos de acesso devem ser recursivos.

Cada *bone* que compõe o esqueleto é um nó desta árvore. Eles podem possuir uma malha (apresentada na secção 3.4) ligada a eles, como é o caso, por exemplo, dos *bones* da mão, do pé, do braço, etc. Por sua vez, existem *bones* que não possuem uma malha ligada a eles, como é o caso do *bone* do centro do personagem e das virilhas, pois os mesmos apenas são úteis para auxilio de posicionamento dos seus nós filhos.

A estrutura de esqueleto do personagem é construída através de métodos de inserção, que são métodos que adicionam um filho a um *bone.* Assim, no construtor do personagem, a criação desta estrutura de esqueleto se dá a partir das folhas da árvore em direção a raiz, isto é, dos membros do personagem até o seu centro. A Figura 17 apresenta o algoritmo em alto nível utilizado para a inicialização do esqueleto.

```
Cria o pé esquerdo
Cria a perna esquerda
Insere o pé esquerdo na perna esquerda
Cria a coxa esquerda
Insere a perna esquerda na coxa esquerda
Cria a virilha esquerda
Insere a coxa esquerda na virilha esquerda
...
```

Figura 17 - Algoritmo para a inicialização do esqueleto

Os *bones* possuem informação da sua posição global (que guarda a informação do *bone* em relação ao mundo/cenário tridimensional) e da sua posição relativa ao seu *bone* pai (nesse caso, o pai é considerado o ponto zero do plano tridimensional). Também guardam os valores das rotações nos eixos x, y e z em um vetor e a ordem dessas rotações em um vetor auxiliar. Além disso, cada *bone* possui uma matriz 3x3 que é a matriz de rotação aplicada sobre o ponto que guarda a posição relativa. Com o auxílio dessa matriz consegue-se passar as informações de rotações para os *bones* filhos.

A matriz se faz necessária pois pode-se aplicar, por exemplo, uma rotação de 15 graus em x, outra de 20 graus em y e uma rotação de 30 graus novamente em x. Se guardássemos apenas um valor para cada rotação, a segunda rotação em x seria incrementada para 45 graus, o que gera um erro, pois uma rotação de 45 graus em x seguida por uma rotação de 20 graus em y gerará um resultado diferente do que se deseja.

Vale ressaltar que para a simples exibição na tela da animação do personagem não haveria necessidade de guardar os pontos de posicionamento global e relativo e nem mesmo a matriz, apenas seria importante o valor das rotações nos eixos em um vetor que mais tarde seria aplicado ao OpenGL. Como a proposta exige o conhecimento da posição dos *bone*s em relação ao mundo para que os cálculos de posicionamento da câmera, levando em consideração a geometria do personagem, sejam possíveis, essas informações são de extrema importância.

A informação de posição em relação ao pai é importante, pois a rotação que for aplicada será feita sobre esse ponto e o cálculo da posição global é dado pela soma dessa posição relativa com a posição global do seu pai.

4.2 Posicionamento da Câmera

Para esta implementação estão disponíveis seis tipos de tomada: *NONE, THIRD_PERSON, FIRST_PERSON, FACE_CLOSE, R_OVER_SHOULDER, L_OVER_SHOULDER*. A primeira opção indica que não há nenhuma tomada específica do personagem sendo realizada, isto é, a câmera está livre. As outras cinco referem-se as tomadas em terceira pessoa, primeira pessoa, *close* da face, tomadas sobre o ombro direito e sobre o ombro esquerdo respectivamente.

As tomadas que estão sucetíveis a serem modificadas pela direção para onde o personagem está olhando (quando se pode direcionar a câmera para essa posição) são as de terceira pessoa e sobre os ombros, já que na tomada de *close* da face a câmera deve voltar-se sempre para a face e na visão em primeira pessoa a direção da câmera já é a direção para onde o personagem está olhando.

Para auxiliar o cálculo de direcionamento da cabeça faz-se necessário um *bone* adicional sobre a cabeça do personagem, que está inicialmente direcionado para frente e ao se aplicar rotações sobre a cabeça esse *bone* auxiliar também é rotacionado e o vetor do seu posicionamento relativo ao pai (cabeça) é o vetor direção para onde o personagem está olhando.

A Figura 18 apresenta o cálculo do posicionamento da câmera para uma tomada em terceira pessoa sem o pedido do diretor para direcionar a câmera ao ponto que o personagem está olhando.

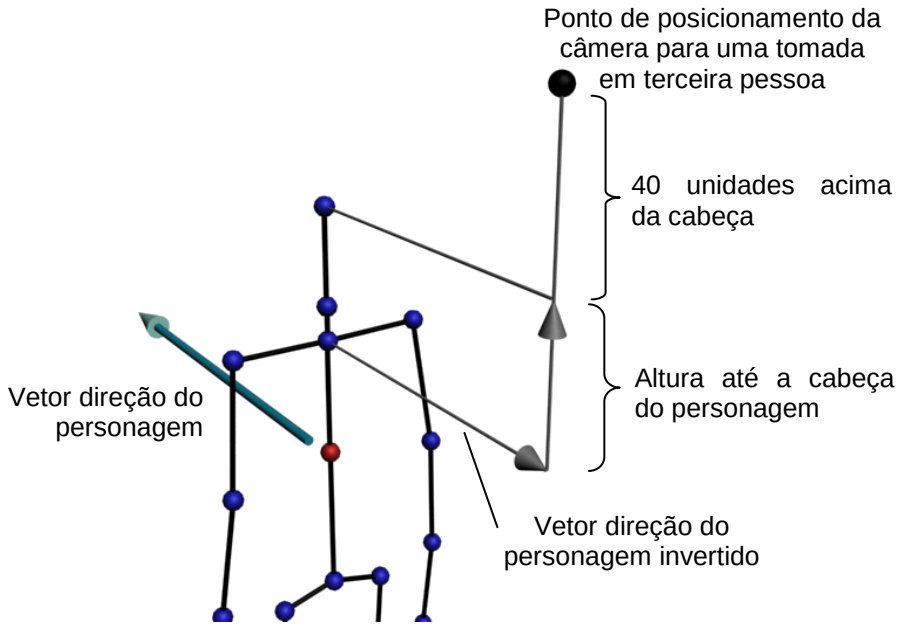

Figura 18 - Detalhe para posicionar a câmera em tomada de terceira pessoa

Para o posicionamento da câmera em uma tomada de terceira pessoa, a câmera é posicionada atrás do personagem (sendo a direção encontrada através do inverso da direção do personagem) e acima da cabeça do mesmo. Observa-se que dispondo-se desta estrutura de esqueleto, pode-se facilmente criar outras configurações de câmera, simplesmente alterando-se as dimensões dos vetores.

A Figura 19 apresenta o cálculo do posicionamento da câmera para uma tomada em primeira pessoa. Este cálculo é feito encontrando-se o ponto intermediário entre o pescoço e a cabeça do personagem e, na seqüência, é realizada a soma de uma parte do vetor auxiliar presente no esqueleto que indica a direção da cabeça.

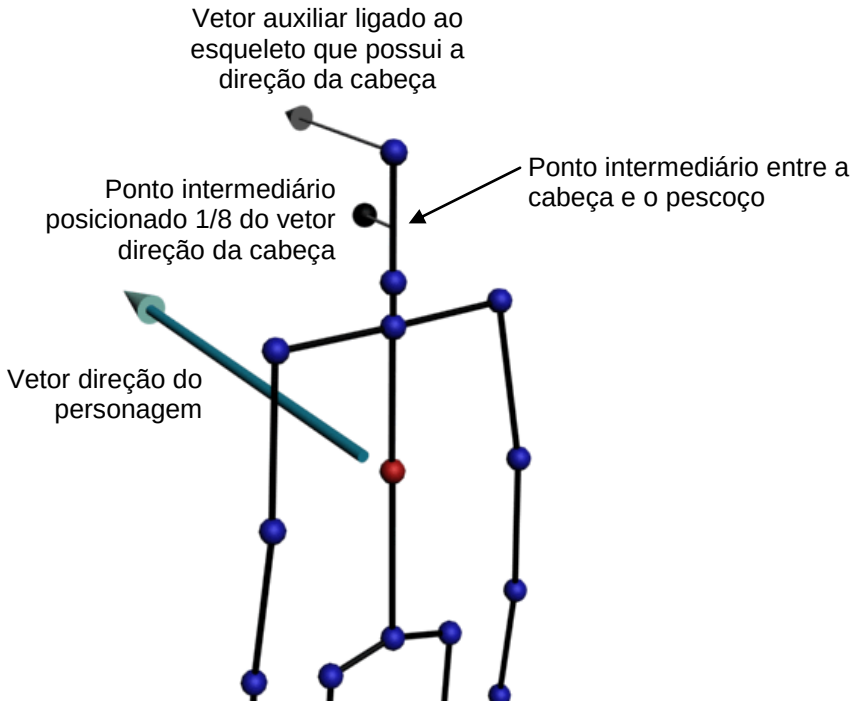

Figura 19 - Detalhe para posicionar a câmera em tomada de primeira pessoa

Para o posicionamento da câmera em uma tomada de *close* da face, o cálculo é semelhante ao do posicionamento em primeira pessoa. Apenas não se divide o vetor auxiliar, deixando-o com seu tamanho original.

A Figura 20 apresenta o cálculo do posicionamento da câmera para uma tomada sobre o ombro direito sem o direcionamento para o ponto onde o personagem está olhando. O cálculo inicia-se com o posicionamento do peito do personagem, movimentando-o no sentido contrário a direção para a qual o personagem está posicionado, somando-se o vetor do ombro direito e a altura da

cabeça e, finalmente, diminuindo-se a altura da câmera para que esta fique em uma posição mais próxima do que seria a visão do personagem. Este cálculo é semelhante para tomadas sobre o ombro esquerdo, onde a diferença fundamentalmente consiste em utilizar o vetor do ombro esquerdo para somar ao ponto que foi encontrado inicialmente.

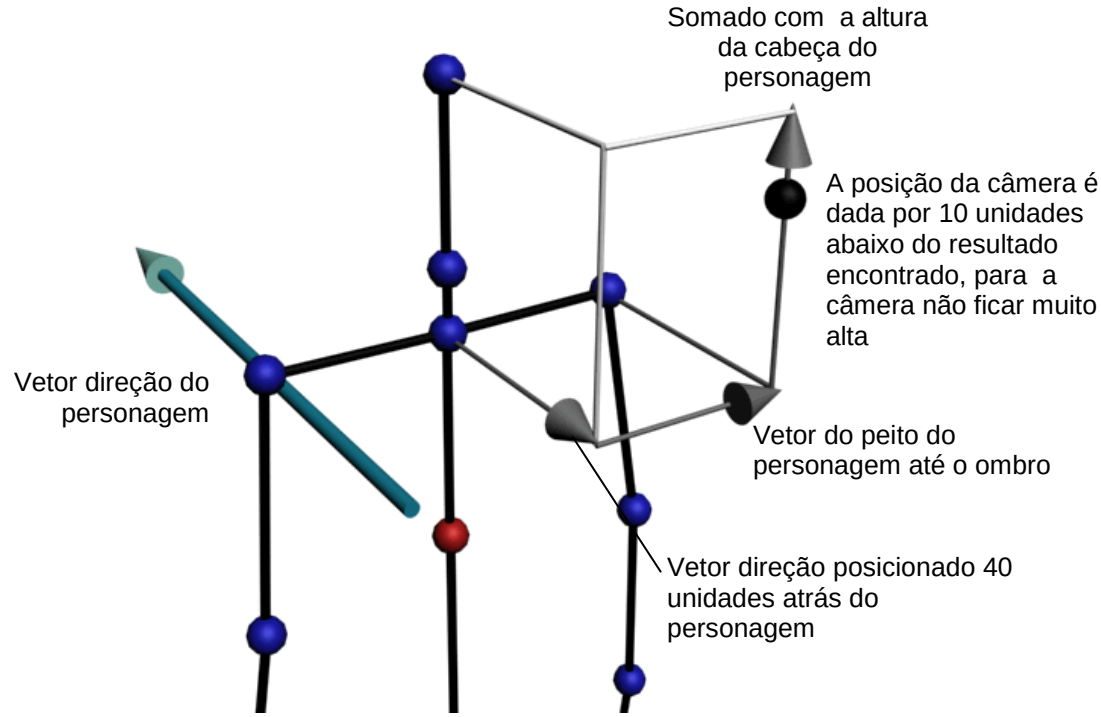

Figura 20 - Detalhe para posicionar a câmera em tomada sobre o ombro sem direcionamento definido

A seção 4.3 apresenta os detalhes para o posicionamento caso a direção da câmera seja para onde o personagem está olhando.

4.3 Direcionamento da Câmera

Após calcular o ponto onde a câmera deve posicionar-se, o personagem precisa também calcular o vetor direção para a câmera (como apresentado na secção 3.6).

Para cada tomada o cálculo deve ser feito de forma diferente, levando em consideração o pedido do diretor para que se necessário, a câmera direcione-se ao ponto para onde o personagem está olhando, mas apenas se a tomada permitir essa liberdade (o que não ocorre nas tomadas de *close* da face ou de visão em primeira pessoa, já que essas tomadas têm direcionamentos específicos).

Para a direção da câmera na visão em primeira pessoa, o vetor utilizado é o mesmo vetor auxiliar do esqueleto que informa a direção para onde a cabeça está virada. Tendo-se esse vetor auxiliar não é necessário nenhum cálculo adicional.

A direção para a tomada de *close* da face também é trivial de ser encontrada, pois é o mesmo vetor auxiliar utilizado para a visão em primeira pessoa, mas dessa vez invertido, isto é, multiplicado por -1.

Na tomada em terceira pessoa há duas possibilidades: a primeira de uma tomada normal, sem a interferência da posição para onde o personagem está olhando e a segunda em que o diretor exige que a câmera volte-se para o ponto que atraiu a atenção do personagem. No primeiro caso, uma boa solução é direcionar a câmera para a parte superior da cabeça do personagem. Deste modo basta encontrar o vetor que tem sua origem na câmera (posição já conhecida, calculada anteriormente) até o topo da cabeça do personagem (posição que pode ser informada pelo esqueleto). O vetor resultante é a direção da câmera para uma tomada em terceira pessoa normal.

No caso de ser necessário que a câmera volte-se para a direção em que o personagem está olhando, basta substituir o ponto da parte superior da cabeça (utilizado no cálculo anterior) pelo ponto para onde a cabeça do personagem está voltada e será encontrado o vetor que tem sua origem na câmera até o ponto de atenção do personagem. Este é o vetor direção da câmera para uma tomada em terceira pessoa direcionada para a visão do personagem.

A Figura 21 apresenta as posições geométricas para uma tomada sobre o ombro do personagem quando o diretor ordena que a câmera direcione-se para o ponto que o personagem está olhando.

Neste caso, a posição da câmera também sofre alterações para que a tomada não perca a sua característica de ser “sobre o ombro”. Faz-se necessário encontrar um ponto acima do ombro onde está sendo feita a tomada. Por este ponto deve passar a visão da câmera sendo que é a partir dele que se consegue encontrar um vetor que vai até o ponto para onde o personagem está olhando. Assim, posiciona-se a câmera na direção contrária a esse vetor somando-se a este o ponto sobre o ombro do personagem. A direção da câmera é o próprio vetor obtido inicialmente.

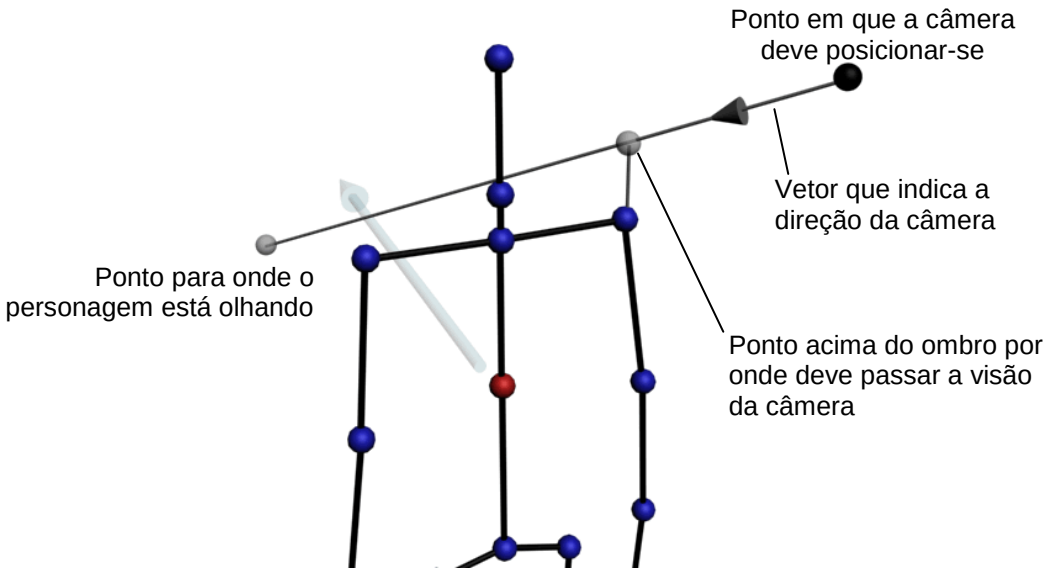

Figura 21 - Direcionamento e posicionamento da câmera em tomada sobre o ombro

A Figura 22 apresenta um exemplo de tomada *Over The Shoulder* pelo sistema implementado.

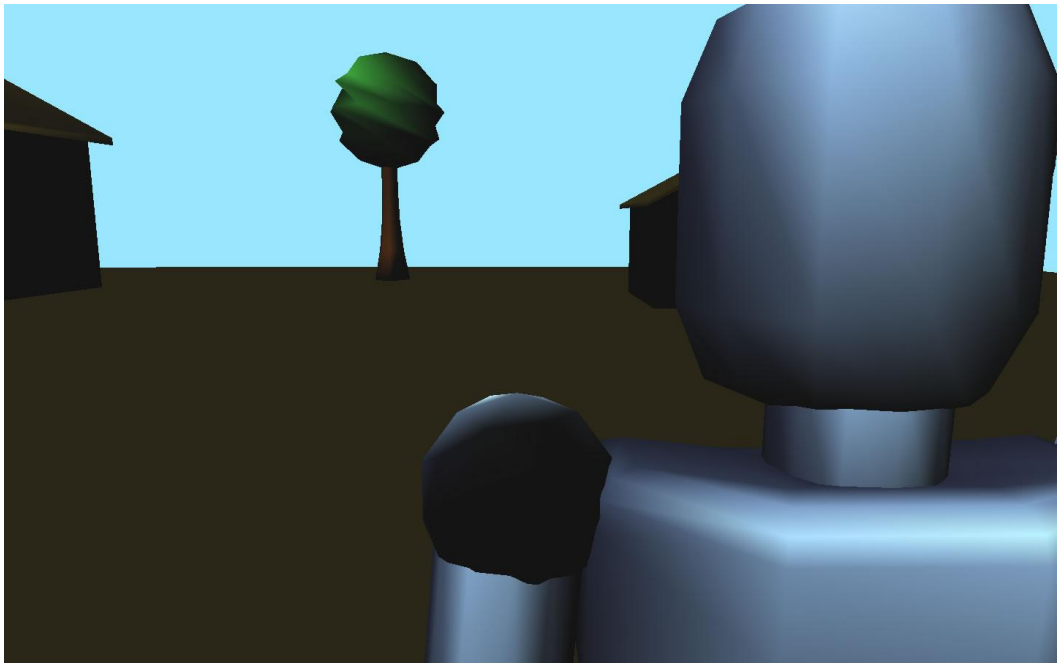

Figura 22 - Tomada Over The Shoulder no sistema proposto

A tomada em terceira pessoa apresenta um cálculo muito semelhante ao da tomada sobre o ombro, quando a câmera deve voltar-se para o ponto que o personagem está olhando. A diferença é o ponto por onde deve passar a visão da câmera, que na tomada em terceira pessoa é substituído pelo ponto do topo da

cabeça do personagem, sendo que os cálculos para a posição e direção da câmera seguem o mesmo algoritmo.

A Figura 23 apresenta imagens que foram geradas pelo sistema proposto e que oferecem exemplos de um close na face (a), uma visão em terceira pessoa (b), uma visão de primeira pessoa (c) e uma tomada com posicionamento livre da câmera, com esta direcionada para a face do personagem (d).

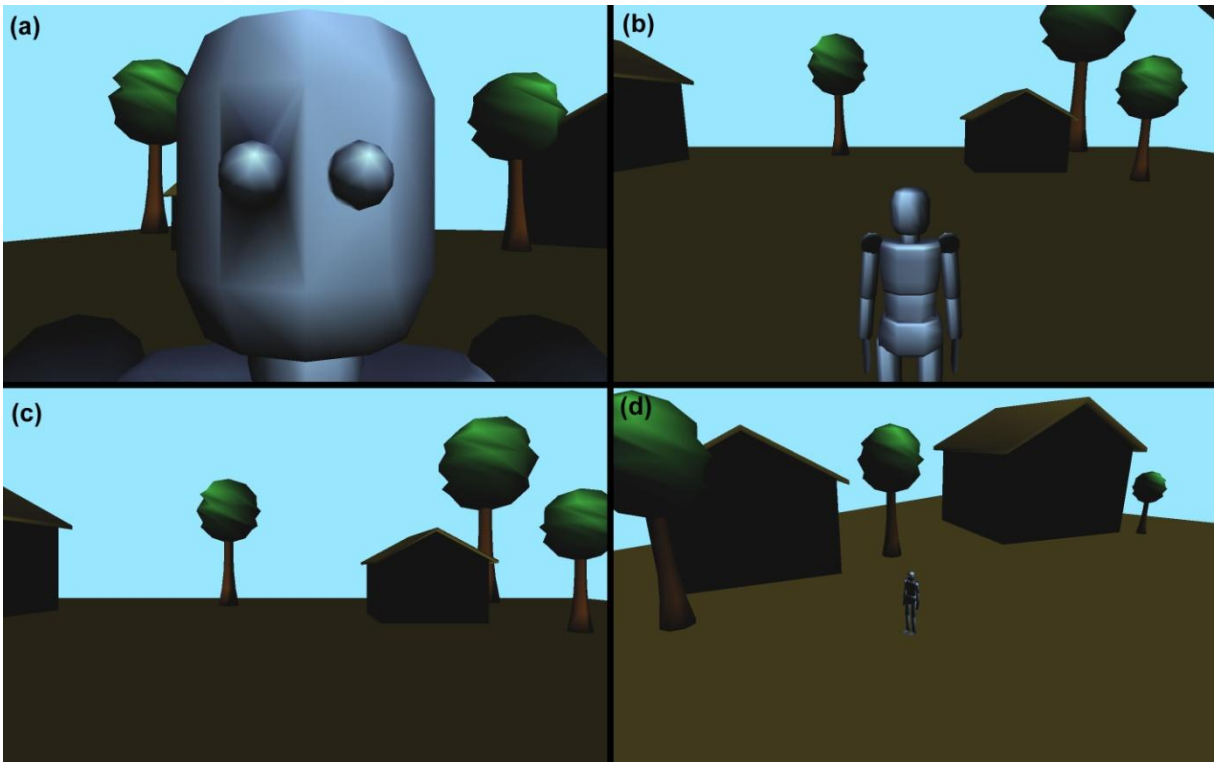

Figura 23 - Algumas tomadas possíveis feita pelo sistema

Baseado no que foi proposto, disponibiliza-se três direcionamentos possíveis para a câmera:

• *What you look:* direciona a câmera para onde o personagem está olhando, como apresentado na Figura 24 (a) em uma tomada em terceira pessoa e na Figura 24 (b) em uma tomada sobre o ombro direito;

• *Where you are going:* direciona a câmera para onde o personagem está virado, isto é, na direção do personagem, como já demonstrado na Figura 22 e Figura 23 (b);

• *Part of body:* direciona a câmera para uma determinada parte do corpo do personagem, como demonstrado na Figura 23 (d).

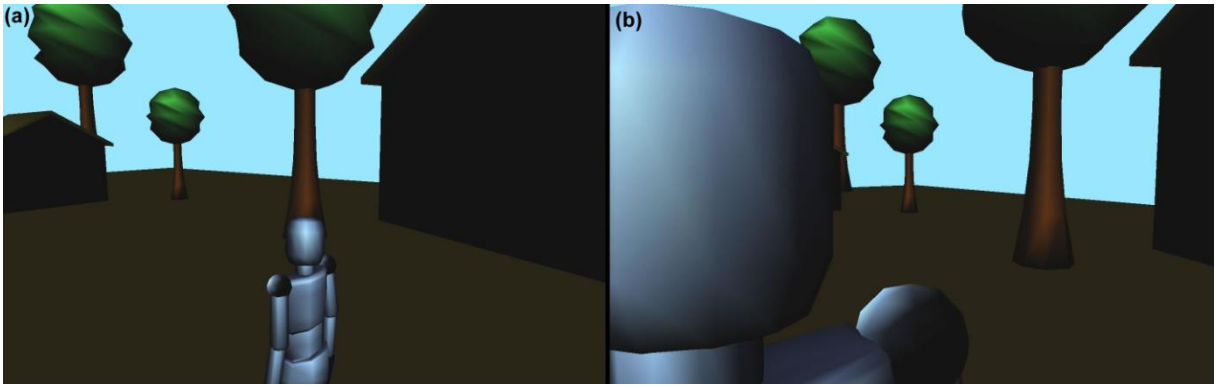

Figura 24 - Direcionamento da câmera para onde o personagem está olhando

A Figura 25 exemplifica um uso mesclado das funções de tomada sobre o ombro, direcionamento da câmera e recuperação de um ponto do personagem. Representa uma tomada sobre o ombro direito do personagem, direcionada para onde ele está olhando que neste caso é a cabeça do outro personagem.

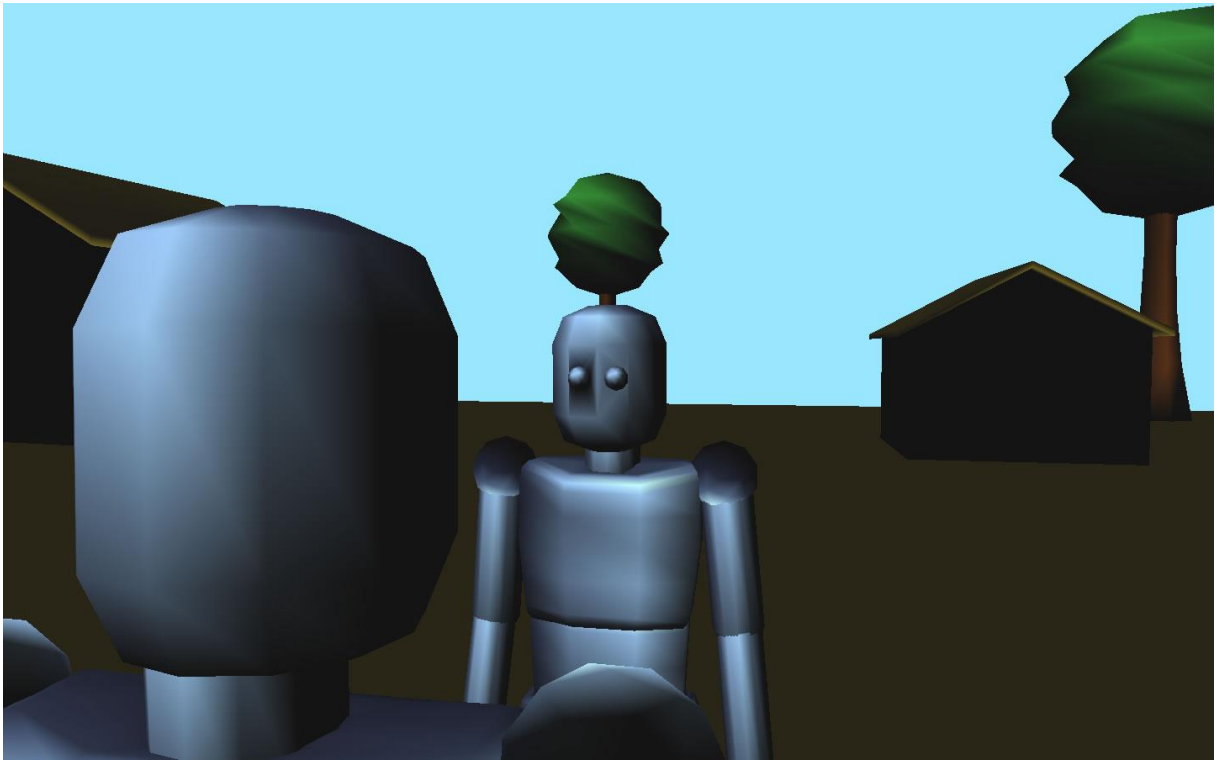

Figura 25 - Tomada sobre o ombro do personagem interagindo com outro

4.4 Formato da Animação

A animação é caracterizada por uma seqüência de posicionamentos do esqueleto do personagem utilizando-se a estrutura de *keyframes.* Desta forma, faz-se necessário guardar as rotações de cada *bone* a cada *keyframe* para posteriormente interpolar as rotações entre os mesmos.

A animação é importada de um arquivo XML, que possui informações das rotações de cada *bone* em cada *keyframe* e o tempo de interpolação entre ambos. A Figura 26 apresenta um código XML de uma animação (animação do personagem parado, respirando) que possui dois *keyframes*.

```
<?xml version="1.0" ?>
<animation>
  <keyframes quantidade="2" cabeca_livre="1">
    <keyframe tempo="2">
     <CENTER px="0" py="0" pz="0"/>
     <CHEST rx="-3.00000000000007" ry="0" rz="0"/>
     <NECK rx="0" ry="0" rz="0"/>
     <HEAD rx="4.99999999999999" ry="0" rz="0"/>
     <R_SHOULDER rx="0" ry="0" rz="0"/>
     <R_ARM rx="0" ry="0" rz="0"/>
     <R_FOREARM rx="0" ry="0" rz="0"/>
     <R_HAND rx="0" ry="0" rz="0"/>
     <L_SHOULDER rx="0" ry="0" rz="0"/>
     <L_ARM rx="0" ry="0" rz="0"/>
     <L_FOREARM rx="0" ry="0" rz="0"/>
     <L_HAND rx="0" ry="0" rz="0"/>
     <WAIST rx="0" ry="0" rz="0"/>
     <R_GROINS rx="0" ry="0" rz="0"/>
     <R_THIGH rx="0" ry="0" rz="0"/>
     <R_LEG rx="0" ry="0" rz="0"/>
     <R_FOOT rx="0" ry="0" rz="0"/>
     <L_GROINS rx="0" ry="0" rz="0"/>
     <L_THIGH rx="0" ry="0" rz="0"/>
     <L_LEG rx="0" ry="0" rz="0"/>
     <L_FOOT rx="0" ry="0" rz="0"/>
    </keyframe>
    <keyframe tempo="2">
     <CENTER px="0" py="0" pz="0"/>
     <CHEST rx="0" ry="0" rz="0"/>
     <NECK rx="0" ry="0" rz="0"/>
     <HEAD rx="0" ry="0" rz="0"/>
     <R_SHOULDER rx="0" ry="0" rz="0"/>
     <R_ARM rx="0" ry="0" rz="0"/>
     <R_FOREARM rx="0" ry="0" rz="0"/>
     <R_HAND rx="0" ry="0" rz="0"/>
     <L_SHOULDER rx="0" ry="0" rz="0"/>
     <L_ARM rx="0" ry="0" rz="0"/>
     <L_FOREARM rx="0" ry="0" rz="0"/>
     <L_HAND rx="0" ry="0" rz="0"/>
     <WAIST rx="0" ry="0" rz="0"/>
     <R_GROINS rx="0" ry="0" rz="0"/>
```

```
      <R_THIGH rx="0" ry="0" rz="0"/>
      <R_LEG rx="0" ry="0" rz="0"/>
      <R_FOOT rx="0" ry="0" rz="0"/>
      <L_GROINS rx="0" ry="0" rz="0"/>
      <L_THIGH rx="0" ry="0" rz="0"/>
      <L_LEG rx="0" ry="0" rz="0"/>
      <L_FOOT rx="0" ry="0" rz="0"/>
     </keyframe>
  </keyframes>
</animation>
```

Figura 26 - Exemplo de XML com script de animação

O nó denominado *keyframes* apresenta informações da quantidade de quadros-chaves que existem na animação, assim como a informação sobre o movimento da cabeça do personagem ser livre ou não para poder ser manipulada pelo diretor.

4.4.1 Transições entre *Keyframes*

Cada animação importada de um arquivo XML é guardada em um objeto "*keyframe"*. Este objeto é composto por vetores bi-dimensionais que guardam as rotações em x, y e z, individualmente, de todos os *bones* em todos os quadros-chaves e possui também vetores que guardam a rotação atual de cada *bone.*

Essa rotação atual é calculada com base na fórmula apresentada na seção 3.3, levando em consideração a taxa de FPS (frames por segundo) para não perder o controle sobre o tempo de duração da animação. A fórmula gerada é:

$$rotação = \frac{rotação\ Objetivo - rotação\ atual}{(tempo\ da\ rotação - tempo\ decorrido) * FPS}$$

É importante ter uma transição suave entre uma animação e outra, isto é, evitar cortes bruscos entre uma animação de corrida e uma animação de luta, por exemplo. Isto ocorre quando simplesmente troca-se a animação que está sendo executada por outra. Quando o personagem encontra-se em uma determinada posição e no quadro seguinte, por causa da troca de animação, está em outra posição totalmente diferente, cria-se a impressão de um "corte" na animação, ou seja, uma descontinuidade no movimento.

Para solucionar este problema na implementação proposta utiliza-se um novo objeto *keyframe* apenas para a transição entre uma animação e outra. Essa

estrutura tem como seu primeiro quadro a posição atual do personagem e como seu segundo quadro a primeira posição da animação para a qual se deseja fazer a transição. O tempo dessa transição deve ser curto (neste caso foi utilizado 0.3 segundos) para não interferir no resultado final da animação.

4.4.2 Editor Básico para Auxílio à Criação de Animações

Para poder criar animações de uma forma mais rápida e intuitiva fez-se necessária a implementação de um editor de animações. Para tanto foi utilizada a linguagem ActionScript em conjunto com o *software* Adobe Flash CS3. A escolha dessa linguagem e *software* se deu por sua facilidade em criação de interfaces de comunicação com o usuário, além de ser orientada a objetos e ser uma linguagem que disponibiliza recursividade, algo necessário para os padrões atuais do projeto.

As classes utilizadas no editor são basicamente as mesmas classes utilizadas no projeto do personagem com algumas modificações necessárias para a tradução do código. A classe de *bones* do personagem está presente para exibir o resultado das rotações do esqueleto na tela assim como as classes de matriz 3x3 e de vetor.

Novas classes foram necessárias para auxiliar a geração do *script* XML, assim como para converter as várias rotações sucessivas sobre um determinado *bone* em apenas uma rotação em x, uma rotação em y e uma rotação em z. Para tanto, no momento da exportação, as rotações são aplicadas sobre a matriz de rotação de cada um dos *bones*, tendo uma matriz final que possui a rotação desejada.

Considerando:

Matriz de rotação em x = $\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos & -sen \\ 0 & sen & cos \end{bmatrix}$.

Matriz de rotação em y = $\begin{bmatrix} cos & 0 & sen \\ 0 & 1 & 0 \\ -sen & 0 & cos \end{bmatrix}$.

Matriz de rotação em z = $\begin{bmatrix} cos & -sen & 0 \\ sen & cos & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

A matriz resultado da multiplicação da matriz x.y.z nessa ordem é

$$\begin{bmatrix} cosy.cosz & cosy.(-senz) & seny \\ -senx.(-seny).(-cosz)+cosx.senz & -senx.(-seny).(-senz)+cosx.cosz & -senx.cosy \\ cosx.(-seny)+senx.senz & cosx.(-seny).senz+senx.cosz & cosx.cosy \end{bmatrix}$$

Dessa forma, ao igualar a matriz de rotação com a matriz de multiplicação consegue-se recuperar os valores de apenas uma rotação em x, uma rotação em y e uma rotação em z, sendo esses os valores exportados para o XML.

As equações criadas para encontrar os valores de x, y e z podem ser descritas como:

$$rotação\ em\ y = \text{acos}\ (celula[0,2] da\ matriz) \tag{4.1}$$

As rotações em x e em z por não serem diretas devem ter dois cálculos, um que define o sinal da rotação e outro que define o valor dessa rotação.

$$sinal\ da\ rotação\ em\ x = -\frac{asen(celula[1,2] da\ matriz)}{\cos(y\ calculado)} \tag{4.2}$$

$$rotação\ em\ x = \frac{\text{acos}\ (celula[2,2] da\ matriz)}{\cos\ (y\ calculado)} \tag{4.3}$$

$$sinal\ da\ rotação\ em\ z = -\frac{asen(celula[0,1] da\ matriz)}{\cos\ (y\ calculado)} \tag{4.4}$$

$$rotação\ em\ z = \frac{\text{acos}\ (celula[0,0] da\ matriz)}{\cos\ (y\ calculado)} \tag{4.5}$$

O valor final para as rotações em x (equação 4.3) e z (equação 4.5) devem possuir o sinal do valor encontrado nas equações 4.2 e 4.4 respectivamente.

A Figura 27 mostra a interface desse editor. A tela está dividida em quatro áreas sendo três delas utilizadas para efetuar as rotações dos *bones* e uma utilizada para configuração do personagem.

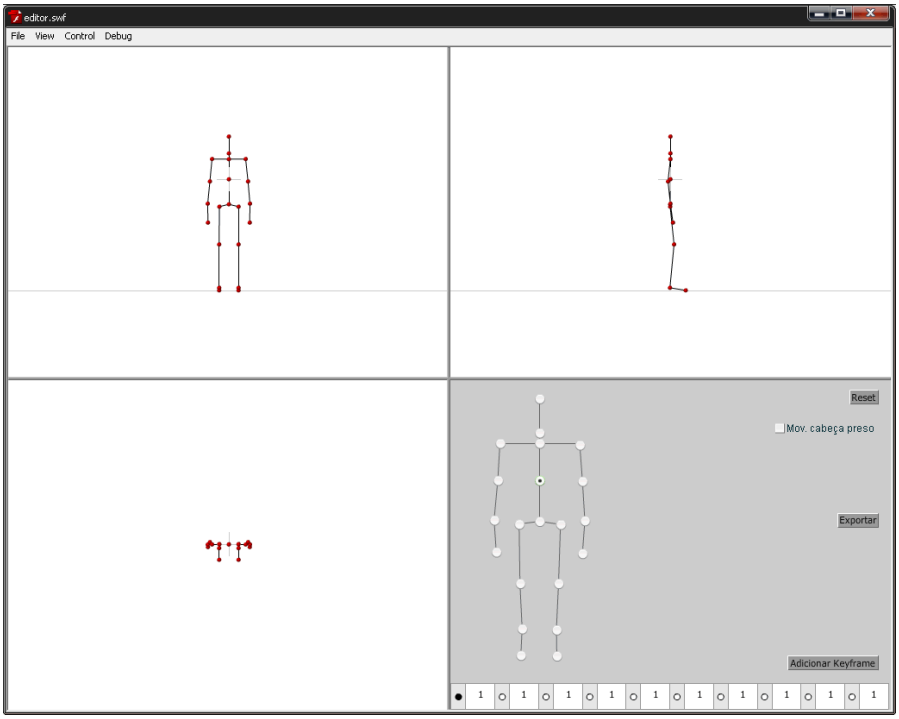

Figura 27 - Interface do Editor

O editor exibe uma visão frontal, lateral e superior do esqueleto do personagem. Na parte inferior direita da tela encontra-se a área em que o usuário pode selecionar o *bone* que deseja movimentar, marcar se a animação da cabeça pode ter seu movimento definido pelo diretor ou se será preso ao personagem, adicionar *keyframes* e definir o intervalo de animação entre eles.

Como o editor não é o foco principal do trabalho, sendo implementado apenas para auxiliar nos testes que precisam ser feitos sobre animações, o mesmo possui algumas limitações se comparados a *softwares* de criação e edição de animações. O editor proposto, por exemplo, não possui cinemática inversa, isto é, as rotações só partem dos nós pais para os filhos, sendo que o contrário não acontece.

Além disso, para facilitar a implementação, as rotações aplicadas sobre um determinado *bone* são desfeitas se forem aplicadas rotações sobre nós ascendentes a ele e assim a animação deve ser construída da raiz em direção às folhas da árvore, que representa a estrutura de esqueleto. Outra limitação é que a animação poderá conter um número máximo de dez *keyframes*.

4.5 Simulando a Câmera

Para fazer uma boa simulação e visualização do que se está desenvolvendo é importante a implementação de uma classe “câmera” responsável pela visualização da cena.

Para este projeto em que a câmera é tratada como um elemento passivo, a mesma será apenas uma abstração da câmera do OpenGL e receberá parâmetros para seu posicionamento através do teclado e do mouse ou mesmo do próprio personagem que tem a capacidade de posicionar a câmera para tomadas específicas (*close* da face, visão em primeira pessoa, visão em terceira pessoa e visão sobre o ombro esquerdo e direito).

A câmera ainda pode receber um parâmetro para configuração da suavidade do seu movimento em relação ao movimento do personagem, evitando, por exemplo, que a câmera fique tremendo demais enquanto o personagem estiver em uma animação de corrida ou forçando esse tremor caso seja necessário aumentar a dramaticidade de uma cena de luta, por exemplo.

4.6 Simulando o Diretor

Como o diretor não é o foco deste trabalho, fazendo-se necessário apenas o conhecimento do seu funcionamento, o mesmo é simulado por entradas no teclado, que ativam comandos de troca de animação do personagem e dos posicionamentos e direcionamentos da câmera, isto é, o próprio usuário é o diretor.

4.7 Testes Sobre o Personagem

Foram criadas quatro animações para fins de testes. Uma delas apresenta o personagem parado, com um pequeno movimento do tórax, criando o efeito de respiração. A segunda animação é do personagem correndo. A terceira apresenta o personagem desferindo um golpe, numa simulação de luta. A última animação desenvolvida apresenta o personagem realizando movimentos de dança.

As cinco tomadas calculadas pelo personagem foram testadas com suas direções padrões e com o direcionamento, definido pelo ponto para onde o personagem deve olhar.

Todas as animações desenvolvidas apresentaram bons resultados desde que o diretor não proponha excessos como, por exemplo, rotações exageradas da cabeça do personagem.

5 CONCLUSÃO

Este trabalho de graduação apresentou a implementação de um personagem que possui esqueleto e que auxilia a câmera a fazer tomadas específicas de si mesmo. Esse tipo de personagem é uma alternativa para a dramatização de histórias que tenham o seu enredo dinâmico, como ambientes *storytelling*.

Com esta proposta reduz-se a dificuldade encontrada por um sistema diretor de posicionar a câmera levando em consideração a geometria do personagem. Como cada personagem detêm características próprias, tanto de forma como de comportamento, ele mesmo pode auxiliar a câmera a fazer tomadas de si, evitando que a câmera necessite conhecer a geometria de todos os tipos imagináveis de personagens, permitindo assim que sejam feitas inclusão de novos personagens nas histórias sem a necessidade de reimplementação do sistema.

Ao concluir o trabalho, os objetivos propostos foram satisfatoriamente atingidos, pois a implementação realizada consegue solucionar o problema proposto.

5.1 Trabalhos Futuros

A proposta apresentada serve como base para pesquisas futuras que possam evoluir cada vez mais o desenvolvimento na área de computação gráfica para storytelling. A seguir apresenta-se alguns possíveis aperfeiçoamentos que podem ser feitos em várias áreas abordadas neste trabalho:

1) Esqueleto com cinemática inversa: O esqueleto pode ser melhorado utilizando técnicas de cinemática inversa, isto é, permitir que um nó filho altere as rotações de um nó pai até um determinado ponto, recriando as limitações do movimento do corpo humano.

2) Editor tri-dimensional: O editor pode ser em três dimensões, sendo que para tanto poderia ser utilizado o OpenGL, o que geraria um resultado semelhante ao 3D Studio Max, porém especializado na modificação e animação do personagem proposto.

3) Aumento da quantidade de tomadas: O número de tomadas que o personagem auxilia a câmera pode ser acrescido. Além disso, o personagem

poderia fornecer parâmetros para a configuração da câmera, como a abertura da lente, por exemplo.

4) Testes de colisão: O personagem pode realizar testes de colisão com o meio e com outros personagens, para que o mesmo possa inclusive implementar animações de queda no estilo *ragdoll* (NATION MASTER*)*.

5) Animação comportamental: O personagem pode possuir sentimentos (EUPHORIA) como tristeza, alegria, euforia, desânimo, etc. E estes podem ser refletidos em sua animação, isto é, uma única animação de correr pode ser transformada em correr triste, correr alegre, correr desanimado, etc.

6) Maior quantidade de esqueletos: Pode-se construir outros esqueletos, com formatos que não sejam humanóides, mas também quadrúpedes, ou alados. Cada esqueleto possui necessidades especiais de posicionamento da câmera para serem solucionadas.

REFERÊNCIAS BIBLIOGRÁFICAS

AZEVEDO, Vinicis Costa de; POZZER, Cesar Tadeu. **Creating a Director for an Interactive Storytelling System**. In: VI Brazilian Symposium on Computer Games and Digital Entertainment, November 2007.

AZEVEDO, Vinicius; FAVERA, Eduardo; POZZER, Cesar Tadeu; CIARLINI, Angelo; FEIJÓ, Bruno; PASSOS, Erick; CLUA, Esteban. **Using Navigation Meshes to Improve Storytelling Dramatization.** In: VII Brazilian Symposium on Computer Games and Digital Entertainment, November 2008.

CAVAZZA, Mark; CHARLES, Fred; MEAD, Steven J.. **Emergent Situations in Interactive Storytelling**. In: SAC '02:PROCEEDINGS FO THE 2002 ACM SYMPOSIUM ON APPLIED COMPUTING, p.1080-1085, Madrid, Spain, 2002. ACM Press.

CHRISTIANSON, D. B., ANDERSON, S. E., HE, L., COHEN, M. F., SALESIN, D. H., WELD, D. S., 1996. **Declarative Camera Control For Automatic Cinematography**. In Proceedings
of AAAI ’96, 148-155.

CIARLINI, Angelo Ernani Maia. **Geração interativa de enredos**. PhD thesis, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1999.

EUPHORIA. **Natural Motion**. Disponível em: < http://www.naturalmotion.com/ euphoria.htm >. Acesso em: 3 nov. 2008.

GOMES, Jonas; VELHO, Luiz. **Computação Gráfica.** Volume 1. Rio de Janeiro: IMPA, 1997.

HE, Li-wei.; COHEN, Michael F. ; SALESIN, David. H.. **The virtual cinematographer**: a paradigm for automatic real-time camera control and directing. In: PROCEEDINGS OF THE 23RD ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES. ACM SIGGRAPH, volumen 30, p. 217{224, August 1996.

HENRY’S, David. **MD2 File Format**. In: David Henry’s homepage. 2004. Disponível em: < http://tfc.duke.free.fr/coding/md2-specs-en.html >. Acesso em: 22 out. 2008.

HENRY’S, David. The Quake II’s MD2 File Format. In: David Henry’s homepage. 2004. Disponível em: < http://tfc.duke.free.fr/old/models/md2.htm >. Acesso em: 22 out. 2008.

HERMANN, Rodrigo de Proença Gomes. **Controle Automático de Câmera em Ambientes Virtuais 3D**. 2005. 55f. Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

LASSETER, John. **Animation Tricks**. In: Tricks to Animating Characters with a Computer. Course 1 at SIGGRAPH, 1994. Disponível em: <http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/principles/lasseter_s94.htm >. Acesso em: 27 out. 2008.

MALVEZZE, Marcelo Curvacho. **Animação em Tempo Real com Keyframe.** 2004. 54f. Trabalho de Graduação – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2004.

MARTINS FILHO, Afonso Rodrigo de Figueiredo. **Estendendo o Logtell: Suporte a Ações entre Personagens Figurantes.** 2007. 58f. Trabalho de Graduação (Ciência da Computação) – Universidade Federal de Santa Maria, Santa Maria, 2007.

MATEAS, Michael; SENGERS, Phoebe. **Narrative Intelligence**. In: AAAI FALL SYMPOSIUM 1999, TECHINICAL REPORT, p. 1-10, Cape Cod. MA, November 1999. AAAI Press.

MATEAS, Michael; STERN, Andrew. **Architecture, authorial idioms and early observations of the interactive drama façade**. Technical report, School of Computer Science, Carnegie Mellon, University, Pittsburgh, PA, December 2002.

NATION MASTER. **Ragdoll Physics.** Disponível em: < http://www.nationmaster.com/encyclopedia/Ragdoll-physics>. Acesso em: 01 dez. 2008.

PIRES, Gabriel Valtes. **Dramatização 3D de Histórias Interativas Geradas Automaticamente.** XV Seminário de Iniciação Científica Puc-Rio, Rio de Janeiro, 2007.

POZZER, Cesar Tadeu. **Um Sistema para Geração, Interação e Visualização 3D de Histórias para TV Interativa**. 2005. 96f. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

SANGOI, Suzana Amaral. **Contribuindo no Realismo da Dramatização de Storytelling com a Criação de Personagens que Demonstrem Emoções Através de Expressões Faciais.** 2008, Trabalho de Graduação (Ciência da Computação) – Universidade Federal de Santa Maria, Santa Maria, 2008.

SILVA, José Carlos Tavares da. **Um modelo para avaliação de aprendizagem no uso de ferramentas síncronas em ensino mediado pela Web**. 2004 p.88-112. 160f. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro,, Rio de Janeiro, 2004.

STURMAN, David J**. A Brief History of Motion Capture for Computer Character Animation.** In: A Brief History of Motion Capture for Computer Character Animation. Course 9 at SIGGRAPH, 1994. Disponível em: <http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/motion_capture/history1.htm >. Acesso em: 27 out. 2008.

TINY XML. **Tiny XML**. Disponível em: < http://www.grinninglizard.com/tinyxml/>. Acesso em: 27 out. 2008.

UCLA LINUX USERS GROUP – MD3 File Format, 2006. Disponível em: <http://www.linux.ucla.edu/~phaethon/q3/formats/md3format.html>. Acesso em: 27 out. 2008.

WIKIPEDIA. **XML**. Disponível em: < http://pt.wikipedia.org/wiki/XML >. Acesso em: 27 out. 2008.