

Projeto de Computação Gráfica

Bruce Christopher Barrera

Email: bruce@acessobrasil.org.br

Leandro Almeida de Araújo

Email: leandro@acessobrasil.org.br

Objetivo

O projeto tem por objetivo transformar a Notação-Libras em uma Animação Real Time Libras utilizando das mais modernas tecnologias de Computação Gráfica. Essa tradução é um processo extremamente complexo por ser um processo não-linear, sem qualquer similar no mundo e referência literária a respeito por se tratar de um processo inédito.

O produto final do projeto será criar uma engine de animação 3D da Notação-Libras que poderá ser utilizada em outros diversos produtos e aplicações visuais, como por exemplo, softwares infantis, leitores de textos online, livros visuais para surdos e tradutores de línguas estrangeiras para a linguagem visual respectiva do país.

Após atingido esses objetivos, um plano futuro seria de estender a aplicação para geração de animação real-time baseado em texto obtido através de reconhecimento de voz.

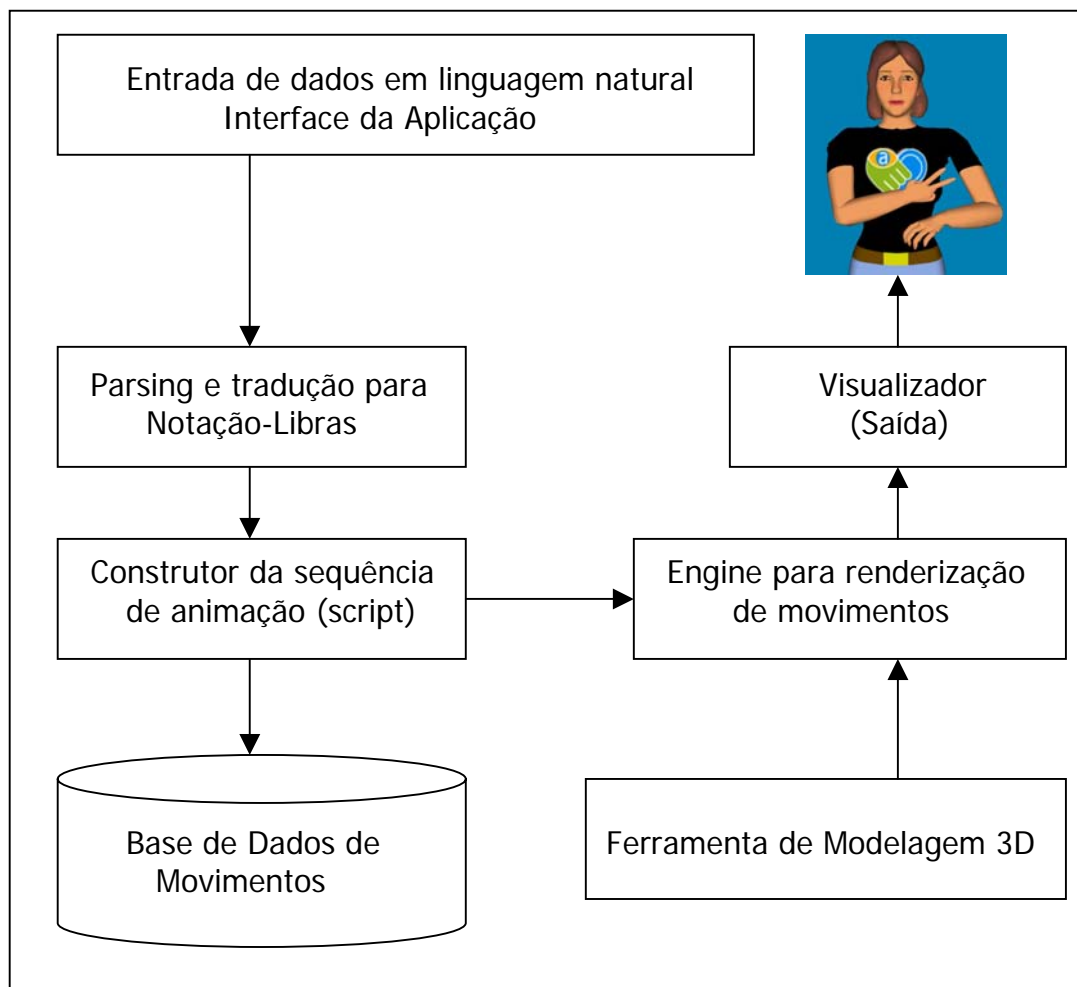


Arquitetura e conceitos do projeto

O projeto consistirá numa junção de várias partes interdisciplinares que serão desenvolvidas independentemente e depois integradas num todo. Ele é composto dos seguintes módulos:

- a. Entrada de dados em linguagem natural (Interface com usuário)
- b. Parsing e tradução da entrada para Notação-Libras
- c. Construtor da sequência de animação
- d. Base de dados dos movimentos
- e. Engine para renderização dos movimentos
- f. Criação do personagem 3D na ferramenta de modelagem
- g. Visualização (saída)

Essas partes podem ser facilmente visualizadas conforme o gráfico abaixo:



Resumo do funcionamento do projeto

Após a entrada da frase em linguagem natural, seja pelo usuário da aplicação ou então através de outro aplicativo (via arquivos texto), será feito o parsing e a tradução para *Notação-Libras* pelo módulo tradutor. A frase resultante será então passada ao módulo Construtor da sequência de animação (script) que se encarregará de analisar essa frase resultante, desmembrá-la em unidades menores e acessar o banco de dados de descrição de movimento em busca dos scripts pré-armazenados de animação.

Com o script montado, o Construtor envia esses script para a engine de renderização que fará o carregamento do personagem 3D e executando a(s) animação(s) descritas no script, resultando assim, no produto final.

Descrição detalhada de cada módulo do projeto

1) Entrada de dados em linguagem natural (Interface da Aplicação)

Entrada dada pelo usuário, digitando-se o texto, ou então através de entrada alimentada por outra aplicação. Como o projeto poderá ser embutido dentro de outras aplicações (uma aplicação que faça leitura de ebooks[7] por exemplo), ela poderá receber o texto a ser mostrado diretamente dessa aplicação através de arquivos texto.

2) Parsing e tradução da entrada para Notação-Libras

Trata-se de um módulo de encarregado de transformar o texto de linguagem natural para a Notação-Libras.

4) Construtor da sequência de animação

Esse módulo recebe a frase em Notação-Libras e busca a sequência de movimentos na base de dados, construindo a sequência da animação.

5) Engine[1] para renderização dos movimentos

Após definida a sequência de animação, o módulo faz todos os cálculos necessários para a renderização dos movimentos. Esses cálculos incluem a leitura do arquivo com o personagem 3D, a criação da lista de vértices e normais[2], faces dos polígonos, ordenação e interpolação[8] dos key-frames e as transformações e cálculos necessários de matrizes para a animação do esqueleto e dos seus vértices interligados.

6) Modelagem 3D

Criação da personagem 3D utilizando-se uma ferramenta de modelagem. Após criado o personagem, é feita uma exportação de suas informações utilizando-se um plugin para o software de modelagem.

7) Visualizador (Saída)

A saída propriamente dita, que consiste na renderização dos quadros da animação na tela, utilizando-se a API[4] OpenGL[9] para isso e as bibliotecas SDL e CAL3D.

Implementação

A conversão da Notação-Libras para uma Animação Real-Time é uma atividade complexa e, portanto, a implementação do projeto foi dividida em etapas. Abaixo segue um quadro com as etapas e as atividades desenvolvidas em cada uma:

ETAPA	PROGRAMAÇÃO	CRIAÇÃO DO MODELO 3D
1 ^a	<ul style="list-style-type: none">- Estudo e pesquisa de linguagens, templates, engines e APIs para utilização no projeto.- Exportação do modelo 3D para arquivo no formato da biblioteca CAL3D.- Criação de protótipo para apresentação.- Análise das atividades desenvolvidas e planejamento futuro.	<ul style="list-style-type: none">- Criação e modelagem do personagem a partir de uma foto real.- Texturização[10]- Criação do esqueleto.- Skinning[15]- Animação.- Exportação.
2 ^a	<ul style="list-style-type: none">- Documentação do sistema- Refinamento da abordagem utilizada para programação 3D.- Criação do engine OOP (Orientada a Objetos) de renderização.- Criação do construtor da sequência de animação.	<ul style="list-style-type: none">- Elaboração e definição do layout da personagem.- Estudo de proporções.- Detalhamento dos sinais, expressões e gestos utilizados na língua de sinais.- Criação da Identidade da personagem.- Criação de um esqueleto facial para expressões faciais.- Criação das configurações de mão utilizados na linguagem visual.- Criação das expressões faciais.
3 ^a	<ul style="list-style-type: none">- Documentação do sistema.- Desenvolvimento da interface com usuário.- Visualizador final.- Testes do sistema.	<ul style="list-style-type: none">- Desenvolvimento de um editor para os sinais primários da linguagem visual.- Desenvolvimento de um editor para expressões faciais.

Ferramentas e tecnologias utilizadas na parte gráfica

O desenvolvimento gráfico do projeto está dividido em duas frentes: a modelagem do personagem 3D e suas animações e a programação gráfica para geração dessas animações em Real-Time.

Como ferramentas para o desenvolvimento, está sendo utilizado o 3D Studio Max 5.1 para modelagem do personagem e para gerar suas animações.

Na parte de programação, a linguagem C++ usando o GNU/C++ como compilador, a API[4] OpenGL[9] para acesso ao hardware da placa de vídeo e as bibliotecas freeware CAL3D para animação esquelética de personagens e a biblioteca SDL para gerenciamento de eventos e janelas do Sistema Operacional.



3ds max™

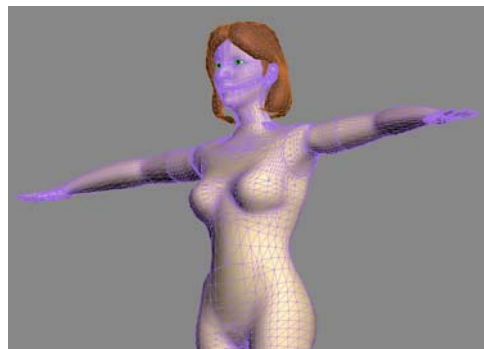


Descrição das atividades

Criação do modelo 3D:

a. Criação e modelagem do personagem.

Devido às especificações do projeto, que incluem animação facial e digital, os quais exigem não só uma representação razoavelmente acurada de um ser humano, como também um elevado grau de detalhismo do modelo, foi utilizada a técnica de modelagem por superfícies NURBS (Non-Uniform Relational Basis Spline), que permite grande precisão na geração de formas orgânicas, bem como controle sobre a qualidade e o grau de detalhe no resultado final.



Apesar da necessidade em se obter um modelo “leve” o suficiente para que fosse animado em tempo real de maneira convincente em um computador de capa cidade

média, a própria exigência do projeto, de representar de forma precisa um grande número de movimentos de forma natural, acarretou a elaboração de um modelo extremamente detalhado de um ser humano.

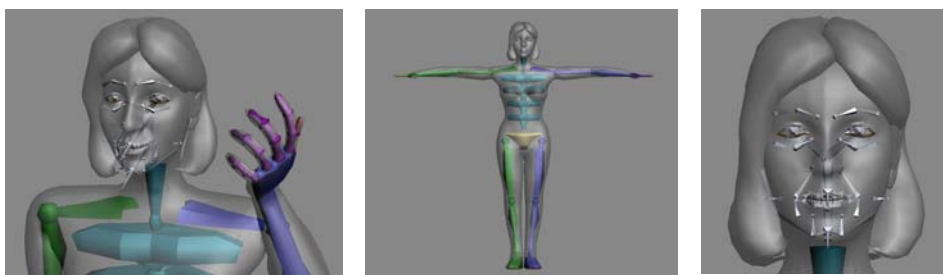
b. Texturização[10].



A fim de se obter um resultado que permitisse a exibição e a animação do modelo com qualidade e fluidez em equipamentos não necessariamente dotados de hardware gráfico de última geração, foi utilizada como textura para o mesmo uma imagem de 256 x 256 pixels - relativamente pequena, levando-se em consideração a capacidade dos melhores dispositivos gráficos do mercado, mas ainda assim suficiente para suprir as necessidades do projeto - e com qualidade de 24 bits (cerca de 16 milhões de cores). O tipo de material empregado não requer um dispositivo para renderização de *shaders* sofisticados.

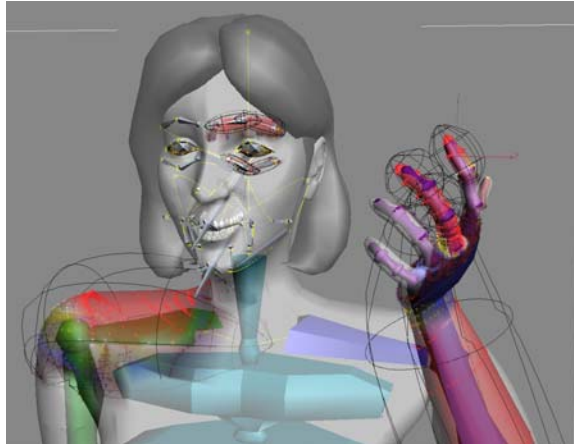
c. Esqueleto e aplicação de deformação esquelética ao modelo.

Ao contrário das técnicas tradicionalmente empregadas em animação facial, tais como o *morphing* - onde inúmeras instâncias do mesmo modelo são empregadas com ligeiras variações de forma a obter uma animação da superfície original interpolando-se entre essas variações, previamente definidas pelo modelador - a técnica empregada neste projeto foi a deformação esquelética, a mesma empregada na animação de todo o corpo do personagem. Entre as vantagens principais dessa alternativa estão a menor necessidade de memória, a possibilidade de se empregar recursos avançados e a possibilidade de se controlar cada elemento da face de forma independente. Assim é possível se contruir uma quantidade praticamente ilimitada de expressões, sem a necessidade de se alterar o modelo original; sem a necessidade de se definir cada expressão previamente, e ainda sem a necessidade de intervenção de um modelador.



O esqueleto foi feito com a ajuda da ferramenta Biped, que faz parte do pacote Character Studio da Discreet. Após contruída a estrutura do corpo, com todos os ossos que controlam a animação do corpo do personagem, utilizando-se o recurso Bones do próprio Discreet 3DSMAX foi construída a estrutura da face, que foi integrada ao esqueleto principal. Essa estrutura facial foi concebida de forma a se obter o máximo de precisão e quantidade de movimentos faciais com o menor número de elementos possíveis, já que cada elemento (osso ou músculo) empregado na animação acarreta um aumento considerável na carga de processamento. Após estabelecida toda a

estrutura do personagem, seu modelo foi associado a mesma através do processo de *skinning*, onde são definidos o grau e a natureza da influência de cada elemento do esqueleto na deformação da malha do modelo. Para esse processo, foi empregado a ferramenta *Physique*, também parte do Character Studio.



d. Exportação.

Foi utilizado o exportador fornecido junto com a biblioteca de animações de personagens CAL3D. O exportador cria um arquivo em formato próprio para:

- Esqueleto (.CSF)
- Malha (.CMF)
- Animação (.CAF)
- Material (.CRF)

Esses arquivos serão lidos pelas rotinas da biblioteca de animação Cal3D uma vez que são arquivos em formato proprietária da mesma.

Programação

a. Estudo e pesquisa de linguagens, templates, engines e APIs[4] para serem utilizados no projeto.

Era necessário definir quais tecnologia, modelos e conceitos a serem utilizados para o desenvolvimento do módulo de programação.

Primeiro, foi cogitado utilizar uma engine 3D que já tivesse a maioria das funções gráficas implementadas. Após vários estudos feitos, nenhuma engine mostrou-se suficientemente completa para tal. Foi decidido criar uma engine[1] proprietária.

A linguagem escolhida foi a C++ por ser uma linguagem extremamente rápida, flexível, de amplo suporte pela comunidade internacional e também por possuir uma grande quantidade de bibliotecas freeware para computação gráfica.

Optou-se por utilizar a API[4] gráfica OpenGL[9] para acesso ao Hardware gráfico por ser uma API[4] amplamente difundida e suportada pela comunidade internacional de desenvolvedores 3D, possuindo atualizações periódicas e muita

documentação disponível. Em contrapartida com sua concorrente, o DirectX, tem como vantagem também ser multi-plataforma.

Para gerenciamento dos eventos do Sistema Operacional e criação da janela da aplicação foi escolhida a biblioteca SDL. Criada por Sam Lantinga, ela encapsula todas as chamadas diretas ao sistema operacional permitindo criar janelas, tratar eventos sem precisar chamar funções específicas do SO. Assim, se a biblioteca for portátil, isto é, existir versões dela para outras plataformas, o código poderá ser apenas recompilado na outra plataforma e já estará pronto para ser rodado.

Então, essa combinação (Engine + OpenGL + SDL + C++) requereu um aprimoramento nas habilidades de programação 3D. Foram feitos muitos testes e estudos até chegar o nível desejado.

b. Exportação do modelo 3D para arquivo em formato proprietário

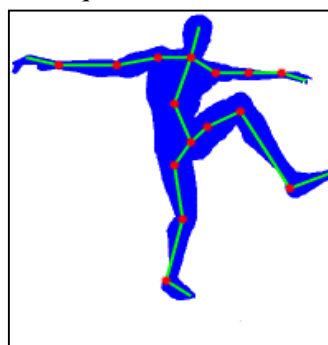
Após modelado o personagem 3D, faz-se necessário obter diretamente da ferramenta de modelagem, todas as informações necessárias para poder animar o personagem. Para isso era imprescindível a construção de um exportador que captasse essas informações e gerasse um arquivo com elas. Para nossa felicidade, a biblioteca de animação de personagens que decidimos utilizar, disponibilizava tal exportador e que atendia as nossas necessidades! Por isso foi adotado o exportador do Cal3D no projeto.

c. Engine de renderização e animação de personagens

A engine está totalmente projetada para seguir o paradigma da Orientação a Objetos. Ela está dividida em pequenos “módulos” que são classes independentes que realizam alguma tarefa específica da engine. Podemos citar a classe de Render que realiza toda a plotagem gráfica na tela e o gerenciamento da janela, a classe de Input que se encarrega em processar os eventos de entrada do teclado e mouse, a classe Texture Manager que guarda a lista de todas as texturas (imagens) carregadas na memória, a classe de Log que provê um log dos eventos e erros para um arquivo texto, etc...

A animação do personagem está sendo feita utilizando a estrutura de animação esquelética onde a malha de polígonos é deformada pelos ossos relacionados. Para isso temos a classe Model que guarda as informações e o estado do personagem 3D, e uma classe Player que controla o tempo e a animação.

Para se conseguir essa animação foi necessário um estudo matemático principalmente para realizar as rotações das juntas. Para isso utilizou-se quaternions que produzem um resultado melhor do que se fossem usados ângulos de Euler.



Para carga dos arquivos e reprodução de animações que não são dinâmicas e expressões faciais, utilizou-se a biblioteca CAL3D que atendeu satisfatoriamente as nossas necessidades.

O movimento dinâmico dos braços foi obtido através do uso da Cinemática Inversa, o mesmo princípio utilizado em projetos de Robótica. Dessa forma foi possível

movimentar o braço para qualquer região do espaço (que estava previamente delimitada através de pontos de controle) dinamicamente.

Glossário

1. <i>Engine</i>	Parte central de uma aplicação, centraliza todas as operações realizadas. Pode ser facilmente reutilizado em outras aplicações.
2. <i>Normal</i>	A normal é um vetor que é perpendicular a um plano dado. Geralmente perpendicular ao plano definido pela face de um polígono.
3. <i>On-the-fly</i>	Instantaneamente, dinamicamente.
4. <i>API</i>	Application Programming Interface, consiste em funções disponíveis para acessar funcionalidades específicas de um hardware ou software.
5. <i>Box modeling, Editable Mesh, Editable poly</i>	Técnicas de modelagem do 3D Studio MAX.
6. <i>Renderização</i>	Gerar a imagem final dos quadros da animação.
7. <i>eBook</i>	Livro eletrônico
8. <i>Interpolação</i>	Gerar valores intermediários num tempo t dado dois valores, inicial e final.
9. <i>OpenGL</i>	API gráfica de acesso direto ao hardware da placa de vídeo.
10. <i>Texturização</i>	Anexar texturas (imagens) ao modelo 3D
11. <i>Malha</i>	Conjunto de polígonos que constituem o modelo 3D e são anexadas ao esqueleto.
12. <i>Low Poly</i>	Baixa resolução de polígonos
13. <i>Medium Poly</i>	Média resolução de polígonos
14. <i>High Poly</i>	Alta resolução de polígonos
15. <i>Skinning</i>	Anexar a malha ao esqueleto.
16. <i>I/O</i>	Entrada e saída

Referências

- [1] Mark DeLoura, *Game Programming Gems 1 & 3* (Charles River Media).
- [2] Mark J. Kilgard, *Textos diversos* (Nvidia Developer Site, <http://developer.nvidia.com>).
- [3] Manson Woo, Tom Davis, Jack Neider, *OpenGL RedBook* (Addison Wesley Publishing).
- [4] Alan & Mark Watt, *Advanced Animation and Rendering Techniques* (Addison Wesley Publishing).

- [5] Nik Lever, *Real-Time 3D Character Animation* (Focal Press).
- [6] Herbert Schildt, *C Completo e Total* (Makron Books).
- [7] André Duarte Bueno, *Programação Orientada a Objetos com C++* (Novatec Editora)
- [8] Game Tutorials, <http://www.gametutorials.com>
- [9] Neon Helium, <http://nehe.gamedev.net>
- [10] Auslan Tuition System, <http://www.cs.uwa.edu.au/~jasonw/AuslanSystem>
- [11] CAL3D Project Page, <http://cal3d.sourceforge.net>
- [12] SDL Project Page, <http://www.libsdl.org>
- [13] Thetos System – Polish Translator, <http://sun.iinf.polsl.gliwice.pl/sign/>