

# UID

# ZINE

2ª EDIÇÃO · AGOSTO / 2007

## **3D GAMESTUDIO**

Cartoon Shader  
aprenda como fazer.

## **3D STUDIO MAX**

Aprenda a fazer um  
Normal Map.

## **Combustion no 3D Max 8**

## **SOFTIMAGE | XSI**

Criando um tapete

## **ALLEGRO**

Partículas Simples.

## **SDL**

Animação de Sprites.

**E MAIS...**

Artigos  
Colunas  
Galeria

# índice

**03**

**REDAÇÃO**

PÁG 4 - ERRATA

**05**

**COLABORADORES**

QUEM FEZ ESSA EDIÇÃO

**07**

**GALERIA DE IMAGENS**

CONHEÇA ALGUNS TRABALHOS DO PESSOAL

**09**

**TUTORIAIS**

PÁG. 09 - 3D STUDIO MAX - NORMAL MAP

PÁG. 17 - SDL - ANIMAÇÃO DE SPRITES

PÁG. 21 - ALLEGRO - PARTÍCULAS SIMPLES

PÁG. 23 - SOFTIMAGE | XSI - TAPETE

PÁG. 27 - 3D STUDIO MAX - COMBUSTION

PÁG. 34 - 3D GAMESTUDIO - CARTOON SHADER

**36**

**ARTIGOS**

PÁG. 36 - PROJETO REAL

PÁG. 43- CONCEITOS 3D

**47**

**COLUNA**

NEM HITLER AGUENTA MAIS!



## EDIÇÃO 2 RELOADED...

*BOM PESSOAL, A SEGUNDA EDIÇÃO TÁ AÍ. DEMOROU UM POUCO MAIS DEVIDO A MIGRAÇÃO DO FORUM DA UNIDEV QUE RENDEU BOAS DORES DE CABEÇA PARA OS ADMINS DA COMUNIDADE. MAS NO FINAL DEU TUDO CERTO E O FÓRUM NOVO ESTÁ AÍ! MUITO MELHOR, MENOS BUGS, MAIS SEGURANÇA E MUITO MAIS RÁPIDO.*

*QUERIA AGRADECER A TODAS AS PESSOAS QUE DIVULGARAM E COMENTARAM SOBRE A REVISTA, BEM COMO TODAS AS SUGESTÕES E CRÍTICAS FEITAS. PODEM TER CERTEZA QUE TODAS FORAM MUITO VÁLIDAS. MAS MUITO MAIS DO QUE CRÍTICAS E SUGESTÕES, PRECISAMOS DE APOIO NA ORGANIZAÇÃO DA REVISTA. PEÇO MAIS UMA VEZ A TODOS QUE QUISEREM COLABORAR COM, ARTIGOS, TUTORIAIS, COLUNAS QUE ENTREM EM CONTATO COMIGO OU COM O RODRIGO FLAUSINO ATRAVÉS DOS E-MAILS: MERKEL.DANIEL@GMAIL.COM E RODRIGOFLAUSINO@YAHOO.COM.BR.*

*POR FIM QUERIA AGRADECER O PESSOAL QUE COLABOROU COM ESSA EDIÇÃO. SEM A AJUDA DE VOCÊS, A REVISTA NÃO TERIA SAÍDO.*

*VALEU MESMO!*

*DANIEL MERKEL*



**VAMOS DOMINAR O MLINDO!!!**

## ERRATA DA EDIÇÃO 01

### PÁGINA 3:

O NOME DO USUÁRIO LORD ETERNAL K FOI GRAFADO ERRADAMENTE. É PAULO DE CAMPOS, E NÃO PAULO NASCIMENTO.  
O SOBRENOME CORRETO DO USUÁRIO RODRIGO\_FL AUSINO É GALDINO, E NÃO GAUDINO.  
O NICK DO USUÁRIO RICARDO RINALDI É RGRDESIGNER, E NÃO RGBDESIGNER.

### PÁGINA 5:

A FICHA DO USUÁRIO LORD ETERNAL FOI GRAFADA ERRADAMENTE. SEGUE FICHA CORRETA.

NOME: PAULO DE CAMPOS  
NICK: LORD ETERNAL K  
CIDADE : FLORIANÓPOLIS-SC  
E-MAIL : LORDETERNAL@LORDETERNAL.COM.BR  
AREA DE INTERESSE: 3DS MAX, TGF, Z-BRUSH, POSER...

### PÁGINA 62 A 71

O EMAIL DO USUÁRIO RGRDESIGNER FOI GRAFADO ERRADAMENTE. O CORRETO É RICARDO.RINALDI@GMAIL.COM

### PÁGINA 77

O SEGUNDO GAME É THE LEGEND OF ZELDA - TWILIGHT PRINCESS, E NÃO THE LEGEND OF ZELDA WIISPORTS - TWILIGHT PRINCESS



VAMOS DOMINAR O MLINDO!!!



**NOME:** BRUNO CROCI  
**NICK:** CROCIBD  
**CIDADE:** GUARULHOS - SP  
**E-MAIL:** CROCIBD@HOTMAIL.COM  
**ÁREA DE INTERESSE:** PROGRAMAÇÃO DE JOGOS (C/C++, JAVA) E PROGRAMAÇÃO WEB (PHP, JAVASCRIPT, FLASH)



**NOME:** DANIEL S.C. MERKEL  
**NICK:** MERKEL  
**CIDADE:** PASSO FUNDO / SÃO LEOPOLDO - RS  
**E-MAIL:** MERKEL.DANIEL@GMAIL.COM / DANIELMERKEL@HOTMAIL.COM (MSN)  
**ÁREA DE INTERESSE:** PROGRAMAÇÃO DE JOGOS (C/C++) OPENGL, PROCESSAMENTO DE IMAGENS, MODELAGEM 3D, 3D STUDIO MAX, WEBDESIGN, DESIGN DIGITAL.



**NOME:** FERNANDO TONON DE ROSSI  
**NICK:** KILL GUNS  
**EMAIL:** MISTICGORRAM@HOTMAIL.COM  
**CIDADE:** CUIABÁ - MT  
**ÁREA DE INTERESSE:** ELETRÔNICA E DESENVOLVIMENTO DE JOGOS.



**NOME:** LUIZ FERNANDO ALVES DA SILVA  
**NICK:** ZIZACO  
**E-MAIL:** ZIZACO@GMAIL.COM



HUMM... ELES SÃO BONS!  
MAS NOSSA EQUIPE É MELHOR!!



**NOME:** RODRIGO FLAUSINO GALDINO DE LIMA  
**NICK:** RODRIGO\_FLAUSINO  
**E-MAIL:** RODRIGOFLAUSINO@YAHOO.COM.BR  
**CIDADE:** VARGINHA - MG  
**ÁREA DE INTERESSE:** MODELAGEM 3D, PROGRAMAÇÃO E GAMEDESIGN



**NOME:** RAFAEL MORAIS  
**NICK:** FUNKDELIC  
**CIDADE:** SÃO PAULO - SP  
**E-MAIL:** FUNKDELIC3D@HOTMAIL.COM  
**ÁREA DE INTERESSE:** GAME ART



**NOME:** RICARDO G. RINALDI  
**NICK:** RGRDESIGNER  
**CIDADE:** BETIM - MG  
**E-MAIL:** RICARDO.RINALDI@GMAIL.COM  
**ÁREA DE INTERESSE:** MODELAGEM 3D, ANIMAÇÃO, FX, GAMES.

**NOME:** YVES NADJARIAN  
**EMAIL:** YVES.NADJARIAN@GMAIL.COM  
**CIDADE:** SÃO PAULO / SP  
**ÁREAS:** PROGRAMAÇÃO DE JOGOS EM DARK BASIC, XNA. PROGRAMAÇÃO: VB.NET, C#, C/C++

**NOME:** ISAÍAS



HUMM... ELES SÃO BONS!  
MAS NOSSA EQUIPE É MELHOR!!

NOME: RAFAEL MORAIS  
CIDADE: SÃO PAULO - SP  
AREA DE INTERESSE: GAME ART



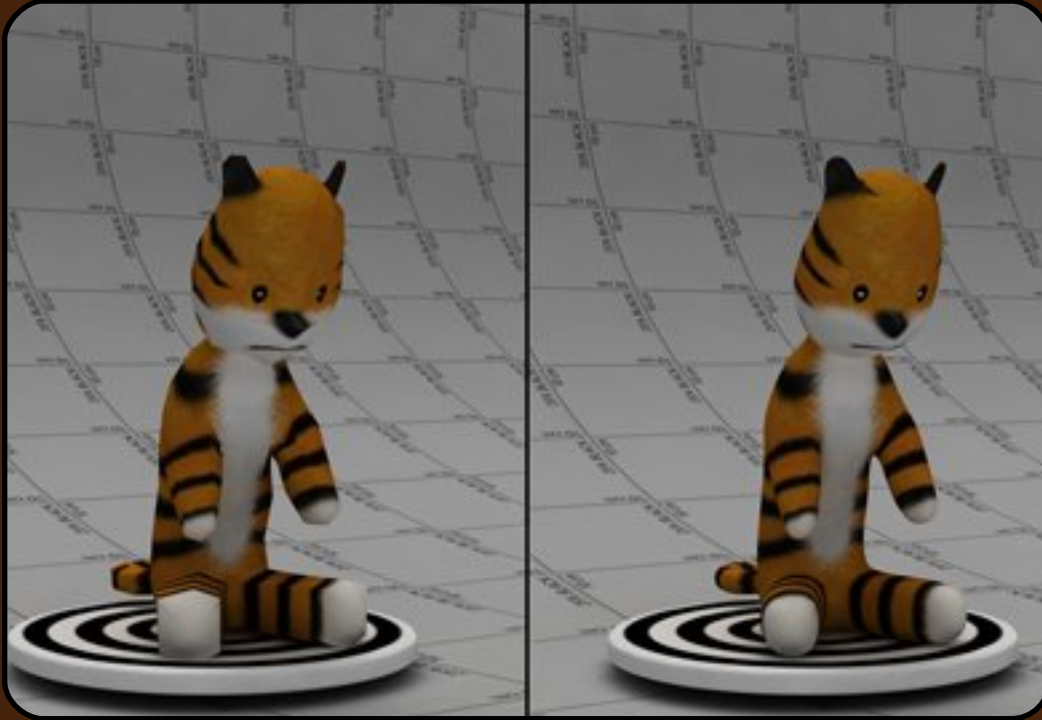
DESCRIÇÃO: NINA LOWPOLY  
SOFTWARE: 3DS MAX

DESCRIÇÃO: ANÃO FERREIRO LOWPOLY  
SOFTWARE: 3DS MAX



TUDO MELH!!  
BUHALHUALHUALHUALHUALH!

NOME: DANIEL MERKEL  
CIDADE: PASSO FUNDO / SÃO LEOPOLDO - RS



DESCRIÇÃO: HAROLDO (LOW E HIGH)  
SOFTWARE: 3D STUDIO MAX

DESCRIÇÃO: PUCCA E GAROU  
SOFTWARE: 3D STUDIO MAX



TUDO MELH!!  
BUHALHUALHUALHUALHUALH!

### **CRIAÇÃO DE NORMAL MAP - PARTE I** POR RAFAEL MORAIS (FUNKDELIC)

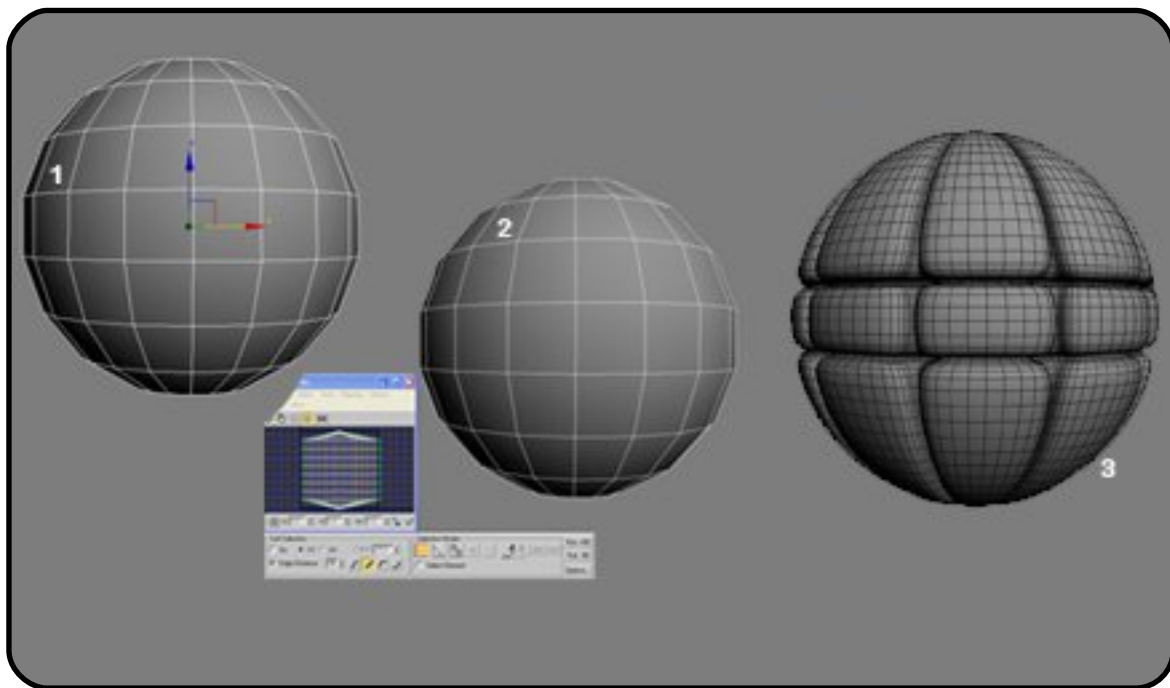


Na primeira parte deste mini tutorial, nos vamos ver o processo de criação de normal map utilizando apenas o 3DStudio Max. Vamos criar um modelo lowpoly e um highpoly e extraíndo o normal map a partir da ferramenta 'RENDER TO TEXTURE'.

Chega de papo, vamos lá!

## CRIAÇÃO DE NORMAL MAP - PARTE I

CONTINUAÇÃO

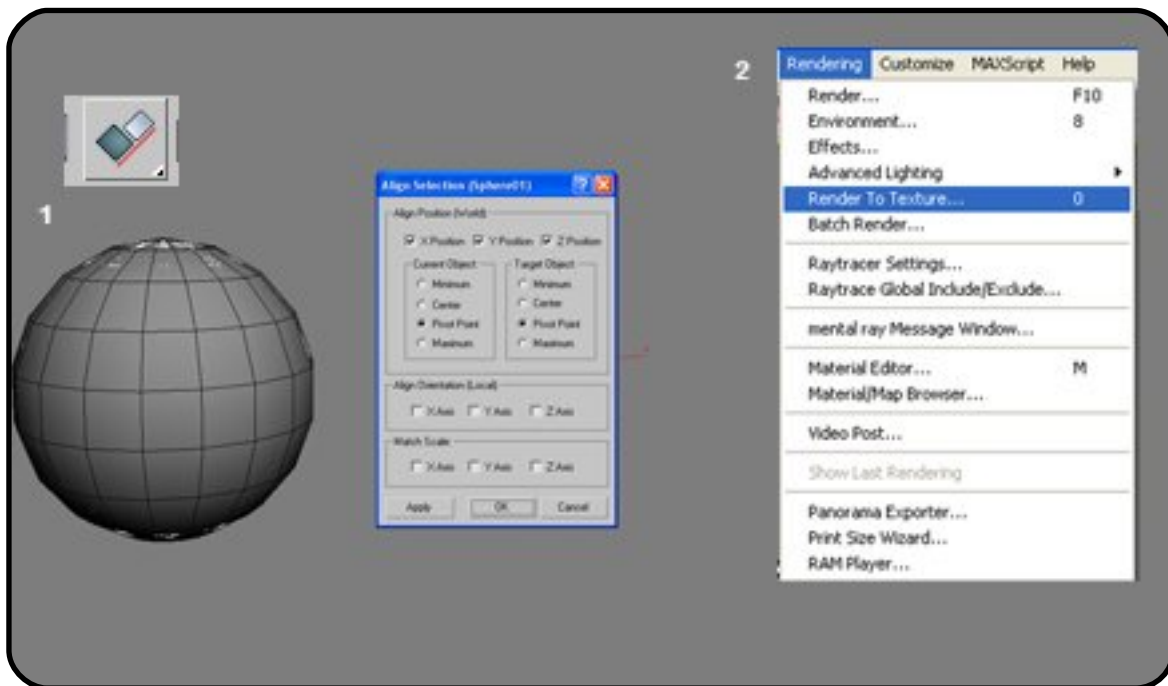


1 - Com o software aberto, crie um primitivo qualquer, eu por exemplo, criei um esfera e apenas diminui um pouco os 'segments' dela, mas sinta-se livre para usar qualquer objeto.

2 - Usando o modificador 'UVW UNWRAP' eu extrai o mapa de UV do meu modelo LOW POLY. Você só precisa do mapa do lowpoly pois é nele que o normal map será utilizado.

3 - Duplicando a esfera e usando o modificador 'TurboSmooth' eu fiz a versão HighPoly e adicionei alguns detalhes simples, para uma melhor visualização no resultado do normal map

## CRIAÇÃO DE NORMAL MAP - PARTE I CONTINUAÇÃO



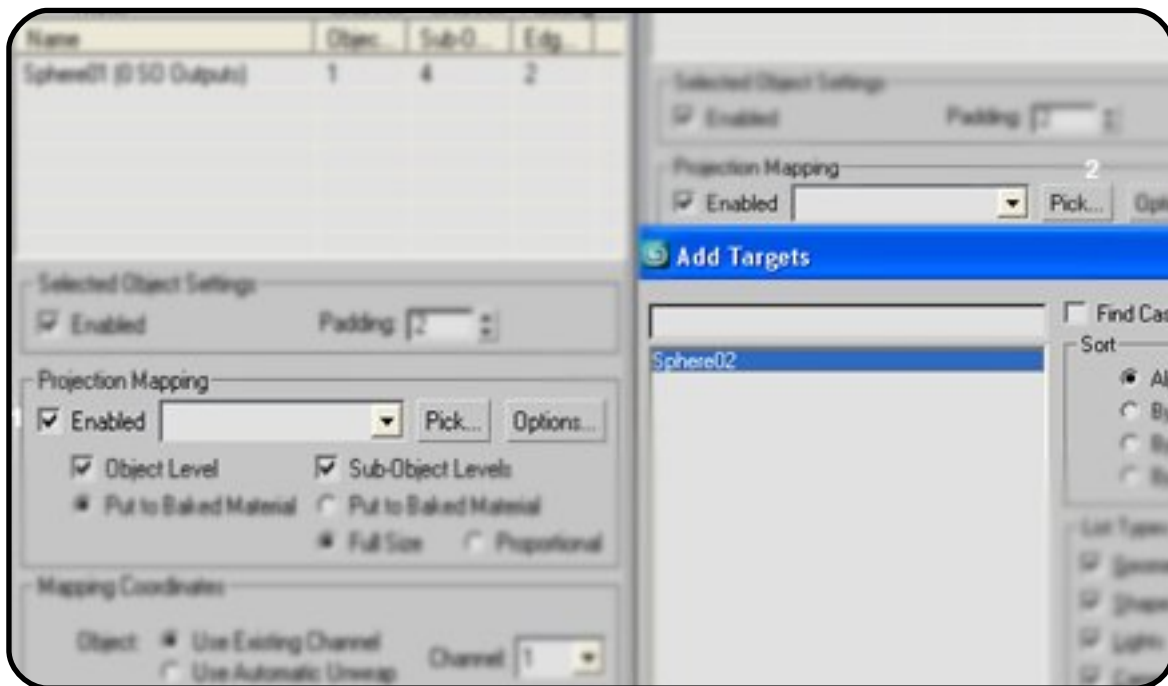
1 - Clique sobre o modelo HighPoly, agora use a ferramenta ALIGN para posicionar o modelo High junto com o Low, de forma que ambos ocupem o mesmo espaço.

2 - Uma vez posicionado, pressione ok para fechar a caixa de dialogo do ALIGN. Agora selecione o MODELO LOWPOLY e aperte 0 ou vá até o menu RENDERING>RENDER TO TEXTURE.

Na primeira vez, você pode ficar meio perdido, pois são varias opções e botões dentro desse menu, mas não "esquente a cabeça", o que agente vai usar é simples, rápido, e não doi tanto assim hehehe :).

## CRIAÇÃO DE NORMAL MAP - PARTE I

CONTINUAÇÃO

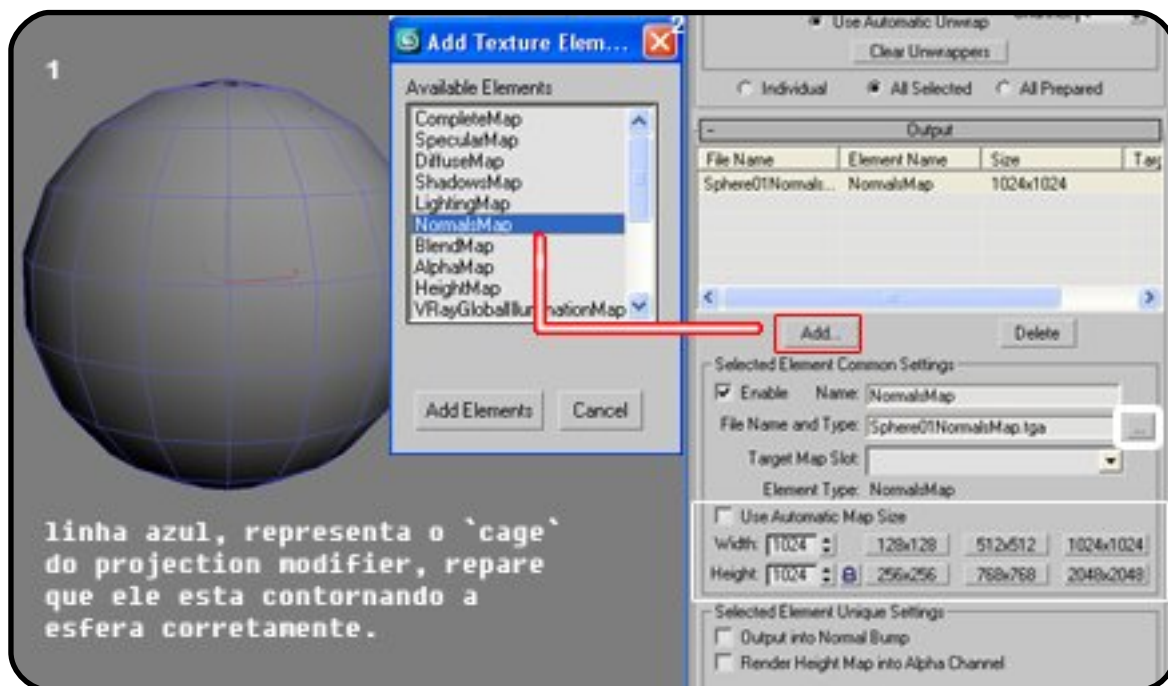


1 - Com o modelo LOWPOLY selecionado, e com o RENDER TO TEXTURE aberto voce vai clicar em PROJECTION MAPPING, e em seguida em PICK.

Agora clique no nome correspondente ao seu modelo HIGHPOLY, clique em ADD para adicionar o modelo HIGHPOLY à projeção.

Agora pode-se notar que foi adicionado um modificador novo, com o nome de PROJECTION, e também que um "cage" azul ficou em volta de seu objeto.

## CRIAÇÃO DE NORMAL MAP - PARTE I CONTINUAÇÃO

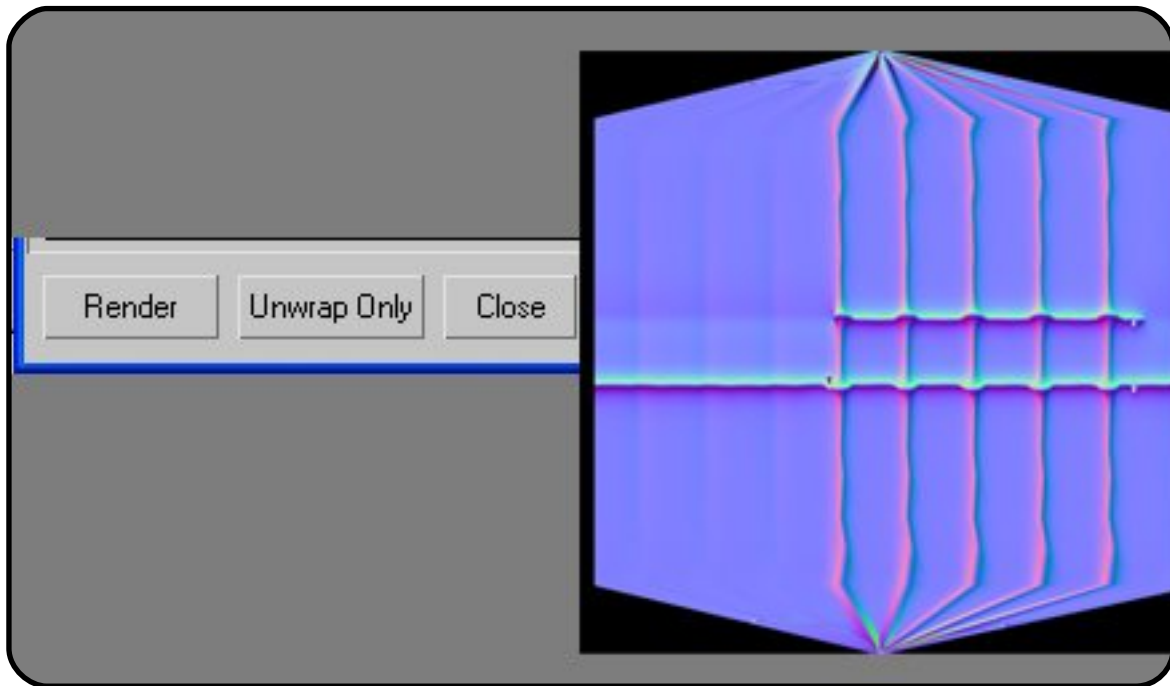


1 - O modificador adicionado serve para você definir como o modelo HIGHPOLY vai ser "representado" pelo LOWPOLY, dentro desse modificador existem opções para voce aumentar ou diminuir a distancia do seu "cage" a melhor forma eh deixar ele o mais proximo possivel e com um formato não tanto bagunçado.

2 - Agora na aba OUTPUT, clique em ADD > NORMALSMAP, ele vai adicionar na lista o mapa a ser criado, mais a baixo voce pode escolher o tamanho do mapa, escolha também o nome e o local onde seu mapa vai ser salvo, de preferência em algum lugar de fácil acesso hehe :)

## CRIAÇÃO DE NORMAL MAP - PARTE I

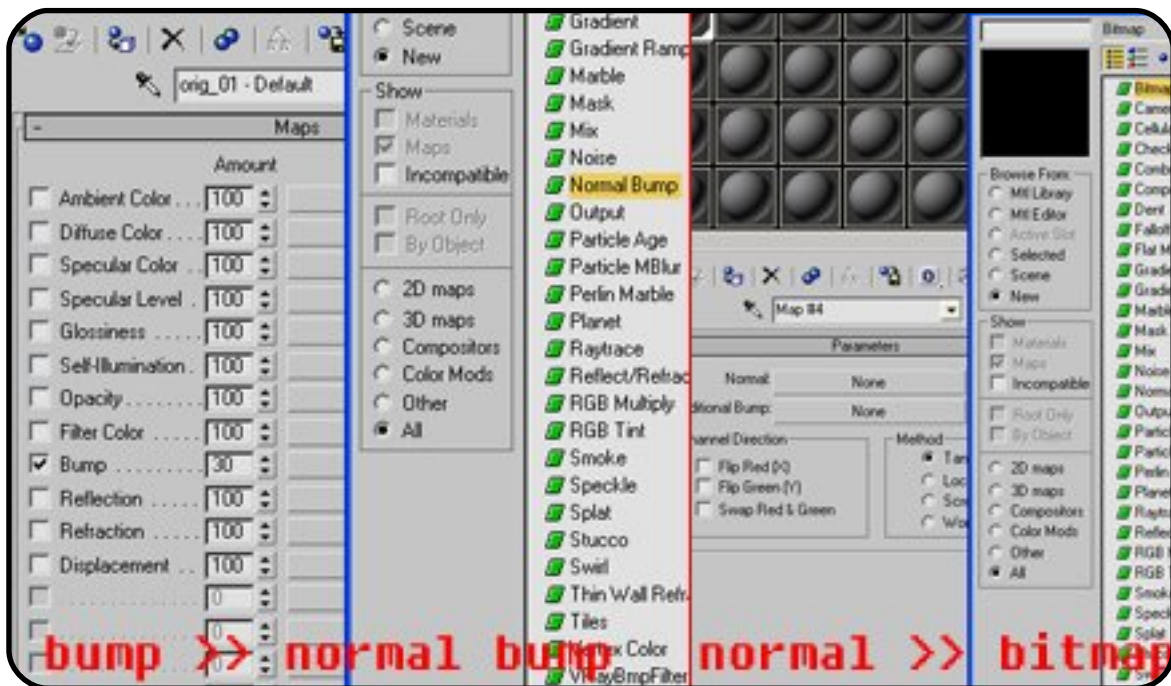
CONTINUAÇÃO



1 - Agora você vai clicar em RENDER, caso aparece uma mensagem, apenas ignore e clica em continue. Você vai notar que uma janela apareceu com seu render nela, novamente, ignore este render pois nao eh o seu normal, é apenas um render normal.

2 - Seu mapa de normal estará salvo no local em que você pré-determinou, no meu caso, a área de trabalho.

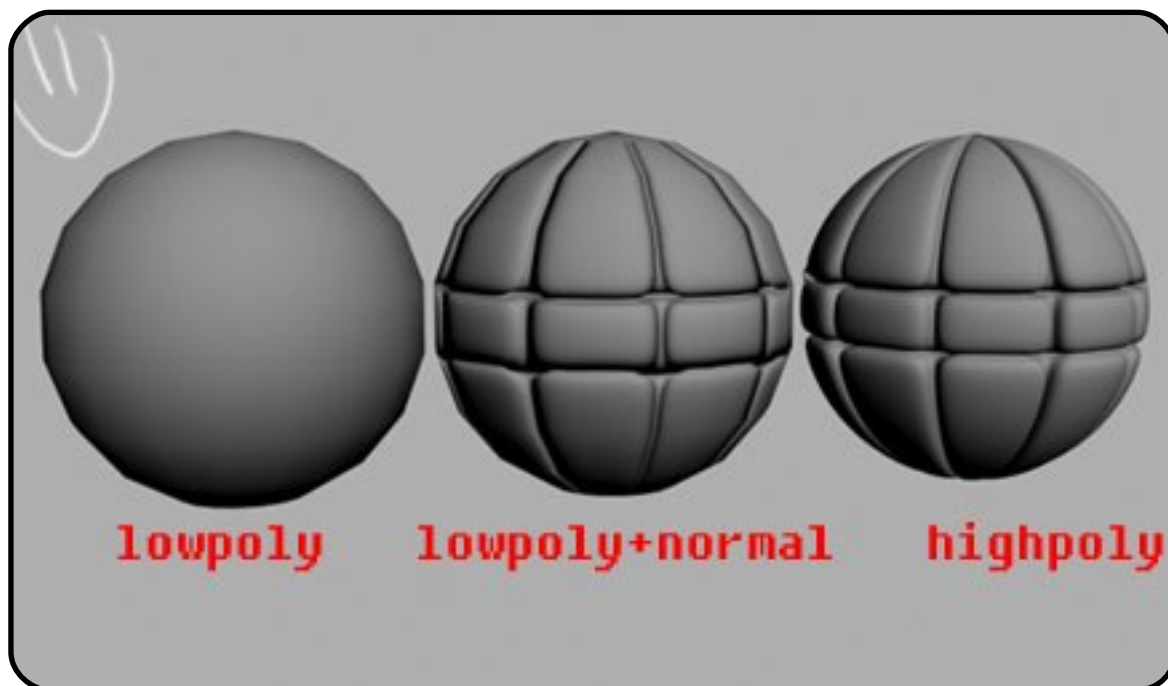
## CRIAÇÃO DE NORMAL MAP - PARTE I CONTINUAÇÃO



1 - Vamos agora aplicar o mapa de normal no seu modelo. Aperte "M" para abrir o MATERIAL EDITOR, clique na aba MAPS > BUMP e adicione o mapa NORMAL BUMP. Dentro do normal bump, clique no slot correspondente ao mapa normal e em seguida escolha o mapa BITMAP, agora adicione o normal que você acabou de gerar.

Mude o valor do bump de 30 para 100 para uma melhor visualização.

**CRIAÇÃO DE NORMAL MAP - PARTE I**  
CONTINUAÇÃO



1 - PRONTO!

E esse é o resultado final do nosso tutorial, pode ver que a qualidade fica muito boa, pois o render to texture do max tira um ótimo normal map, na minha opinião, melhor que de plug-ins como o Zmapper do Zbrush.

Espero que este tutorial esteja claro para todos que vão ler.

Na próxima edição agente vai um pouco mais além e eu vou mostrar como usar o xNormal um software gratuito e muito potente para criação de normais a partir de fotos.

Bom é isso galera!

Qualquer dúvida pode me mandar um email: [funkdelic3d@hotmail.com](mailto:funkdelic3d@hotmail.com)

## ANIMAÇÃO DE SPRITES COM SDL

POR FERNANDO TONON DE ROSSI (KILLGUNS)

Esse tutorial é uma continuação do “SDL - Primeiros Passos” apresentado na 1ª edição da revista.

Nesse tutorial pretendo ensinar como animar as sprites e como detectar entradas do mouse.



### Pré-requisitos:

- Conhecimento básico da linguagem C;
- Conhecimento apresentado no tutorial anterior;
- Compilador com a biblioteca instalada.

Vamos fazer estrelas e bolas seguirem o mouse e girar ou parar quando clicamos um botão do mouse.

Utilizaremos as sprites do site reiner's tilesets, encontradas no endereço:  
<http://reinerstileset.4players.de/animatedE.htm>.

### Carregando as sprites:

O primeiro passo é carregar as sprites, para isso utilizaremos a função `SDL_LoadBMP`.

```
SDL_Surface *Estrela_Bola[36];
char sprite[20];

for(int m = 0; m < 36; m++)
{
    sprintf(sprite, "groggy00%d.bmp", m);
    Estrela_Bola[m] = SDL_LoadBMP(sprite);
};
```

Vamos deixar o fundo transparente, o fundo dessas imagens é azul claro (220,239,255).

```
for(int m = 0; m < 36; m++)
{
    SDL_SetColorKey(Estrela_Bola[m], SDL_SRCCOLORKEY | SDL_RLEACCEL,
    SDL_MapRGB(Estrela_Bola[m] ->format, 220,239,255));
};
```

### Verificando ações do mouse:

Agora já temos todas as sprites carregadas, vamos detectar a posição do mouse com a função `SDL_GetMouseState`.

```
int X=0,Y=0;
SDL_GetMouseState(&X,&Y);
SDL_Rect rect_imagem;
rect_imagem.x = X;
rect_imagem.y = Y;
```

## ANIMAÇÃO DE SPRITES COM SDL

### CONTINUAÇÃO

Vamos criar uma pilha de eventos para gerenciar os eventos do mouse.

```
SDL_Event evento;
bool Botao = false;

if(SDL_PollEvent(&evento))
{
    switch(evento.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            Botao=true;
            break;
        case SDL_MOUSEBUTTONUP:
            Botao=false;
            break;
    }
}
```

Imprimindo a animação:

Já sabemos a posição do mouse e qual botão foi clicado, agora basta fazermos a animação.

Para isso vamos declarar uma variável que servirá de contador e vai registrar em qual posição está a animação.

```
int Anim=0;
```

Agora vamos aumentar o valor se o mouse estiver liberado e não aumentar se algum botão estiver pressionado.

```
if(Botao==false)
{
    if(Anim<35)
        Anim++;
    else
        Anim = 0;
}
```

Já fizemos o controle da animação, agora vamos imprimi-la na tela e pronto!

`SDL_FillRect(Tela, 0, SDL_MapRGB(Tela->format, 0, 0, 0));` // Imprime um quadrado preto no fundo da tela, isso define a cor do fundo.

```
SDL_BlitSurface(Estrela_Bola[Anim],NULL,Tela,&rect_imagem);
SDL_Flip(Tela);
```

## ANIMAÇÃO DE SPRITES COM SDL

### CONTINUAÇÃO

Código completo utilizado nesse tutorial (continuação do tutorial apresentado na primeira edição):

```
#include <SDL.h>

int main(int argc, char** argv)
{
    SDL_Init(SDL_INIT_VIDEO|SDL_INIT_AUDIO);

    atexit(SDL_Quit);

    SDL_Surface *Tela;

    Tela = SDL_SetVideoMode(800, 600, 16, SDL_SWSURFACE|SDL_FULLSCREEN);

    SDL_Surface *Estrela_Bola[36];
    char sprite[20];

    for(int m = 0; m < 36; m++)
    {
        sprintf(sprite, "groggy00%d.bmp", m);
        Estrela_Bola[m] = SDL_LoadBMP (sprite);
    };

    for(int m = 0; m < 36; m++)
    {
        SDL_SetColorKey(Estrela_Bola[m], SDL_SRCCOLORKEY | SDL_RLEACCEL ,
            SDL_MapRGB(Estrela_Bola[m] ->format, 220,239,255));
    };

    int X=0,Y=0;
    SDL_GetMouseState(&X,&Y);
    SDL_Rect rect_imagem;
    rect_imagem.x = X;
    rect_imagem.y = Y;

    SDL_Event evento;
    bool Botao = false;

    if ( SDL_PollEvent ( &evento ) )
    {
        switch(evento.type)
        {
            case SDL_MOUSEBUTTONDOWN:
                Botao=true;
                break;
            case SDL_MOUSEBUTTONUP:
                Botao=false;
                break;
        }
    }
}
```

## ANIMAÇÃO DE SPRITES COM SDL

CONTINUAÇÃO

```
int Anim=0;

if( Botao == false)
{
    if(Anim<35)
        Anim++;
    else
        Anim = 0;
}

// Imprime um quadrado preto no fundo da tela, isso define a cor do fundo.
SDL_FillRect(Tela, 0, SDL_MapRGB(Tela->format, 0, 0, 0));

SDL_BlitSurface(Estrela_Bola[Anim],NULL,Tela,&rect_imagem);

SDL_Flip(Tela);
return 0;
}
```

### Referências Bibliográficas:

SDL API - SDL Documentation Wiki – Disponível em:

<[http://www.libsdl.org/cgi/docwiki.cgi/SDL\\_20API](http://www.libsdl.org/cgi/docwiki.cgi/SDL_20API)> Acesso em: 21/05/2007.

Reiner's tilesets – Disponível em: <<http://reinerstileset.4players.de/animatedE.htm>> Acesso em: 02/06/2007.



ENTENDELI?  
OU QUER QUE EU DESENHE?

## PARTICULAS SIMPLES EM ALLEGRO

BRUNO CROCI (CROCIDB)



Olá pessoal, tudo bom? Hoje venho apresentar um tutorial simples de Partículas com efeito de chuva, em Allegro e C. Com ele é possível entender o conceito de partículas, e assim, deixá-las mais sofisticadas para colocar em seu jogo.

Bom, chega de papo e vamos para o que interessa!

Primeiramente precisamos ter uma partícula como um objeto, que tem posição e velocidade, veja a seguinte estrutura:

```
/* Estrutura da partícula */
typedef struct _tParticle
{
    int x, y; // Posicao atual da partícula
    int vel; // Velocidade com que ela se desloca
}tParticle;
```

A partir desta estrutura, criamos um vetor de partículas, assim:

```
/* Vetor de particulas */
tParticle parts[NUM_PARTICLES];
```

Precisamos também de um bitmap, que conterá a imagem da partícula, você pode carregar ele de um arquivo pré-definido, ou fazer ele com o allegro mesmo, eu criei uma função que desenha a partícula e retorna o bitmap, veja:

```
BITMAP* doParticle()
{
    /* Cria bitmap */
    BITMAP* teste = create_bitmap(1, 2);
    /* Pinta ele de uma cor */
    clear_to_color(teste, makecol(200,200,255));
    /* Retorna ele */
    return teste;
}
```

Criei um bitmap global:

```
/* BITMAP que contem a imagem da partícula */
BITMAP *particle;
```

E então desenhei a partícula nele:

```
/* Chama a funcao que desenha a partícula */
particle = doParticle();
```

Agora nós criamos as partículas e damos a elas posições aleatórias, veja a função create\_particle:

## PARTICULAS SIMPLES EM ALLEGRO

CONTINUAÇÃO

```
void create_particle()
{
    /*
     * Aqui ele inicia todas as particulas com posicoes
     * e velocidade aleatórias.
     */
    for (i = 0; i < NUM_PARTICLES; i++)
    {
        parts[i].x = rand() % 640;
        parts[i].y = rand() % 480;
        parts[i].vel = (rand() % 3) + 1;
    }
}
```

Agora vamos dividir os processos das partículas em partes, atualizar suas posições e velocidade e exibir na tela, `update_particle()` e `render_particle()` respectivamente:

```
void update_particle()
{
    /* Atualiza todas as particulas */
    for (i = 0; i < NUM_PARTICLES; i++)
    {
        parts[i].y += parts[i].vel;
        parts[i].vel = (rand() % 2) + 1;
        if (parts[i].y > 480)
        {
            parts[i].x = rand() % 640;
            parts[i].y = rand() % 100;
            parts[i].vel = (rand() % 3) + 1;
        }
    }
}

void render_particle()
{
    /* Exibe-as na tela */
    for (i = 0; i < NUM_PARTICLES; i++)
    {
        draw_sprite(buffer, particle, parts[i].x, parts[i].y);
    }
}
```

No `update_particle` nós incrementamos ao Y de cada partícula a sua velocidade atual, se sua posição verticalmente for maior que o tamanho da tela, então ela não aparece mais, aí mandamos ela lá pra cima, para poder cair novamente.

E no `render_particle` apenas desenhamos na tela um bitmap `particle` para cada partícula em suas respectivas posições.

Aí basta chamar essas funções no loop principal. Veja o exemplo que vem junto com a revista.

Viram como trabalhar com partículas não é tão difícil? Este exemplo é bem simples, mas a partir dele você pode fazer efeitos de neve, efeitos de sangue, vazamento de água, bom, use a sua criatividade e conhecimentos de física (para os deslocamentos das partículas).

Qualquer dúvida, poste no fórum da UniDev! Obrigado.

**TUTORIAL PARA SOFTIMAGE | XSI - CRIANDO UM TAPETE COM HAIR**

POR RICARDO G. RINALDI (RGRDESIGNER)



Para criarmos o tapete utilizaremos a ferramenta Hair no intuito de simular suas fibras. Para tal, utilizaremos um mapa de texturas para os parâmetros de cores dos fios.

Antes de qualquer coisa, abra uma nova cena.

- Na Barra de Ferramentas “Model” (tecla de atalho “1”), crie um cubo (Get > Primitive > Polygon Mesh > Cube);

- Na janela Cube, altere o valor de Length para 1;

- Na Barra de Comandos, em Transform, nos campos referentes à escala, informe os valores 5, 1 e 3 respectivamente para os eixos X, Y e Z – os valores são referentes às dimensões do tapete de 5,0 m x 3,0 m x 1,0 m de espessura;

Como um tapete não pode ter espessura de 1,0 m, temos que realizar uma “extrusão” da face superior no eixo Y negativo. Faça assim:

- Com a ferramenta de seleção de faces por “raycast” (tecla “U”), selecione a face superior;

- Com a face superior selecionada, utilize o comando de translação para o eixo Y de modo que a espessura do objeto seja bem fina;

- Atribua ao objeto referente ao tapete, a projeção de textura XZ (Get > Property > Texture Projection > XZ);

- Atribua o material Lambert ao objeto (Get > Material > Lambert);

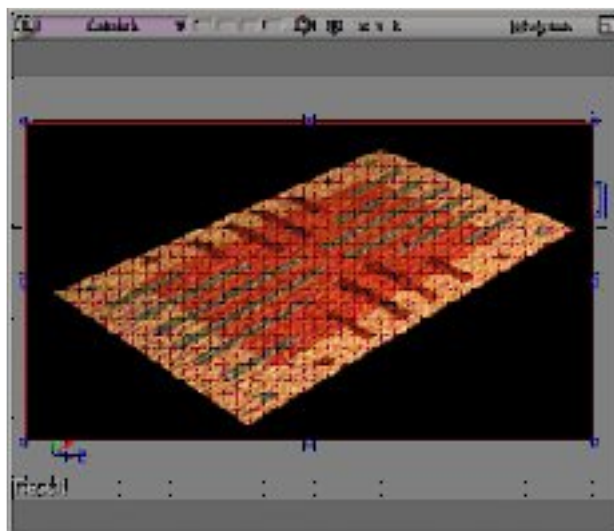
- Na janela Lambert, atribua uma imagem para Difuse (clique no ícone de “tomada” do lado direito das palhetas de cores de Difuse e escolha Image) – veja a baixo:



## TUTORIAL PARA SOFTIMAGE | XSI - CRIANDO UM TAPETE COM HAIR

CONTINUAÇÃO

- Na janela Image, clique em New > New from File... e escolha uma figura que se adeque ao tapete;
  - Novamente na janela Lambert, clique no ícone de “tomada” do lado direito das palhetas de cores de Ambient e escolha Clips > Tapete\_tga;
  - Adicione uma Point Light na cena, pouco acima do objeto referente ao tapete;
- O objeto “Tapete” deve se apresentar com o aspecto semelhante ao da ilustração ao lado:



- Alterne para a Barra de Ferramentas “Hair” (teclas Ctrl + 2);
- Novamente com a face superior do objeto selecionada, vá em Create > Hair > From Selection;
- Vá em Display > Render Hair > Show Render Hair;
- Re-escale o tamanho dos fios (Modify > Scale);
- Na janela Hair, na aba General, altere o valor de Render Settings > Total Hairs para 50000 – caso tenha fechado a janela, basta pressionar “Enter” com o objeto “Hair” selecionado;
- Ainda na mesma aba, em Hair Multiplicity > Strand multiplier, altere o valor do campo para 2,0;
- Na aba Effects, em Thickness, altere o valor do campo Root para 1,5 e do campo Tip para 0,5;
- Feche a janela Hair;

Temos agora que conectar um mapa de textura aos parâmetros de cores de Hair.

## TUTORIAL PARA SOFTIMAGE / XSI - CRIANDO UM TAPETE COM HAIR CONTINUAÇÃO

Faça da seguinte forma:

- Selecione o objeto emissor de Hair e vá em Get > Property > Texture Map > Texture Map, na Barra de Ferramentas “Model”;

- Na janela Texture Map, selecione um arquivo de imagem para o mapa da seguinte forma:

- No campo referente a Clip, escolha a imagem referente à textura para o tapete;

- Abra o Explorer e expanda a árvore do objeto emissor de Hair, para que seja localizado o ícone Texture Map;

- Ainda no Explorer, selecione o objeto emissor de Hair e o objeto Hair – para selecionar mais de um objeto no Explorer, mantenha a tecla “Ctrl” pressionada;

- Alterne novamente para a Barra de Ferramentas “Hair” (Ctrl + 2);

- Clique no botão Transfer Map (Modify > Transfer Map);

- No Explorer, clique sobre Texture Map do objeto emissor, como mostrado ao lado:

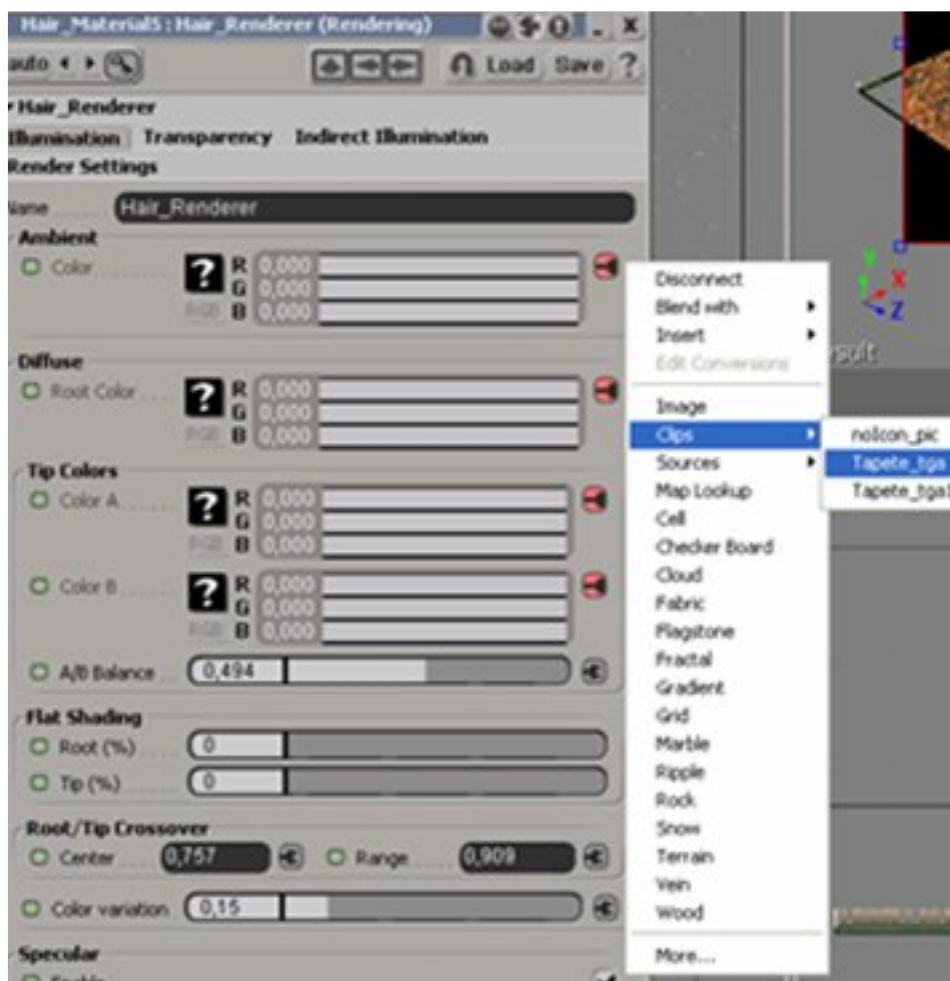
- Selecione apenas o objeto Hair;

- Na Barra de Ferramentas “Hair”, vá em Modify > Shader;

- Em Hair\_Renderer, clique no ícone de “tomada” do lado direito das palhetas de cores de Ambient > Color e escolha Clips > Tapete\_tga (ou no nome da textura que você está utilizando) – veja ao lado:

- Repita o procedimento para Diffuse > Root Color e para Tip Colors > Color A e Color B;

- No campo Flat Shading, informe o valor 15 para Root (%) e 75 para Tip (%);



## TUTORIAL PARA SOFTIMAGE / XSI - CRIANDO UM TAPETE COM HAIR CONTINUAÇÃO

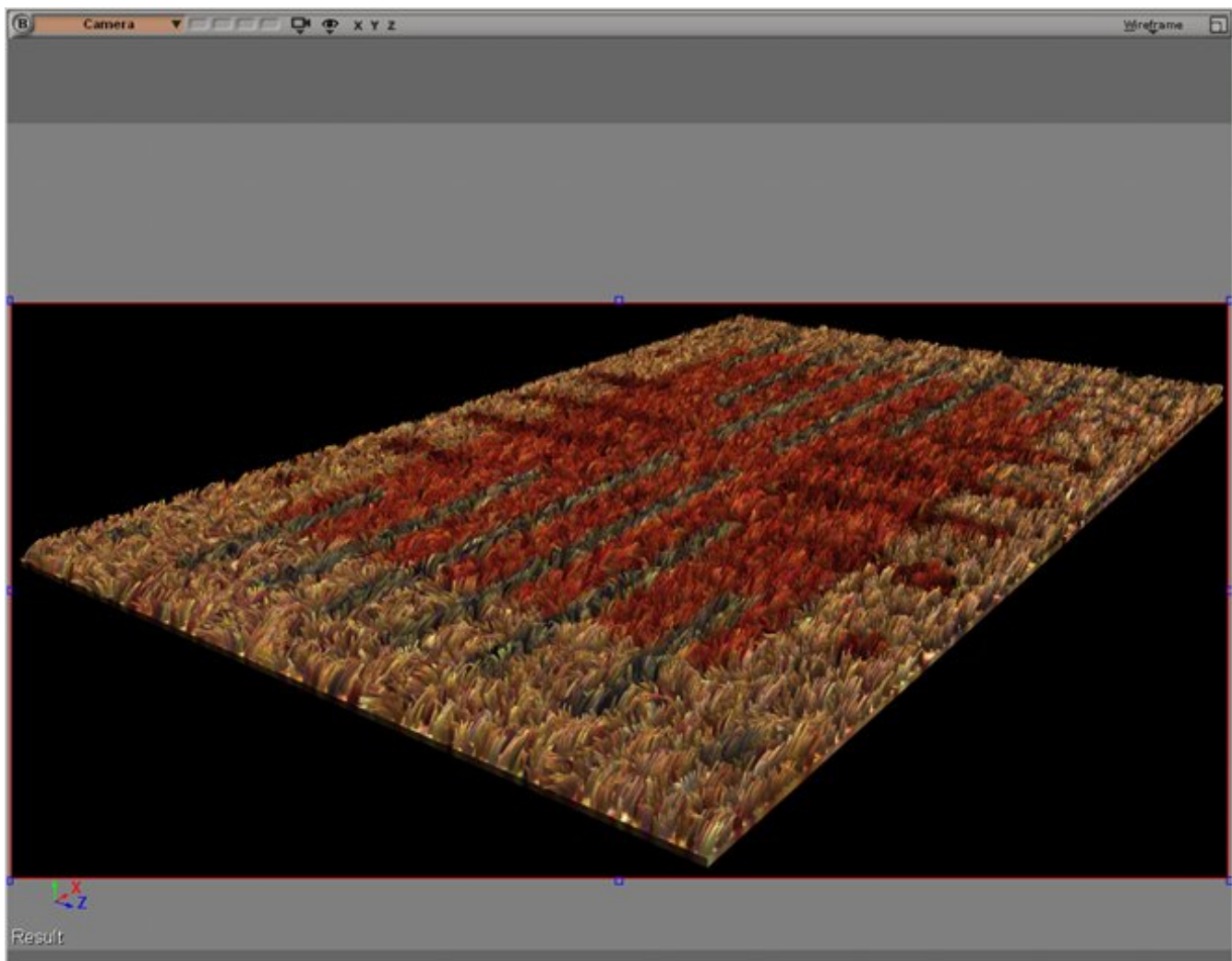
- No campo Specular, desmarque a caixinha referente a Enable;
  - Na aba Transparency, desabilite Enable;
  - Na aba Render Settings, informe o valor 1 para o campo Normal Blend;
- Pronto! A modelagem do tapete está finalizada.

O aspecto final da modelagem “renderizada” deve se assemelhar com o da figura ilustrada abaixo:

É importante lembrar que o uso da ferramenta Hair deve ser usado com parcimônia, pois seu uso acarreta num aumento de pontos a serem levados em conta pelo software e, por sua vez, gera mais cálculos e maior tempo de render.

Uma correta texturização e um uso adequado de Bump Mapping podem substituir o uso de Hair, dependendo do tipo de exposição da cena.

O interessante desta aplicação é a coloração variada de cada fio, ganha graças ao mapeamento da textura.



**COMBUSTION EM 3DS MAX 8**

ISAÍAS

Fala aí galera! Tudo tranqüilo? Este é o primeiro tutorial de uma série de artigos que estarei abordando sobre modelagem e animação em 3Ds Max 8. Segue abaixo a representação:

Tutorial 1: Explosão 3D realísta (Combustion);

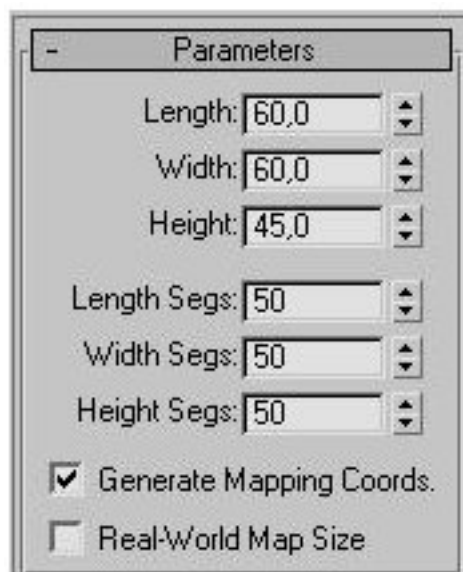
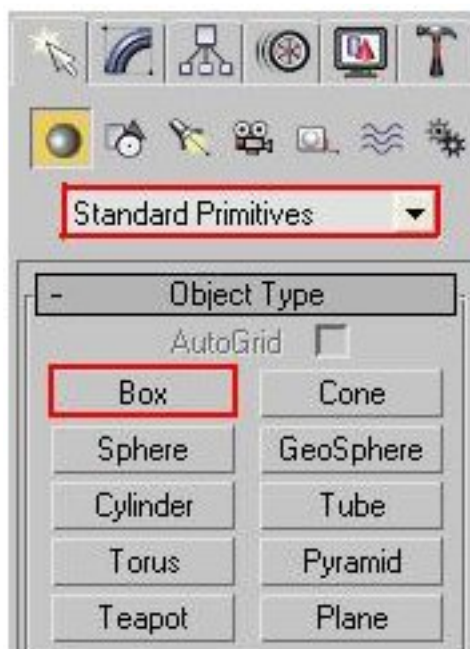
Tutorial 2: Sistema de partículas;

Tutorial 3: Modificadores;

Tutorial 4: Efeitos especiais;

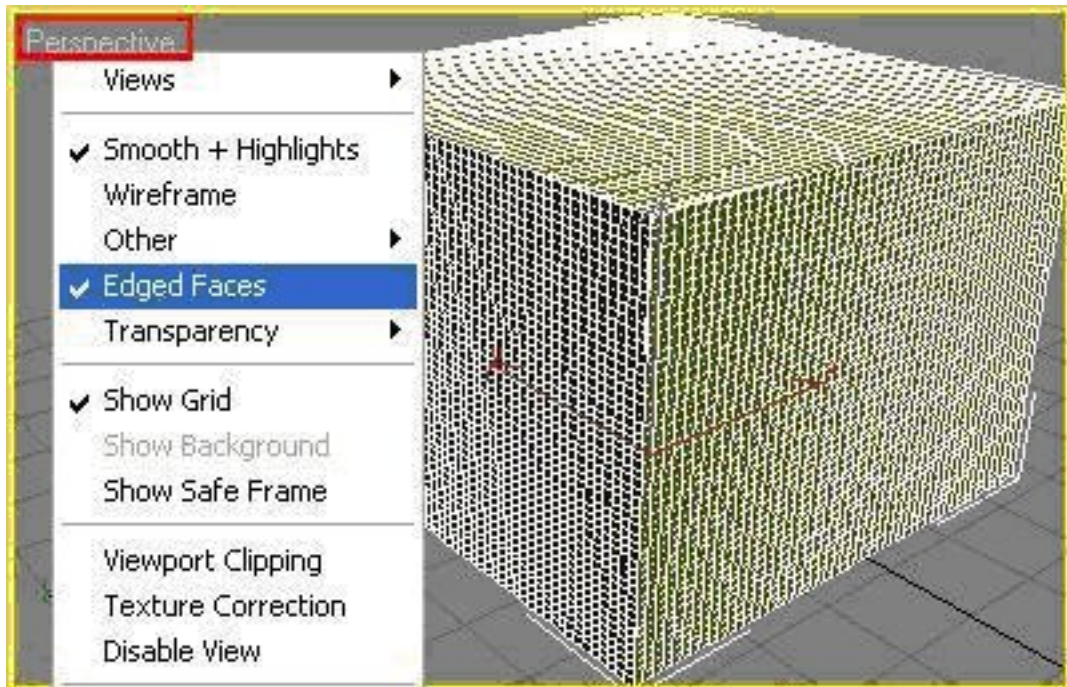
Para esta edição da udzine, preparei um material falando sobre como criar explosões tridimensionais realistas no 3ds max. Este efeito pode perfeitamente ser adaptado nos mais diversos projetos, por exemplo, eu, desenvolvi uma animação de uma nave espacial explodindo. Vou tentar apresentar de um modo mais dinâmico e simples, para até mesmo quem nunca usou o max conseguir acompanhar.

Ótimo, vamos lá. Para iniciar, crie um box na Viewports perspective, com as dimensões e números de segmentos representado na figura abaixo:

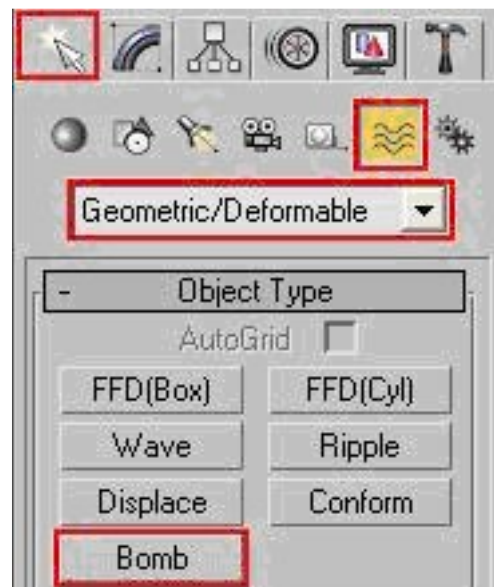


## COMBUSTION EM 3DS MAX 8 CONTINUAÇÃO

Clicando com o botão direito do mouse em perspective, e marcando edge faces você terá essa visão do objeto criado.



Excelente, agora vamos criar a bomba, que será responsável por deformar o nosso objeto em pedaços. Vá ao menu Create, clique Space Warps , selecione Geometric/Deformable e opte por bomb. Desenhe a bomb bem no centro de seu box.



## COMBUSTION EM 3DS MAX 8

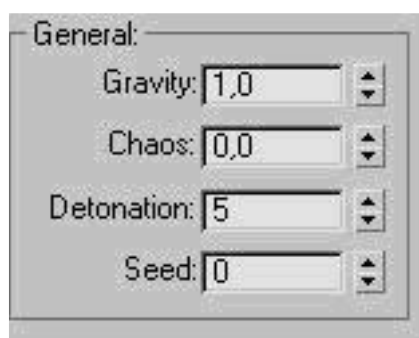
CONTINUAÇÃO

Bom, para não abordarmos assuntos que não é de nosso interesse, não vamos fazer a texturização da caixa e nem aplicar luzes. Presumo que você esteja familiarizado com estes detalhes. Caso contrário, entre em contato que o ajudarei!

Voltando ao assunto. Agora é necessário linkar a nossa bomba ao nosso objeto criado, certo? Para isso vá ao toolbar superior e clique em bind Bind to space Warp, selecione a bomb e arraste marcando a caixa. Pronto! A caixa já sofrerá as ações da bomb.



Para testá-la, clique em play e veja a explosão, estas partículas que voam bruscamente, são conseqüências dos números de segmentos que deixamos acima 50, 50, 50. Selecione a bomb, e vá ao painel modify, e altere os valores de gravity, chãos, seed, o campo detonation provavelmente estará em 5, ou seja, a explosão começará no frame 5, fique a vontade para alterar

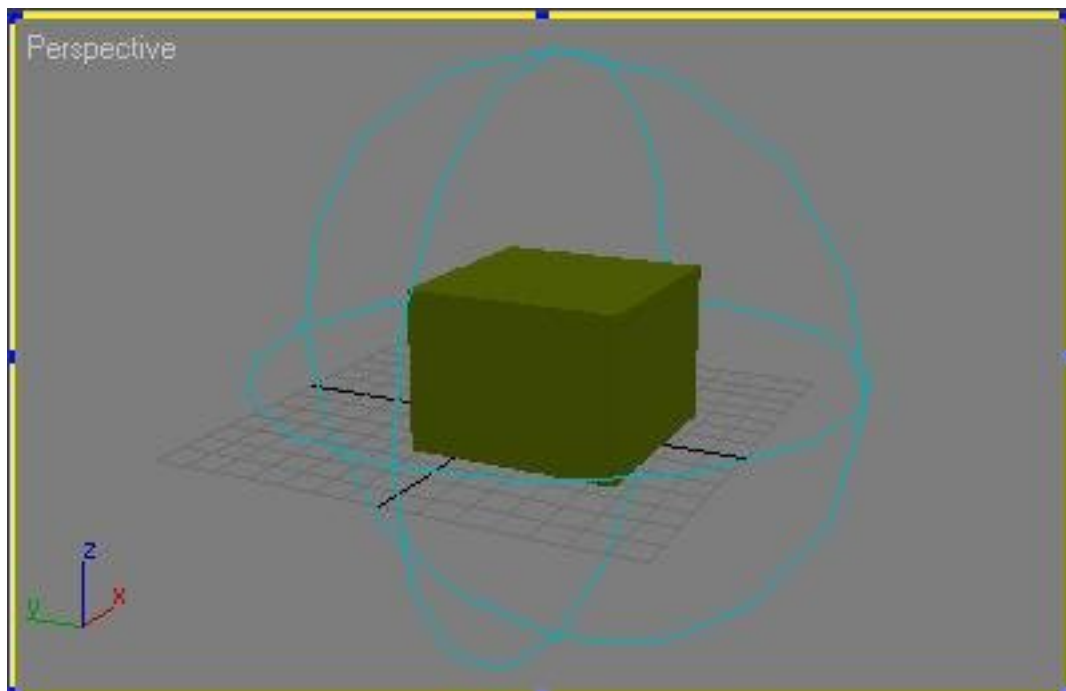
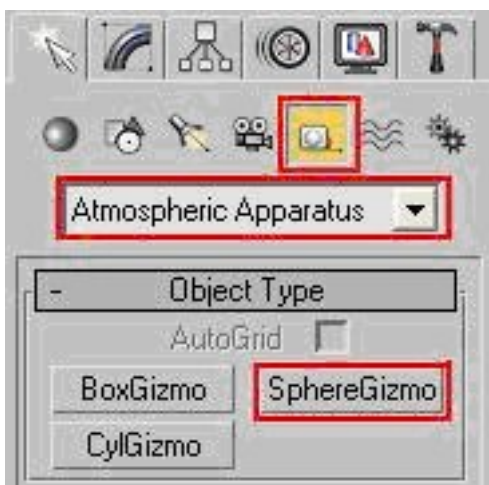


Ótimo, excelente. Já temos o controle de nossa explosão, agora o que nos falta é o fogo para passar a impressão de explosão.

## COMBUSTION EM 3DS MAX 8

CONTINUAÇÃO

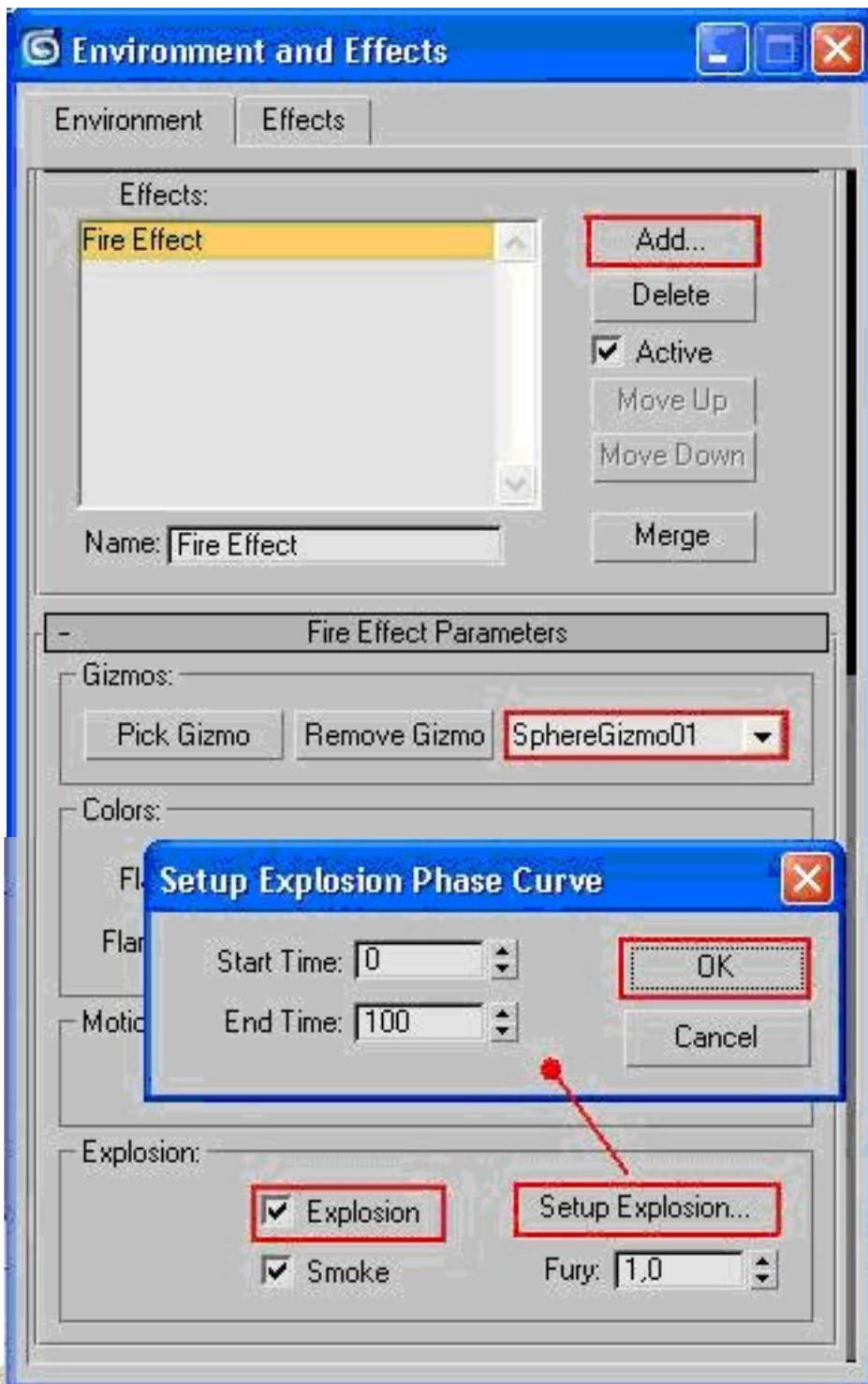
Este efeito é criado com ajuda de um objeto Helper chamado Gizmo. Para criar este objeto, vá até o painel de controle selecione Create, depois helper e selecione atmospheric apparatus, e opte por SphereGizmo. Desenhe a SphereGizmo de modo que cubra a caixa. O raio dela é muito importante, pois ela será responsável pelos danos



## COMBUSTION EM 3DS MAX 8

CONTINUAÇÃO

Feito isso, temos que habilitar o 3ds max para o fogo ser exibido, vamos então até Rendering/Environment, ou simplesmente tecle 8, adicione Fire Effect, ou seja, o fogo; marque Pink Gizmo, tecle H, e selecione SphereGizmo01, logo mais abaixo, marque a opção explosion, entre em setup explosion (esta parte é indispensável) e clique em ok. Desmarcando a opção smoke podemos tirar o efeito de fumaça, lembrando que podemos também alterar a fury da explosão. Observe a figura a seguir:

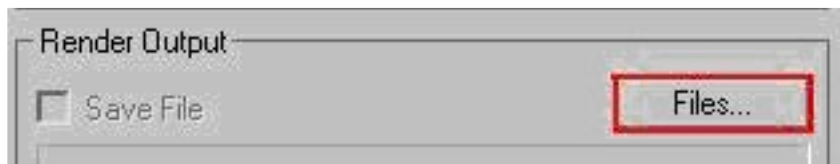
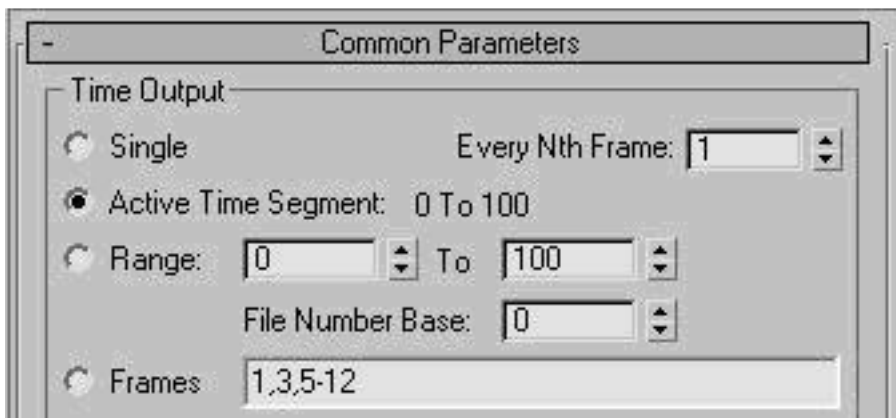


## COMBUSTION EM 3DS MAX 8

CONTINUAÇÃO

Ótimo, pressione F9 no frame 15, e verá um bloco se partindo e o fogo o envolvendo...

Agora, se quiser renderizar a animação com seus 100 frames, (é claro que vai querer), entre em Render Scene Dialog, ou pressione F10. Marque a opção Active Time Segment, e verifique se as opções effects e Atmospherics estão marcadas.



Salve essa animação em uma pasta vazia como seqüência de imagens TGA. Agora poderá estar se perguntando: Mas imagens? Queria um vídeo! Calma! Este método tem a vantagem de renderizar tudo em arquivos de imagens, caso pisque a energia, ou dê qualquer outro problema lá no frame 95, você não terá que renderizar tudo novamente, é só marcar a caixa Range e marcar o frame que quer renderizar, no caso o frame 95 a 100. Depois de renderizado, abra a pasta e veja um monte de arquivos, digo, 100 tga. Para criarmos um vídeo dele é simples, vamos usar um programa gratuito e excelente pra esse tipo de trabalho, o VirtualDub, faça o download. Depois de instalar, abra o programa, vá em file e escolha open vídeo file, escolha o arquivo tga inicial e dê ok.

Para o seu vídeo final não ficar muito grande e pesado, vá ao menu vídeo e escolha compression, escolha um codec que lhe agrade, agora é só ir ao file e clicar em Salve As AVI, e deixe renderizar por poucos segundos.

## COMBUSTION EM 3DS MAX 8 CONTINUAÇÃO

Pronto, sua explosão já está feita!!!

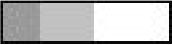
Agora lembre-se, não se limita a isso, fique livre pra criar texturas, iluminação, um céu, etc's. Veja como poderás chegar em um cenário trabalhado, e claro, consumirá um maior tempo de render.



**3D GAMESTUDIO - CARTOON SHADER**

DANIEL MERKEL (MERKEL.DANIEL@GMAIL.COM)

Para utilizar o efeito cartoon em seus modelos no 3d gamestudio 6.22 (ou superior) cole o código abaixo no seu arquivo wdl principal e utilize uma imagem semelhante a esta abaixo com o nome de toonlookup.png.



Não esqueça de colocar a ação "action cartoon" no seu modelo lá no wed.

```

action cartoon
{
my.material=mtl_toon;
}

bmap bmp_toonlookup=<toonlookup.png>;

material mtl_toon
{
    skin1=bmp_toonlookup;
    effect =
    "
matrix matWorldViewProj;
matrix matWorld;vector vecSunDir;
texture entSkin1;texture mtlSkin1;
technique t0{
pass p0 // shade it
{
    zwriteenable= true;
    zenable = true;
    Texture[0]=<mtlSkin1>;
    ColorOp[0]=Modulate;
    ColorArg1[0]=Texture;
    ColorArg2[0]=Texture;
    AddressU[0]=Clamp;
    AddressV[0]=Clamp;
    AddressW[0]=Clamp;
    TexCoordIndex[0]=0;
    Texture[1]=<entSkin1>;
    ColorOp[1]=Modulate;
    ColorArg1[1]=Texture;
    ColorArg2[1]=Current;
    TexCoordIndex[1]=1;
    VertexShaderConstant[0] = <matWorldViewProj>;
    VertexShaderConstant[4] = <matWorld>;
    VertexShaderConstant[7] = <vecSunDir>;

```

## 3D GAMESTUDIO - CARTOON SHADER

CONTINUAÇÃO

```

VertexShader =      asm
    {
        vs_1_0
        dcl_position  v0
        dcl_normal    v3
        dcl_texcoord0 v7
        dcl_texcoord1 v8
        mov oT2.xy, v7
        dp4 oPos.x, c0, v0
        dp4 oPos.y, c1, v0
        dp4 oPos.z, c2, v0
        dp4 oPos.w, c3, v0
        dp3 r0.x, c4.xyz, v3.xyz
        dp3 r0.y, c5.xyz, v3.xyz
        dp3 r0.z, c6.xyz, v3.xyz
        dp3 oT0.x, r0.xyz, -c7.xyz
        mov oT1.xyz, v8
    };
}

pass p1 // ink it
{
    CULLMODE=CW;
    vertexShaderConstant[0]=<matWorldViewProj>;
    vertexShaderConstant[16]=0.8; // grossura do
contorno

    vertexShader =      asm
    {
        vs_1_0
        dcl_position v0
        dcl_normal v3
        dcl_texcoord v7
        mov r0,v0
        mul r1,c16.x,v3
        // Scale the normal
        add r0.xyz,r0.xyz,r1.xyz
        m4x4 oPos,r0,c0
        // Transorm position to clip space
        mov oD0, c0
        mov r0,c0
    };
}
};
}

```

## COMO FAZER UM PROJETO REAL

POR LUIZ FERNANDO ALVES DA SILVA (ZIZACO).



Muita gente deve achar esse artigo algo totalmente desnecessário. Pra alguns pode até ser, mas algo que já presenciei muitas vezes é algum projeto em desenvolvimento sendo abandonado, descontinuado ou simplesmente esquecido.

Aqui vai uma coletânea de dicas sobre como manter seu projeto (ou futuro projeto) em constante desenvolvimento e como torná-lo possível.

### Romance e Realidade

Essa é a primeira coisa que condena qualquer projeto! Antes mesmo de começar o projeto você já pode constatar que ele vai morrer.

Vou explicar a teoria com base a um evento literário:

### O Romantismo

Na literatura brasileira(sim, sou antipatriota ;D) os artistas escreviam em um estilo chamado "Romantismo", aonde tudo era lindo e maravilhoso! Romances prometidos e finais felizes. Tudo um mar de rosas.

Na literatura romantismo as coisas eram perfeitas como deveriam ser, os personagens das historias literários não tinham defeitos,

eram heróicos e puros, além de que tudo sempre acabava bem no final. O mundo era perfeito, pacífico, todas as pessoas do mundo são boas e querem o bem estar coletivo.

Após vários anos, percebeu-se de que o romantismo não passava de conto-de-fadas aonde coisas totalmente irreais e incabíveis aconteciam.

Voltando ao mundo dos games. Sim, isso se reflete nos games!

Muita gente vive no romantismo da coisa, acreditam que podem fazer jogos perfeitos, revolucionários e inimagináveis! Que mesmo sem ter terminado nem um jogo se quer e mesmo tendo mínima experiência, acreditam que podem criar um jogo da ultima geração! Aham que vão terminar seus jogos em pouco tempo, ou pior, acham que vão terminar em 7 anos, mas acreditam fielmente que vai valer a pena pois será "o melhor jogo do mundo".

Outro pensamento extremamente romancista é o de criar um MMORPG(Mult Massive On-line RPG, ou simplesmente RPG-Online!). Eu confesso que é muito triste você ver uma equipe, ou pior ainda, uma pessoa apenas, iniciando um projeto de um RPG On-line Massivo. Isso é o extremo do romantismo.

## COMO FAZER UM PROJETO REAL CONTINUAÇÃO

Agora vem a parte chata de tudo isso: O Romantismo não é bom! Pelo contrario, é péssimo!

Assim como um conto de fadas, o romantismo é algo irreal, inatingível, inexistente. É triste pra você, pra mim, e pra todos os desenvolvedores de jogos que não trabalham numa empresa especializada com equipe, dinheiro, e experiência no mercado.

### O Realismo

A evolução literária do romantismo foi o "Realismo".

No realismo, as coisas eram reais do dia dia de uma pessoa comum.

Na literatura realista as coisas não são perfeitas, os personagens tem defeitos, muitas vezes eram covardes, e nem sempre o final é feliz. O mundo não é perfeito, as pessoas não são todas boazinhas, existe corrupção, bandidos e tudo que é Real. Lógico que também existiam coisas boas mas não em exagero, como no já morto romantismo.

Voltando ao mundo dos games, mais uma vez.

Existem muitas pessoas(a maioria) que não admitem o realismo, justamente por ele ser a triste verdade, mas sim:

Você não vai fazer um jogo perfeito, nem revolucionário, nem algo inimaginável. Seu jogo não vai ter os últimos recursos gráficos nem de núcleo. Você não vai terminá-lo em um piscar de olhos e muito menos vai agüentar 7 anos trabalhando nele pra ele sair extraordinário.

E muito menos será um RPG On-line massivo! Caia na real!

Achou ruim? Agradeça se você terminar seu jogo. Porque nem isso os romancistas conseguem!

### Fatos Realistas

A chave para finalizar um projeto de jogo é não ser romântico quanto ao que você pretende fazer. Não imagine um super-jogo aonde o jogador pode fazer tudo que quiser, ou um jogo graficamente ótimo.

Agora como praticamente todos os romancistas não aceitam o realistas, vou ter o difícil trabalho de mostrar o quanto eles estão errados. Lógico que eu não vou convencer ninguém, mas não vão dizer que eu não tentei.

Analise os fatos a seguir:

## COMO FAZER UM PROJETO REAL

### CONTINUAÇÃO

\* Nunca nenhum jogo do brasil teve nível de um jogo do exterior, principalmente se feito por alguma empresa. E não, o seu não vai ser o primeiro! Caia na real!

\* Se você tem contato com alguma comunidade de desenvolvedores de jogos, você constatará que: Todo mundo fala que sabe, mas ninguém nunca terminou um projeto.

\* Se você tem contato com alguma comunidade de desenvolvedores de jogos, você constatará que: Todo mundo fala que vai fazer um jogo, mas nunca ninguém faz.

\* Se você tem contato com alguma comunidade de desenvolvedores de jogos, você constatará que: Apesar dos fatos acima, muita gente começa projetos, mas sempre abandonam. Pode ver que depois de um tempo eles param de atualizar sobre informações sobre os projetos, e depois ninguém nem lembra mais!

\* Existem centenas de desenvolvedores de jogos no brasil, mas existem poucas dezenas de jogos que foram produzidos aqui.

\* Se você for examinar essas poucas dezenas observe que que todos são jogos simples, simples do tipo que um romancista acharia "bobo"

\* Se você for examinar essas poucas dezenas de jogos que foram concluídos,

constará que metade deles tiveram algum lucro considerável

\* Você pode contar nos dedos da mão quantos jogos foram feitos com "super-engines" gratuitas como Ogre

\* Você pode contar nos dedos do cotovelo(zero) quantos jogos foram feitos com "super-engines" gratuitas como Ogre no brasil.

\* Desses poucos jogos que foram concluídos praticamente nenhum foi feito por uma pessoa só.

\* Desses poucos jogos que foram concluídos praticamente todos foram renumerados e tiveram um investimento inicial que saiu do bolso de alguém.

\* Não, esse alguém não foi o ministério da cultura brasileiro. Caia na real!

\* Você mora no brasil

Se isso não foi o bastante pra você. Pare de ler esse artigo, continue por nunca terminar seus projetos, e continue iniciando um novo projeto por mês ou semana, afinal você é brasileiro e não desiste nunca!

## COMO FAZER UM PROJETO REAL CONTINUAÇÃO

### Iniciando um projeto

Se você percebeu que não adianta viver nos contos-de-fadas romancistas de fazer um super-jogo. Aqui vão dicas de como iniciar um projeto serio e real.

### O jogo será gráfico?

Um jogo gráfico é qualquer jogo que possua imagens em movimento(sejam 2D ou 3D), vários modelos 3D ou varias sprites 2D. Se sim, você precisara de alguém para cuidar dessa parte, e dependendo do nível gráfico, você precisara de uma equipe para isso. Exemplos: de jogos não gráficos são o Civilization 2 ou -, Um jogo de tabuleiro, Um jogo de estratégia que não seja em tempo real, Um tetris, Um adventure ao estilo CarmenSanDiego, etc... (todos bobos pra um romancista, apesar de que eles nunca fizeram nem um desses.)

Tente evitar níveis gráficos exagerados. Isso é uma coisa que condena a maioria dos projetos. Saiba que existem outras maneiras de deixar um jogo interessante e atrativo.

Todos os exemplos citados de jogos não gráficos, são de extremo sucesso e geraram muito mais lucro do que "Sandy & Junior O Game"(Um jogo gráfico produzidos no brazil). Agora se você é do tipo "a única coisa que deixa um jogo bom é o gráfico" desista de desenvolver jogos, principalmente no brazil.

### O Sistema de Jogo

O que você pode fazer no jogo final? Pense em coisas simples, porem divertidas, os jogos da serie "Tycoon" por exemplo. Muito cuidado nessa parte, cuidado para não voar alto de mais(ser romancista). Um jogo com a liberdade do "GTA: SanAndreas" é algo não muito realista.

É muito importante que você faça um bom Design de jogo, uma documentação que contenha tudo que você pode fazer no jogo, como as coisas reagirão, etc.

Eu publiquei um artigo (<http://zizaco.wordpress.com/tutoriais/design-document/>) sobre como fazer um bom design de jogo, se você quiser dar uma boa lida ajuda bastante, mas lembre-se de manter sempre os pés no chão, e ser realista, principalmente por que o design de jogo influi diretamente nas ferramentas que você ira usar para desenvolvê-lo.

Um jogo com muitas possibilidades, escolhas, capacidades e recursos de qualquer tipo, requererá uma engine ou linguagem mais potente, o que cai em outro buraco romancista.

## COMO FAZER UM PROJETO REAL

### CONTINUAÇÃO

Dependendo das possibilidades do jogo você precisara de mais de um programador para a sua equipe.

#### Ferramentas e Softwares

Aqui é um ponto muito importante! Examine bem o seu Design, e veja quais ferramentas você ira desenvolver o projeto.

Escolha a ferramenta mais simples o possível dentro daquilo que você precisa.

Evite sempre que possível super-engines! sejam elas gratuitas ou não. Examine que todos os jogos que foram completos em super-engines alem de terem equipes muito numerosas, eram renumerados. alem de serem de fora do brasil. Tenha sempre em mente que o que vale é o produto final, não importando as ferramentas no processo de criação. Não sinta vergonha de não usar uma linguagem de baixo nível ou uma super-engine em seu projeto! Tenha em mente que quanto mais poderosa, mais complicada de se usar. Portanto querer usar uma engine muito poderosa é uma atitude romancista.

#### Formando a Equipe

Se seu projeto se adequou em alguma parte que mencionou que você vai precisar de uma equipe, isso vai ser muito importante pra você. Agora se o seu projeto for algo mais simples, ou não gráfico, nem sempre uma equipe é

necessaria.

Formar uma equipe não é uma tarefa fácil, muitos pensam que é só convidar alguém que entenda do assunto, que ira fazer tudo pra você como um escravo, a troca de nada. O que é um completo engano!

Primeira coisa a saber: Ninguém ira se juntar a sua equipe a troca de nada, isso mesmo, ninguém! Você tem que necessariamente ter algo a oferecer(não necessariamente dinheiro), seja a curto ou a longo prazo.

Uma das coisas mais garantidas para atrair alguém para sua equipe é dinheiro (Lógico!), mas como você é um brasileiro, você com certeza ira tentar oferecer alguma outra coisa no lugar de dinheiro. Então eu recomendo você a oferecer uma boa proposta. Sim, isso mesmo, uma boa proposta convence qualquer um. Um bom projeto de jogo, organizado, com um bom design, com uma historia legal e realista. Isso é o suficiente para você conseguir formar sua equipe, mas a sua proposta tem que ser boa, bem apresentada, você tem que explicar detalhadamente como você pretende que seja o funcionamento do jogo, dizer quais são seus planos para comercializá-lo depois de pronto (se você for comercializá-lo). Mostre que é um projeto serio, isso é o mais importante, pois ninguém(serio) vai querer fazer parte de uma equipe cuja a proposta seja:

## COMO FAZER UM PROJETO REAL CONTINUAÇÃO

"Quero fazer um jogo de tiro, vai ser muito legal, preciso de modeladores, e desenhistas. Não é renumerado, dividiremos os lucros quando o jogo ficar pronto."

Alem de isso ser patético, provavelmente a pessoa que faz uma proposta dessas é mais um romancista que não deve ter nenhum design do projeto para mostrar em sua proposta.

Outra coisa importante é analisar as pessoas que gostaram da sua proposta e querem fazer parte da equipe. Pois por mais que pareça que "quanto mais melhor", as coisas não são bem assim. Existem muitas pessoas que não levam um projeto a serio, entram em uma equipe por "embalo" e depois simplesmente abandonam ou esquecem dela. Ao fazer a proposta deixe bem claro de que é um projeto serio, e que a equipe deve agir de acordo.

Outra coisa é que existem muitas pessoas que se acham qualificadas para exercer uma determinada tarefa (principalmente modeladores 3D) quando na verdade não são muito experientes, e acabam por atrasar todo o projeto. Selecione bem as pessoas, saiba que ninguém precisa ser um "expert", afinal vocês são realistas(tomara) e não querem fazer um super-jogo romancista, mas as pessoas precisam no mínimo saber a respeito da atividade que estão exercendo.

Apos a sua equipe formada, faça reuniões

regularmente, mantenha sempre a comunicação e sempre informe todos de cada progresso do projeto detalhadamente. Afinal sem eles o projeto não anda e a motivação deles é ver o projeto indo para frente.

### Recompensa

Isso é o que garante que o seu projeto continue andando.

Você e toda a sua equipe(se você tiver uma) precisam de uma recompensa, lógico, afinal você é um ser humano! Mas não entenda mal, quando eu digo recompensa não quis dizer dinheiro em especifico.

Dinheiro é sim uma recompensa, mas não a única existente, existe também a gratificação desde ver o personagem que você modelou em ação no jogo, ate ouvir usuários e\ou jogadores dizendo que acharam o seu jogo legal.

Uma boa idéia é expor o progresso do seu projeto para um publico de outros desenvolvedores de jogos ou de jogadores, de forma que eles possam fazer comentários, tanto negativos como positivos. Isso é bastante motivante para todos da equipe. Afinal ver que os usuários se preocupam com o produto final (fazendo criticas) ou elogiando as coisas boas do projeto, faz você não se sentir sozinho e ajuda muito te motivando.

## COMO FAZER UM PROJETO REAL CONTINUAÇÃO

Talvez comentários sejam uma das melhores recompensas, só ficando atrás para um usuário final dizendo o quanto gostou de ter comprado\jogado o seu jogo, mas é lógico que você só vai ouvir isso se você tiver no mínimo terminado seu jogo ou distribuído uma demonstração, durante a produção do jogo, você estará limitado a mostrar como anda o progresso e o que já esta pronto.

### Considerações Finais

Bem acho que é isso.

Se você era um romancista, espero que você tenha revisto seus conceitos. Se não era, tomara que esse artigo tenha ajudado em algo.

Muito importante ressaltar que esse artigo foi inscrito em cima de algumas idéias pessoais minhas, você não é obrigado a concordar em nada que foi escrito aqui.

Esse artigo foi baseado na matéria Ant-Engavetamento escrita por Luan dos Santos (<http://pandadev.wordpress.com/2007/06/20/artigo-anti-engavetamento/>)

## CONCEITOS 3D

YVES NADJARIAN

Resolvi escrever este arquivo antes de começar a escrever sobre programação em Dark Basic pois pude constatar a dificuldade de pessoas em lidar com as três dimensões sem poder vê-las. Então vou começar com o básico do 3D, que poderá servir para modeladores iniciantes também.

Em minha aula de computação gráfica, foi passado rapidamente para os alunos como funcionam os famosos eixos X, Y e Z, e como o desenho em três dimensões é feito baseado neles. E tivemos que fazer um exercício em um programa de rasterização através de comandos.

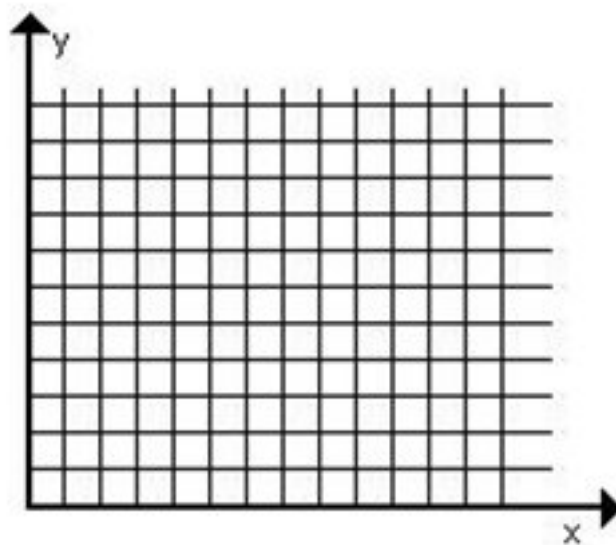
O programa era um editor de texto e os comandos parecidos com C/C++ eram digitados para criar objetos, luzes, cameras, etc. Claro que os que já conheciam algo do tipo, ou já haviam modelado, mesmo que visualmente como no Blender, consegue ter uma visão melhor do espaço quando se trata apenas de variáveis.

Cheguei a esboçar uma grade que representava três dimensões para tentar mostrar a alguns colegas como tudo funciona. E aqui vamos nós.

O bom e velho plano cartesiano

A grande maioria de nós chegou a estudar o famoso plano cartesiano. É a partir dele que temos uma idéia de como se posicionam objetos no espaço 2D, e mais à frente 3D.

Ele é representado normalmente pelo seguinte desenho:



### CONCEITOS 3D CONTINUAÇÃO

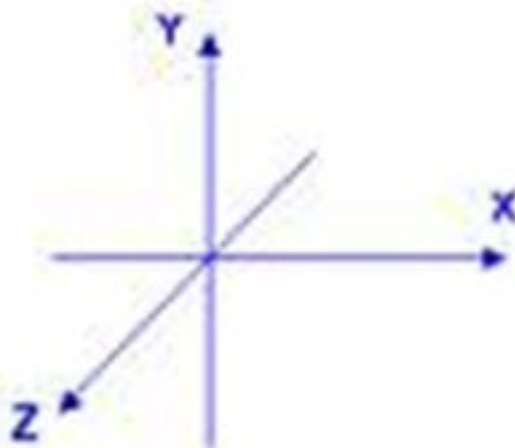
O eixo X, é usado para posicionamento horizontal, enquanto o eixo Y usado para posicionamento vertical. E é muito importante lembrar isso. Muitas pessoas confundem como que o plano cartesiano é desenhado, pensando que estão vendo o plano por cima, quando na verdade a visão é pela frente.

Então imagine: se no plano junto ao eixo X, for colocado um objeto qualquer, significa que o objeto está tocando o chão. E ao move-lo pelo eixo Y, você estaria tirando o objeto do chão e fazendo-o flutuar. Alguns confundem o plano achando que seria algo como um tabuleiro de xadrez, movendo o objeto pela profundidade.

E é ai que entra nosso terceiro eixo, o Z. E é ai que a diversão começa. Lembre de jogos como Street Fighter 2 por exemplo. Ao aplicar um soco em um personagem, bastasse que o soco alcançasse o outro personagem, não importava se ao invés de pegar na cara, o soco fosse mal calculado e pegasse somente de raspão ao lado do rosto. Não havia a profundidade!

O Mario somente podia passar pelos inimigos pulando por eles. Porque não apenas dando um passo pro lado?

Com os jogos 3D, isso mudou, e para isso agregamos ao plano cartesiano o eixo Z:



## CONCEITOS 3D

### CONTINUAÇÃO

Então objetos agora possuem 3 variáveis de posicionamento, não somente duas. Porém com a vinda do terceiro eixo, chegou também outros componentes que não devem ser desprezados.

Embora no começo, tais itens não existissem nos jogos em 3D, se hoje um jogo não possuí-los, não terá a menor qualidade!

### Luzes

O conceito de luzes em ambientes 3D para os jogos foi inicialmente para que aquela sombra igual para todos personagens em forma de círculo pudesse melhorar. Embora calcular sombras requer um bom esforço do processador (se você é iniciante nesta área saiba que não são tudo flores, devemos nos preocupar com performance) e eu já tenha visto jogos em 3D com as mesmas sombras em formas de círculo embaixo dos personagens, esta idéia foi aproveitada para outros usos (também na área de modelagem).

Pare para pensar. Na verdade nós não vemos a luz. Nunca passou por sua cabeça que nos filmes espaciais em que o Sol aparece, nós vemos o Sol, e também a Terra sendo iluminada por ele. Mas não vemos a Luz passando e fazendo o seu caminho até a Terra. A luz só é percebida quando é refletida pelos objetos que estão em seu caminho. Por isso que não adianta apontar uma lanterna pro céu, vai continuar tudo escuro, pois nada está refletindo a luz.

Em um mundo 3D acontece da mesma forma, e com isso se tira alguns proveitos, por exemplo, se você fez o terreno todo com belas texturas que teve muito trabalho de criar, e quer que o terreno pareça estar a tarde, ou a noite, não precisa ser refeito, e colocar outras texturas com outras cores. Aplique a luzes de forma e cor correta que os efeitos serão os mesmos.

Claro que existem outros usos para luzes em 3D, mas isso já elimina as dúvidas de pessoas que não faziam a menor idéia do que se tratava. E principalmente: "Porque não estou vendo a minha luz?"

## CONCEITOS 3D CONTINUAÇÃO

### Câmera

Já que a profundidade será usada, alguns conceitos mudam de quando os jogos eram apenas em 2D. Lembre-se de jogos de corrida 2D, que até tentavam imitar pistas 3D. Na verdade o que acontecia era que o carro estava sempre no mesmo lugar, e o fundo se movia para dar a impressão de que o carro estava andando pela pista.

Em um jogo 3D, imagine ficar calculando as possíveis variáveis de uma pista em 3 dimensões... Portanto o que ocorre, a pista não se move fica parada, como na vida real. Quem se move é o carro, como na vida real. E porque você sempre vê o carro? Porque existe uma câmera seguindo-o como se fosse em uma corrida na vida real!

A câmera, se bem usada, pode ter efeitos ótimos nos jogos. Caso você já tenha jogado Need for Speed, deve-se lembrar de quando o carro pulava o tempo ficava mais lento e ele era visto meio que de frente. Na verdade a câmera que estava atrás do carro pula para outra posição e mostra o carro de outro ângulo. Coisa que seria praticamente impossível se ao invés de usar câmeras, ficassemos rotacionando os objetos.

Pense na câmera como uma de verdade em um filme, onde os objetos não se preocupam em como aparecerão, e sim a câmera que escolhe o ângulo.

### Ação!

Agora que uma visão do mundo 3D no escuro está mais fácil, estamos prontos para o próximo passo. Um passo por vez. Elaborarei os próximos artigos a fim de produzir a princípio pequenos jogos em Dark Basic, claro que a lógica do jogo é algo que será adquirido junto, mesmo que sua linguagem preferida seja outra.

Até a próxima,

Yves Nadjarian

**NEM HITLER AGUENTA MAIS**

POR YVES NADJARIAN (YN).

Não é de hoje que já se comenta sobre os jogos que possuem como tema a segunda guerra mundial estão saturados no mercado. As mesmas armas, os mesmos gritos dos alemães, e nenhuma novidade.

Aliás, não há novidades. Tudo sobre este tema já foi abordado, do Dia D, ao fim da guerra. Tenho que assumir, a primeira vez que vi *Medal of Honor: Frontline* em um Playstation 2, justamente sobre o Dia D, fiquei abismado com a qualidade, e a melhora na guerra.

Os jogos de guerra de Playstation One era apenas você de seu país contra o exército inteiro do oponente que estranhamente mandava somente grupos de 5 ou 6 soldados por vez. Ora, aquilo não era uma guerra era treinamento, e de escoteiro ainda! E para tentar encobrir isto os produtores tentavam por um pouco de humor quando um inimigo ou outro era pego de surpresa urinando em um poste.

Com o novo console a gama de efeitos aumentou, explosões, barcos, aviões, mas parou por ai. Não há muito mais o que criar depois disso.

Uma nova missão, ou um cenário um pouco diferente já não compensa mais para outro jogo abordando o mesmo tema.

Talvez as produtoras resistam e lancem

mais jogos ainda para os novos consoles como Playstation 3, Wii e Xbox 360, mas acredito que dificilmente um jogo neste estilo fará sucesso a longo prazo.

Todos vibraram com os primeiros jogos neste estilo. Poder reviver um pouco de uma guerra que tantos ouvimos falar, mas devem existir mais de vinte títulos. Todos parecidos!

Se Adolf soubesse que depois da guerra tanta programação envolvesse um tema tão familiar ele cobraria royalties!