Deformação em Carros de Corrida

Roberta Claudino Barreto Pessanha Gomes
Marcos Rangel Júnior
Sicília Ferreira Ponce Pasini Judice
Ítalo de Oliveira Matias

Universidade Candido Mendes – Campos, Brasil

Figura1: Efeito de deformação obtido neste trabalho.

RESUMO

No mundo real, estamos em constante interação com diversos tipos de objetos. A cada interação é gerada uma determinada reação, cuja intensidade varia de acordo com os elementos envolvidos. A exemplo de tal pode-se citar a deformação causada por um choque entre corpos. Em aplicações gráficas, elementos presentes na física podem ser estudados e utilizados, a fim de aproximar o comportamento de tais aplicações com o mundo real. É com base neste cenário que este trabalho propõe o estudo e desenvolvimento de uma técnica de deformação aplicável a carros de corrida baseada no método warping.

Palavras-chave: deformação, jogos eletrônicos, simulação computacional, warping.

Contatos
{roberta.claudino, mrangel, siciliajudice}@gmail.com
italo@ucam-campos.br

1. Introdução

Os corpos interagem de diversas formas no mundo real. Uma dessas muitas formas é a *colisão*, que vem a ser um evento isolado no qual uma força age com intensidade relativa em cada um dos corpos que interagem por um dado intervalo de tempo [Justino 2001]. Os corpos envolvidos em uma colisão sofrem distorções em suas formas, tal evento é denominado *deformação*.

Pode-se citar diversas situações em que podemos aplicar colisões e seus efeitos na área de games, tais como: deformação em tecidos [Terzopolous et al. 1987], em personagens (esqueletos, faces) e em veículos, entre outros. O objetivo desse estudo é simular o efeito da deformação decorrente da colisão parcialmente elástica entre veículos, especificamente entre carros de corrida. Para realização deste trabalho, foi utilizado um jogo desenvolvido por [Rangel Junior et al. 2006; Azeredo et al. 2006].

A seguir serão apresentados maiores detalhes sobre o estudo desenvolvido. No tópico 2 será falado sobre ferramentas de desenvolvimento. O tópico 3 abordará a deformação em jogos eletrônicos. O tópico 4 tratará das técnicas de deformação geométrica. No tópico 5 será apresentado o método de deformação adotado. O tópico 6 trará a conclusão do trabalho.

2. Ferramentas de Desenvolvimento

As ferramentas, ou *toolkits*, incorporam uma parte do conhecimento exigido na criação de um game, não sendo necessário ao desenvolvedor entender plenamente as etapas para sua realização. Existem várias ferramentas auxiliares ao desenvolvimento de um jogo eletrônico, a ressaltar: game engine, motores de som, física, inteligência artificial, dentre outros.

2.1 Motores

2.1.1 Motores Gráficos

De acordo com [Boni 2005], um motor, ou engine, gráfico tem como objetivo explorar ao máximo a capacidade de processamento da placa de vídeo para poder realizar as etapas necessárias de renderização na tela, seja de um game ou qualquer outra aplicação gráfica. São exemplos de motores gráficos: Fly3D [Policarpo 2003], Irrlicht Engine [Gebhardt 2004] e Ogre3D [THE OGRE TEAM 2001].

Neste trabalho foram analisadas as características de cada um dos motores gráficos, tais como: documentação disponível, preço de licença para uso, estabilidade, confiabilidade e outras. Após tal análise, optou-se pelo uso do Ogre3D, pois além de possuir uma vasta documentação e um fórum para discussão, este motor é gratuito.

2.1.2 Motores Físicos

O motor físico é responsável pela parte física de uma aplicação. Efeitos como colisão, gravidade, elasticidade e qualquer outra forma de interação de objetos ficam a cargo desse motor. Como os motores gráficos, também existem vários motores físicos disponíveis, a citar: ODE [Smith 2003], Havok [HAVOK 1999], Tokamak [Lam 2003] e Newton [Jerez and Suero 2003], dentre tantos outros. Tais opções também foram analisadas seguindo basicamente os mesmos critérios citados anteriormente e a escolhida para o estudo foi o ODE, por também possuir vasta documentação. Outros motivos para a escolha do ODE como motor físico a ser utilizado foi sua grande difusão, credibilidade e facilidade de integração com o Ogre3D.

2.2 OgreODE

O OgreODE é uma das diversas ferramentas de auxílio para aplicações feitas no Ogre3D. É uma biblioteca feita em C++ capaz de integrar o sistema gráfico do Ogre com a engine física ODE. Tal ferramenta provê acesso a todas as primitivas da engine física e ainda traz alguns objetos pré-feitos, como carros, e suporte a criaturas articuladas (ou ragdolls), como visto em [OGRE3D WIKI 2000].

Com os veículos pré-feitos (ou prefab_Vehicles) trazidos pelo OgreODE, é possível agregar a um projeto carros, caminhões, monociclos ou quaisquer outros objetos que possuam comportamento semelhante a estes citados. Tais veículos trazem certos atributos, como por exemplo: massa, suspensão, centro gravitacional e coeficiente de atrito das rodas, entre outros.

3. Deformação em Jogos

No mundo dos games, exige-se cada vez mais dos recursos dos computadores para processar algoritmos que tornam as aplicações mais realistas. Nem sempre a evolução do hardware é capaz de acompanhar a evolução da demanda por realismo. Tal fato provoca a busca por formas de implementar o que se deseja lidando com as limitações dos equipamentos computacionais.

A deformação está presente em boa parte dos games de simulação de veículos. Alguns exemplos a serem salientados de aplicações que contenham simulação de deformação de veículos são: Carmageddom [STAINLESS SOFTWARE 1997], Driver [REFLECTIONS INTERACTIVE 1999] e GTA3 [DMA DESIGN 2001]. Apesar de ser possível presenciar deformação na maior parte dos games de simulação de veículos, esta não é vista com muita freqüência em outras aplicações voltadas ao entretenimento.

4. Método de Deformação Adotado

A deformação desenvolvida neste estudo tem como base o método de *Warping* e baseada em outros estudos de deformação em veículos como em [Rodrigues et al. 2005]. Tal método foi utilizado porque se mostrou de fácil implementação, forneceu resultados satisfatórios e também por conta de seu tempo de processamento. Enquanto outros métodos são capazes de produzir efeitos mais realistas em se tratando de deformação de corpos, o *Warping* tende a ser mais rápido por conta da flexibilidade de escolha de sua função de deslocamento. Este método também trabalha somente com os vértices já existentes no corpo, não sendo necessária a criação de novos vértices. A preservação do número de vértices em jogos é importante devido ao fato de serem aplicações que necessitam de respostas rápidas de processamento, e quanto maior o número de pontos em uma cena, maior o tempo gasto para processá-los.

4.1 Descrição do Algoritmo

O algoritmo implementado é acionado quando uma colisão entre veículos é detectada. A partir da detecção, seguem-se os passos abaixo:

-O ponto central da colisão é localizado em cada veículo;
- verifica-se o impacto, a direção do choque e a constante dos materiais envolvidos;
-são localizados os vizinhos do ponto central do impacto que estiverem dentro de um raio de tolerância;
-cada vizinho é deslocado segundo a função de deslocamento, que leva em consideração a

distância do ponto central, o impacto e a direção da colisão;
-atualiza-se a malha do veículo.

O ponto central de uma colisão é fornecido pelo OgreOde, uma vez que este é capaz de detectar colisão entre corpos e calcular o centro da colisão. Define-se como "ponto central", o ponto em comum durante a colisão entre objetos. Durante a colisão de carros, vários pontos são retornados sequencialmente, fazendo o algoritmo ser executado a cada vez que o evento de colisão ocorre.

A direção e o impacto da batida também são fornecidos pelo motor no momento em que tal ocorre. Tais valores são levados em consideração porque é importante saber em que eixos o ponto será deslocado. Quanto maior o impacto, maior a profundidade da batida. Também é necessário levar em consideração a constante dos materiais envolvidos na colisão, que representa a dureza dos mesmos. Para materiais mais rígidos, utiliza-se uma constante com valor maior e o inverso ocorre para materiais com menor rigidez. Nesse estudo, adotou-se o valor 0,7 como constante da carcaça dos veículos. Tal valor foi determinado após testes de colisão.

Especificamente neste estudo, a tolerância adotada para identificação dos vizinhos foi de 0,10 unidades no eixo x, 0,40 no eixo y e 0,15 no eixo z. Esses valores foram escolhidos levando em consideração o tamanho do veículo no ambiente, a distância inicial entre os vértices e o número dos mesmos no objeto. Tal tolerância faz com que os pontos que possuam algum dos valores em x, y ou z, cujo módulo da distância dos valores nos respectivos eixos do ponto central seja menor que o valor da tolerância sejam classificados como vizinhos a serem deslocados. Por exemplo, se o ponto central é (5,5; 4; 6,2) e o ponto em análise é (7; 4,2; 5), então tal vértice é classificado como vizinho, pois no eixo y, o módulo da diferença entre os valores dos pontos é igual a 0,2 e, portanto, menor que a faixa de tolerância em y.

Com os vizinhos identificados, é feito o deslocamento de cada um observando-se sua distância do ponto central do choque e a constante *k* atribuída ao material a ser deformado. Quanto mais próximo ao centro, maior será o deslocamento do vizinho e o inverso é válido para pontos mais distantes. Tal lógica faz com que o deslocamento se atenue com a proximidade de cada vértice com o centro da batida, fornecendo um efeito mais realista à deformação. O cálculo feito para saber o valor exato do deslocamento é feito a partir da determinação do módulo da diferença em cada eixo do vizinho para o vértice central:

$$X = X + (Tx - |Xc - X| \cdot k)$$
$$Y = Y + (Ty - |Yc - Y| \cdot k)$$
$$Z = Z + (Tz - |Zc - Z| \cdot k)$$

onde X, Y e Z correspondem aos valores nos respectivos eixos do ponto em análise; Tx, Ty e Tz correspondem às tolerâncias adotadas para cada eixo; Xc, Yc e Zc correspondem aos valores nos respectivos eixos do ponto central da colisão; e k corresponde ao valor da constante do material.

Tendo em mãos tais diferenças, observa-se a direção da colisão. Se o choque for frontal, por exemplo, o deslocamento se dará principalmente em torno do eixo z (vide Figura 2).

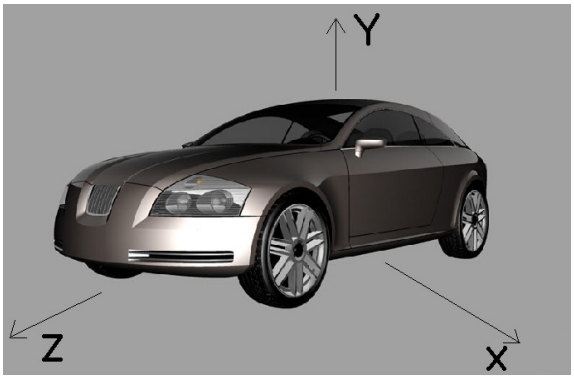

Figura 2: Representação dos eixos locais de um objeto

Após o deslocamento de todos os vértices eleitos como vizinhos do vértice central da colisão, a malha de pontos dos veículos envolvidos no choque é atualizada e o algoritmo chega ao fim. A atualização da malha é necessária para demonstrar o deslocamento dos pontos além de garantir a visualização da deformação para o usuário.

5.Resultados Obtidos

O algoritmo implementado tem por finalidade simular o efeito de deformação em veículos, trazendo o mínimo possível de perda de processamento. A técnica eleita como base se mostrou condizente com os requisitos buscados para implementação e gerou resultados satisfatórios. É importante ressaltar que, embora existam técnicas que sejam capazes de fornecer efeitos de deformação mais realistas, muitas vezes são mais custosas computacionalmente. Se levarmos em conta o custo x benefício do algoritmo desenvolvido, este se mostrou satisfatório no objetivo proposto com perdas bem pequenas de processamento. Abaixo seguem alguns dados mais concretos para verificação dos resultados:

Tabela 1: Número de Frames por Segundo (FPS) no momento de colisão

Colisão	FPS
Sem deformação	82
Com deformação	81

Figura 3: Veículo antes da colisão

Figura 4: Efeito de deformação obtido no projeto

Os testes mostrados acima foram feitos em uma máquina com processador Pentium IV 3.0 GHz, com 512 MB de memória RAM e uma placa de vídeo NVIDIA GeForce4 MX 4000 com 128 MB de memória.

6.Conclusão

Apesar da aparência da deformação não ter sido a melhor possível, tendo em vista o resultado de trabalhos publicados na área, esta se mostrou satisfatória para seu fim. Após a análise das limitações dos games com relação à exigência de processamento, muitas vezes torna-se necessário abrir mão de certa perfeição para chegar a um ponto de equilíbrio ente custo-benefício das técnicas aplicadas nos mesmos.

Referências

AZEREDO, T.R., VIEIRA, P.S., SANTOS NETO, A.A., MATIAS, I.O. *Inteligência Artificial aplicada à tomada de decisão em jogos eletrônicos.* 2006. 77 f. Monografia (Bacharel em Ciência da Computação) – Universidade Candido Mendes – Campos, RJ. 2006.

BONI, G., 2005. *Engines*. Unidev. Disponível em: <http://www.unidev.com.br/artigos/Enginesgustavoboni000.asp>. Acesso em: 15 jul. 2006.

DMA DESIGN, 2001. *GTA3*. Disponível em: <www.rockstargames.com/grandtheftauto3/>. Acesso em: 30 jul. 2006.

GEBHARDT, N., 2004. *Irrlicht Engine*. Disponível em: <http://irrlicht.sourceforge.net>. Acesso em: 20 out. 2005.

HAVOK, 1999. *Havok Physics*. Disponível em: <http://www.havok.com>. Acesso em: 08 jun. 2006.

JEREZ, J., SUERO, A., 2003. *Newton Game Dynamics*. Disponível em: <http://www.newtondynamics.com>. Acesso em: 08 jun. 2006.

JUSTINO, M. J., 2001. *Colisões*. Laboratório Virtual de Física. Disponível em: <http://www.virtual.unilestemg.br/laboratorio/colisoes.html>. Acesso em: 22 mai. 2006.*

LAM, D., 2003. *Tokamak Game Physics SDK*. Disponível em: <http://www.tokamakphysics.com>. Acesso em: 08 jun. 2006.

OGRE3D WIKI, 2000. *OgreODE*. Disponível em: <http://www.ogre3d.org/wiki/index.php/OgreODE>. Acesso em: 16 jun. 2006.

POLICARPO, F., 2003. *Fly3D Game Engine*. Disponível em http://fabio.policarpo.nom.br/fly3d/index.htm. Acesso em 26 jul 2006.

RANGEL JUNIOR, M., GOMES, R.C.B.P., JUDICE, S.F.P.P., MATIAS, I.O. *Simulação computacional aplicada a jogos eletrônicos***.** 2006. 74 f. Monografia (Bacharel em Ciência da Computação) – Universidade Candido Mendes – Campos, RJ. 2006.

REFLECTIONS INTERACTIVE, 1999. *Driver: You Are the Wheelman*.

RODRIGUES, T., PIRES, R., DIAS, J. M., 2005. *D4MD – Deformation system for a vehicle simulation game*. SBGames.

SMITH, R., 2003. *ODE*. Disponível em: <http://www.ode.org>. Acesso em: 01 nov. 2005.

STAINLESS SOFTWARE, 1997. *Carmageddon*, Disponível em: <www.sci.co.uk>. Acesso em: 30 jul. 2006.

TERZOPOLOUS, D., PLATT, J., BARR, A., FLEISCHER, K., 1987. Elastically Deformable Models. ACM SIGGRAPH Computer Graphics. Volume 21, Edição 4, p. 205-214.

THE OGRE TEAM, 2001. *Ogre3D*. Disponível em: <http://www.ogre3d.org>. Acesso em: 20 out. 2005.