

Aspectos Fundamentais da Criação de Jogos em Shockwave 3D

RANGEL JUNGLES DOS SANTOS
LAI / rangel_jungles@yahoo.com.br

ANDRÉ LUIZ BATTAIOLA
LAI / albattaiola@ufpr.br

RAFAEL PEREIRA DUBIELA
LAI / rafaeldubiela@hotmail.com

LAI-Laboratório de Animação Interativa
Departamento de Design – Universidade Federal do Paraná
Rua General Carneiro, 460, 8º andar, sala 814, Curitiba, Paraná

Resumo

O formato Shockwave 3D é uma das tecnologias de publicação de conteúdos tridimensionais mais promissoras existentes hoje no mercado. Sua implementação independente de plataforma, bem como a sua natural vocação para a publicação na Internet são, sem dúvida, seus principais atrativos. Sob essa perspectiva, indagações a respeito da viabilidade da construção de jogos nessa tecnologia e do grau de dificuldade encontrado nessa tarefa, têm sido feitas. Assim como especulações a cerca das suas limitações, possibilidades e peculiaridades. Neste contexto, o presente artigo expõe as informações a respeito dessa tecnologia obtidas durante a produção da fase tridimensional do Edugraph: Jogo educacional voltado a disseminação dos conceitos fundamentais da Computação Gráfica, cuja extensão foi realizada no Projeto de Pesquisa LudicLearning. Tais informações se restringem somente aos aspectos mais básicos da produção de um jogo nessa tecnologia, tais como, a exportação de modelos e cenários, movimentação de personagens, manipulação de ciclos de animação e detecção de colisão. Sem menção a quaisquer outras características da produção de um jogo. Sem focar aspectos mais avançados como a implementação de ambientes multi-player e de inteligência artificial.

Palavras-chaves: *Shockwave 3D, Jogos para Web, Jogos Educacionais.*

1. Introdução

O *Edugraph* é um jogo educacional voltado ao ensino de conceitos fundamentais da Computação Gráfica [1]. Ele é composto por fases bi e tridimensionais, originalmente implementadas na plataforma *Windows*. Visando melhorar a sua divulgação e também desatrelá-lo da plataforma (possibilitando assim a sua aplicação mais efetiva no ensino a distância), uma versão para *Web* foi idealizada e implementada no *LudicLearning*¹.

Essa nova versão é composta, a exemplo da original, por fases bi e tridimensionais. Sendo que este artigo relata mais especificamente os resultados obtidos na criação da fase tridimensional do jogo, desenvolvida na tecnologia *Shockwave 3D*. Ela é caracterizada pela existência de um enredo, no qual um personagem (*Edugraph*) se vê preso em uma sala (denominada cubo) da qual só poderá sair após ordenar corretamente, por meio de um painel, a inserção das caixas (denominadas chaves) nos seus respectivos encaixes (denominados fechaduras).

Para a sua realização foram estudadas:

¹ LudicLearning - projeto de pesquisa financiado pelo CNPq (Processo número 552173/2002-9).

- Características gerais do *Director Studio MX* e da sua linguagem de programação, Lingo,
- Exportação de modelos gerados no *3D Studio Max 5.1* para o formato *Shockwave 3D*, por meio do Plasma 1.0;
- Produção de cenas em terceira pessoa compostas por cenários e personagens tridimensionais;
- Associação de eventos de teclado a modificações na cena;
- Manipulação de ciclos de animação;
- Detecção de colisão.

As informações e as conclusões obtidas com esse estudo são descritas nos tópicos a seguir.

2. Características gerais do *Director Studio MX*

Os jogos convencionais (produzidos para execução em sistemas específicos) são compostos por três partes principais: roteiro, interfaces visual e auditiva e motor [2]. A exemplo deles, a fase tridimensional do jogo Edugraph desenvolvida para *Web* foi criada a partir da elaboração de um roteiro e de interfaces visuais e auditivas. Contudo, ao invés de um motor convencional, foi utilizado um software originalmente projetado para a elaboração de interfaces interativas para CDs e DVDs, além de sites e quiosques na Internet [3].

O *Macromedia Director Studio* dispõe de recursos para a manipulação de uma grande variedade de mídias, tais como animações e multimídia (filmes em flash, gifs animadas e apresentações do *Microsoft Power Point*), imagens (bmp, gif, jpeg, lrg – xRes -, *Photoshop 3.0* ou superior, *MacPaint*, png, tiff, pict e targa), arquivos multi-imagem (flc e fli do Windows e Pics e Scrapbook do Macintosh), sons (aiff, wav, *mp3 audio*, *Shockwave Audio*, *Sun Au* sem compressão e ima comprimido), vídeos (*QuickTime 2, 3, 4 e 6*, avi e *RealMedia*), textos (rtf, html e ASCII), paletas (PAL e *Photoshop CLUT*) [4].

A manipulação de conteúdos em *Shockwave 3D*, desta forma, é apenas uma das funcionalidades deste software, que também possibilita a inserção de controles típicos de

navegação na internet e criação de formulários dinâmicos, criação de aplicações multi-usuário, dentre outras coisas.

A Lingo.

O *Director Studio* possui uma linguagem de programação orientada a objetos denominada Lingo, a qual, a exemplo da interface gráfica do software é inteiramente baseada na abstração “filme”. Tal abstração norteia o desenvolvimento do conteúdo, com isso a produção de códigos se dá em função da necessidade de tratamento de eventos típicos de um filme. A exemplo, “entrada em quadros” (*EnterFrame*) e “término do filme” (*EndMovie*).

Outra característica importante desta linguagem é que ela possui duas sintaxes distintas. Uma mais parecida com a linguagem natural, denominada verbal e outra mais próxima das linguagens de programação convencionais, denominada sintaxe ponto [5]. A sintaxe verbal é recomendada para a fase de aprendizado da criação de *scripts* de controle, enquanto que a sintaxe ponto é recomendada para programação avançada. A produção de códigos com a utilização simultânea de ambas as sintaxes é permitida sem qualquer ressalva. Contudo, tal prática não é recomendada por questões de legibilidade.

Além disso, em Lingo os códigos produzidos podem ser divididos em quatro tipos básicos de *scripts*: de filme, de comportamento, anexados a membros de elenco e parentais. Cada qual enfocando alguns aspectos do “filme”. Não existe uma regra formal para a divisão de atribuições para os diferentes tipos de *script*. Há somente a recomendação de que as funcionalidades sejam agrupadas de acordado com a sua origem.

Com isso os “scripts de filme” realizam costumeiramente tarefas relacionadas com o controle do fluxo de execução do “filme” (determinando quais quadros serão exibidos a cada instante de tempo entre outras coisas). Enquanto que os “de comportamento” realizam o controle das propriedades dos elementos presentes na cena a que estão associados. Os “anexados a membros de elenco” controlam as propriedades das mídias

presentes no elenco. E os parentais são destinados à definição de objetos.

Apesar dos diferentes enfoques, os diversos tipos de *script* da Lingo possuem a mesma estrutura de organização. As variáveis globais (antecedidas pela palavra reservada *global*) são declaradas logo no topo do arquivo, antes dos manipuladores de eventos (*handlers*)². A declaração deste tipo de variável é necessária somente para a definição do seu escopo, pois a Lingo não é uma linguagem tipada. Esta característica aliada ao fato de que não há diferenciação na escrita com maiúsculas e com minúsculas, torna a linguagem imprópria para a criação de códigos extensos e complexos.

A definição das classes de objetos, bem como das superclasses, feita nos *scripts* parentais segue essa mesma regra. As variáveis globais do *script*, designadas agora pela palavra reservada *property* (equivalentes às propriedades das classes da orientação à objetos), são declaradas logo no início do arquivo, antes dos *handlers* que nesse caso equivalem aos métodos do objeto.

O construtor da classe não é diferenciado sintaticamente dos demais métodos. A única diferença é a utilização do comando *new* na descrição dos parâmetros do *handler* para a alocação de uma nova instância da classe. A criação de um método destrutor não é necessária, pois a atribuição do valor *void* à um objeto ocasiona a sua desalocação. A utilização de uma superclasse para a definição de uma classe se dá com a criação de uma variável global especial, designada pelo nome reservado *ancestor*.

Devido à indistinção entre métodos (*handlers*) públicos e privados e também devido à falta de restrição de acesso às propriedades dos objetos, o encapsulamento provido pela Lingo é ineficaz, dado que é possível se enviar mensagens por métodos privados dos objetos e também se alterar diretamente as propriedades da classe, sem o envio de mensagem. Por essa razão, alguns consideram que a Lingo não implementa a orientação à objetos no sentido estrito do

conceito. Porém se consideradas algumas práticas de programação (tais como a de não utilização dos recursos que atentam contra o encapsulamento e herança), a orientação a objetos existente na Lingo pode ser utilizada sem maiores problemas.

Em virtude da inexistência da declaração de variáveis locais, da não tipificação das variáveis e da possibilidade da utilização da escrita em maiúsculas e minúsculas, a Lingo não é recomendada para a elaboração de jogos com controle muito complexos. Pois a depuração e a manutenção dos códigos produzidos sob essas condições, podem depender muito tempo. Entretanto, a construção de jogos com essa característica não é impossível, vez que as dificuldades encontradas podem ser superadas por fases de depuração maiores e mais elaboradas e pela geração de documentação adequada às necessidades do processo de manutenção.

3. Exportação de Modelos

Os modelos e cenários da fase tridimensional do *Edugraph* foram construídos utilizando-se o *Discreet 3D Studio Max 5.1*, software de modelagem tridimensional conhecido por prover muitos recursos para o desenho e a animação de cenas. A escolha desta aplicação, que não possui a funcionalidade de exportação de cenas para o formato *Shockwave 3D*³ implicou na necessidade de adoção de um conversor de formato. Para essa tarefa foi empregado o software *Discreet Plasma 1.0*, aplicação de modelagem tridimensional, capaz de importar cenas criadas no *3D Studio Max 5.1* e exportar conteúdos para *Shockwave 3D*, apesar de possuir recursos de edição limitados.

Com isso a etapa de modelagem do conteúdo que idealmente deveria ser realizada diretamente em *Shockwave 3D*, foi realizada em duas fases distintas: a modelagem propriamente dita (realizada no *3D Studio Max*) e a exportação para *Shockwave 3D* (por meio do *Plasma 1.0*).

Devido a alguns problemas detectados no *Plasma*, a exportação do personagem e do

² A declaração de variáveis globais pode também ser feita no corpo dos *handlers* que às utilizam, porém essa não é uma prática convencional.

³ A versão 6 do *3D Studio Max* contém recursos para a exportação de cenas para *Shockwave 3D*.

cenário foi problemática. Isso aconteceu não somente porque a exportação de animações de personagens complexos (compostos por esqueleto, malhas de polígonos e texturas), não pode ser realizada sem a criação de um agrupamento único, mas, principalmente porque o *Plasma* não é capaz de importar animações e alguns parâmetros de finalização de cenas (*rendering*) do *3D Studio Max*. Em razão disso, a produção de animações em *Shockwave 3D* é um processo trabalhoso, visto que os modelos são criados no *3D Studio Max* enquanto que a adição de movimentos é feita no *Plasma* [6].

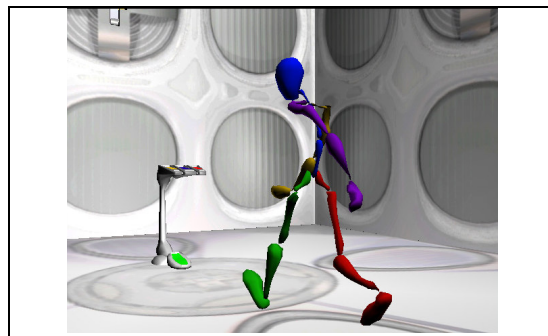
O fato de o *Plasma* possuir ferramentas para a edição de conteúdo tridimensional nem sempre é suficiente para sanar esse problema, posto que os recursos disponíveis nesse software são limitados e podem não permitir a modelagem de certos objetos, sendo necessário assim o emprego do *3D Studio Max*. Além disso, a utilização de algumas dessas ferramentas pode ocasionar anomalias nos conteúdos exportados. Este problema foi detectado quando da exportação do ciclo de caminhada do personagem. Situação em que o simples fato de o braço e da perna esquerda terem sido produzidos a partir da cópia e do espelhamento do braço e da perna direita, respectivamente, causou a distorção completa desses membros na animação exportada.

Em vista destes fatos, pode-se dizer que a exportação dos modelos gerados no *3D Studio Max 5.1* para o formato *Shockwave 3D*, por meio do *Plasma 1.0* é uma etapa crítica do processo de criação de um jogo nessa tecnologia, pois para que ela seja realizada de forma bem sucedida é necessária a observação de um conjunto razoável de condições, como aquelas citadas anteriormente, sendo que muitas não são facilmente compreendidas.

4. Produção de cenas tridimensionais

A fase tridimensional criada é composta por um conjunto de elementos, quais são: uma sala fechada, um painel, cinco encaixes, uma caixa e um personagem. A seguir há uma figura que demonstra o ambiente citado:

Figura 1 - fase tridimensional



A criação da cena tridimensional que comporta essa fase poderia ser feita com a modelagem em um único arquivo de todos esses elementos, porém, para facilitar a administração do conteúdo e garantir que este sempre será exibido numa mesma condição inicial, optou-se pela divisão dos modelos em arquivos distintos e a geração da cena dinamicamente. Assim, a sala, os encaixes e a caixa foram modelados em um único arquivo, enquanto que o painel e o personagem ganharam arquivos próprios.

Isso é possível graças a recursos disponíveis no *Director Studio MX* de criação dinâmica de mídias e de cópia de modelos entre arquivos. O trecho de código abaixo demonstra como elas foram processadas a fim de gerar a cena apresentada.

Fragmento de Código 1 – criação da cena

```
-- Criando a nova cena
cena = new(#shockwave3d, member(1))
cena.name = "Cena Principal"
...
-- Criando referências para os membros de elenco que
-- contem modelos que serão usados na cena
cubo = member("Sala")
edu = member("Personagem")

-- Copiando os modelos para a cena:
repeat with i = cubo.model.count down to 1
  cena.cloneModelFromCastMember →
(cubo.model(i).name, cubo.model(i).name, cubo)
...
end repeat
```

O emprego desse método para a criação da cena revela uma característica interessante do *Director*: a possibilidade de se produzir cenas inéditas a partir da utilização de modelos existentes em bibliotecas de conteúdo. Isso é especialmente interessante quando da elaboração de jogos com muitas fases,

cenários e personagens. Nessa situação, com a aplicação deste método, não seria necessário dispor de arquivos com as fases completas, mas somente dos elementos necessários para criá-las. Eles poderiam ser criados isoladamente e reunidos em bibliotecas de modelos, a serem utilizadas conforme a demanda de apresentação.

5. Associação de eventos de teclado à modificações na cena

Para a manipulação do personagem na cena criada, foi utilizada uma interface composta por dez teclas. Quatro delas destinadas à movimentação do personagem no mundo tridimensional, outras quatro à rotação da câmera e duas para o acionamento do painel (uma para ordenar o acionamento dos botões e outra para o acionamento do pedal). A associação das mudanças na cena com os respectivos eventos do teclado foi realizada por meio da verificação do estado das teclas a cada “quadro” do “filme”, em um algoritmo de *polling*.

Tal implementação não é a ideal, mas foi empregada porque o tratamento de eventos de teclado não suporta o pressionamento simultâneo de duas ou mais teclas. Situação bastante comum na movimentação do personagem no mundo tridimensional.

Com isso, a cada início de “quadro” uma rotina de tratamento da interface é executada. Nela o estado de cada tecla da interface é testado, determinando a execução do código de tratamento adequado. A seguir há dois trechos de código que demonstram a chamada e a rotina de tratamento da interface criada:

Fragmento de Código 2 – tratamento de eventos do teclado

```
-- Processamento realizado a cada frame.
On enterFrame
  -- Chamada da rotina de tratamento da interface.
  CapturarComandos()
  ...
end enterFrame
...
-- Handler responsavel por capturar e tratar os
-- comandos do teclado.
On capturarComandos
  if the keyPressed <> EMPTY then
    -- Se a tecla "A" for pressionada, rotacionar o
    -- conjunto (personagem e câmera) para a
    -- esquerda
```

```
if keyPressed("A") then
  cena.group("Conjunto").rotate(0,0,1,#self)
  transf_inversa = #rot_dir
end if

-- Se a tecla "D" for pressionada rotacionar o
-- conjunto (personagem e câmera) para a direita.
If keyPressed("D") then
  Cena.group("Conjunto").rotate(0,0,-1,#self)
  Transf_inversa = #rot_esq
end if
...
end capturarComandos
```

A impossibilidade da utilização do evento de teclado para o tratamento da interface criada não constitui um empecilho de grandes proporções, posto que não é complexo substituí-lo por um código de *polling*. Além disso, o processamento adicional originado dessa substituição não implica em perda significativa no desempenho da aplicação.

6. Manipulação dos ciclos de animação

Conforme estabelece o roteiro, o personagem possui liberdade para executar 5 ações distintas. São elas: andar para frente, andar para trás, permanecer parado à espera de comandos, esticar o braço direito para pressionar um dos botões do painel e esticar a perna direita para pisar no pedal do painel.

Foram criadas animações cíclicas a fim de se representar cada uma destas ações. Com isso, a ação de andar para frente, por exemplo, é composta por um ciclo de 28 quadros, nos quais o personagem descreve todos os movimentos de uma caminhada.

A manipulação adequada dessas animações é de fundamental para se obter um jogo de boa qualidade. Por essa razão a preocupação de se atribuir ao estado momentâneo à animação correspondente é especialmente importante. No desenvolvimento da fase tridimensional do *Edugraph*, optou-se, a exemplo da criação da cena, por gerar uma biblioteca de conteúdo, a qual contém todos os ciclos de animação do personagem (*walk cycle*) dispostos em diferentes arquivos. Isso é possível porque da mesma forma que os modelos, elas podem ser excluídas e copiadas entre arquivos *Shockwave 3D*. Essa situação possibilita a substituição das animações do personagem conforme a

necessidade, sem comprometer o desempenho da aplicação.

A comutação utilizada não possui suavização, apesar da possibilidade dada no *Director* de se mesclar animações. Nesse processo são produzidos movimentos híbridos a partir da combinação linear dos conteúdos no fim e no início das animações que estão deixando e começando a serem executados, respectivamente. Para tanto é empregado um fator que estabelece a contribuição que os movimentos de cada animação têm sobre a mescla, além do tempo em que ela está em vigor.

Outro recurso disponível, cujo uso visa tornar a movimentação dos personagens mais próximo do real, é o mapeamento de movimentos. Nele movimentos compostos são produzidos a partir de movimentos básicos com a designação de quais são as origens dos movimentos dos diversos “ossos” do “esqueleto” do personagem. Um exemplo simples da aplicação desse recurso é a produção de um movimento de “correr atirando” à partir dos movimentos “correr” e “atirar” existentes. Ela pode ser feita em tempo de execução, conforme a demanda, sem a necessidade da modelagem prévia da ação.

Tais recursos permitem a produção de ações mais harmoniosas para os personagens, na medida em que removem as transições abruptas entre animações. Incoerências como a de um personagem não poder sentar-se e olhar para a esquerda simultaneamente, sendo que pode fazê-lo isoladamente, também são eliminadas, tornando mais reais as ações dos personagens. O que corrobora em muito o emprego de animações por esqueletos.

7. Detecção de Colisão

Em *Shockwave 3D* o tratamento das colisões entre modelos pode ser realizado de duas formas: pela escrita integral de códigos de tratamento ou pela utilização da resolução automática. Destas, a escrita de códigos certamente é a opção mais trabalhosa, posto que é necessário controlar os mais diversos aspectos da colisão, contudo, existem algumas funcionalidades que facilitam a elaboração destes códigos. Entre elas, podem-se mencionar as que identificam quais modelos

colidiram, os pontos de contato entre eles e a normal da colisão.

A resolução automática ao contrário requer bem menos trabalho do programador, pois realiza a interrupção da transformação geométrica que causou a colisão sem que seja necessária a escrita de código para isso. As únicas tarefas do programador na sua utilização, além de adicionar e habilitar o modificador colisão, são determinar a “embalagem” envolvente do modelo (a qual pode ser caixa, esfera ou a própria malha do objeto) e o *handler* que contém o tratamento da colisão, que nesse caso se resume a determinar qual será o comportamento dos modelos envolvidos no evento.

Na fase produzida foi utilizada a resolução automática em conjunto com o controle de execução da animação do personagem. Isso foi necessário porque a resolução automática não pára a execução da animação do modelo que colidiu. Como consequência, novas colisões podem acontecer enquanto o tratamento (reposicionamento do personagem no mundo 3D) está sendo feito. Nesse caso, em virtude do reposicionamento do personagem ser realizado por transformações geométricas, o tratamento dado será falho, vez que o personagem não poderá se afastar do objeto com que colidiu, permanecendo assim grudado a ele.

A utilização do controle da animação sana esse problema ao “paralisar” o personagem enquanto o tratamento da colisão não foi concluído, dado que os movimentos oriundos da animação causadores das novas colisões deixam de ser executados.

Desta forma pode-se dizer que a detecção e o tratamento das colisões em *Shockwave 3D* é relativamente simples, porém deve-se ressaltar o elevado custo computacional dessa operação. Na aplicação desenvolvida, a queda de desempenho foi bastante acentuada com a utilização desses recursos. Observou-se também que esta queda tende a ser ainda maior com o aumento do número e da complexidade dos modelos presentes na cena.

Por outro lado, um ponto positivo observado foi o fato do resultado ter sido satisfatório com a utilização de todos os tipos de “embalagens de modelos”. A diferença de desempenho observada na aplicação de cada

uma delas não foi muito significativa, apesar de a complexidade da utilização da malha ser certamente maior do que a da esfera e da caixa envolvente.

8. Conclusão

Um estudo realizado a fim de se identificar as possibilidades e as dificuldades da utilização da tecnologia *Shockwave 3D* para a elaboração de jogos para internet, foi apresentado nesse artigo. Nele, além de uma descrição da ferramenta e da linguagem de programação utilizada para a formulação do jogo, 5 aspectos básicos foram abordados dando-se ênfase as suas características e aos seus problemas.

As principais conclusões obtidas a respeito de cada um desses aspectos e também da ferramenta e da sua linguagem de programação foram as de que a exportação de modelos gerados no *3D Studio Max 5.1* para o formato *Shockwave 3D*, por meio do *Plasma 1.0* é um processo complicado, apesar da consistência do formato e da qualidade do software de modelagem utilizado. São necessários conhecimentos avançados a respeito das limitações do software de conversão de formato para que a conversão seja bem sucedida. Razão pela qual essa é, talvez, a etapa mais crítica da elaboração de um jogo.

A produção de cenas em terceira pessoa compostas por cenários e personagens tridimensionais é uma etapa relativamente simples da criação do conteúdo. Ao contrário da exportação, ela não reserva qualquer segredo ou dificuldade.

A possibilidade da criação de bibliotecas de mídias torna a elaboração do jogo mais fácil e eficiente.

Os recursos disponíveis para a associação de eventos de teclado às modificações na cena, ao contrário, não são tão funcionais. Por isso é necessária uma adaptação no código de controle para a sua realização adequada. Contudo, como tal adaptação pode ser facilmente implementada sem aumento do custo computacional, a complexidade dessa etapa pode ser considerada baixa.

Não tão simples, a manipulação de ciclos de animação não chega a ser uma etapa crítica da criação do jogo. A dificuldade encontrada

na sua implementação é alta, mas em função dos recursos disponíveis para a sua estruturação, não se pode dizer que ela é a mais complicada.

A detecção de colisão não representa problema considerável à produção do jogo, desde que se entendam as suas peculiaridades.

Apesar do objetivo central do *Director* não ser a produção de um jogo, ele é consideravelmente funcional para esta aplicação. A sua interface, bem como a abstração que a norteia se enquadra perfeitamente nas necessidades do programador para a geração de um conteúdo lúdico, animado e interativo. A sua linguagem nativa, porém, possui um conjunto de características indesejáveis para a realização dessa tarefa. O fato de não ser tipada, de não distinguir entre maiúsculas e minúsculas, e, principalmente, apresentar vulnerabilidades no processo de orientação a objetos, a tornam imprópria para a criação de aplicações de grande porte ou mesmo de enredos complexos. Ressalte-se, porém, que esse quadro não impede a elaboração de jogos com tais características. Antes disso, apenas tornam mais difícil a realização dessa tarefa.

Com base nas informações acima descritas, pode-se concluir que a produção de jogos em *Shockwave 3D* com a utilização do trio, *3D Studio Max 5.1*, *Plasma 1.0* e *Director Studio MX*, é possível, no entanto, com a ressalva de que por questões de desempenho e de complexidade de implementação, a criação de conteúdos mais sofisticados não é recomendada, pois os resultados obtidos podem ficar aquém do esperado.

9. Referências

- [1] Battaiola, A. L.; Dubiela, R. P.; Martins, F. E.; Vieira, T. V. & Santos, R. J. d. "Teaching Computer Graphics in a Ludic Learning Environment". World Congress on Engineering and Technology Education – WCETE. São Paulo, SP. março de 2004.
- [2] Battaiola, A. L.; Elias, N. C.; Domingues, R. d. G.; Assaf, R. & Ramalho, G. L. "Desenvolvimento da Interface de um Jogo Educacional com Base em Interfaces de Jogos". V Symposium on Human Factors in Computer Systems da SBC, 2002.
- [3] Alleson, A.; Baumann, J.; Bhangal, S.; Blaha, T.; Cameron, A. & outros. "Director 8.5 Studio". Livro. agosto de 2001
- [4] Armstrong, J.; Brown, G.; Gowin, S. & Statler, T. "Macromedia Director MX – Using Director MX";

Manual. Macromedia, Inc. San Francisco, CA. dezembro de 2002.

[5] Armstrong, J.; Brown, G.; Gowin, S. & Statler, T. "Macromedia Director MX - Lingo Dictionary"; Manual. Macromedia, Inc. San Francisco, CA. dezembro de 2002.

[6] 'Plasma Tutorial – Discreet'. Manual. maio de 2002

[7] The artists, animators, and, programmers of the epic software group. "Macromedia Director Game Development – From Concept To Creation". Livro. Epic Software Group,

[8] Macromedia, Inc. <http://www.macromedia.com> (17/08/2004)

[9] Discreet, Inc. <http://www.discreet.com> (12/08/2004)

[10]Epic Software Group, Inc. <http://www.epicsoftware.com/> (05/02/2004)