Tela de Alfinetes Virtual

Rui Varela[1] e J. M. S. Dias[2,3]

[1]*Av. General Roçadas, 173 4º Esq., 1170-160 Lisboa, Portugal*

[2]*MLDC, Microsoft Language Development Center, Lisboa, Av. Prof. Doutor Aníbal Cavaco Silva, Edifício Qualidade C1, C2, TagusPark, 2744-010 PORTO SALVO, Portugal, http://www.microsoft.com*

[3]*ADETTI/ISCTE, Edifício ISCTE, Av. das Forças Armadas, 1600-082 Lisboa, Portugal, http://www.adetti.pt*

Sumário

Este artigo descreve um sistema de realidade virtual 3D não imersiva, inspirada na tela de alfinetes tradicional de Alexandre Alexeieff. *O sistema disponibiliza primitivas que permitem construir figuras geométricas controlando a posição de cada "alfinete" numa tela virtual e representar ainda, imagens reais em níveis de cinzento, transpondo o valor da luminância associado a cada pixel, para a posição do correspondente "alfinete". É possível criar uma animação em tempo real, com base num conjunto de imagens chave, utilizando um modelo de interpolação linear ou elástico. A aplicação foi construída pensando na sua extensibilidade, sendo possível adicionar novas funcionalidades/primitivas/animações. Para interagir com a aplicação, o utilizador pode usar uma consola semelhante à encontrada nos jogos do tipo FPS ("First Person Shooter"). A plataforma usada para desenvolver a tela de alfinetes digital foi o MX Toolkit, um ambiente C++ desenvolvido no nosso laboratório, orientado para a construção de aplicações em realidade virtual, aumentada e mista. Este artigo descreve a arquitectura do sistema, apresenta as funcionalidades disponibilizadas pela aplicação e conclui apresentando resultados de um estudo de usabilidade para avaliar o grau de satisfação da técnica.*

1. Introdução

A tela de alfinetes é um artefacto que tem fascinado animadores e o público em geral, quer pelas animações proporcionadas por Alexandre Alexeieff e sua esposa Claire Parker (*National Film Board of Canada*) [NFB], inventores da técnica, quer pelos mais recentes brinquedos para crianças que honram a sua invenção.

A animação "*The Nose*", de Alexeieff e Parker [Wiki], foi desenvolvida com uma tela de 240.000 alfinetes, sendo cada alfinete cuidadosamente reposicionado em altura para cada nova imagem, com uma ferramenta especialmente criada para o efeito. Ao lado da tela estava um foco luminoso cuja luz, ao incidir nos alfinetes de forma rasante, definia uma sombra. A variação da altura dos alfinetes correspondia assim, a uma variação de um nível de cinzento entre o preto e o branco, correspondente á sua sombra projectada.

Actualmente, podemos encontrar alguns autores cujas animações tentam recriar o efeito inventado por Alexeieff e sua esposa. O objectivo da aplicação aqui apresentada não é apenas criar uma animação numa tela de alfinetes digital, mas também proporcionar essa animação em tempo real. Pretende-se assim um protótipo capaz de mostrar o efeito realizado em tempo real que possa, em futuras versões, vir a suportar a entrada de dados através de uma câmara de vídeo.

O artigo está organizado da seguinte forma. Na secção 2 damos a conhecer os trabalhos passados relacionados de alguma forma com tela de alfinetes virtual. Na secção 3 explicamos o âmbito do projecto e quais os seus requisitos. Passamos então para a secção 4 onde explicamos a tela de alfinete virtual e como está estruturada através de um grafo de cena. Na secção 5 demonstramos a arquitectura funcional da tela, aqui explicamos o conceito de formas desenháveis e como é aplicado a um modo de desenho permitindo criar animações. Na secção 6 explicamos como conseguimos melhorar a performance da tela usando níveis de pormenor e optimização da organização espacial. Na secção 7 explicamos os modelos matemáticos usados para implementar as animações na tela. A estrutura dos ficheiros de animações usados pela aplicação é explicada na secção 8. A avaliação da usabilidade da aplicação é apurada na secção 9 e finalmente, na secção 10 apresentamos as conclusões e alguns passos futuros.

2. Trabalho Relacionado

Desde que a técnica de animação por tela de alfinetes foi introduzida por Alexeieff e Parker, podemos identificar essencialmente o trabalho de Pedro Faria Lopes [Lopes] nesta área, como um marco a assinalar. Lopes simulou, fielmente, a maneira de animar usada por Alexeieff e Parker, usando o mesmo princípio das sombras, mas desta vez criadas de forma digital. Lopes desenvolveu um sistema de animação por computador denominado "Tela de Alfinetes Digital" [Lopes92], que disponibilizava um conjunto de ferramentas que permitiam manipular facilmente os pinos da sua tela de alfinetes. No sistema de Lopes, uma imagem é inicialmente sintetizada usando as ferramentas que permitem manipular alfinetes ou conjuntos de alfinetes, um exemplo é o típico rolo, que permitia atrair ou repelir um conjunto de alfinetes. Após a primeira imagem ter sido gerada, o processo de animação passa então por fazer sucessivos ajustes à tela. Para a síntese de imagem, Pedro Faria Lopes usou inicialmente um algoritmo bastante simples: percorre-se todos os pinos da tela e para cada um, calcula-se a direcção e comprimento da sombra com base na posição da fonte de luz, então projecta-se a sombra a

partir da posição do pino. Lopes passou então para um algoritmo mais complexo que evitava o *aliasing*, produzindo imagens com melhor qualidade. Esta técnica não tinha como objectivo produzir animações em tempo real, e a carga computacional necessária para fazer face a esse desafio também não o permitia na altura (início da década de 90). Lopes introduz também a noção de tempo no seu sistema, usando interpolação de estados de tela (altura dos pinos). Assim, dado um conjunto de estados chave da tela, o mecanismo de interpolação produzia os estados intermédios das animações.

Ainda como referência e como influência adicional deste projecto, temos o videoclip "Only" do grupo musical Nine Inch Nails [NIN]. Neste vídeo, um "Pin Art" (nome da tela de alfinetes vulgarmente vendida no mercado) é animada tendo como base os movimentos do vocalista do referido grupo.

Figura 1: ***Imagem retirada de "Art Toy" do video Only de Nine Inch Nails***

Neste trabalho, a síntese de imagem, bastante realista, não é realizada em tempo real, tendo a animação da tela sido levada a cargo por uma empresa de renome na animação por computador: "digitaldomain" [DD].

3. Requisitos da Tela de Alfinetes Virtual

O Tela de Alfinetes Virtual surge no âmbito de trabalho académico, no contexto da disciplina de Programação 3D, integrada no plano curricular do 5º ano da licenciatura de IGE (Informática e Gestão de Empresas), leccionada no ISCTE (Instituto Superior de Ciências do Trabalho e da Empresa), no ano lectivo 2006/2007. O sistema desenvolvido, tinha de obedecer a um conjunto de requisitos definidos nesse contexto:

- *Estruturação espacial do grafo de cena e remoção de objectos não visíveis;*
- *Técnicas de aceleração gráfica 3D de forma a proporcionar animação e visualização em tempo real;*
- *Níveis de pormenor ("Level of Detail");*
- *Simulação de Física de Newton ao nível da partícula.*

4. A Abstracção Tela de Alfinetes

Os alfinetes presentes nas telas de Alexeieff e Parker são transpostos para o mundo 3D, como a repetição de um dado modelo no mundo virtual. De facto, o que se representa na aplicação é uma matriz bidimensional de alfinetes abstractos, sendo o alfinete concreto, um modelo 3D externo carregado durante a execução.

Numa estrutura de grafo de cena, cada elemento geométrico e topológico incluído no mundo, é posicionado, escalado e orientado através da agregação desse elemento a um nó de transformação geométrica. Assim, cada alfinete corresponde ao modelo 3D base (que comporta a geometria e topologia do alfinete), agregado a um nó de transformação. Por sua vez, estes nós de transformação estão associados, de uma forma hierárquica a outros nós e assim sucessivamente. A estruturação e organização correcta dos nós na tela de alfinetes provoca um aumento do número de imagens por segundo calculadas pelo sistema, o que é essencial para atingirmos uma animação fluida, ou dita de tempo real (com um tempo entre imagens geradas inferior a 20 ms). Abordaremos, mais à frente, as técnicas de aceleração usadas neste sistema.

Tendo a matriz de alfinetes sido criada, o esboço de formas na tela é atingido movendo os pinos no seu grau de liberdade único (vertical): de facto os pinos só se movem na dimensão vertical no referencial próprio da tela, estando fixos nas outras duas.

De uma maneira simples podemos comparar a matriz de alfinetes à memória de imagem (*framebuffer*) que encontramos nos sistemas gráficos com conversão por varrimento. Podemos assim aplicar os algoritmos de criação de primitivas gráficas 2D, desenvolvidos nos tempos pioneiros da Computação Gráfica. Neste contexto, o nosso sistema disponibiliza duas primitivas 2D básicas: o algoritmo de desenho de linhas e o de círculos, ambos de *Bresenham* [HILL99] (Figura 1.).

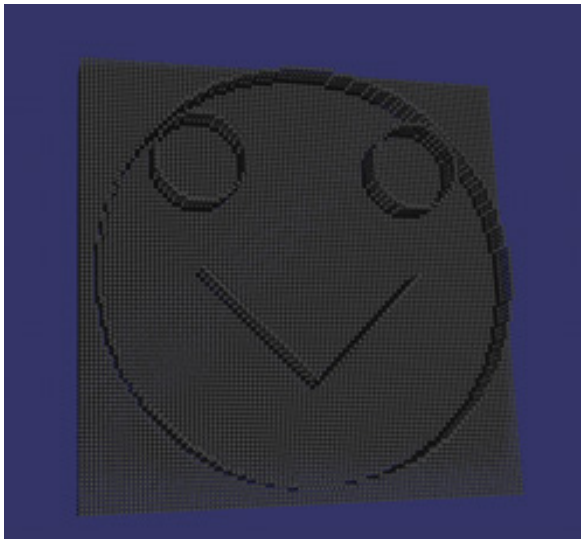

Figura 2: ***Um desenho geométrico simples (smiley) criado na Tela de Alfinetes Virtual com as primitivas 2D círculo e linha.***

5. Arquitectura Funcional do Sistema

A plataforma usada no desenvolvimento do sistema foi a MX Toolkit [Dias03]. Esta plataforma apresenta-se bastante robusta e flexível para a criação de aplicações na área de realidade virtual e aumentada, incorporando, entre outros módulos, aquele que é mais relevante para o caso do presente artigo, o *openscenegraph*, um grafo de cena 3D com um SDK C++ de alto desempenho [OSG].

5.1 Formas Desenháveis

Usando os princípios de *Programação Orientada por Objectos,* definiu-se a estruturação das formas abstractas que provocam alterações na posição dos pinos da tela. Estas formas tomam o nome de *Drawables* (Figura 3). Na sua base está a classe abstracta *Drawable* e dela derivam as classes *Line*, *Circle* e *LuminanceImage*. Estas são assim as primitivas disponibilizadas pela aplicação. A incorporação de novas primitivas torna-se possível porque corresponde a uma nova extensão da classe *Drawable* ou alguma que dela descenda. A primitiva *LuminanceImage* carrega uma imagem de um ficheiro e calcula o valor da luminância associado a cada *pixel* (Figura 4). Esse valor é, então, traduzido proporcionalmente para a posição vertical do respectivo alfinete (Figura 5).

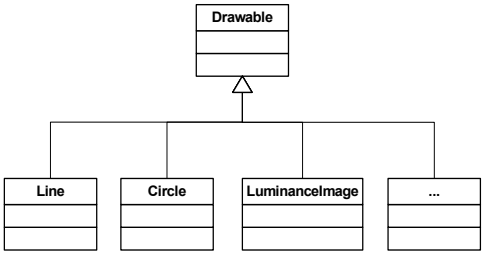

Figura 3: ***Diagrama de classes (UML) de um Drawable.***

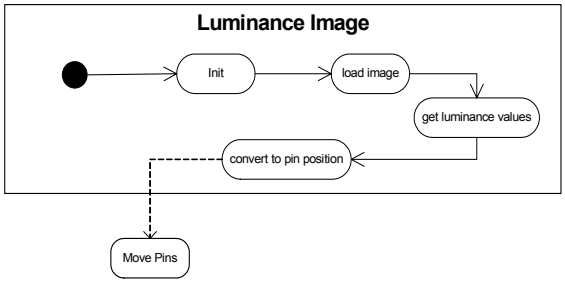

Figura 4: ***Sequência de carregamento de uma imagem.***

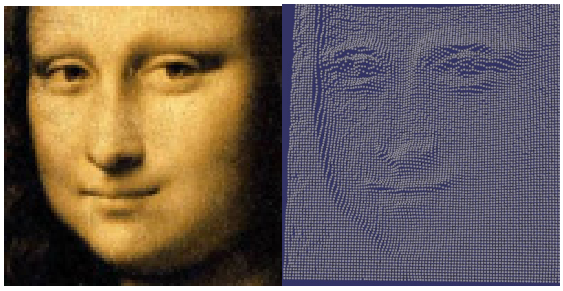

Figura 5: ***Tela Virtual (à direita) gerada usando os valores da luminância da imagem da esquerda.***

5.2 Modos de Desenho

Na secção anterior foram enunciados os objectos que são responsáveis por provocar alterações nas posições verticais dos alfinetes. No entanto, esses objectos não interagem directamente com a tela de alfinetes. Entre esses objectos e a tela, encontramos o conceito de "maneira de desenhar na tela" que define, efectivamente, o modo como um pino se move, ou seja, como passa da posição vertical anterior para a posição seguinte.
Esta abordagem permite, de forma fácil, incorporar novos modos de animação em tempo real, aplicados á tela virtual. O sistema disponibiliza então 4 modos de desenhar (Figura 6.):

- Modo Absoluto, onde cada alfinete actualiza a sua nova posição de forma absoluta, proporcionalmente à luminância do pixel da imagem de entrada correspondente.
- Modo Relativo, onde cada alfinete actualiza a sua nova posição X_t, "relativamente" à posição anterior X_{t-1}, dado um acréscimo α, $X_t = X_{t-1} + \alpha$
- Modo em Animação Linear, onde o deslocamento da posição chave anterior X_{t-1}, para a posição chave seguinte X_t, demora um tempo *t*. As posições intermédias são definidas pela interpolação linear entre as duas posições chave.
- Modo em Animação Elástica Linear, onde a transição entre duas posições é ditada pela simulação de uma mola elástica de Hooke (linear) amortecida [MPL].

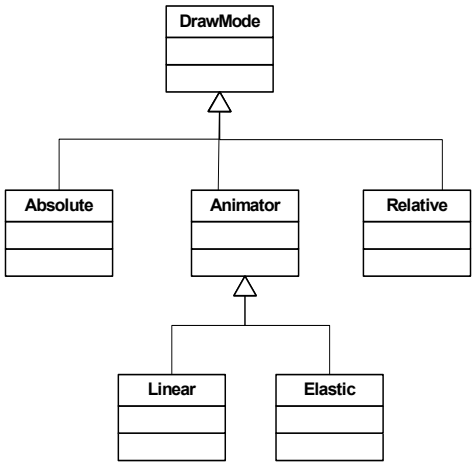

Figura 6: ***Diagrama de classes (UML) do DrawMode***

5.3 Animação na Tela de Alfinetes Virtual

No nosso sistema, a animação em tempo real consegue-se com a execução de primitivas em instantes temporais pre-definidos (Figura 7).

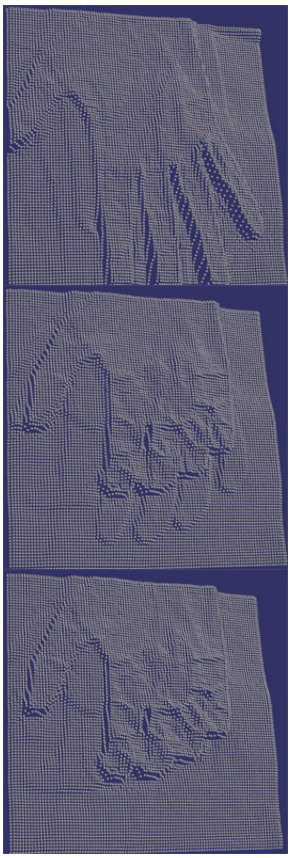

Figura 7: Imagens da ***Animação de uma mão usando o modo linear que interpola duas imagens chave.***

A aplicação disponibiliza apenas um tipo de animação, baseada na primitiva *LuminanceImage* (Figura 8). No entanto é facilmente extensível para outras primitivas/listas de primitivas de animação

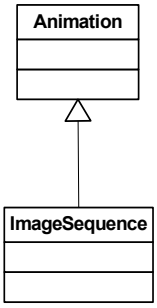

Figura 8: ***Diagrama de classes (UML) de Animation.***

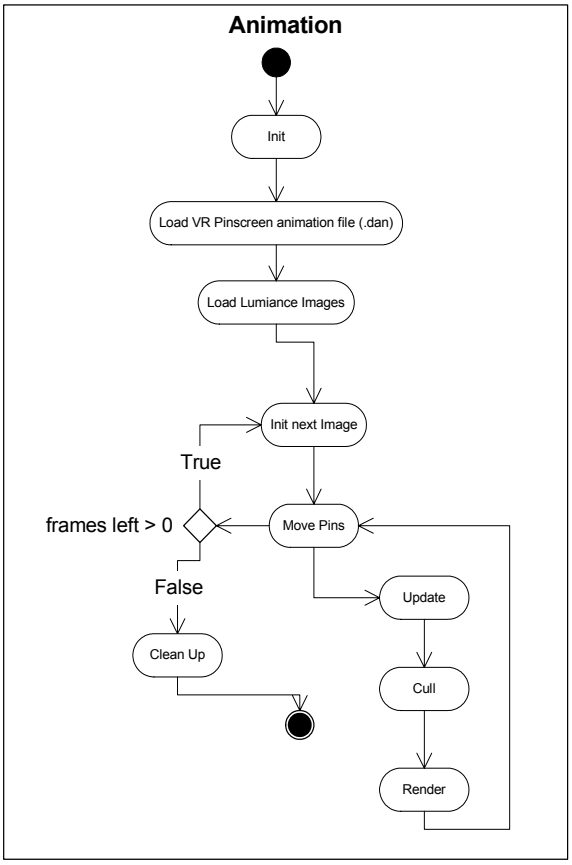

Figura 9: ***Sequência de animação***

5.4 Posicionamento dos Alfinetes

Quando uma imagem (*LuminanceImage*) é carregada, é calculado o valor da luminância de cada *pixel*. Este valor é então, convertido para uma posição vertical de alfinete, usando uma proporção simples entre a escala da luminância e o movimento máximo possível de um alfinete. Este movimento máximo corresponde a 2 vezes o seu tamanho. A aplicação permite também, que o utilizador defina explicitamente um valor para esta variável.

6. Técnicas de Aceleração Gráfica 3D

6.1 Organização do Grafo de Cena

O *openscenegraph* é um grafo de cena 3D genérico que disponibiliza ferramentas que ajudam a aceleração da aplicação gráfica. Dentro destas, podemos considerar como a mais importante, a técnica bem conhecida de "*view frustum culling"* que permite remover, de forma eficiente, grande parte da geometria da cena que se encontra fora da pirâmide de visualização. No caso do *openscenegraph*, este algoritmo de simplificação exige que seja criada uma hierarquia de esferas envolventes dos nós do grafo. Explicar esta técnica sai fora do âmbito deste artigo, remetendo-se o leitor para a excelente descrição em [Moller02]. No entanto, esta técnica funciona de forma ineficiente, se o grafo de cena

que lhe é fornecido estiver, por natureza, mal gerado organizado e mal balanceado. De seguida apresenta-se uma situação que ilustra este problema.

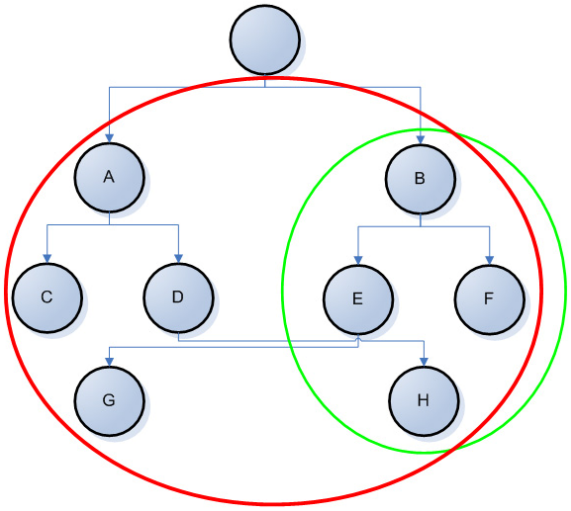

Figura 10: ***Grafo de cena não óptimo.***

Assumindo cada círculo como cada nó da cena e, a posição de cada círculo na imagem como a posição real do nó, podemos identificar os nós G e H que, para efeitos de "*frustum culling*" diminuem a eficiência da árvore. A *esfera envolvente* verde corresponde à *esfera envolvente* óptima para a árvore B. No entanto, como o nó G está logicamente associado ao nó E e a sua posição está mais próxima dos nós E/D, a *esfera envolvente* de B passa a ser a esfera envolvente vermelha. O mesmo acontece para A e seus descendentes. Neste caso as *esferas envolventes* de A e B são iguais, e ambas são passadas à fase de síntese de imagem, mesmo que uma das árvores esteja fora da pirâmide de visualização. A eficiência de *culling* desta árvore podia ser melhorada associando G a D, e H a E. Neste caso, para a árvore B, a esfera envolvente passava a ser a verde ao invés da vermelha, e na fase de *culling* a árvore A seria podada, eliminando-a do processo de síntese de imagem.

De forma a optimizar a disposição da matriz de alfinetes foi idealizado um algoritmo que organiza a matriz, criando uma hierarquia de nós associada. A tela tem duas dimensões que não são alteradas durante as animações, respeitando a colocação dos alfinetes. A tela tem ainda uma terceira dimensão para o deslocamento dos pinos. Para a organização espacial, interessa criar uma estrutura eficiente para a colocação dos pinos, o que torna a estrutura a operar, uma estrutura 2D. O algoritmo de organização espacial tem como parâmetro o número de nós por dimensão. Significa que se o valor for 2, cada nó da árvore a ser construída, vai ter 2 * 2 nós descendentes.

O algoritmo, dado uma matriz original de "alfinetes", cria uma outra matriz em que cada nó é pai de *n* elementos da matriz original, sendo o valor do *n,* ditado pelo parâmetro: número de nós por dimensão. Este procedimento é então executado de novo mas agora na matriz acabada de criar, e assim sucessivamente até se criar uma tabela com apenas um nó, que se torna a raiz da estrutura. O parâmetro *n* define o tamanho do bloco a operar em cada nível da estrutura. A 1D, o algoritmo cria uma árvore do tipo da Figura 11 (assumindo 2 como número de nós por dimensão).

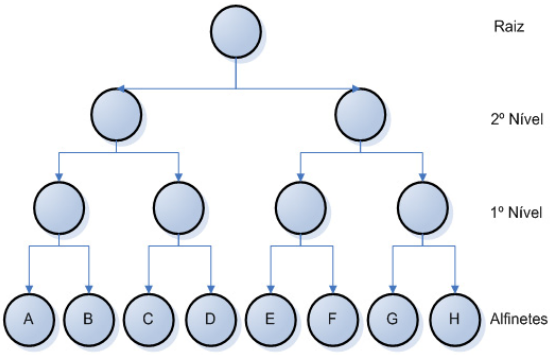

Figura 11: ***Árvore criada a 1D***

No exemplo da Figura 11, as folhas são os círculos marcados com letras e representam cada alfinete, o algoritmo começa por pegar nos 8 alfinetes e na primeira iteração agrupa-os, pela sua posição, dois a dois, gerando o 1º nível da estrutura, este nível têm então apenas 4 nós. O algoritmo prossegue de forma idêntica para gerar o segundo nível, mas desta vez usa os nós do 1º nível para operar. O algoritmo prossegue de igual forma até o número de nós a operar ser 1, neste caso para e este último nó torna-se a raiz da estrutura. Para a síntese de imagem a raiz da estrutura é passada ao motor gráfico do *openscenegraph* que usa as *esferas envolventes* de cada nó para saber se ele está visível ou não. Assim se, a título de exemplo, os alfinetes E, F, G e H estiverem fora da pirâmide de visualização, torna-se necessário apenas 1 único teste no 2º nível, para conseguir descartar a árvore correspondente.

6.2 Nível de Pormenor (*Level Of Detail*)

A técnica de nível de pormenor é usada para aceleração gráfica 3D. Ela funciona por disponibilização de várias cópias do mesmo objecto, em que cada cópia difere da outra no sentido em que a complexidade ao nível da geometria diminui. O objectivo é substituir um objecto bastante complexo por uma cópia do mesmo, mas mais simples, quando estamos a uma distância suficiente para que não se consigam notar diferenças apreciáveis entre os dois.

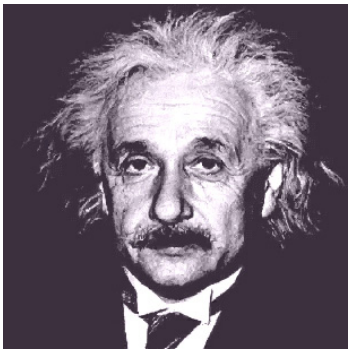

Figura 12: ***Imagem Original.***

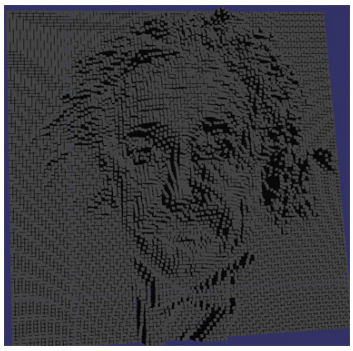

Figura 13: ***Imagem na Tela de Alfinetes Virtual com todo o pormenor.***

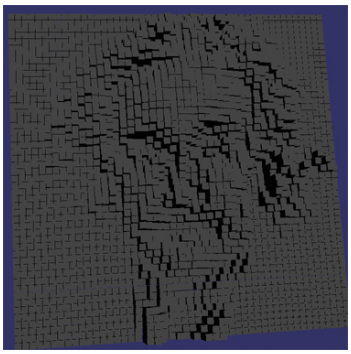

Figura 14: ***Imagem na Tela de Alfinetes Virtual com o 1º nível de pormenor.***

A tela de alfinetes virtual suporta nível de pormenor. No entanto a técnica funciona de uma maneira diferente da usual. Neste caso um objecto não tem diversas cópias. O que acontece é que para um bloco de, por exemplo 4 alfinetes, cria-se um nó LOD (*Level Of Detail*) que substitui os mesmos 4 alfinetes. O nó LOD é um alfinete escalado em duas dimensões (transversal e longitudinal) da tela.

A aplicação permite que se defina o número de "níveis de pormenor", bem como a distância entre cada nível. A construção de nós LOD é hierárquica e pode ser activada ou desactivada na aplicação. O tamanho de cada bloco LOD (nº de alfinetes por bloco), é ditado pelo número de nós por dimensão, e o número de níveis da hierarquia é ditado pelo número de níveis de pormenor.

A técnica de LOD permite aumentar bastante o número de imagens por segundo (*fps*), porque reduz significativamente a geometria desenhada. Imaginando 100x100 alfinetes, com blocos de 4 alfinetes e somente um nível de pormenor, é possível ter uma tela a ser simulada como se fosse uma tela de 50x50 alfinetes, sentindo-se apenas o "peso" dos 100x100 na transição entre nível.

7. Modelos de Animação

7.1 Movimento Linear

Como já foi referido, o sistema disponibiliza um modelo de animação baseado na interpolação linear entre duas posições chave. A interpolação linear, dada uma origem e um destino, cria as posições intermédias entre os dois. Cada posição é gerada face um parâmetro t, que é normalmente dependente de uma variável tempo e que varia entre 0 e 1.

Dados $\boldsymbol{O}$ (origem) e $\boldsymbol{D}$ (destino), calcula-se o vector $\boldsymbol{V}$, diferença entre os dois pontos:

$$V = D - O$$

Define-se, então, a função de interpolação linear da seguinte forma:

$$P(t) = O + V.t$$

Com $t = 0$,

$$P(0) = O$$

Com $t = 1$,

$$P(1) = D$$

Com t a tomar um valor entre 0 e 1, surge uma posição entre $\boldsymbol{O}$ e $\boldsymbol{D}$.

7.2 Movimento Elástico

Outro modelo de animação, baseado em simulação Física, é o conhecido modelo "*mola - massa*", que se define por uma mola linear com atrito.

O modelo original de molas de *Hooke* é, vectorialmente:

$$F = -k.X$$

Este modelo diz que a força da mola é proporcional ao seu deslocamento da posição de equilíbrio:

- $\boldsymbol{X}$ é o vector cuja norma é a distância entre os dois pontos que definem a mola, menos a sua distância de equilíbrio.
- k é a constante elástica da mola que define a rigidez da mesma.

Podemos adicionar a este modelo linear um modelo de atrito linear simplificado, que impede que a mola balance eternamente, obtendo o seguinte modelo vectorial:

$$F = -k.X - b.V$$

- b é o coeficiente de atrito.
- $\boldsymbol{V}$ é a velocidade relativa entre os dois pontos da mola.

Soluções analíticas existem para casos particulares de equações diferenciais ordinárias da forma anterior. No entanto uma solução numérica é talvez mais simples de desenvolver, utilizando o método de integração de Euler ou de Runge-Kutta de 4ª ordem [Press92].

8. Ficheiro de Animação

A animação da tela é definida num ficheiro de texto com três secções. A primeira, tem informação relativa ao tipo de animação e ao directório base das imagens. A segunda secção, inclui a listagem de todas as imagens chave presentes na animação. A terceira e última secção, define os instantes temporais chave de animação: mapeia um instante temporal chave a um índice da lista de imagens.

```
swing
Data\Images\hand\
4
100x100_hand_00.jpg
100x100_hand_01.jpg
100x100_hand_02.jpg
100x100_hand_03.jpg
4
1.5 0
3 1
4.5 2
6 3
```

Figura 15: ***Ficheiro de animação***

9. Avaliação da Usabilidade do Sistema

9.1 Medotologia

Para verificar que o Tela de Alfinetes Virtual é realmente uma aplicação capaz de proporcionar uma experiência visualmente agradável e apelativa, conduzimos um teste de avaliação da usabilidade do sistema, para obter os níveis de satisfação dos sujeitos do teste.

O teste foi realizado num computador portátil Pentium M com 2 GByte de memória RAM e um processador gráfico integrado ATI Mobility Radeon X700 com 64MiB de memória.

Os sujeitos do teste totalizaram 10 estudantes do ISCTE não pagos para o efeito, com idades entre os 20 e 29 anos e formações académicas distintas. Todos os elementos partilhavam uma característica em comum, que era a falta de contacto com ambientes de Realidade Virtual e a falta de conhecimentos ao nível de computação gráfica.

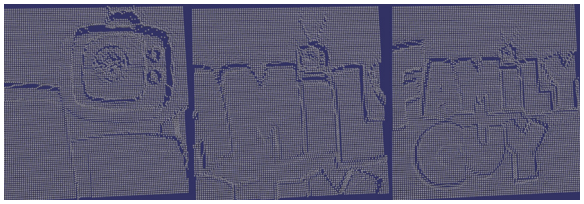

Figura 16: ***Clip de Abertura do Family Guy***

O teste foi conduzido por um elemento que desenvolveu a aplicação e o sujeito do teste não teve interacção com o sistema. O teste começou com a uma breve explicação do que é uma tela de alfinetes e como ela funciona.

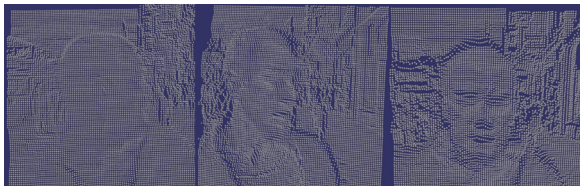

Figura 17: ***Clip de Trainspotting***

De seguida apresentaram-se uma série de fases de demonstração, focando a visualização de imagens com esta técnica, a apresentação de animações com métodos de interpolação linear e elástica linear e a reprodução de sequências de imagens, convertidas para o modo de visualização da tela de alfinetes virtual. Estas demonstrações visaram testar a aplicação e o seu impacto na satisfação percebida pelos sujeitos:

1. Smiley apresentado na **Figura 2**.

2. Apresentação das imagens *Mona Lisa* e *Einstein.* **Figura 5** e **Figura 13**.

3. Apresentação da Imagem *Einstein* com nível de pormenor. **Figura 14**.

4. Transição entre *Mona Lisa* e *Einstein* usando o modelo de animação elástico.

5. Animação de uma mão, usando o modo linear, com um conjunto de 11 imagens. Imagens dessa animação podem ser encontradas na **Figura 7**.

6. Reprodução de uma sequência de imagens retiradas da série de animação - *Family Guy* **Figura 16**.

7. Reprodução de uma sequência de imagens retiradas do filme – *Trainspotting* **Figura 17**.

Após a conclusão da apresentação, cada sujeito do grupo alvo foi convidado a responder um inquérito para avaliar o grau de satisfação e a atractividade visual relativamente ao que haviam acabado de ver. Nesse inquérito constavam as seguintes questões:

1. Na escala de [1-6], como classifica a 1ª fase, a nível de atractividade visual. [1 nada, 6 muito].

   Resultado : 3,3

2. Na escala de [1-6], como classifica a 2ª fase, a nível de atractividade visual. [1 nada, 6 muito].

   Resultado : 4,2

3. Na escala de [1-6], como classifica a 3ª fase, a nível de atractividade visual. [1 nada, 6 muito].

   Resultado : 3,5

4. Na escala de [1-6], como classifica a 4ª fase, a nível de atractividade visual. [1 nada, 6 muito].

   Resultado : 5

5. Na escala de [1-6], como classifica a 5ª fase, a nível de atractividade visual. [1 nada, 6 muito].

   Resultado : 4,8

6. Na escala de [1-6], como classifica a 6ª fase, a nível de atractividade visual. [1 nada, 6 muito].

   Resultado : 5,4

7. Na escala de [1-6], como classifica a 7ª fase, a nível de atractividade visual. [1 nada, 6 muito].

   Resultado : 5,7

8. Na escala de [1-6], sentiu dificuldades a reconhecer as imagens projectadas. [1 muito difícil, 6 muito fácil].

   Resultado : 5,6

9. Na escala de [1-6], acha que a aplicação tem potencial para ajudar a desenvolver conteúdos criativos, [1 pouco, 6 muito].

   Resultado : 5,0

Os resultados foram obtidos calculando a média para cada questão. A classificação mais baixa, nas questões de atractividade visual, foi obtida no *smiley*. Esta fase é, com efeito, a menos atractiva a nível visual. A complexidade do *smiley* é também muito reduzida, daí a sua reduzida atractividade face às outras fases. A classificação mais elevada foi obtida na reprodução da sequência do *trainspoting*. Nesta sequência podemos identificar um indivíduo a correr e, mais tarde, a ir ao encontro de um veículo em movimento e como consequência do embate, a cair no chão. Esta sequência apresenta-se mais complexa e, ao nível do estímulo visual, funcionou bastante bem.

Calculando a média dos resultados obtemos um indicador que pode funcionar como apreciação global da aplicação. O valor da média é **4.72**, concluindo que a utilização da aplicação teve uma apreciação positiva.

	1	2	3	4	5	6	Resultados
1ª Questão	0	2	3	5	0	0	**3,3**
2ª Questão	0	0	0	8	2	0	**4,2**
3ª Questão	0	2	3	3	2	0	**3,5**
4ª Questão	0	0	0	2	6	2	**5**
5ª Questão	0	0	0	4	4	2	**4,8**
6ª Questão	0	0	0	1	4	5	**5,4**
7ª Questão	0	0	0	0	3	7	**5,7**
8ª Questão	0	0	0	0	4	6	**5,6**
9ª Questão	0	0	0	3	4	3	**5**

Tabela 1: ***Resultados dos inquéritos***

10. Conclusões e Trabalho Futuro

Neste trabalho, apresentámos um sistema que simula em tempo real a tela de alfinetes de Alexandre Alexeieff e Claire Parker, permitindo a criação de animação através de primitivas simples e de um mecanismo de interpolação linear ou elástico. Testes de usabilidade, visando avaliar o grau de satisfação e a atractividade visual que resultam da visualização de um conjunto de 7 demonstradores, mostraram um grau de satisfação positivo.O número de alfinetes da tela condiciona a resolução das imagens sintetizadas.

Desta forma as técnicas de aceleração são muito importantes para conseguirmos produzir animação em tempo real. Uma maior capacidade computacional disponível, significa que o sistema pode lidar com um maior número de pinos, e um melhor desempenho dos modelos de animação. Assim sendo, o uso das facilidades das GPU, para o posicionamento dos alfinetes, poderá melhorar o desempenho da tela virtual. Nesse sentido, prevemos o desenvolvimento dos algoritmos de animação utilizando os s*haders* das placas gráficas, usando *vertex shaders* para o posicionamento dos vértices da geometria de cada alfinete. Neste caso, será então apenas necessário enviar à placa gráfica a posição no eixo de movimento dos alfinetes. A incorporação de um modelo de sombras na tela também será um dos passos futuros no sentido da criação de animações mais realistas, de acordo com a abordagem inicial de Alexeieff e e Parker, seguida também por Pedro Faria Lopes. Alguns testes foram feitos com a técnica de *Shadow Maps* exemplificada no *openscenegraph*, ainda não satisfatórios. Outro passo futuro será introduzir uma componente de cor em cada alfinete, permitindo animações coloridas já desenvolvidas por Lopes [Lopes92]. Neste caso, os modelos de animação poderiam funcionar não só para o reposicionamento dos alfinetes mas também para a transição da cor. Acreditamos também que a aplicação possa ter utilidade na criação de conteúdos criativos, síntese de imagem não realista, ambientes de realidade aumentada e representação de objectos virtuais em tempo real.

11. Referências

[DD] *"Digital Domain"*, Consultado a 18 de Junho de 2007, http://www.digitaldomain.com/.

[Dias03] Dias, J., M., S., Santos, P., Monteiro, L., Silvestre, R., Bastos, R., "Developing and Authoring Mixed Reality with MX Toolkit", ART03, The Second IEEE International Augmented Reality Toolkit Workshop, Tokio, Japão, 6 Outubro 2003.

[DSB03] Dias J. M. S., Santos P., Bastos R.: "*Developing and Authoring Mixed Reality with MX Toolkit*". ART03*,* 2003.

[Eberly] Eberly, D. H., 2004, "Game Physics", Morgan Kaufmann.

[GOG] *"Graffer, Gpring Physics, Integration Basis"*, Consultado em Janeiro de 2007, http://www.gaffer.org/game-physics/integration-basics/.

[HILL99] Hill, F. S, , *"Computer graphics using OpenGL*", Prentice Hall, 1999, pp. 554-561.

[Lopes] Lopes, P.F. 1999, *"The Pinscreen in the Era of the Digital Image",* Consultado em Janeiro de 2007, http://www.writer2001.com/lopes.htm.

[Lopes92] Lopes, P.F., Gomes, M. R., *"A Computer Model For Pinscreen Simulation : A New Animation Paradigm"*. Computer Graphics Forum, Volume 11, Issue 1, 1992, pp. 31-42.

[Moller02] Moller T. A., Haines E.:*"Real-Time Rendering",* A.K. Peters Ltd., 2002, pp. 363-368.

[MPL]*"MyPhysicsLab Simple Spring"*, Consultado em Janeiro de 2007, http://www.myphysicslab.com/spring1.html.

[NFB] *"Techniques, Pinscreen"*, Consultado em Janeiro de 2007, http://www.nfb.ca/animation/objanim/en/techniques/pinscreen.php.

[Nin] "Visual", Consultado a 18 de Junho de 2007, http://www.nin.com/visuals/.

[OSG] *"OpenSceneGraph"*, Consultado a 18 de Junho de 2007, http://www.openscenegraph.org/.

[Press92] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P., *"Numerical Recipes in C"*, Cambridge, 1992, pp. 710-714.

[Wiki] "PinScreen animation", Consultado em Janeiro de 2007. http://en.wikipedia.org/wiki/Pinscreen_animation.

12. Anexo Funcionalidades Extra

12.1 Scripting

A aplicação disponibiliza um pequeno e leve sistema de *scripts*. Sucintamente um *script* é uma lista de comandos que o utilizador pode passar na consola.

12.2 Consola

A aplicação disponibiliza uma consola para interface com o utilizador. Esta disponibiliza os seguintes comandos:

- **/help**
- **/quit**
- **/list** - lista os comandos disponíveis
- **/clear** - apaga a lista de mensagens.
- **/echo *<msg>***
- **/version** - mostra a versão.
- **/set** *<var name> <value>* - define o valor de uma variável. Sem parâmetros lista a variaáveis.
- **/rebuildPinScreen** - reconstrói a tela de alfinetes.
- **/drawPin** *<col> <line> <pos>* - move o pino [col, col] o valor da posição.
- **/drawLine** *<orig col> <orig line> <dest col> <dest line> <pos>*
- **/drawCircle** *<center col> <center line> <radius> <pos>*
- **/loadLuminanceImage** *<file>*
- **/run** *<script>*
- **/play** *<anim>*
- **/stop** - para a animação.

Estão disponíveis as seguintes variáveis:

- **ps_lod** - número de níveis de detalhe.
- **ps_lod_distance** - distância entre níveis de lod.

- **ps_divisions** - número de nós por divisão por dimensão.
- **ps_pin_name** - nome do ficheiro com o modelo do alfinete.
- **ps_v_pins** - número de alfinetes na vertical.
- **ps_h_pins** - número de alfinetes na horizontal.
- **ps_v_spacing** - espaçamento entre os pinos nas vertical.
- **ps_h_pins** - espaçamento entre os pinos na horizonal.
- **ps_draw_mode** - modo de desenho *<absolute, relative, linear, elastic>*
- **ps_max_movement** - movimento máximo na dimensão de movimento dos pinos *<0 = auto>*
- **ps_linear_duration** - duração de cada mudança de posição para o modo linear.
- **ps_pin_mass** - massa do alfinete para o modelo elástico.
- **ps_spring_stiffness** - rigidez da mola no modelo elástico.
- **ps_spring_damping** - atrito da mola no modelo elástico.

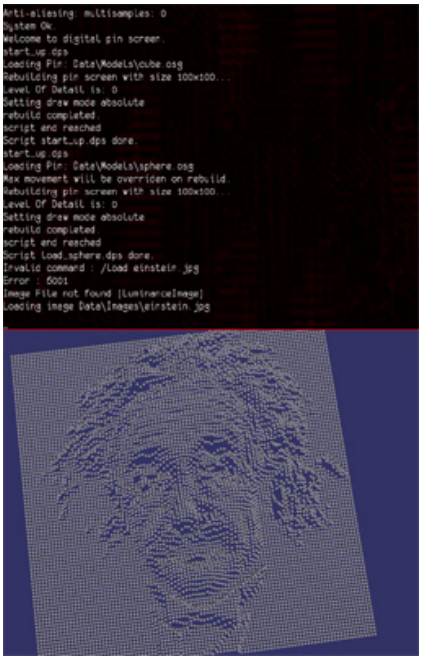

Figura 18: ***Consola clássica de videojogos***