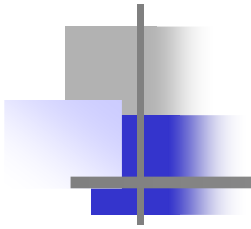
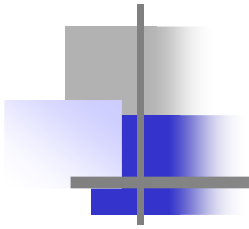
Visualização, Simulação e Games

Esteban Walter Gonzalez Clua
Instituto de Computação – UFF
esteban@ic.uff.br



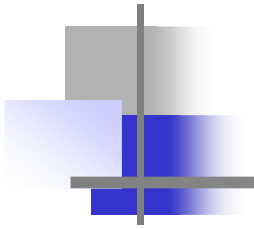
Parte 1 – Conceitos de Real Time Rendering

a. Pipeline Gráfico
b. Triangle Strips
c. Métodos de Culling
d. Level of Details
e. Per vertex ilumination
f. Introdução a API´s Gráficas
g. XNA (baseado em DirectX)



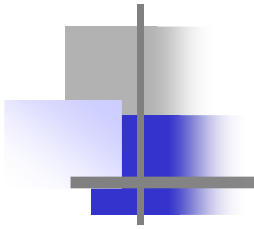
Parte 2 - Arquitetura de GPU´s

a. Arquitetura de Hardware
b. Programação de GPUs
c. Vertex Programming
d. Pixel Programming
e. General Pourpose GPUs



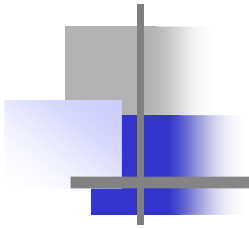
Parte 3 - Arquitetura de Game Engines

- Conceitos de Engenharia de Software para Tempo Real
- Arquitetura de Frameworks
- Arquitetura Ferramental
- Sistemas Distribuídos e Multi-cores
- Programação de Threads



Parte 4 - Real-Time Physics

- Algoritmos de Colisão
- Algoritmos de Corpos Rígidos
- Tratamento de Sistemas de Partículas
- Arquitetura de Physics Processor Units
- Ageia / Physics



Parte 5 - Tratamento de Inteligência Artificial

- Mecanismos para inserir IA em Sistemas Tempo Real
- Interpretadores de Scripts
- Máquinas de Estados

Bibliografia

BÁSICO

Zerbest, S. and Düvel, Oliver, **3D Game Engine Programming,** Premiere Press, ISBN: 1-59200-351-6

Randima, Fernando (editor), **GPU GEMS I and II,** Addison Wesley

Eberly, David H. **3D Game Engine Architecture: Engineering Real-Time Applications with Wild Magic** The Morgan Kaufmann Series in Interactive 3D Technology

COMPLEMENTAR

E. Azevedo e A. Conci, **Computação Gráfica: Teoria e Prática** Editora Campus, ISBN 85-352-1252-3.

A. H. Watt, F. Policarpo **3D Games - Real-time Rendering and Software** Addison-Wesley, ISBN: 0201-61921-0.

Finney, Kenneth C., **3D Game Programming All in One** Pemiere Press, ISBN: 1-59200-136-X

McShaffry, Mike **Game Coding Complete** Paraglyph Press, ISBN: 1-932111-75-1

Sherrod, Allen **Ultimate 3D Game Engine Design & Architecture**

Maurina, Edward F. **The Game Programmer's Guide to Torque: Under the Hood of the Torque Game Engine**

Harrison, Lynn Thomas **Introduction to 3D Game Engine Design Using DirectX 9 and C#**

Webgrafia

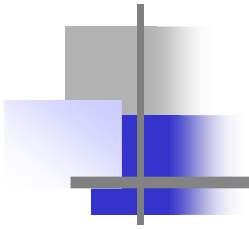
Eventos::
Game Developers Conference (www.gdconf.com)
SBGames (www.sbgames.org.br)
Virtual Game Developer Conference (www.vgdc.net)

Web sites::
Unidev (www.unidev.com.br)
GameDev (www.gamedev.net)
Gamasutra (www.gamasutra.com)
Makegames (www.makegames.com)
Flipcode (www.flipcode.com)

Magazines:
Game Developer Magazine (www.gdmag.com)

Groups::
International Game Developers Association (www.igda.com)

Resources::
Microsoft (www.microsoft.com/directx)
ATI (www.ati.com)
NVidia (www.nvidia.com)
XNA (www.xnatutorials.com)



Objetivos do Curso

Avaliação do Curso

Parte 0 - Visão Geral

Esteban Walter Gonzalez Clua

Para quê você quer fazer um game?

- Só por prazer... (porque tudo tem que ter razões econômica$?)

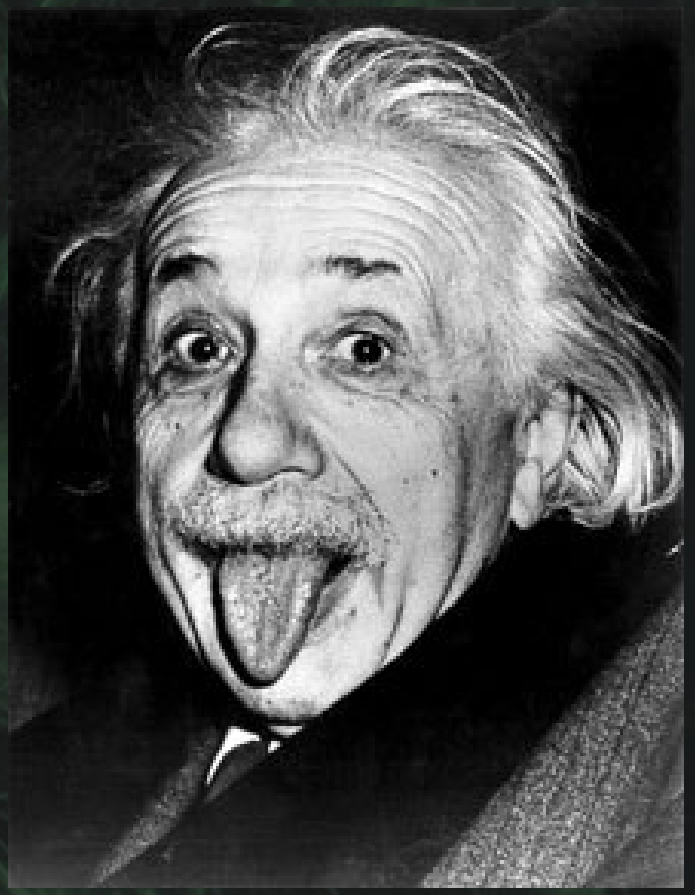
- Quero aprender a usar ferramentas novas e estudar novas tecnologias

- Quero usar tecnologias de games para outros fins (simulação, educação, visualização científica, etc...)

- Quero entrar na indústria de games

1 - Definia seu perfil

- Contador de estórias
- Ilstrador
- Artista
- Programador aprendiz
- Programador Hard Core
- Grana



2- Criar as Estórias

- Estórias = experiências (uma estória boa é aquela que sentimos vontade de recontar para outros...)

- Emoções

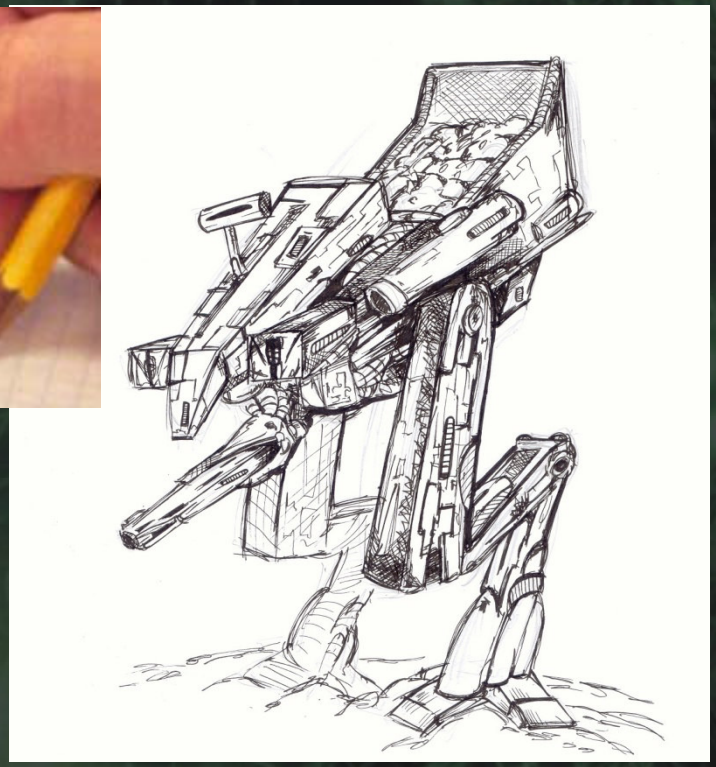
- O legado de nossa Vida é uma estória. A Vida das pessoas são Estórias… O que Seriam duas pessoas Com estórias iguais?

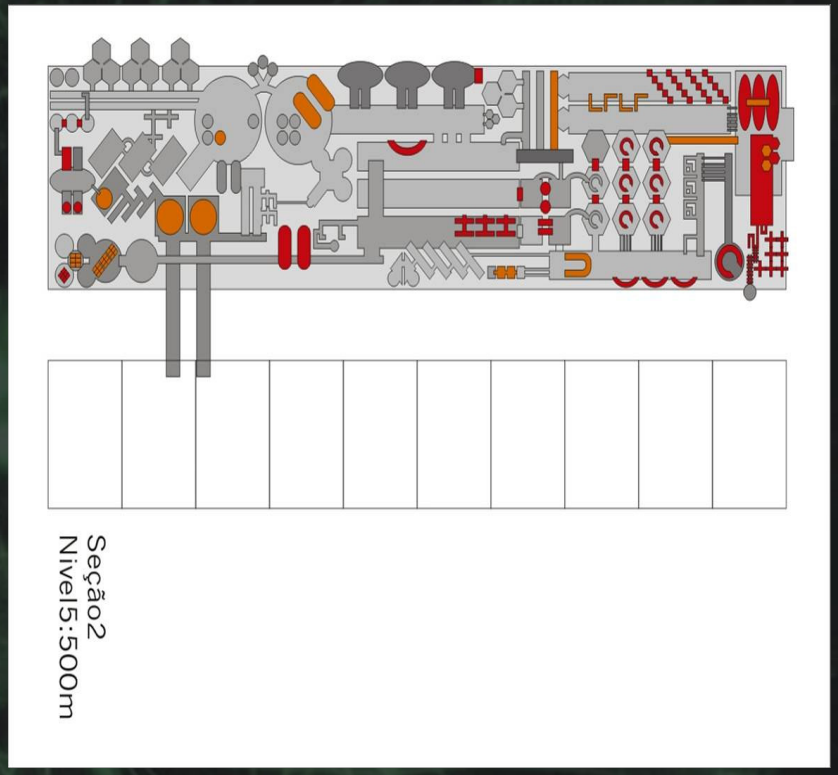
Exemplo

Computador sabe contar estórias?

3- Agora precisamos entender sua estória...

- É aqui que o bixo pega... Arte Conceitual



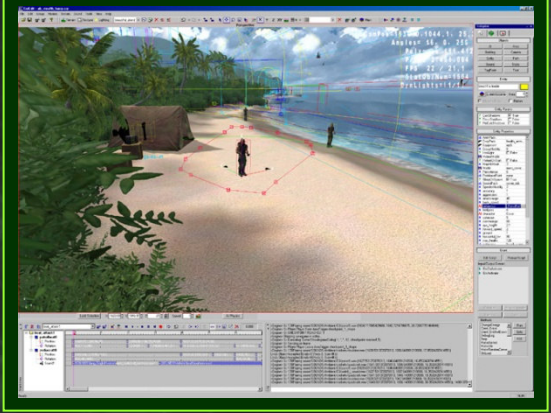


Ferramentas

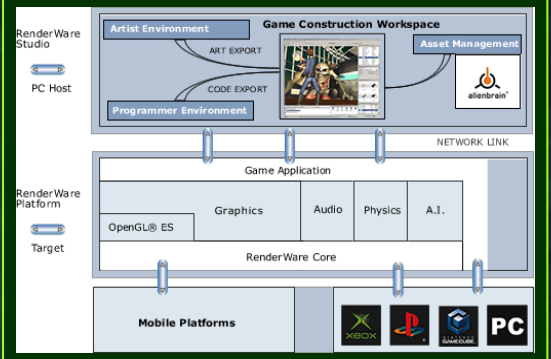
Quando se inventou o computador, criou-se uma máquina a mais, quando se criou o compilador, criou-se uma nova era tecnológica.

4- engine...

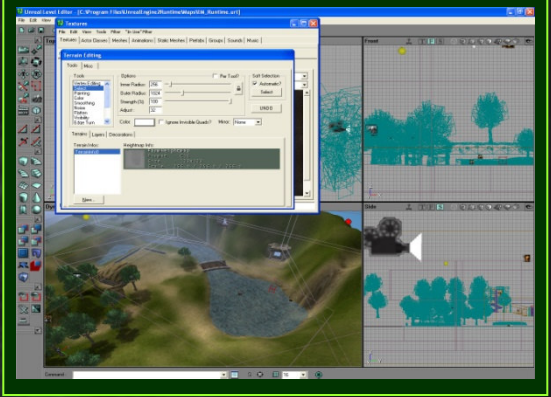
www.crytek.com



www.renderware.com

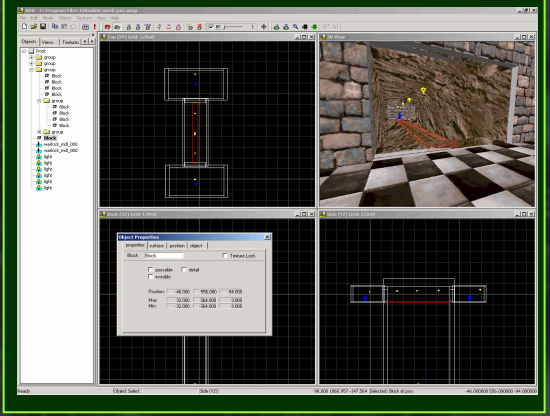


www.unrealengine.com



www.ogre3d.org

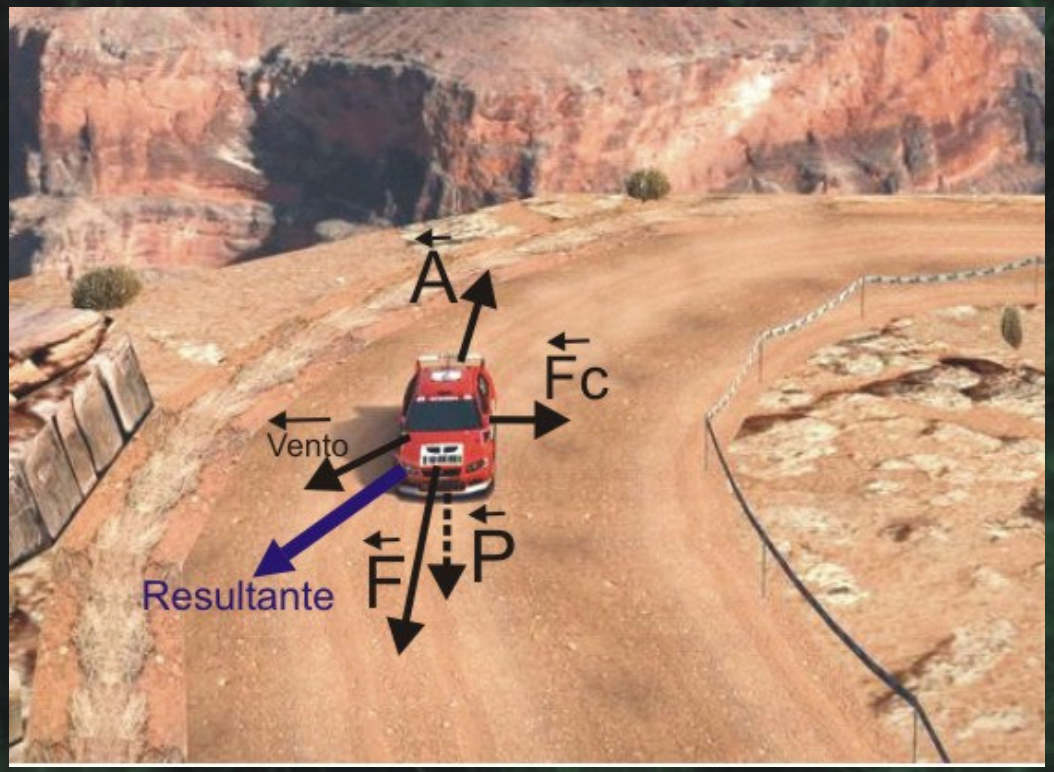
www.3dgamestudio.com



www.garagegames.com

4- engine...

Engine - Física



Física

- Métodos de detecção de colisão
- Dinâmica de corpos rígidos
- Física de Partículas
- Física de corpos flexíveis

Física

AGEIA

Home | About AGEIA | News | Technology | Partners | Developers

NovodeX home | downloads | support | tools & middleware | solution providers

NovodeX Tools By AGEIA

Our Collision Detection Makes Quite an Impact.

Download Rocket Now

Download NovodeX SDK

NovodeX Tools Feature Set

Support for Multi-Threading
Take advantage of next-generation multiprocessing systems with the new multi-threaded API in the NovodeX SDK.

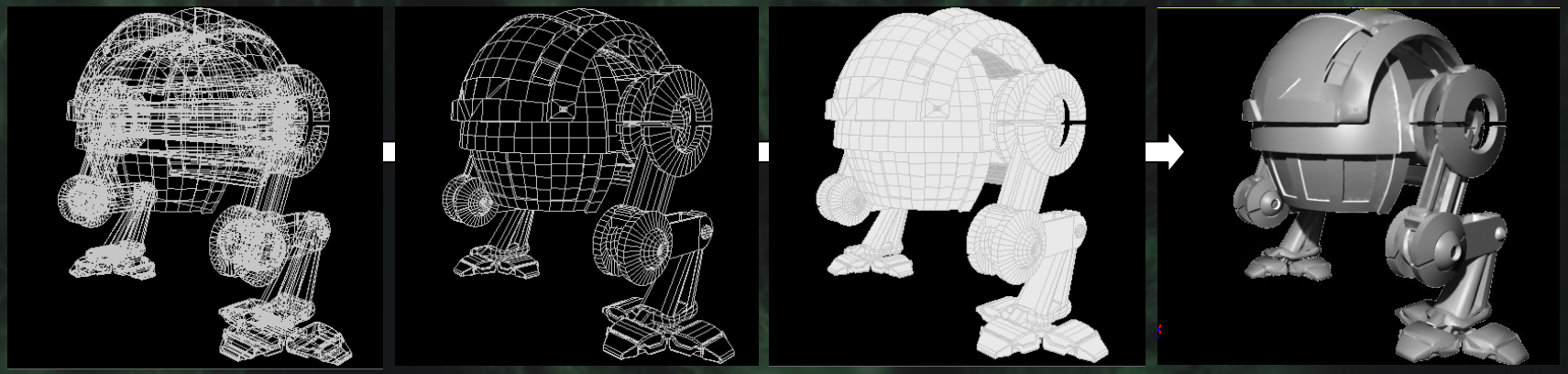
Improved High-Speed Collision Detection
The industry's fastest and most comprehensive collision system is more robust than ever.

Multi-Platform Support
The NovodeX SDK now supports development from PC to console.

For Commercial Game Development and Game Engine inquires to leverage NovodeX Physics SDK or NovodeX Rocket please register and download the NovodeX SDK or contact Developer Relations Our North American or European office will follow up with you on your inquiry promptly.
For non-game inquires for the NovodeX SDK please contact our business development department at bizdev@ageia.com

Inteligência Artificial

Engine - Renderização



Transformações 3D
Projeção 3D → 2D

Culling

Clipping

Shading
Texturização

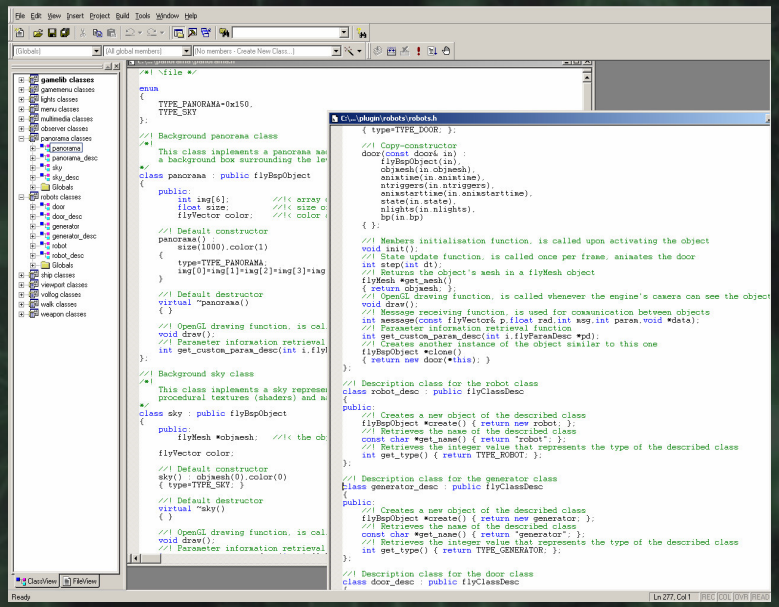
programa de vértices

programa de Pixels

Engine - Real Time shaders

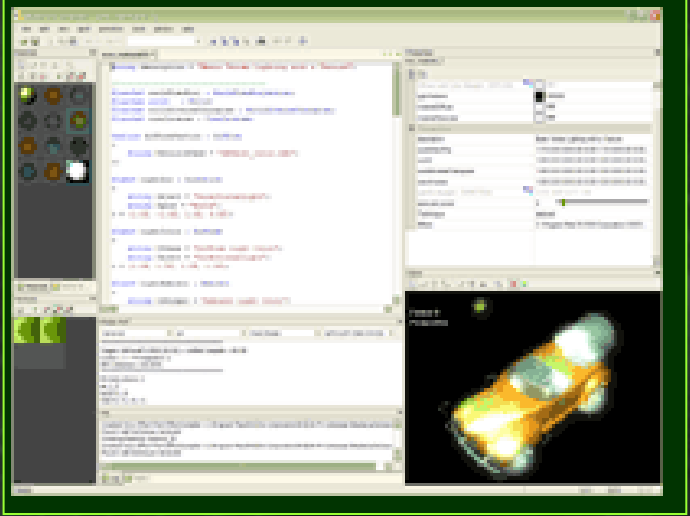
5- Agora a plataforma de programação

- Compilador, APIs, plataforma

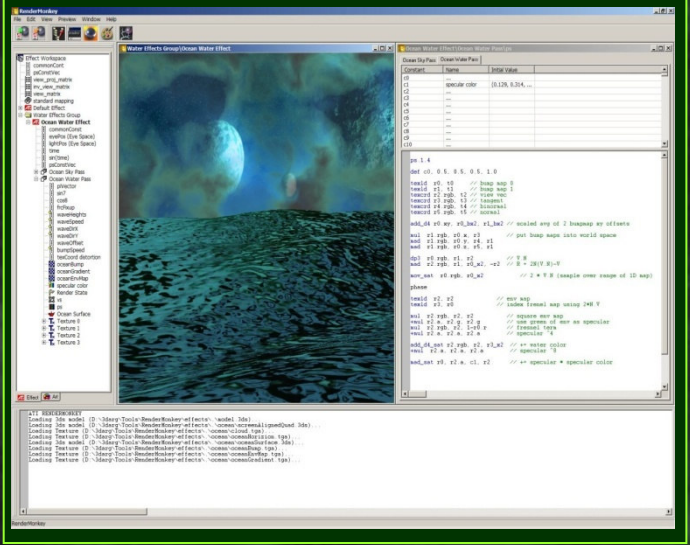


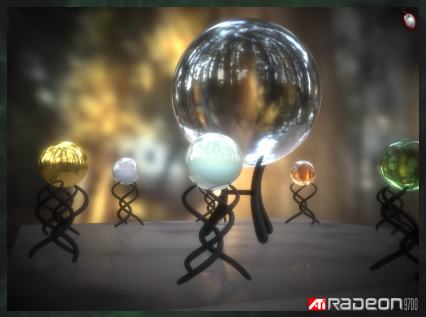
5- Ainda falando de programação

NVídia FX Composer



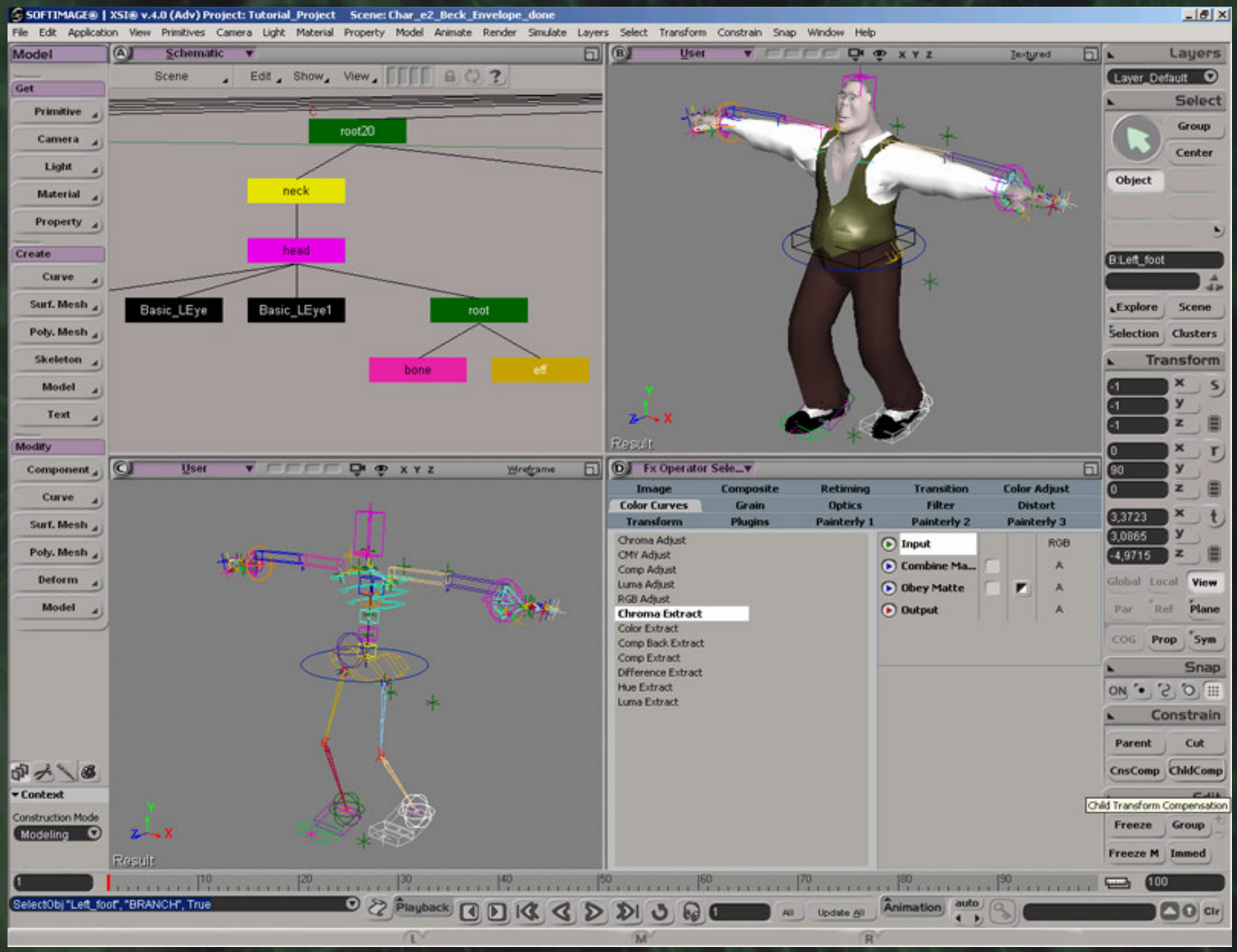
ATI Rendermonkey





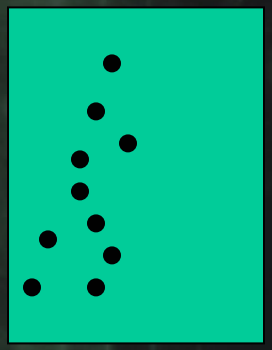
- *Shaders: HLSL , CG, OpenGL Shader Language*

6- Modelagem



Esteban Walter Gonzalez Clua

6- Animação

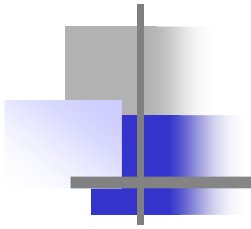


Necessidade
Da animação

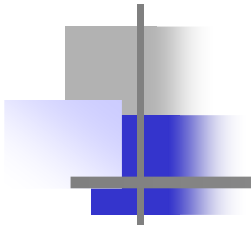
Eu acredito

Eu não acredito

Grau de fidelidade da modelagem



Parte 1 – Conceitos de Real Time Rendering



a. Pipeline Gráfico

Pipeline Gráfico

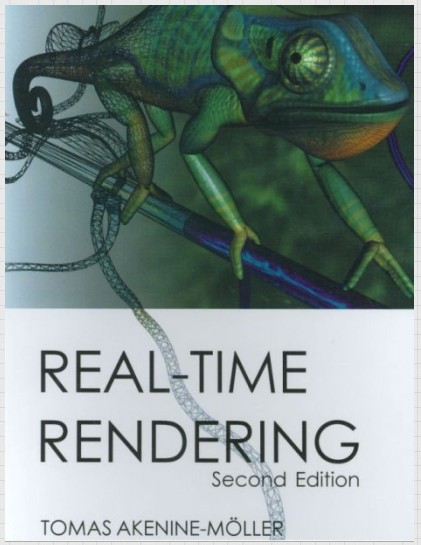
- Pipeline / Estágios

- Gargalo

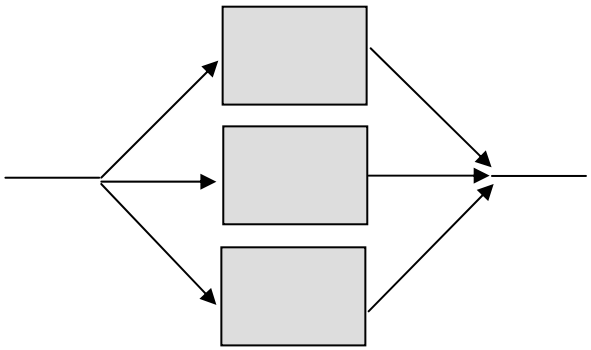
- Otimização

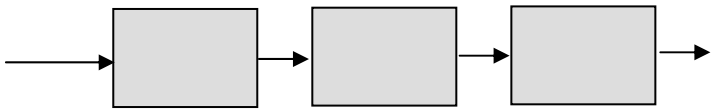
- Tipos de Processamento Paralelo

referência

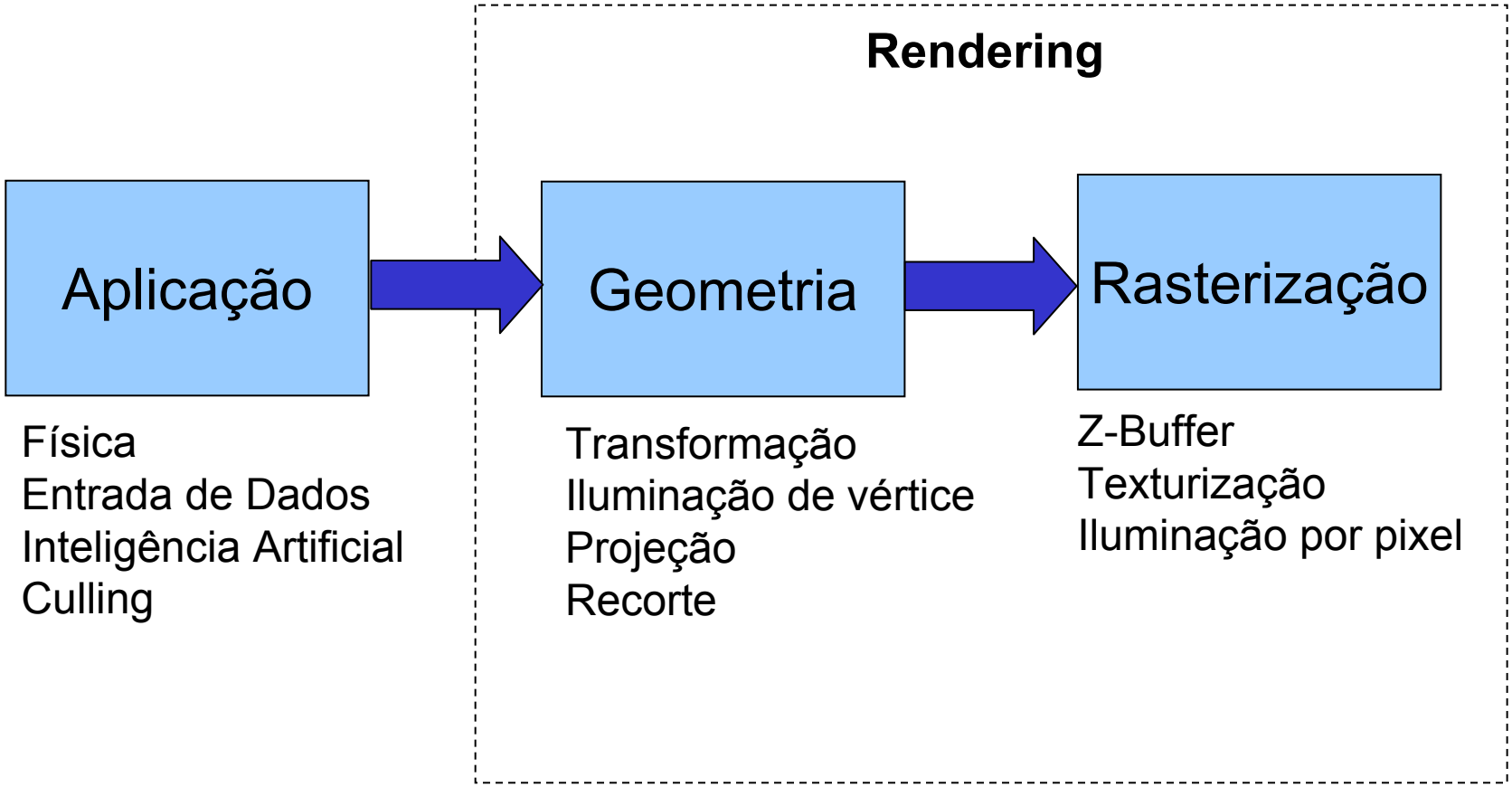


Real Time Rendering – Second Edition
Akenine-Möller, Haines

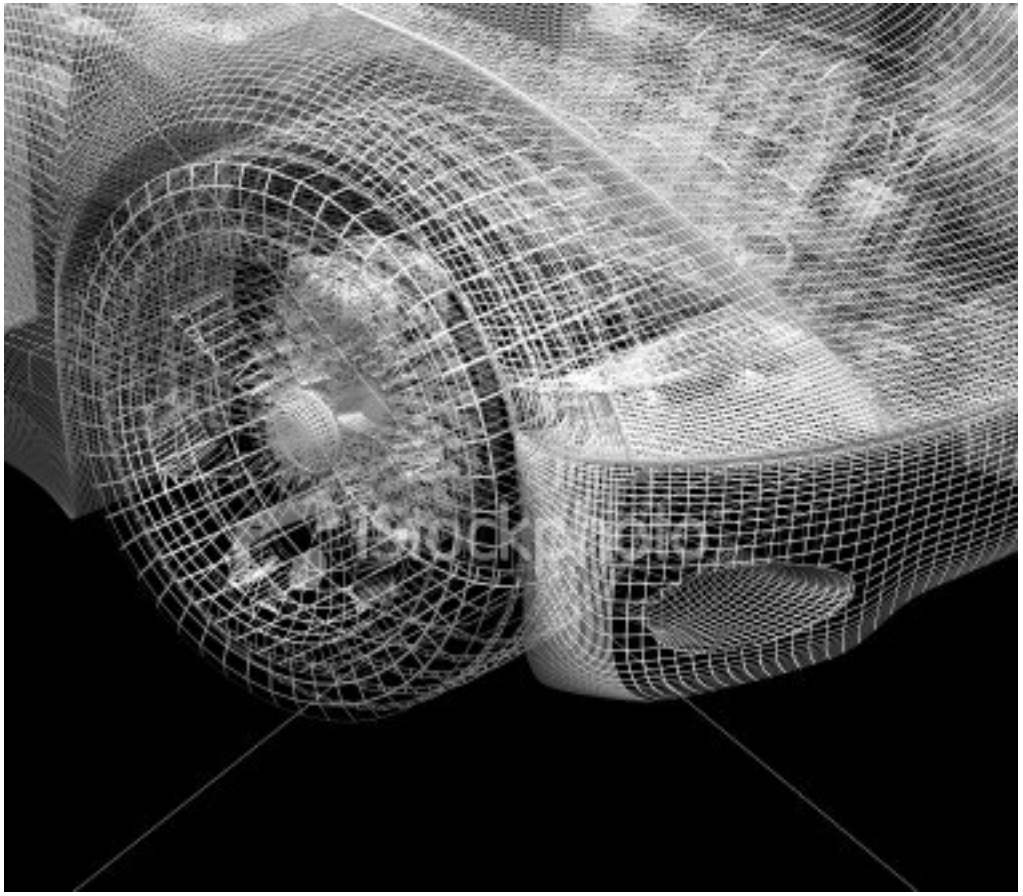




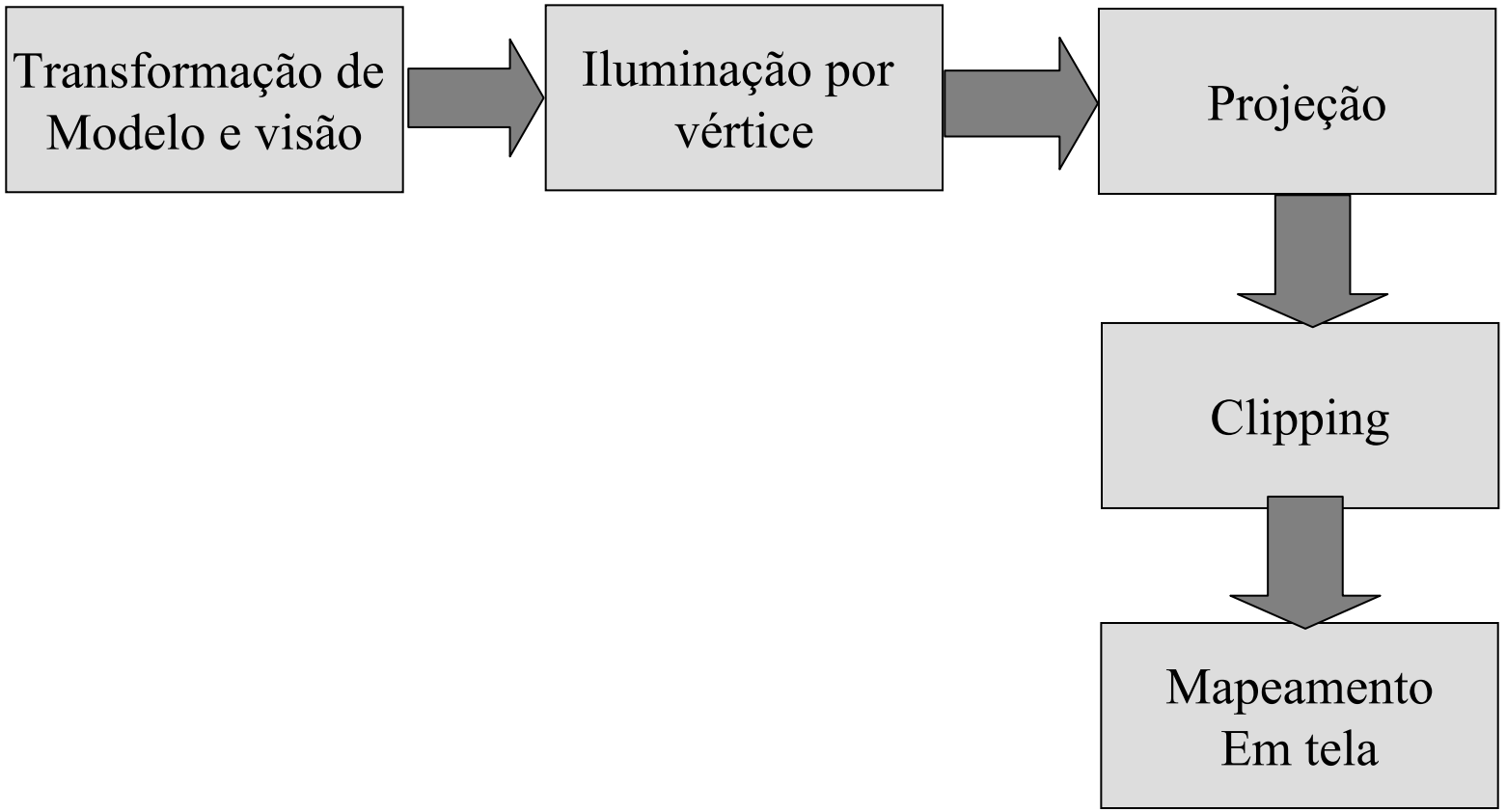
Pipeline Gráfico



Representação de modelos geométricos

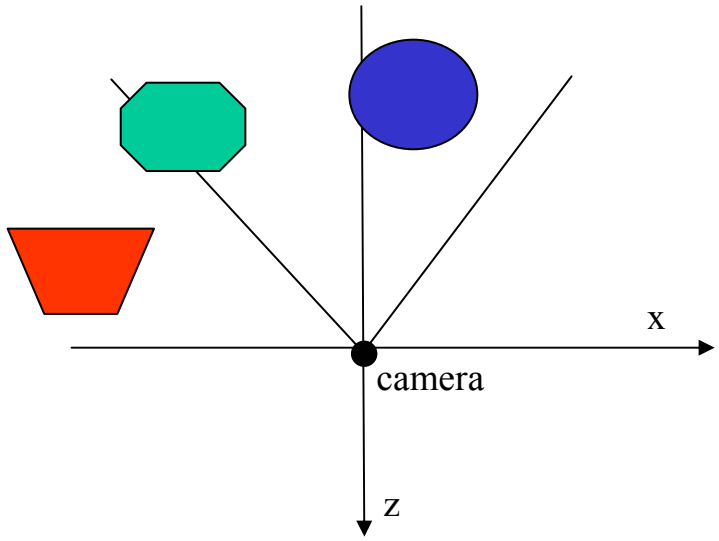


Estágio de Aplicação



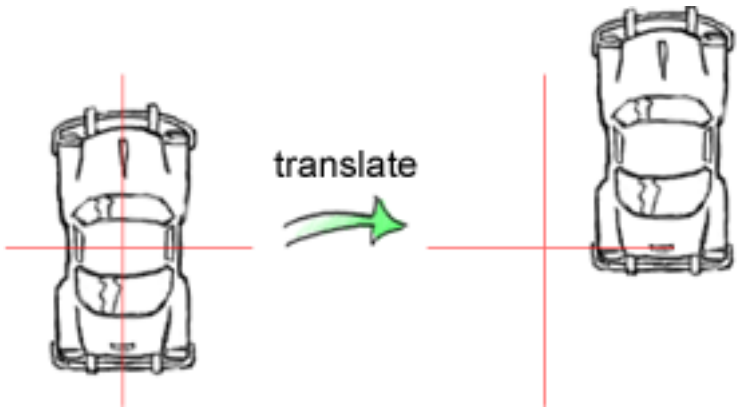
Aproximadamente 100 operações de ponto flutuante para esta aplicação

Transformação de Modelo e Visão



Transformação

$$T_{\mathbf{v}} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

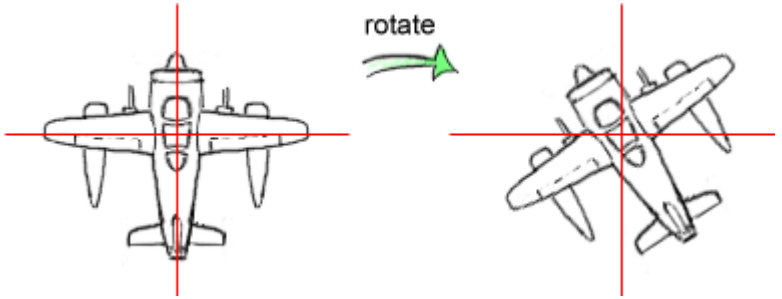


$$T_{\mathbf{v}}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix} = \mathbf{p} + \mathbf{v}.$$

T(v)

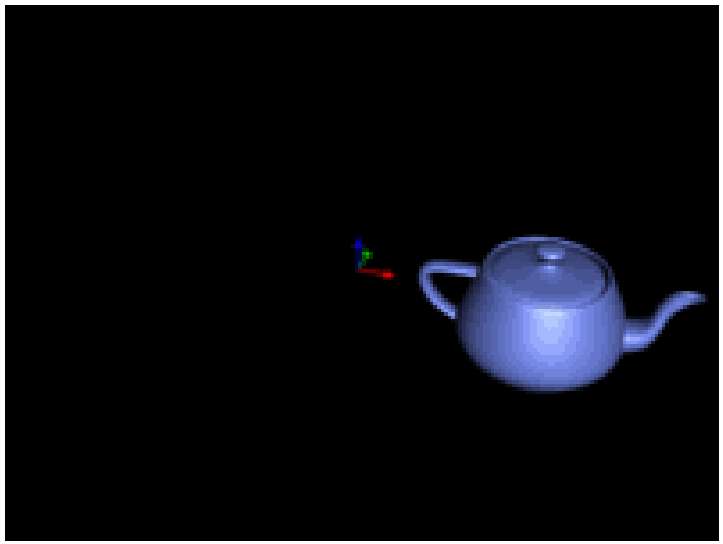
Rotação

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

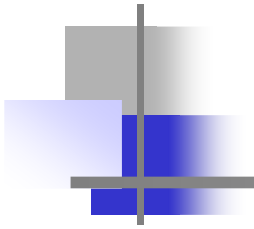


Rotação

$$R_x R_y R_z = \begin{bmatrix} \cos\alpha\cos\beta & -\cos\gamma\sin\alpha + \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\gamma\sin\beta + \sin\alpha\sin\gamma & 0 \\ \cos\beta\sin\alpha & \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & \cos\gamma\sin\alpha\sin\beta - \cos\alpha\sin\gamma & 0 \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Escala

$$\begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



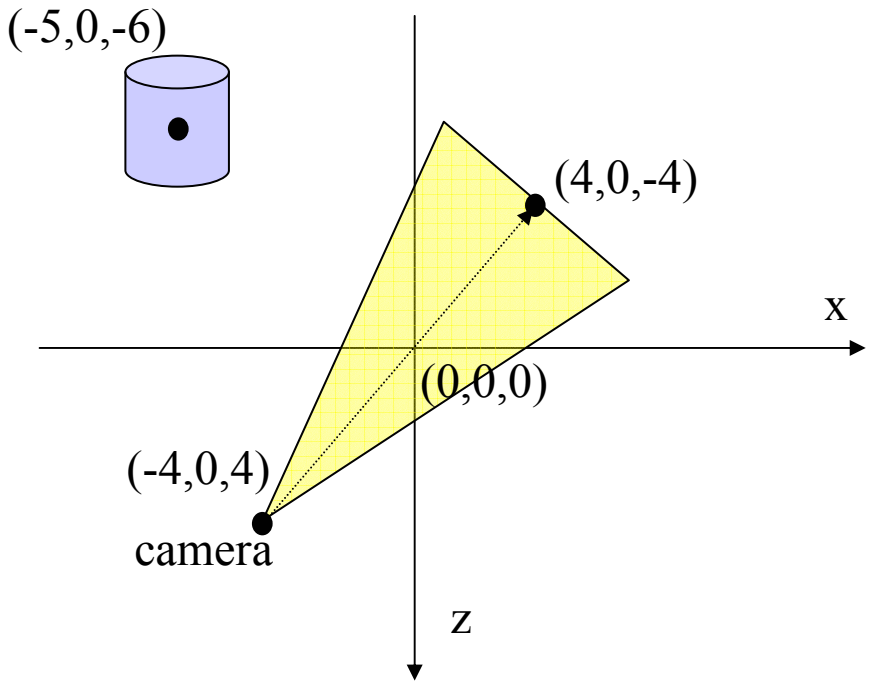
Composição de Transformações

Como rotacionar um objeto ao redor de um ponto p?

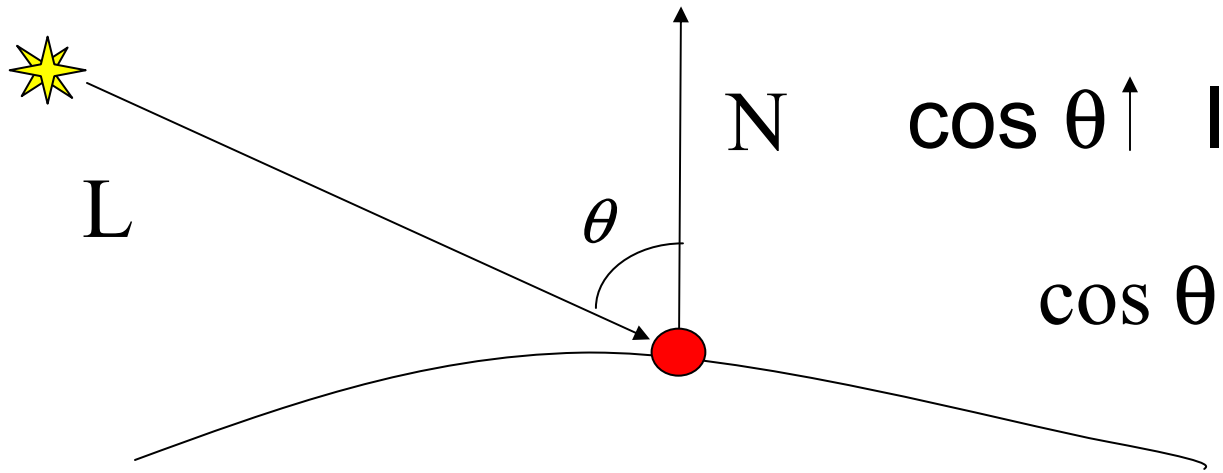
$T(p).R_z(\alpha).T(-p)$

Exercício

Crie uma matriz de transformação para eye space para a cena descrita abaixo:



Iluminação por vértice



$\theta\uparrow$ Iluminação$\downarrow$

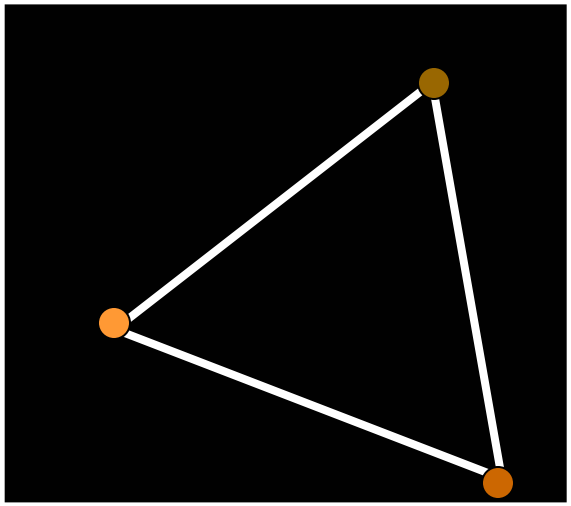
$\cos\theta\uparrow$ Iluminação$\uparrow$

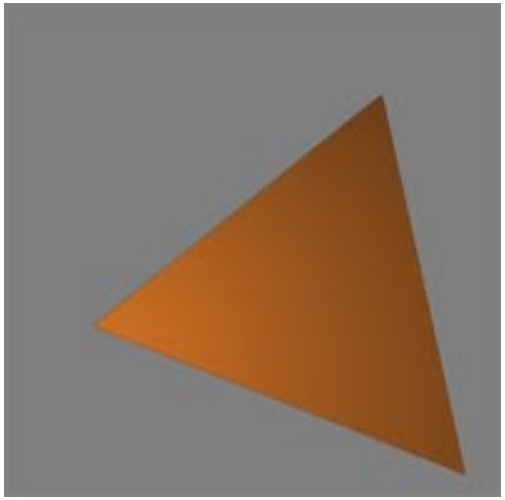
$\cos\theta = L \cdot N$

Itotal = *Iambiente* + *Idifusa* + *Iespecular*

$$I_{\text{total}} = I_a K_a C_d + f_{\text{at}} I_{\text{luz}} [K_d C_d (\mathbf{N} \cdot \mathbf{L}) + C_s K_s (\mathbf{R} \cdot \mathbf{V})^{n_s}]$$

Iluminação por vértice



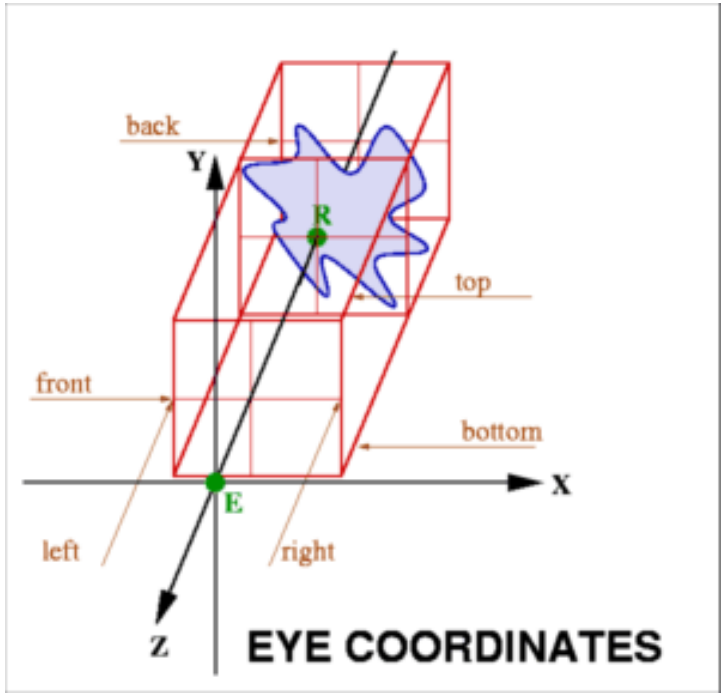


Projeção

Projeção Ortográfica

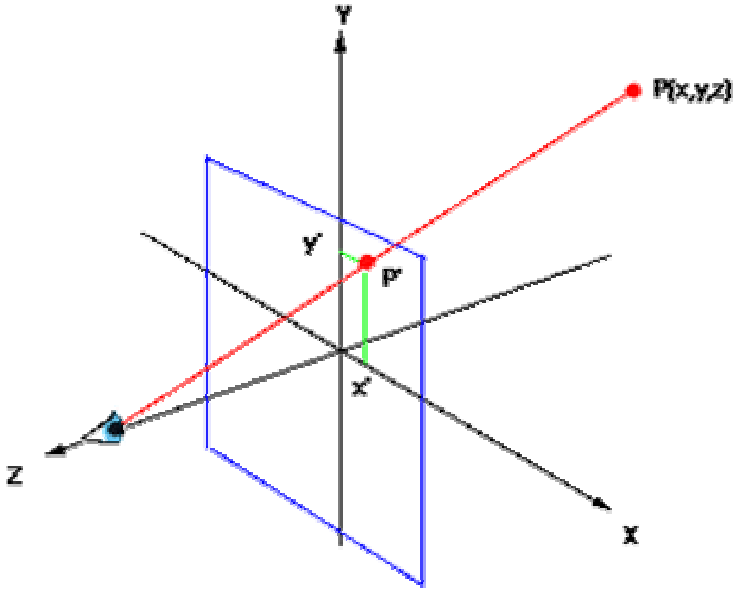
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

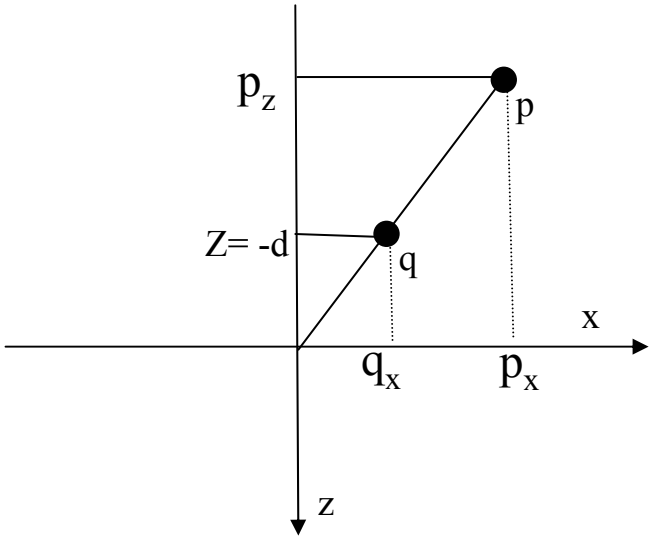
Assumindo que os vértices estão em coordenadas de *eye space*



A matriz não possui inversa, pois a determinante é nula. Assim, esta é uma transformação sem “volta”

Projeção



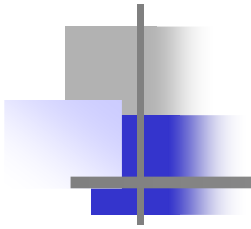


$$\frac{q_x}{p_x} = \frac{-d}{p_z}$$

$$q_x = -d\,\frac{p_x}{p_z}$$

Projeção Perspectiva

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix}$$

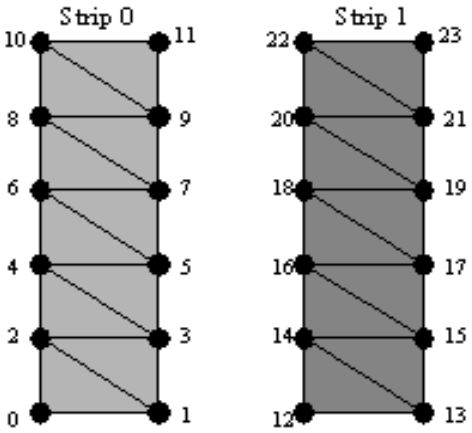


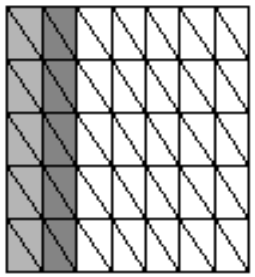
b. Triangle Strips

Idéia fundamental: minimizar volume de vértices e consequentemente, minimizar cálculos de iluminação, normais, clipping, etc.

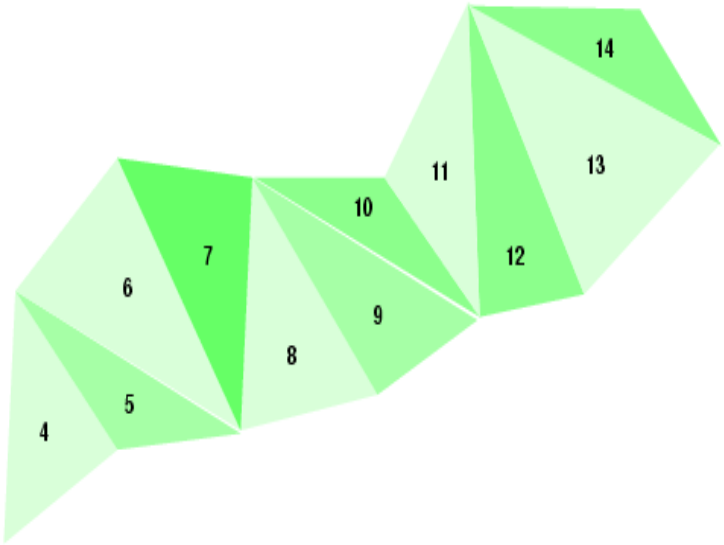
Triangle Strips

Strips: É possível descrever um triângulo com menos de 3 vértices?





Problema

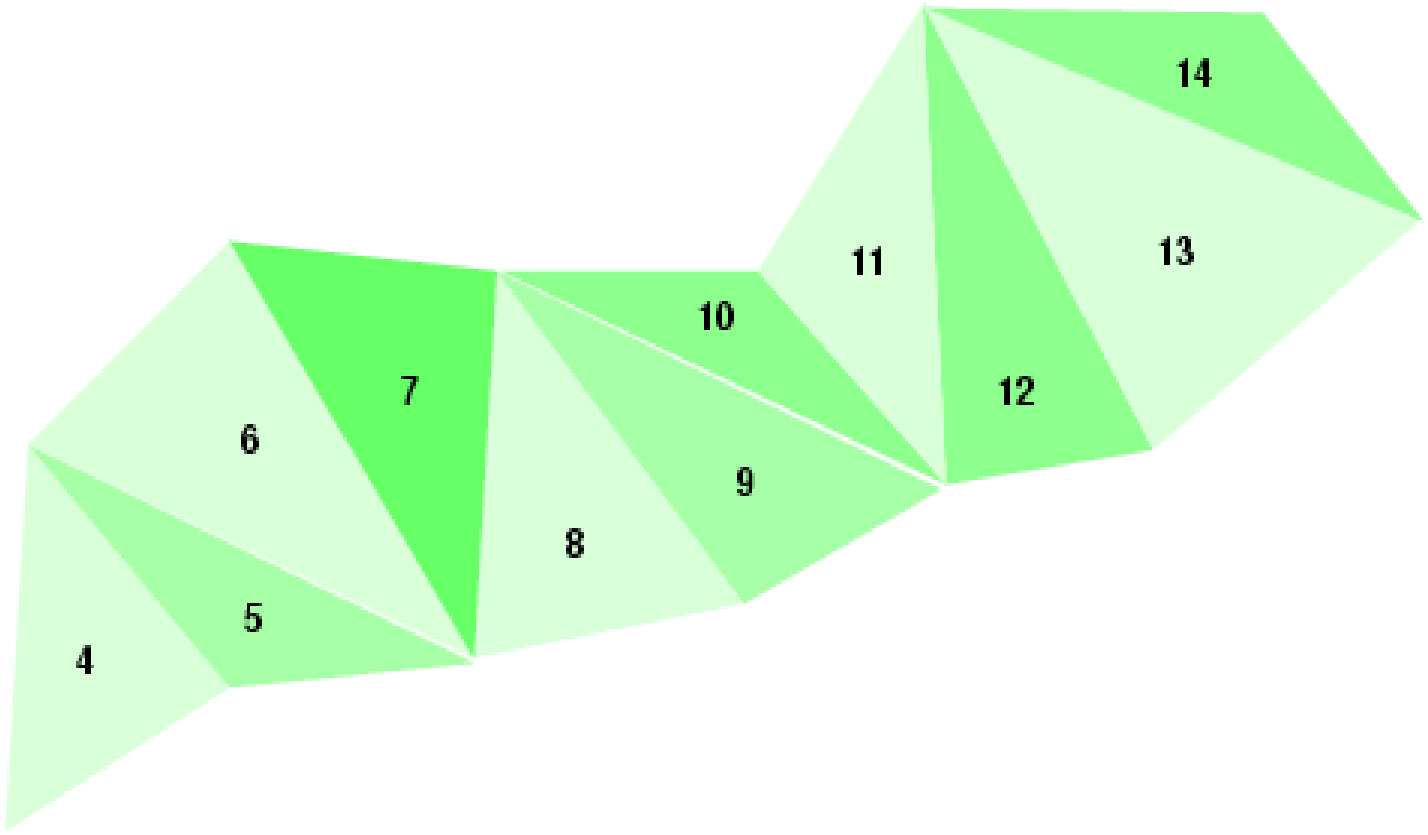


Para n triângulos, n+2 vértices

Cada Triangulo: V_i, V_{i+1}, V_{i+2}

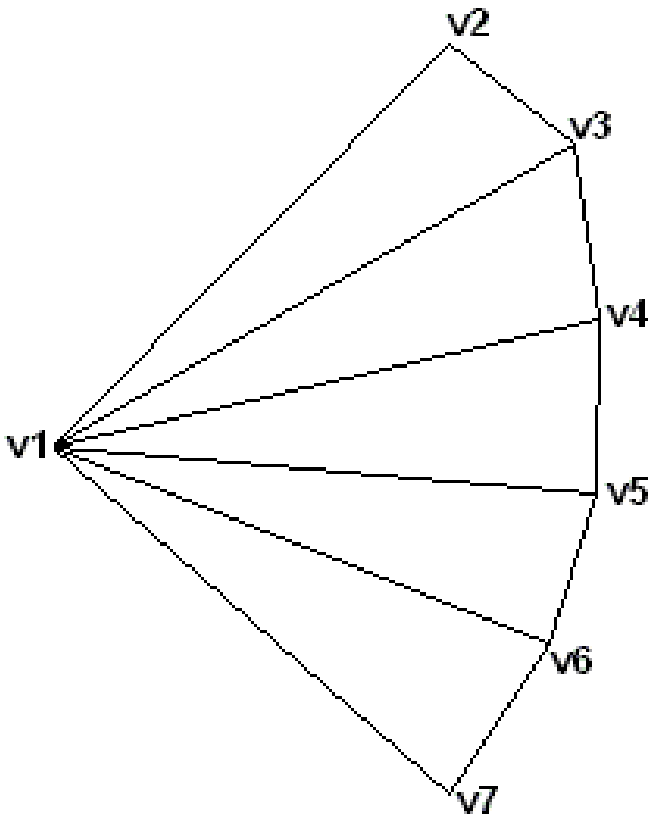
Triangle Strips

Problema



Triangle Fans

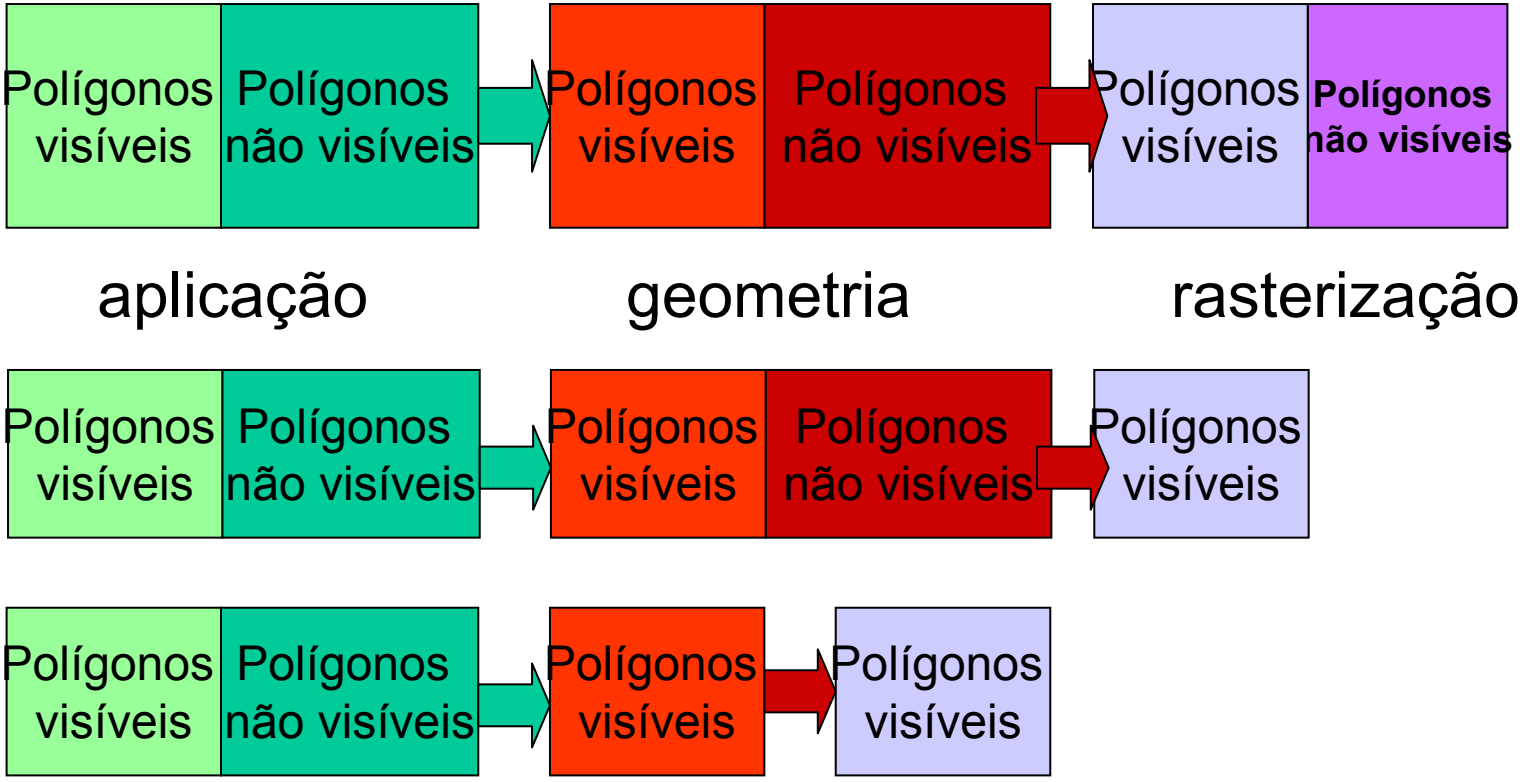
Cada Triangulo: V_1, V_{i+1}, V_{i+2}



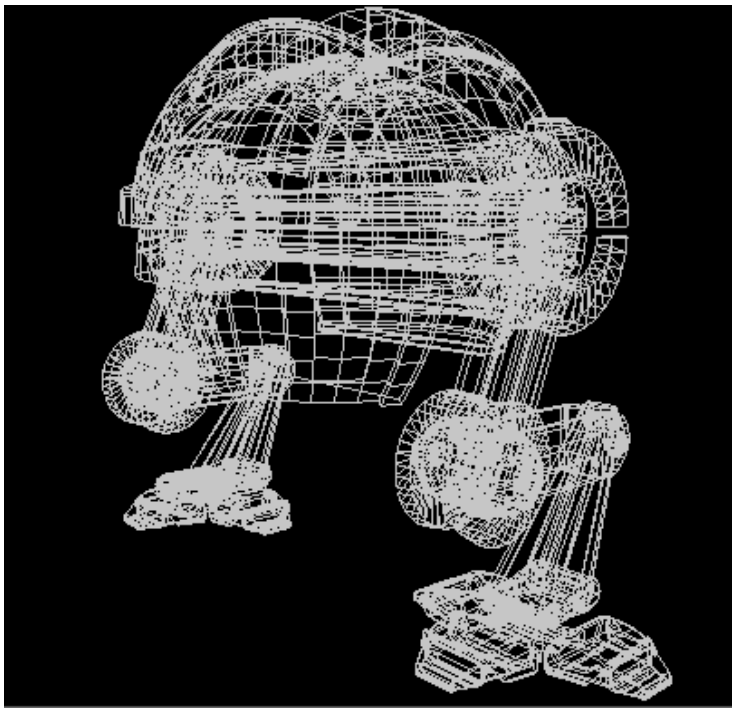
c. Métodos de Culling

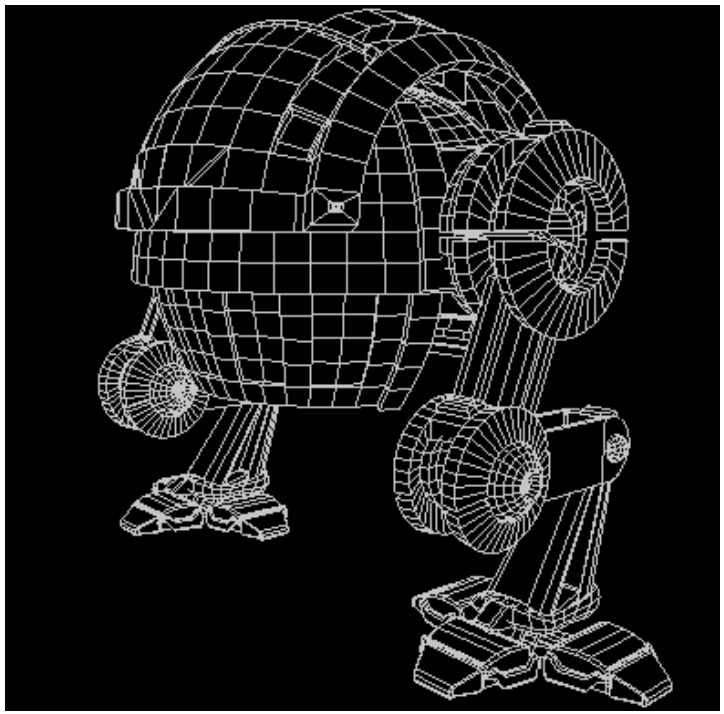
Culling

- Estrutura de Dados
- Estágio do Culling
- Gargalo de Culling



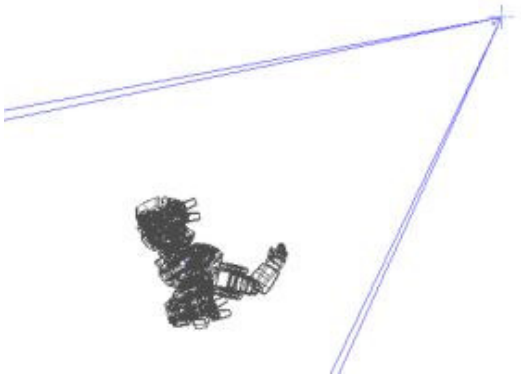
Backface Culling



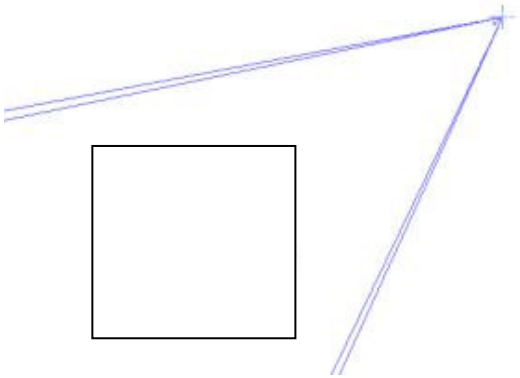


Estágio de Aplicação x Estágio de Geometria

Culling baseado em estruturas hierárquicas

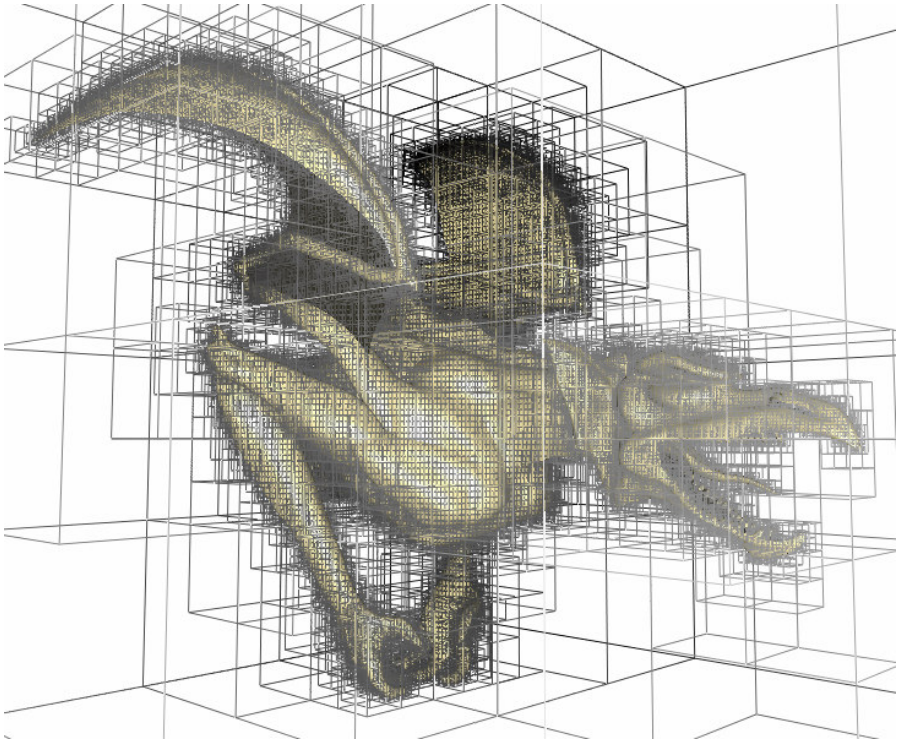


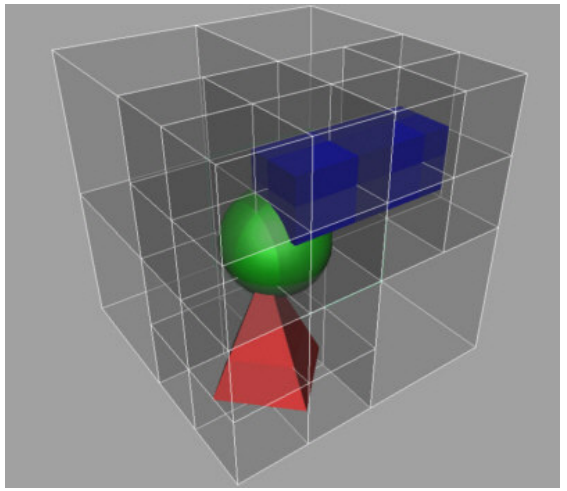
Campo de Visão

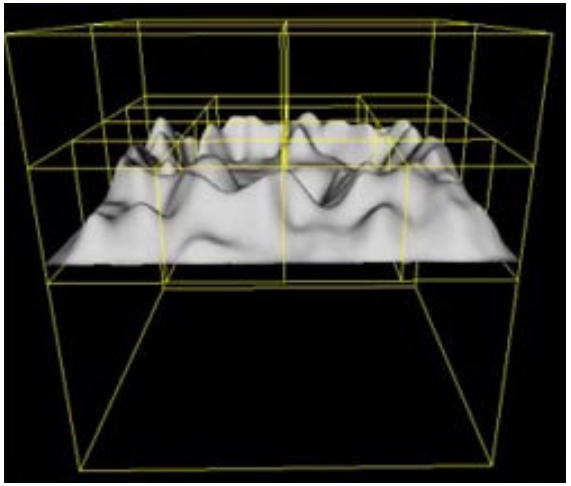


Bounding Volumes

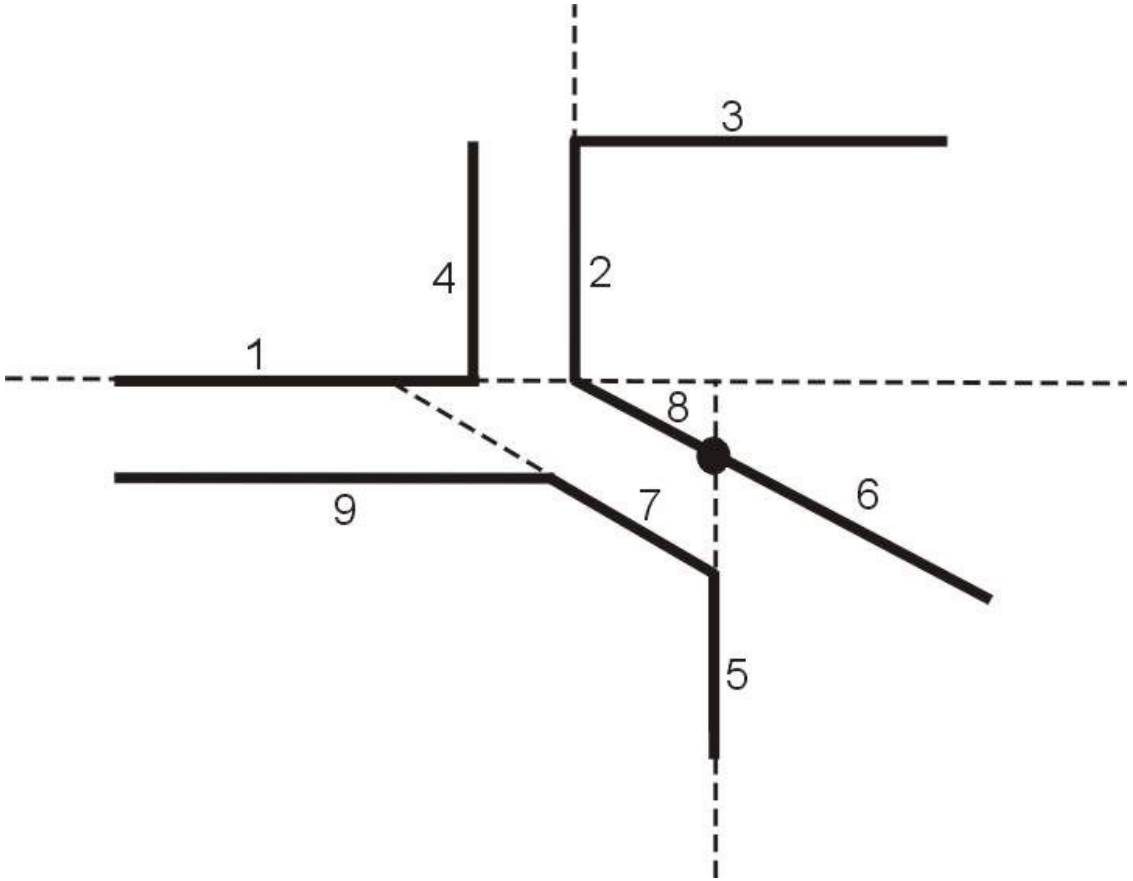
Octree – Estrutura Hierárquica



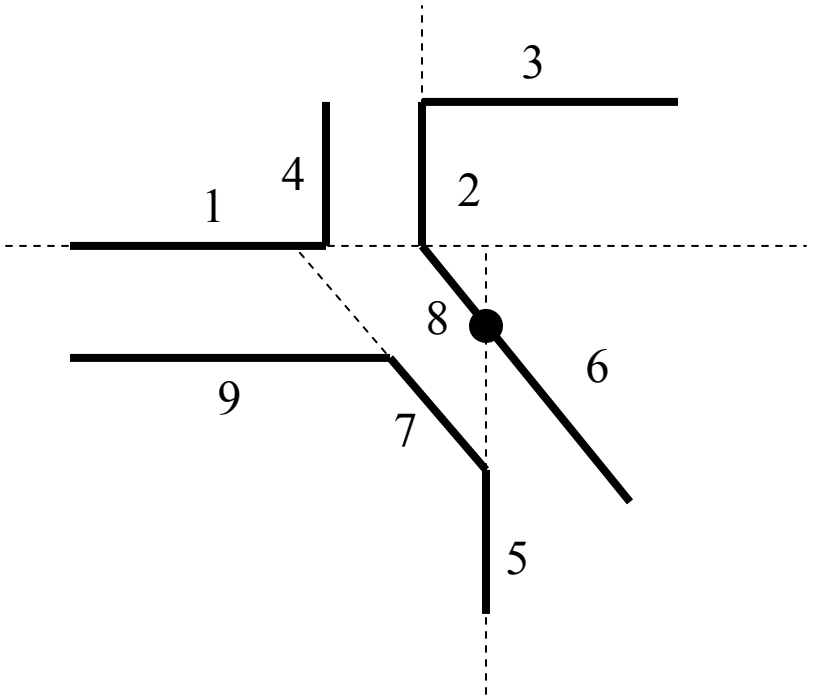


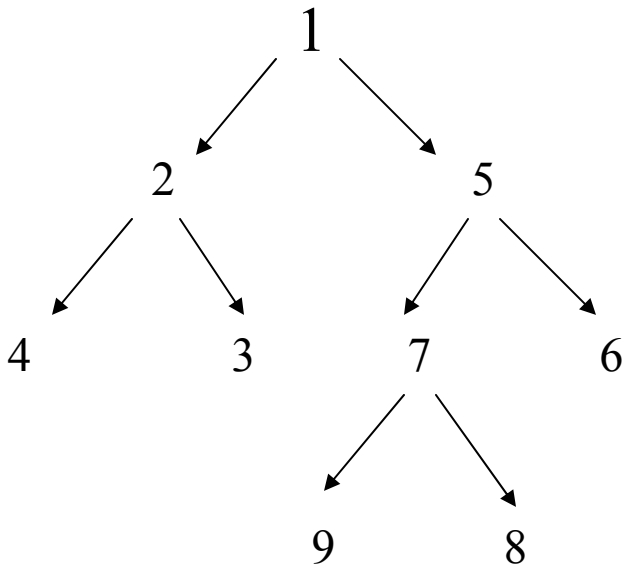


BSPs



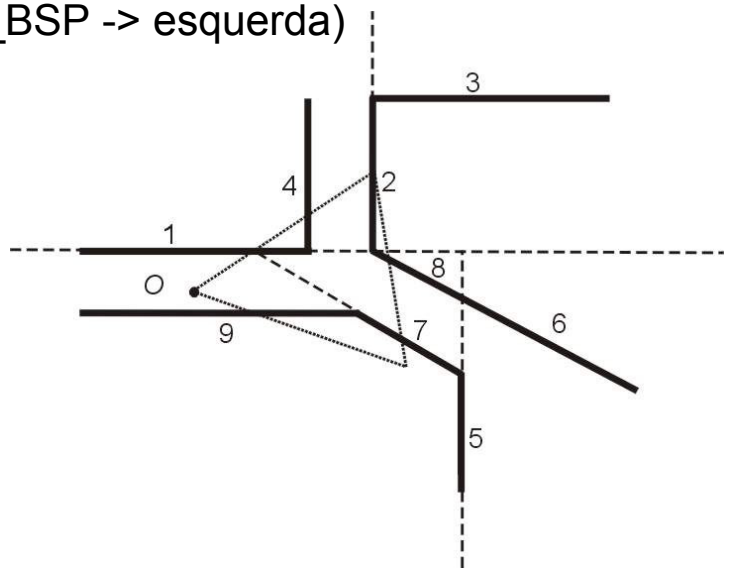
BSPs



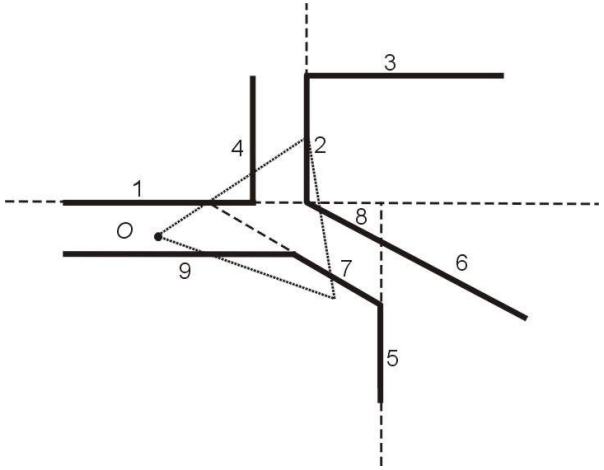


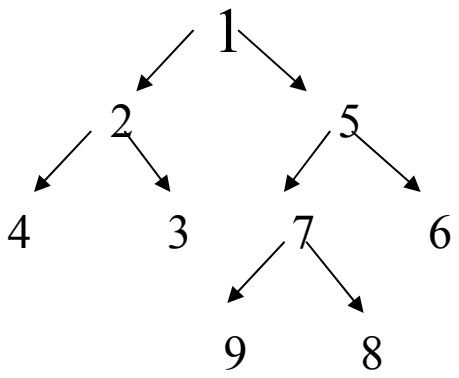
BSPs - Visualização

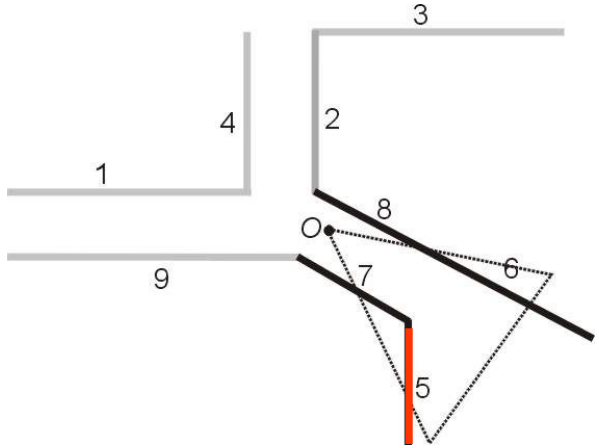
```
Desenha_BSP (O, no_Arvore_BSP)
    Se (no_Arvore_BSP é folha)
        Plota_Poligono (no_Arvore_BSP)
    Senão
        Testa de que lado O está em relação ao plano de no_arvore_BSP
        Se O estiver à direita do plano
            Desenha_BSP (O, no_Arvore_BSP -> esquerda)
        Plota_Poligono (no_Arvore_BSP)
            Desenha_BSP (O, no_Arvore_BSP -> direita)
        Se O estiver à esquerda do plano
            Desenha_BSP (O, no_Arvore_BSP -> direita)
        Plota_Poligono (no_Arvore_BSP)
            Desenha_BSP (O, no_Arvore_BSP -> esquerda)
```



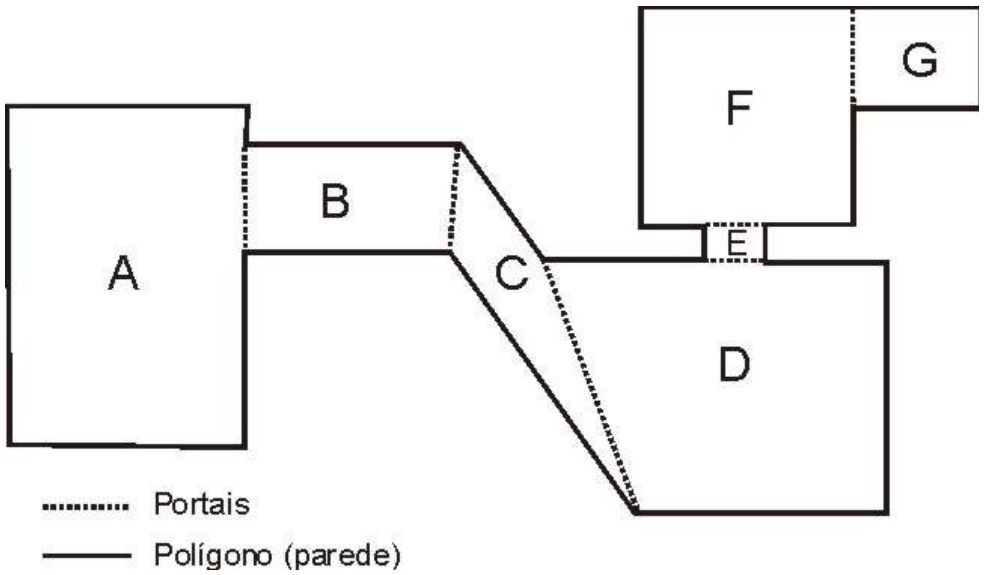
BSPs - Visualização







Portais



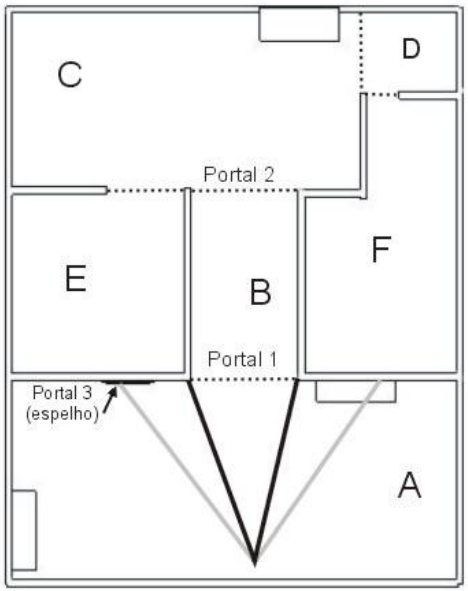
PVS

	A	B	C	D
A	1	1	0	0
B	1	1	1	1
C	0	1	1	1
D	0	1	1	1

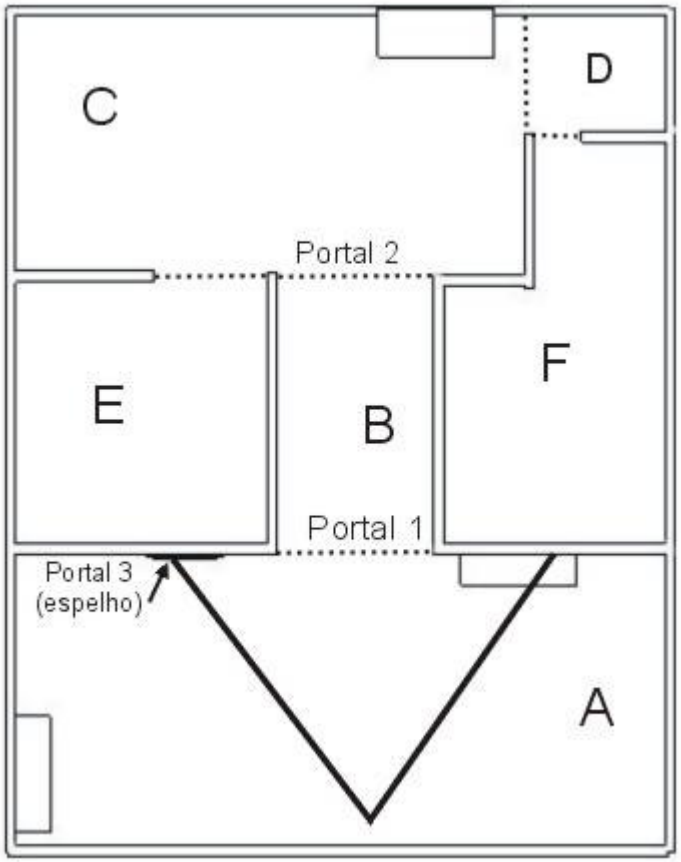
Portais - Visualização

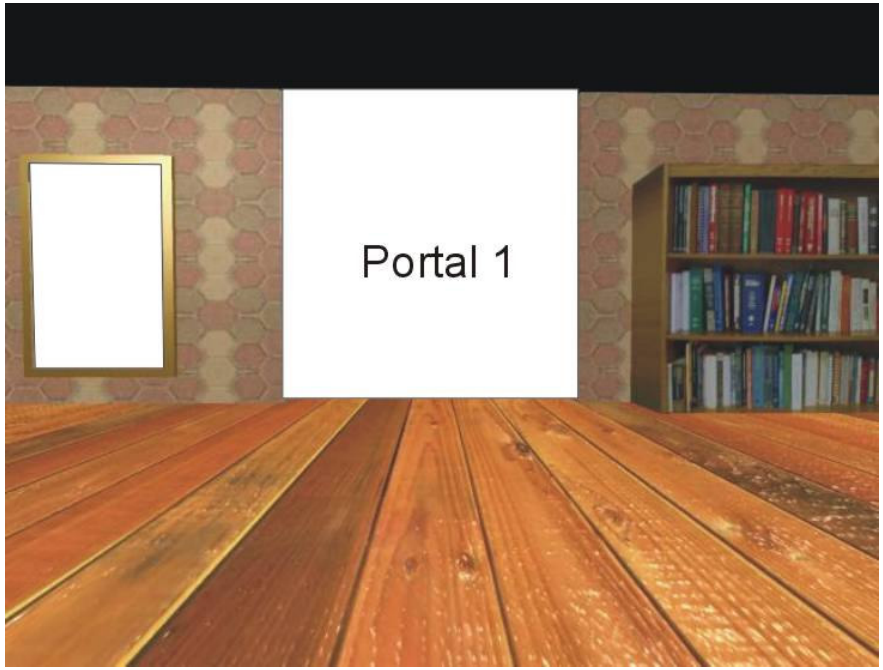
```
Render (Celula* celula, Camera* camera, View* view)
{
   Para todos os polígonos da celula onde se encontra o observador faça
            Se o polígono não é portal então
                  Plote o polígono clipado para a tela
            Senão
                  Crie um novo view utilizando o portal e o view corrente
                 render (célula_vizinha, camera, novo_view)
}
```

1) Qual a condição de parada da recursão?
2) Onde entram as PVS nesta história?

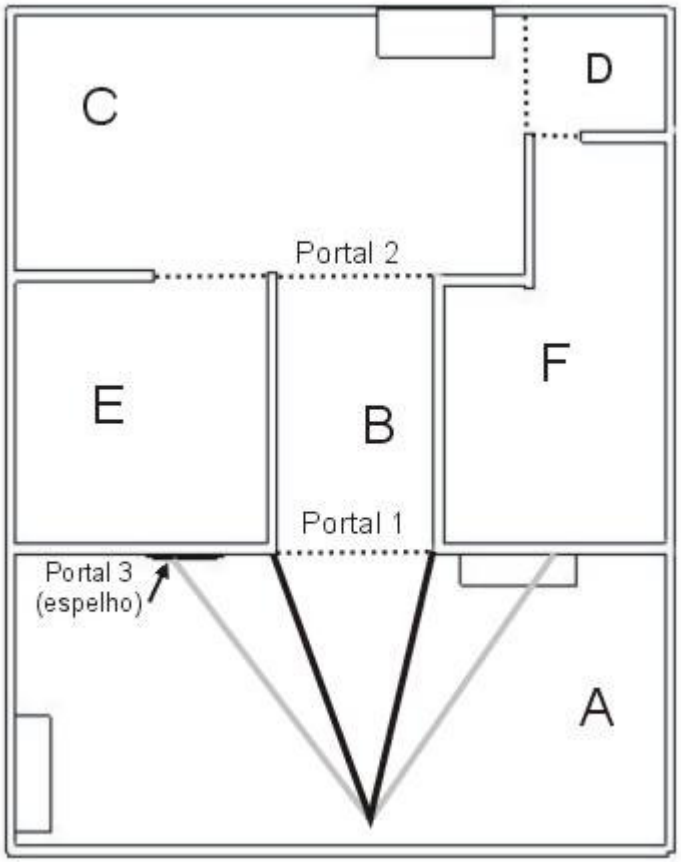


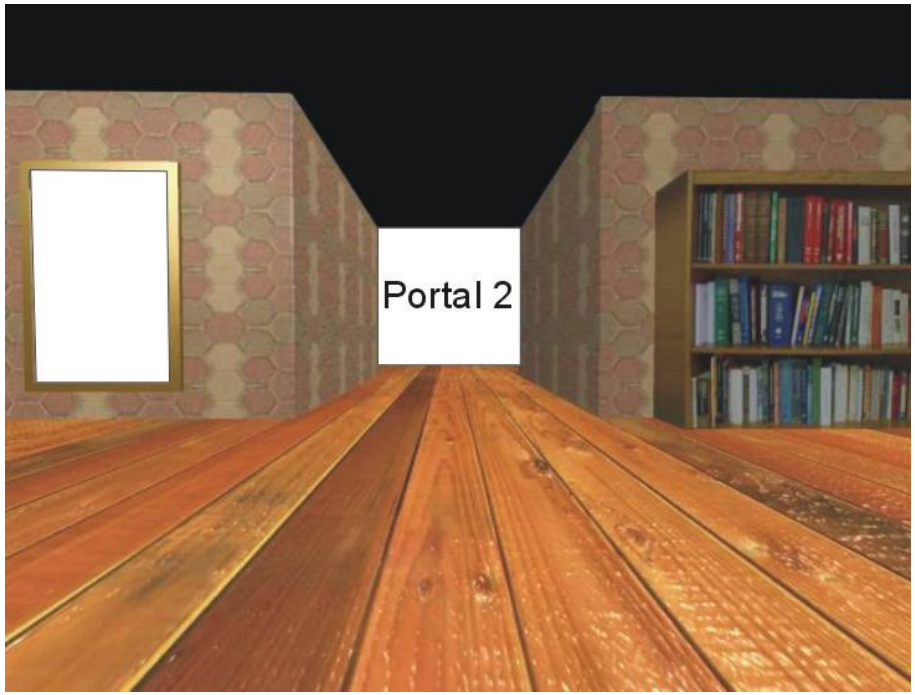
Portais - Visualização



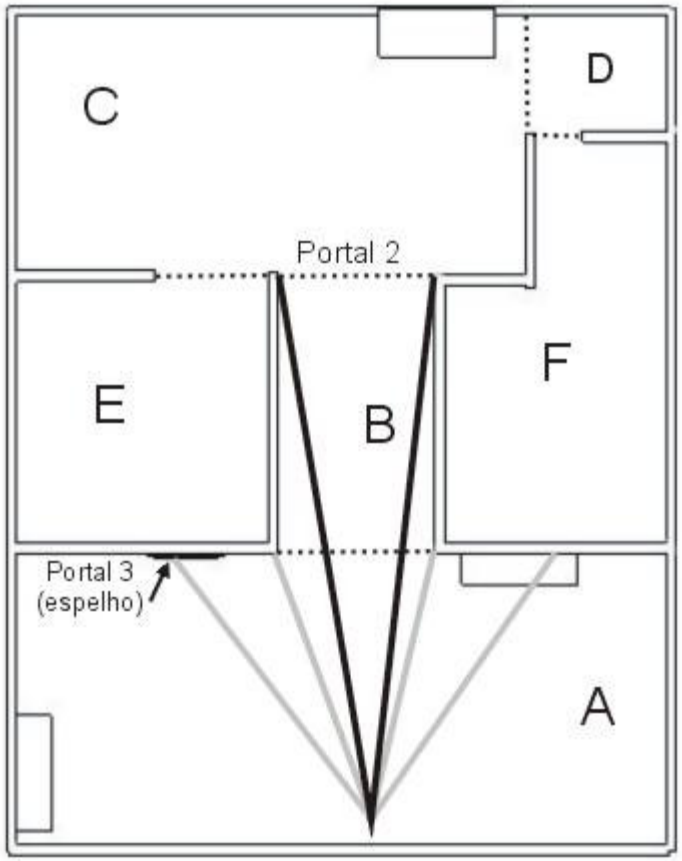


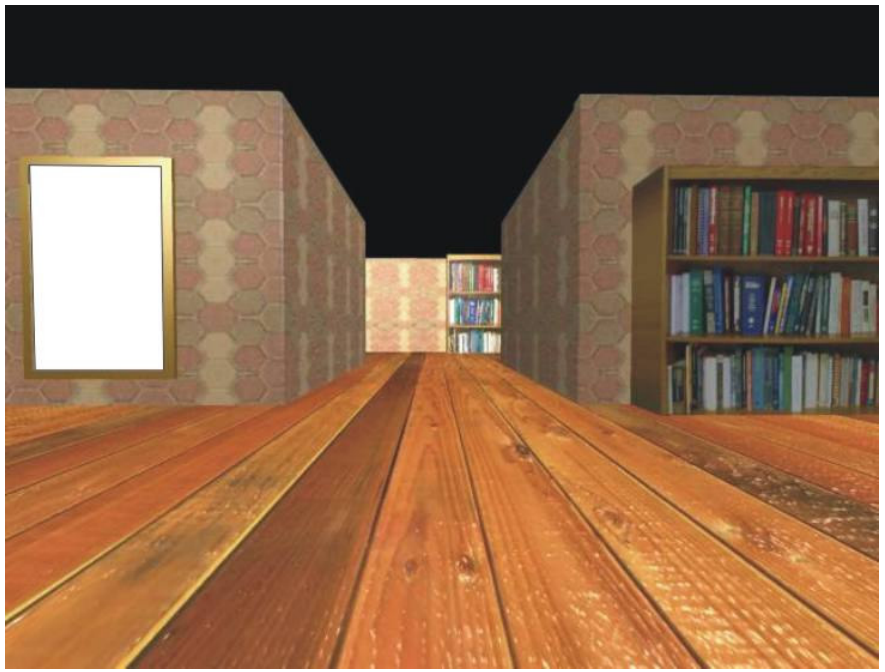
Portais - Visualização



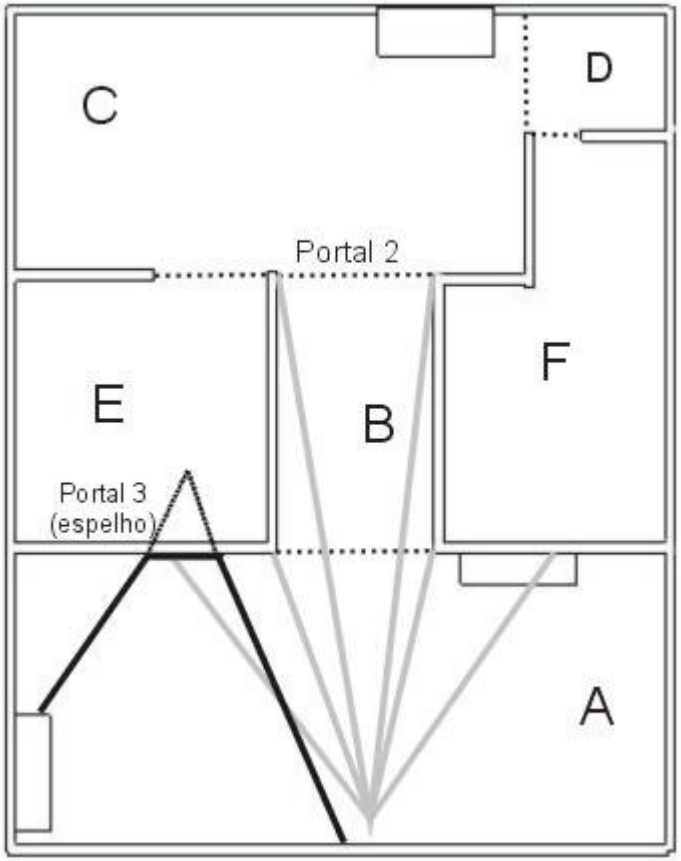


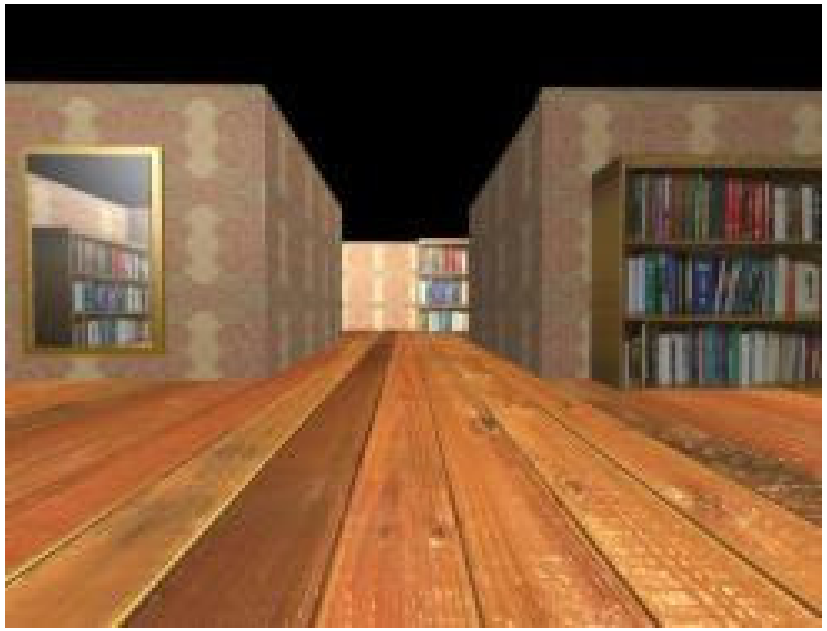
Portais - Visualização





Portais - Visualização





d. Level of Details

Idéia fundamental: simplificar modelos de acordo com distância

d. LOD Switching

Efeito de Popping

d. LOD Switching

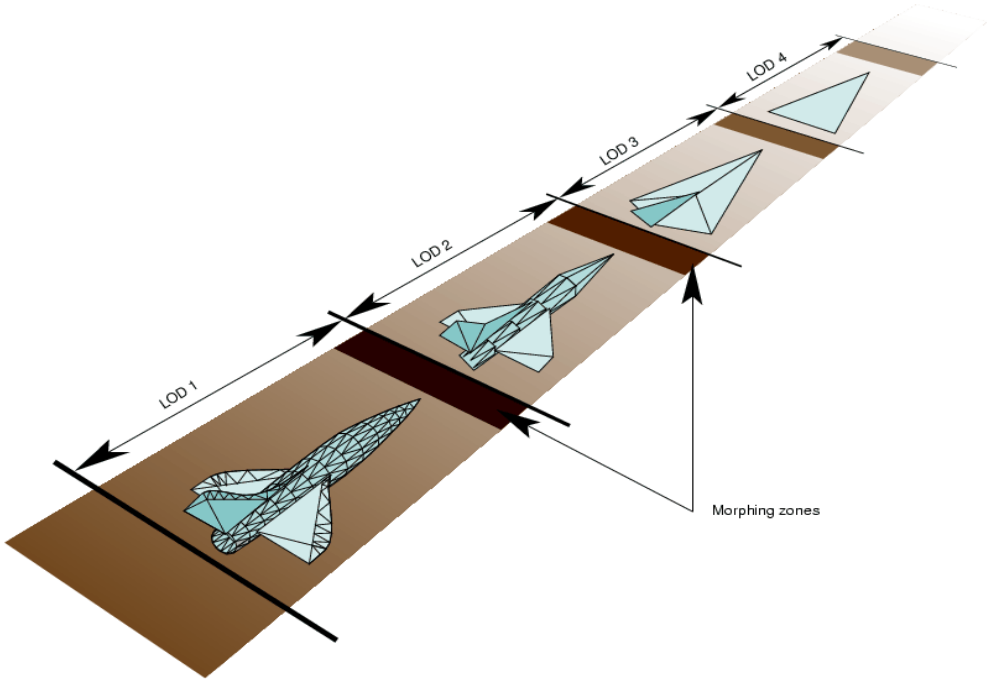
1) Geometrias Discretas de LOD

- Permite indexação direta de triângulos pela DMA *(Direct Memory Access)*

d. LOD Switching

2) Blend LODs

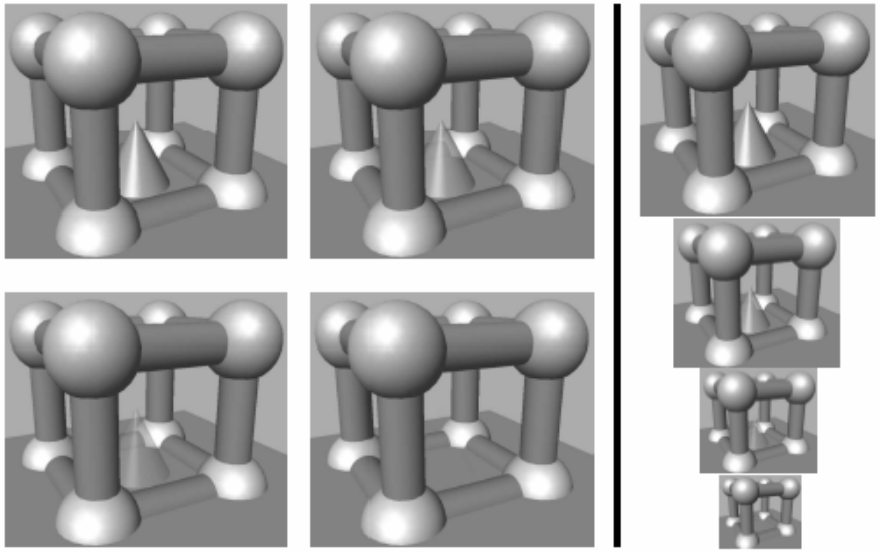
- 2 etapas de rendering
- Z Buffer desligado para o LOD original
- Encarecimento do Pipeline



d. LOD Switching

2) Alpha LODs

- Objeto vai desaparecendo gradativamente
- Alpha aumenta `a medida que camera se afasta

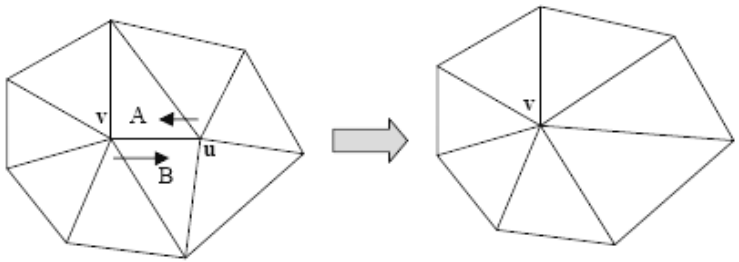


d. LOD Switching

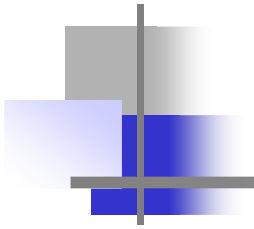
2) Continuous LODs (CLOD)

- Redução Dinâmica
- Problemas: desconhecimento dos modelos, recuperação, dificuldade de lidar com strips, problemas com texturas

- Algumas arestas são selecionadas
- De acordo com critérios, os vértices podem ser animados de forma a colapsarem
- Quando dois vértices estão muito próximos, ocorre um colapso e novos polígonos surgem



Muito útil para terrenos

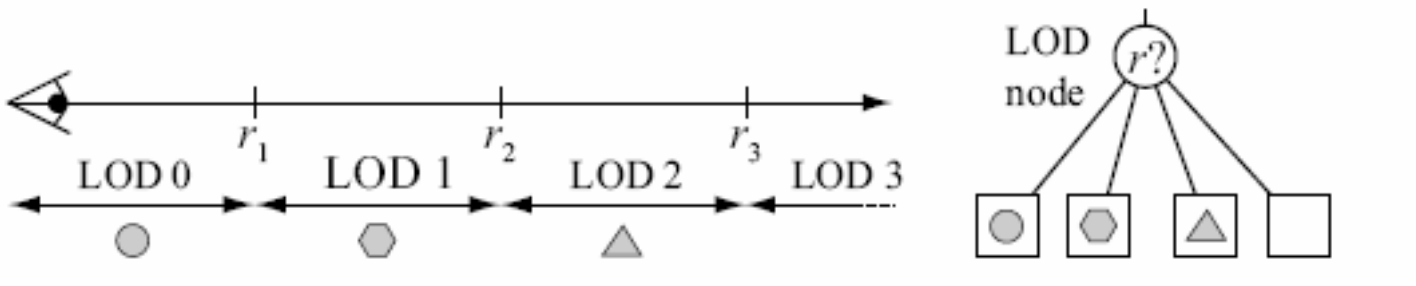


d. LOD Selection

Análise da *Benefit Function*

d. LOD Selection

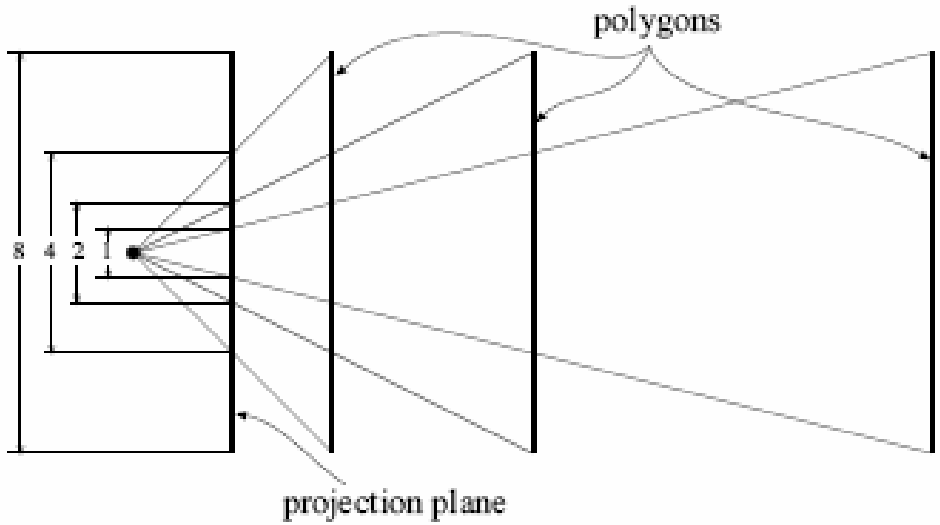
Range Based



d. LOD Selection

Projected Area-Based

$$p = \frac{nr}{\mathbf{d} \cdot (\mathbf{c} - \mathbf{v})}.$$



p: área projetada
n: distância do observador ao plano de projeção
r: raio da bounding sphere
d: vetor observador normalizado
c: centro do bounding sphere
v: posição do observador
d.(c – v): projeção do centro da esfera ao vetor de observador -> distância ao vetor de observação

d. LOD Selection

Outros métodos:

-Importância de um objeto

- Focus based Selection

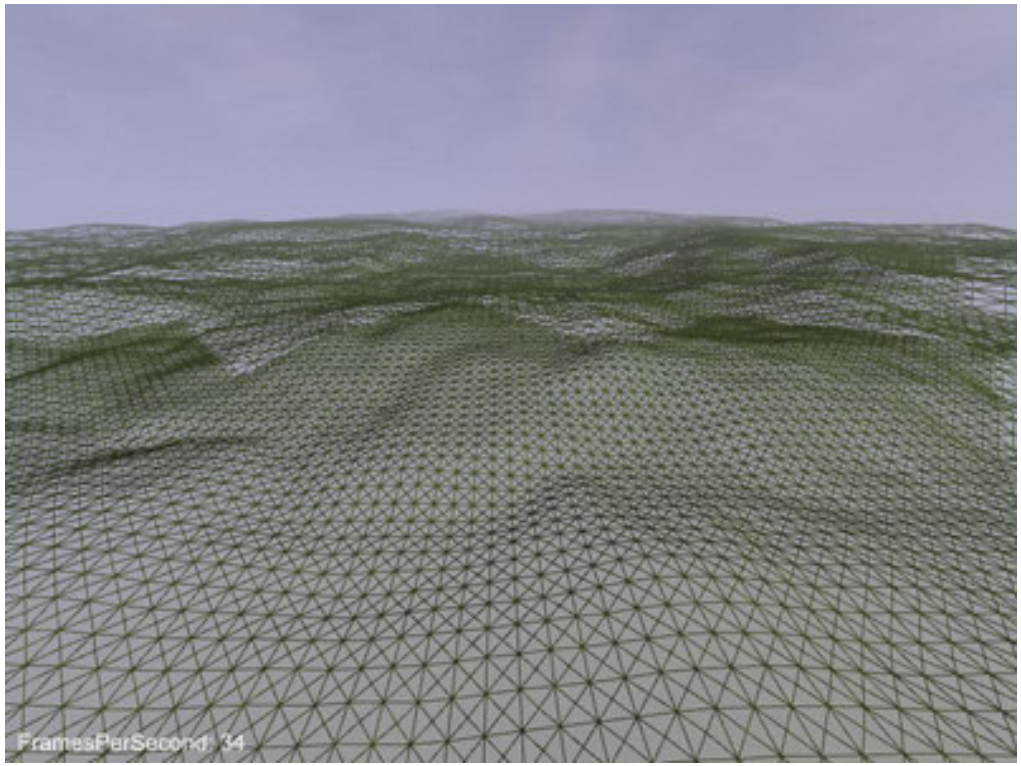
- Motion Based Selection

- Polygon Budget

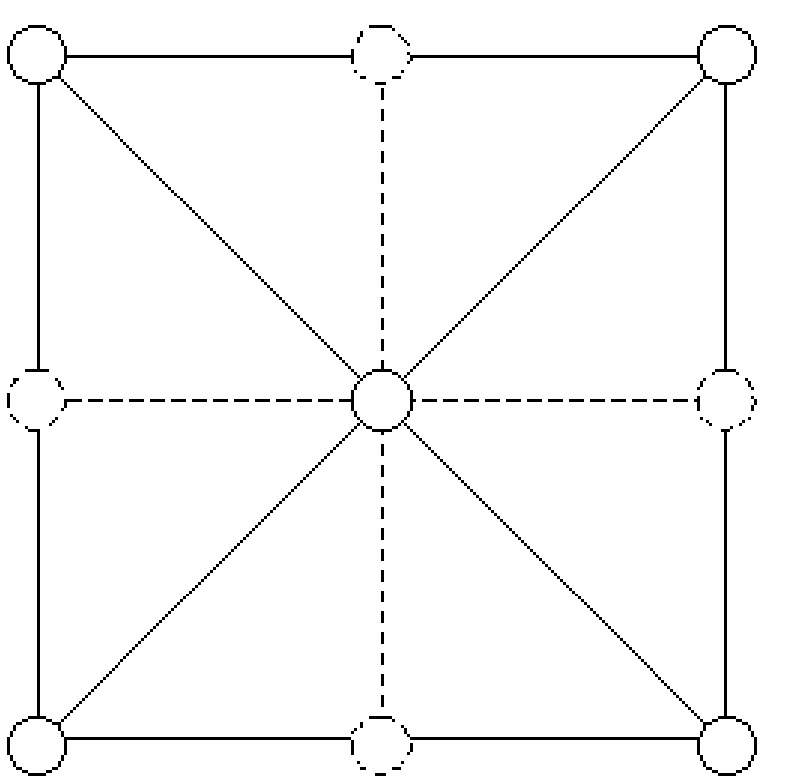
d. LOD em terrenos

Problema: ordem grande de magnitude: ex. 1m a 100000m

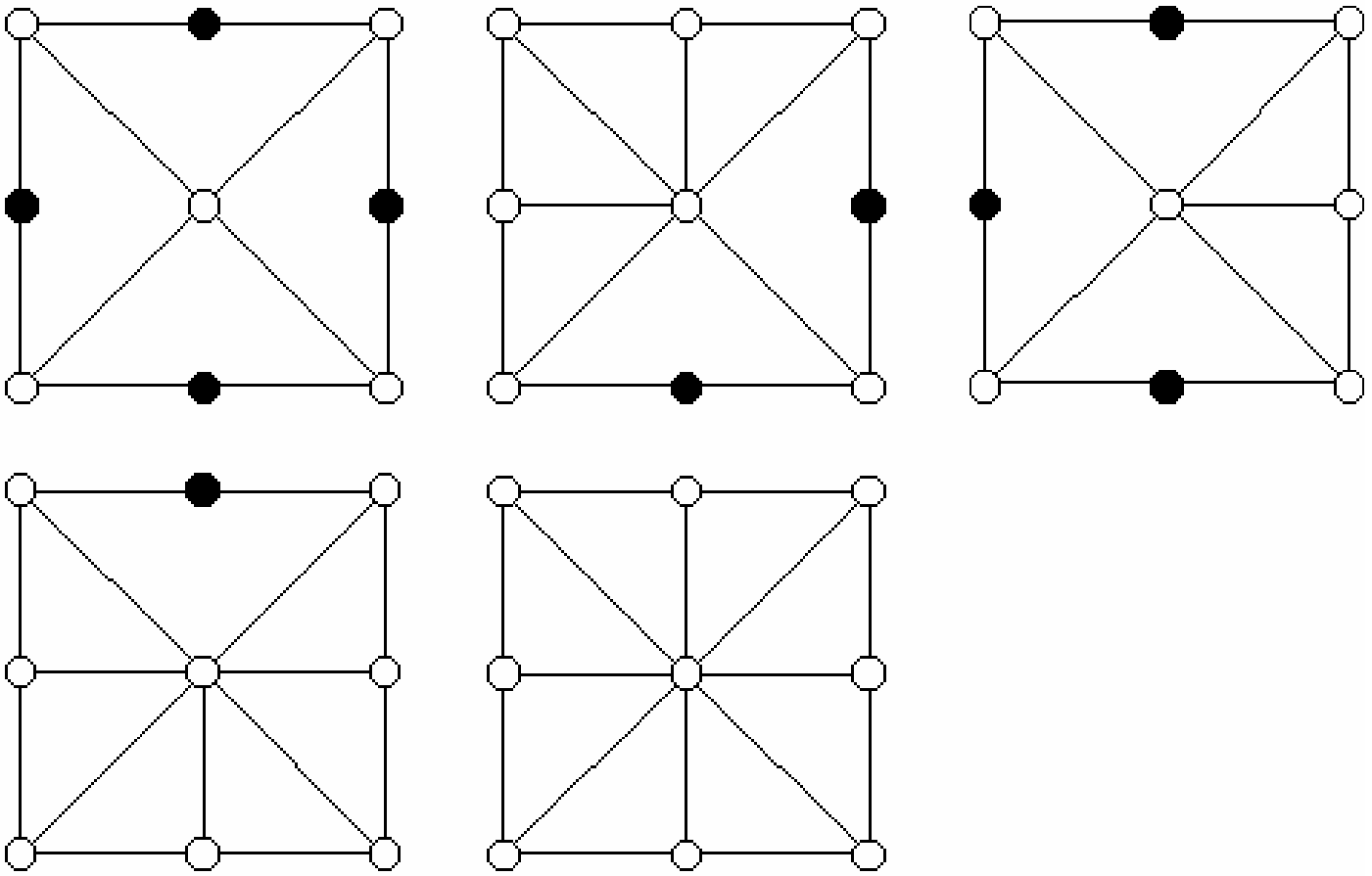
Tratamento Força Bruta



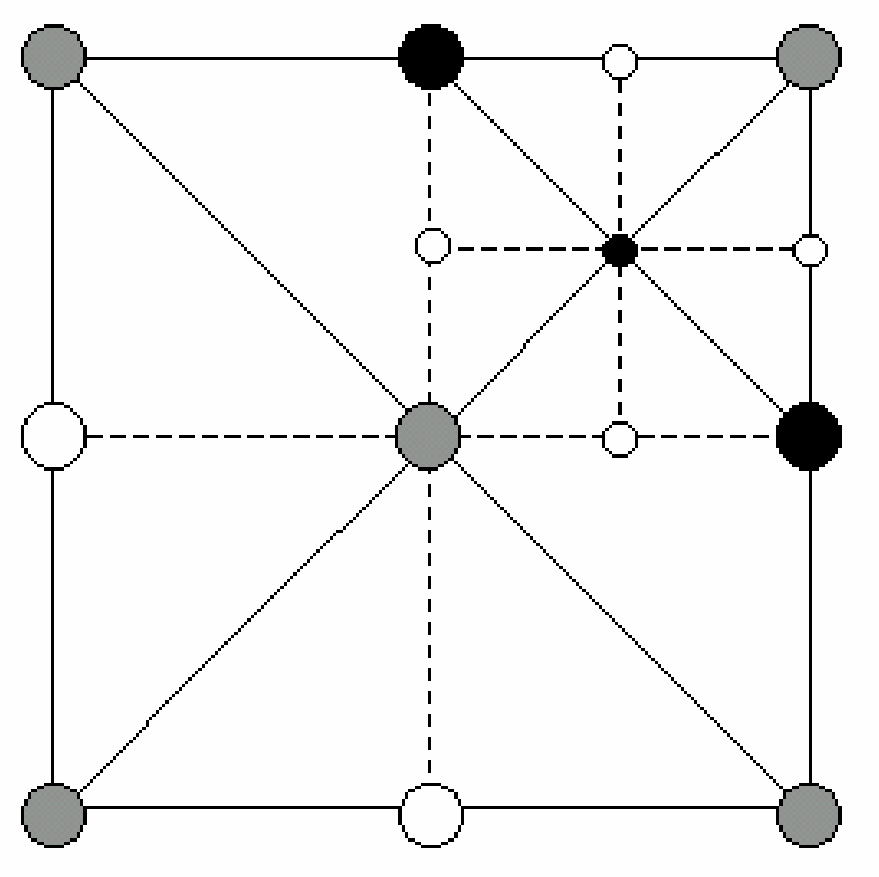
d. LOD em terrenos



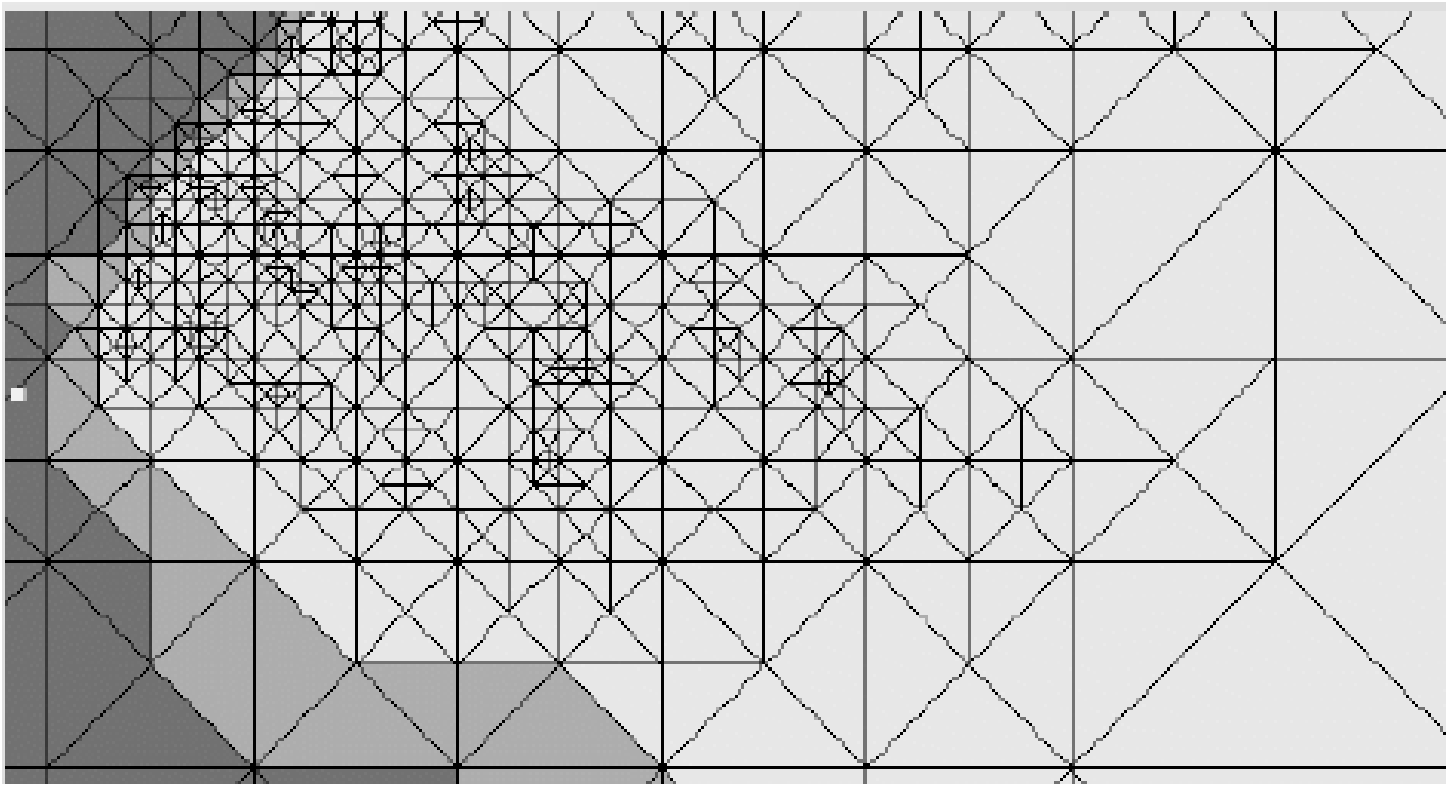
d. LOD em terrenos

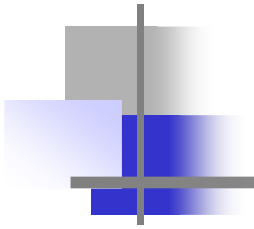


d. LOD em terrenos

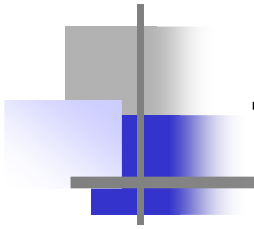


d. LOD em terrenos





Mais uma última coisa sobre Geometria... (prometo)



Texturas

Importância das texturas

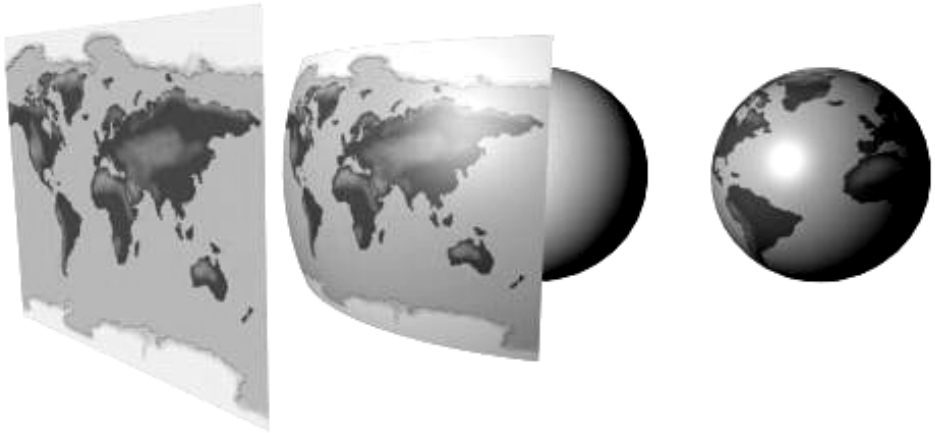
Redução da Geometria da cena

Importância das texturas

Ambientação do jogo

Definição e tipos de texturas

Bidimensional 2D => 3D

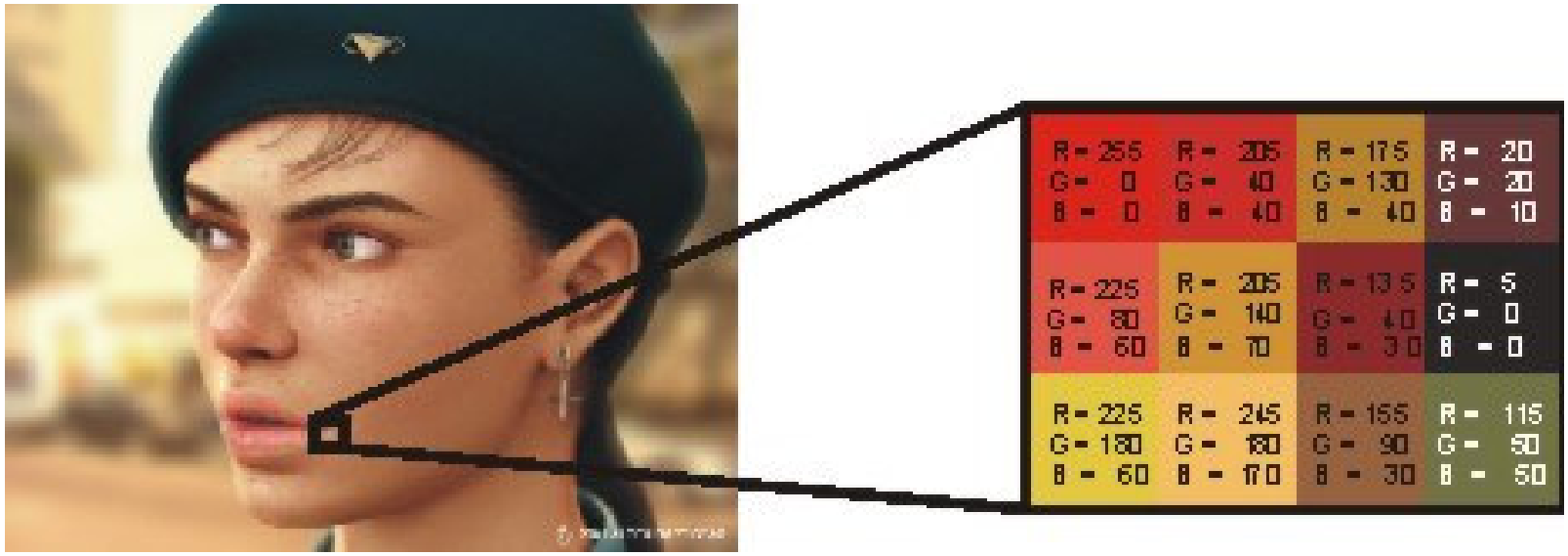


Tridimensional 3D => 3D

Imagem

Texel

Matrizes de texels

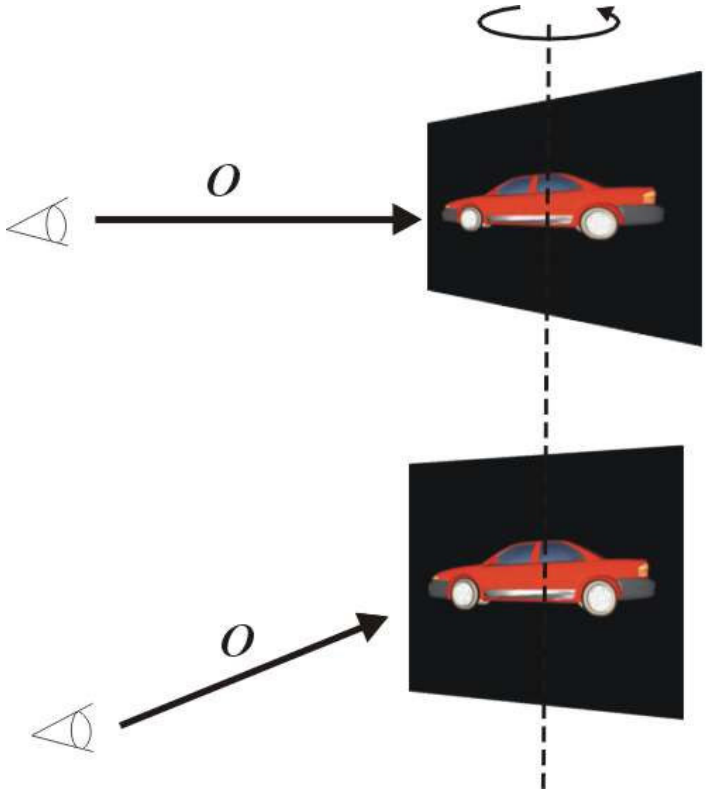


Imagem

Alpha Channel (32 bits)



Sprites



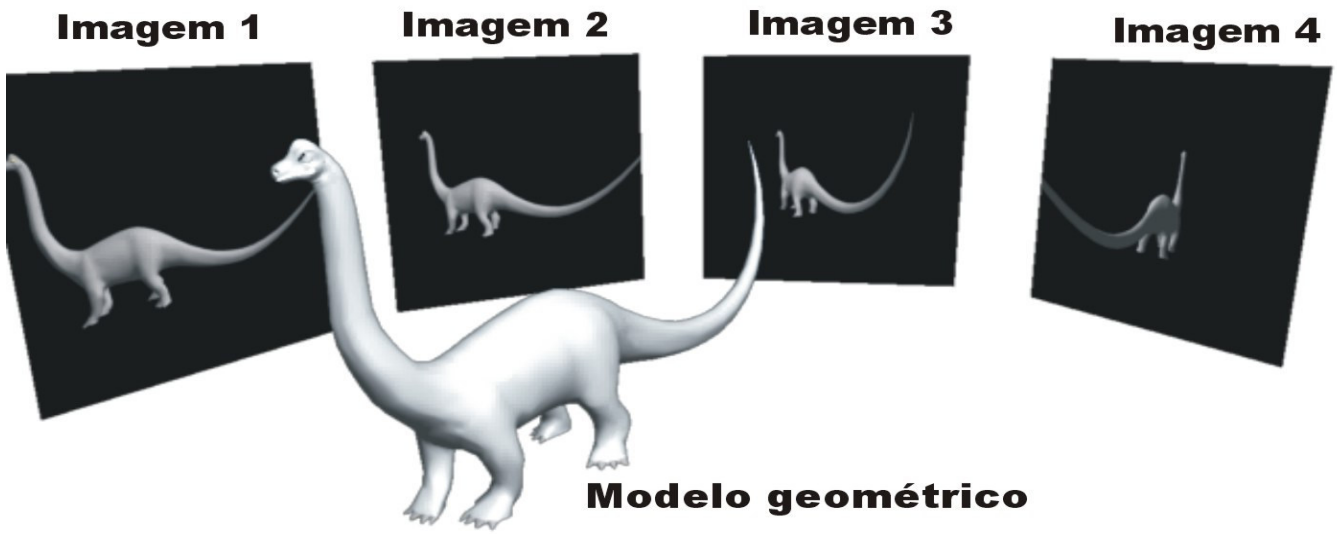
Operador R(E, O)

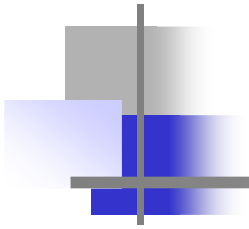
Garante a propriedade:

$\theta = \cos^{-1}(O.N_E) = 0$

Problemas deste tipo de sprite

Sprites Alternáveis





Diversos tipos de aplicação de textura

Cor da superfície

Mapa de reflexo (cromo)

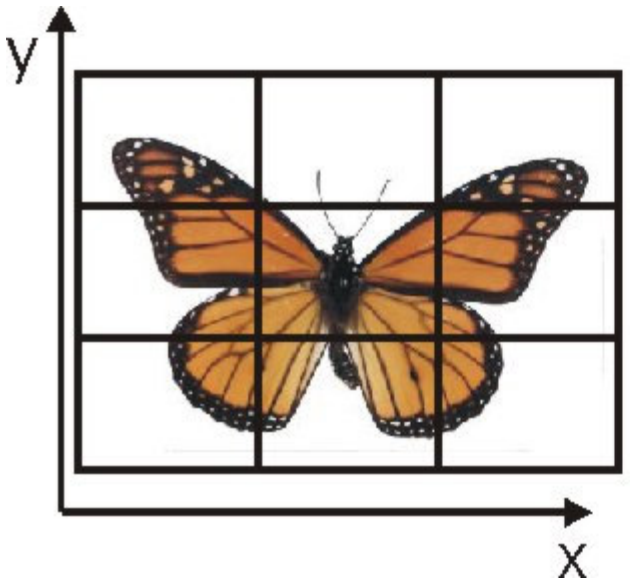
Mapa de opacidade (grade)

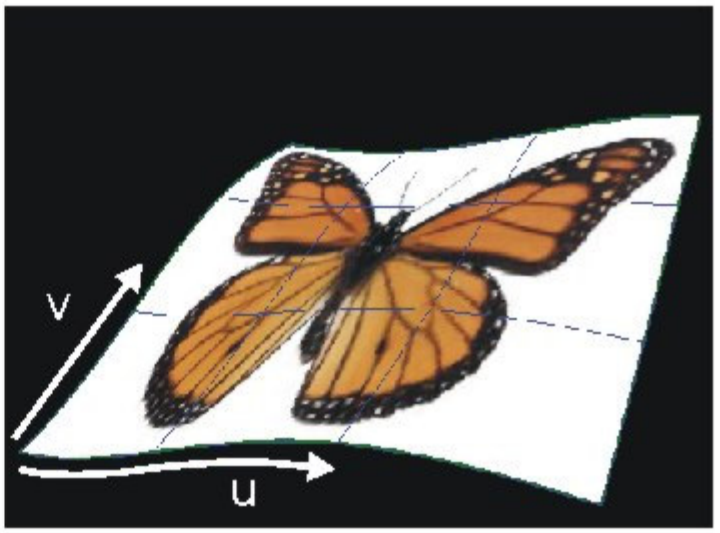
Mapa de relevo (laranja)

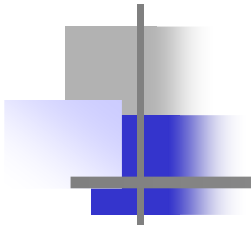
Projeção da Textura

Problema: Aplicar um objeto 2D sobre outro 3D

Parametrização de uma superfície

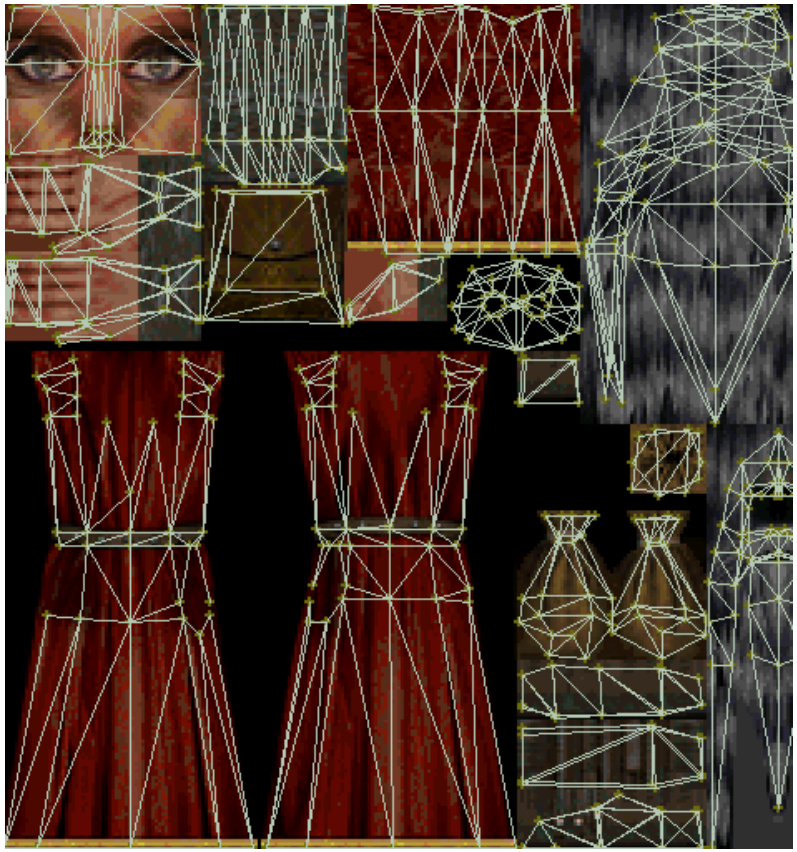




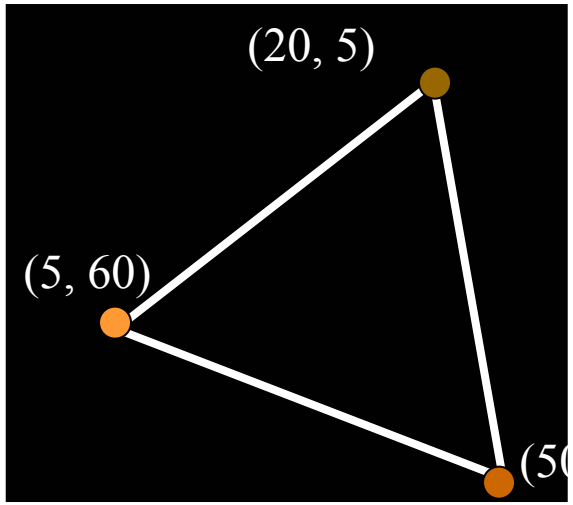


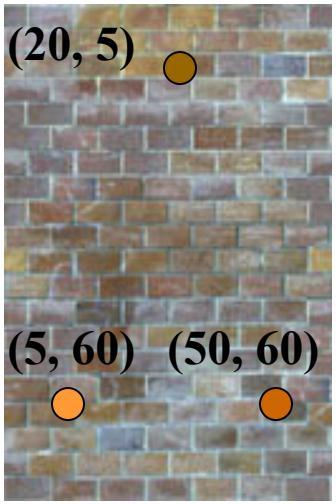
Projeção da Textura - Exemplo de esfera

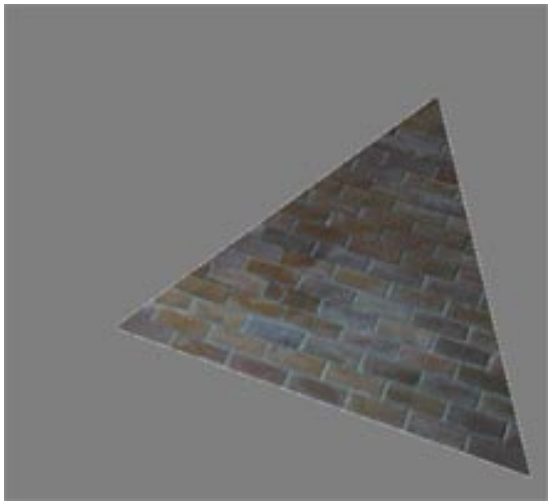
Métodos mais comuns de aplicação de texturas em jogos

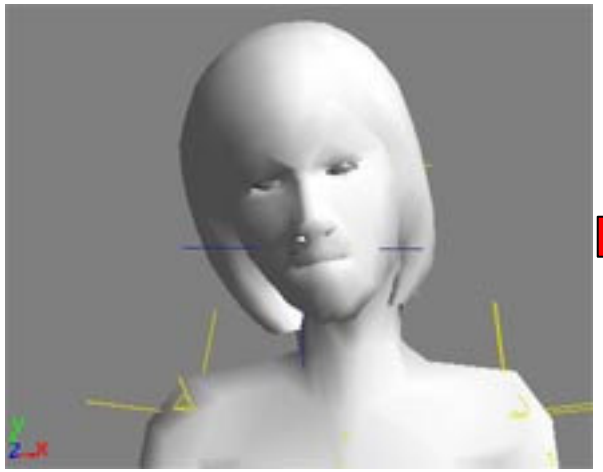


Mas como funciona no Pipeline Gráfico TR?





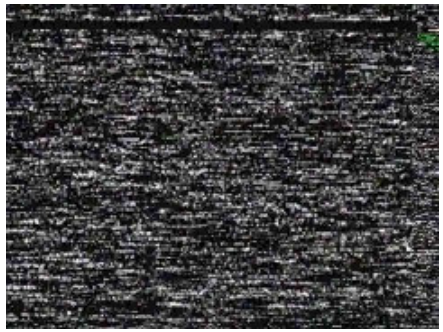




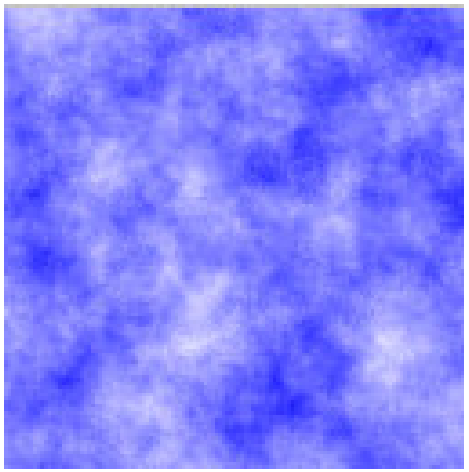
Texturas Procedurais

Funções Básicas (contra exemplo: função random)

- Pseudo-randomicidade;
- Não pode haver periodicidade de padrões;
- Funções devem ser estacionárias e isotrópicas.

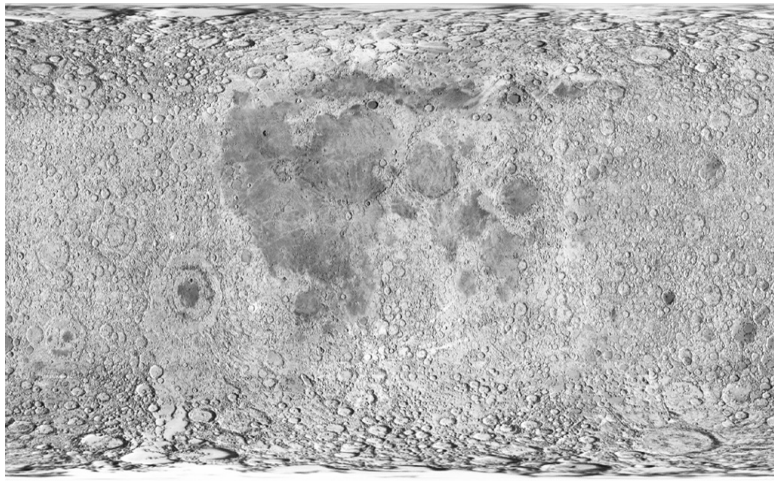


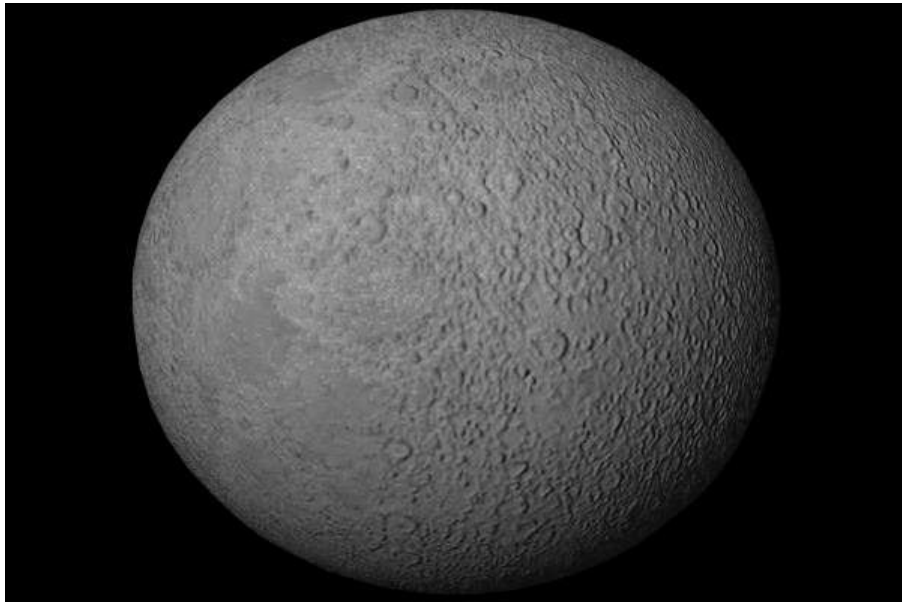
Texturas Procedurais



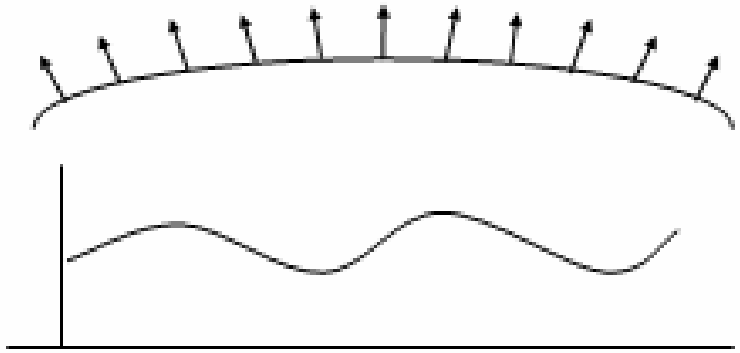
Implementações em hardware

Bump Mapping



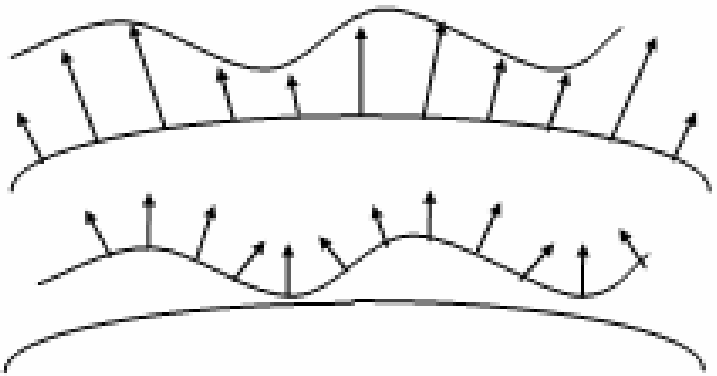


Bump Mapping



Superfície original O(u)

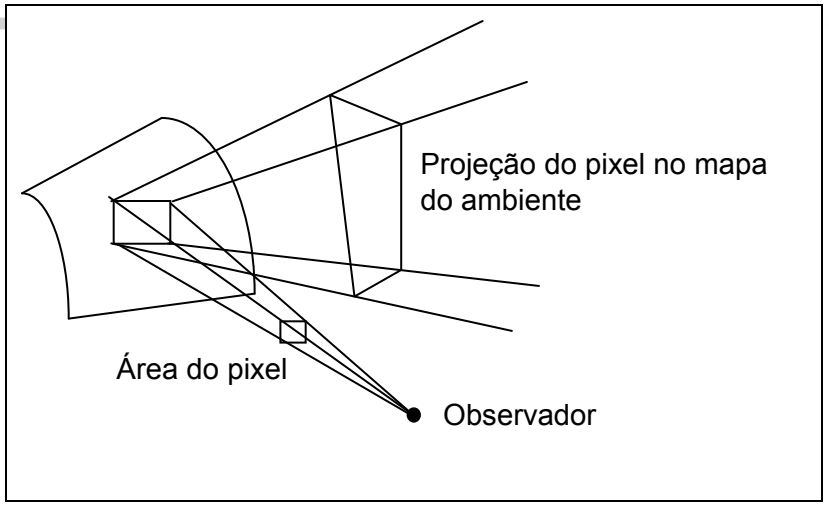
Uma bump map B(u) qualquer



Ajustando O(u) usando B(u)

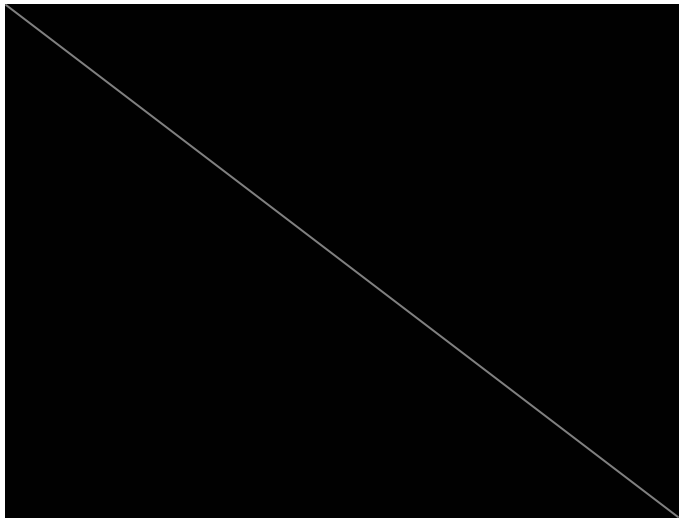
Novos vetores da superfície O(u)

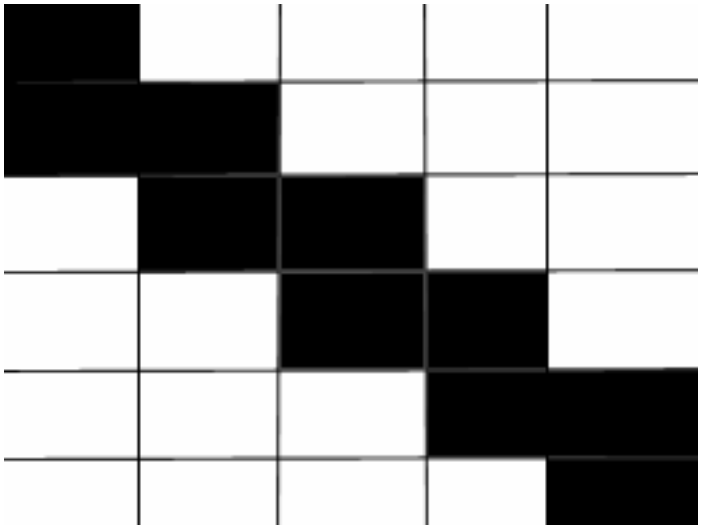
Enviroment Mapping



Aliasing e métodos de correção (amostragem)

Contínuo x Discreto

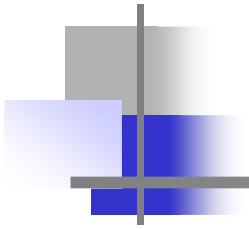




Problema: texturas pequenas para áreas grandes

Light Maps

Cor_Pixel = Texel x Light_Map;



f. Introdução a APIs gráficas

APIs Gráficas e OpenGL

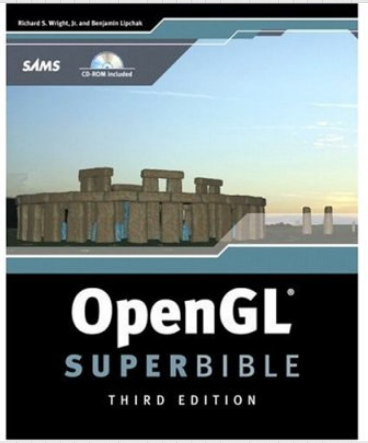
- Conceitos Básicos

- OpenGL x DirectX

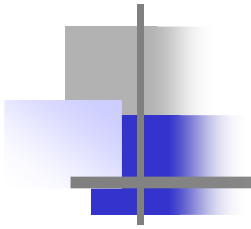
- GLUT

- Renderização baseada em estados acumulativos

referência



OpenGL SuperBible (3rd Edition)
Richard S Wright, Benjamin Lipchak



Windows Programming

Arquitetura Básica

Callback Function

WinMain

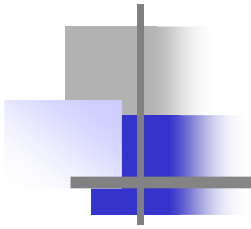
Aplication

Callback Function

```
LRESULT CALLBACK MainWindowProc (  ... )
{
        // iniciar handlers


        switch (Msg)
        {
                case WM_CREATE:
                // Cria Janela e Render Context

                case WM_CLOSE:
                // Libera aplicação

                case WM_SIZE:
                // Reescala janela

                case WM_KEYDOWN:
                // verifica tecla usada...

        }
}
```

WinMain Function

```
Int WINAPI WinMain (...)
{

// Definição de Classes do Windows
// Registrar janela

// Criar Janela

Init_OpenGL
While (!Fim)
{
        OpenGL_Render ();
        SwapBuffers ();

}

Close_OpenGL();
```



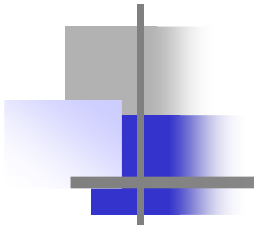
OpenGL – Conceitos Básicos

- Rendering Context

- Loop de renderização

- Back Buffer x Front Buffer

- Função de Renderização



OpenGL

1.Inicialização

Criação da Janela e do rendering context, estados iniciais, inicialização de outros componentes

2. Laço Principal

Entrada de dados, Física, IA, Renderização

3. Finalização

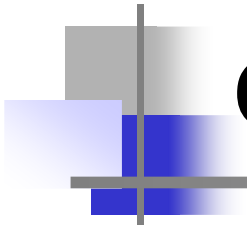
Liberação de recursos

OpenGL - Inicialização

```
// inicializa a Aplicação Gráfica.

void InitOpenGL(...)
{
        // Inicializa a OpenGL.
        g_hWnd = window;
        context = wglCreateContext(...); // cria o rendering context
        SizeOpenGLScreen(width, height); // determina o tamanho da janela

        // Inicializa a física
        InitPhysics();

        // Inicializa a IA.
        InitIA();
        ...
}
```

OpenGL - Laço

```
void MainLoop()
{
        While (TRUE)
                if (GetMessage(msg))
                {
                        if (msg == QUIT) // mensagem para terminar programa
                        break;

                // Se a mensagem é uma notificação que uma tecla válida
                // foi pressionada,  haverá um tratamento específico para esta ação.
                // O motor deverá logo em seguida aplicar os passos de física e IA e
                // renderizar a cena.
                if (msg == KEY_XXX)
                        HandleKey_XXX(...);
                        ...
                RunPhysics();
                RunAI();
                RenderScene();
                }
}
```



OpenGL – Funções de Estados

glEnable (GLenum cap)

glDisable (GLenum cap)

glIsEnabled (GLenum cap)

OpenGL - Rendering

```
void RenderScene()
{
        // Configura câmera.
        glLoadIdentity();
        gluLookAt(0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
        SetupCamera();

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        glEnable(GL_ATRIBUTO_XXX); glDisable(GL_ATRIBUTO_YYY);

        glBegin(GL_TRIANGLES);
                glColor3f(1.0, 0.0, 0.0);
                glVertex3f(2.0, 0.0, 0.0);
                glColor3f(0.0, 1.0, 0.0);
                glVertex3f(0.0, 2.0, 0.0);
                glColor3f(0.0, 0.0, 1.0);
                glVertex3f(0.0, 0.0, 2.0);
        glEnd();

        ... // Muda estados, plota mais polígonos…

        // Depois de renderizar tudo realiza o swap de buffers.
        SwapBuffers();
}
```

OpenGL – Triangle Strips

```
…

for (int x = 0; x < 3; x++)
                {
                glBegin(GL_TRIANGLE_STRIP);
                for (int z = 0; z < 3; z++)
                {
                                glVertex3f(x, 0.0, z);
                                glVertex3f((x+1.0), 0.0, z);
                                glVertex3f(x, 0.0, (z+1.0));
                                glVertex3f((x+1.0), 0.0, (z+1.0));
                }
```

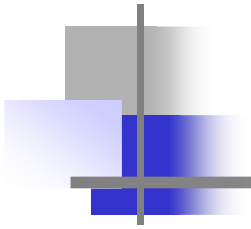
O que este trecho de código faz?...

OpenGL - Finalização

```
void ReleaseEngine(...)
{
        // Deleta o rendering context.
        wglCreateContext(...);

        // Finaliza a física (veja Seção 3.3.3).
        ReleasePhysics();

        // Finaliza a IA.
        ReleaseAI();
        ...
}
```

OpenGL – Tudo junto

```
int main(...)
{
        // Chama função para criar janela no Windows.
        window = CreateMyWindow(...);

        // Inicializa sobre a janela recém criada.
        InitOpenGL(...);

        // Chama o laço principal.
        MainLoop();

        // Finaliza.
        ReleaseEngine(...);
}
```

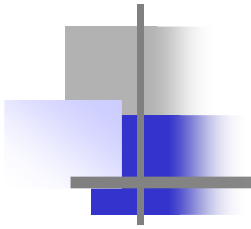


OpenGL - Tutoriais

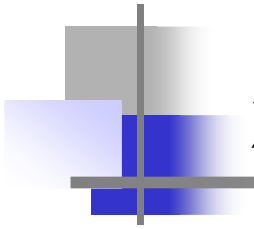
http://nehe.gamedev.net/lesson.asp?index=01

http://www.unidev.com.br/artigos.asp?id=26&cat='OpenGL'

Exercício: Fazer uma aplicação que projete um cubo 3D e faça-o girar com as setas do teclado



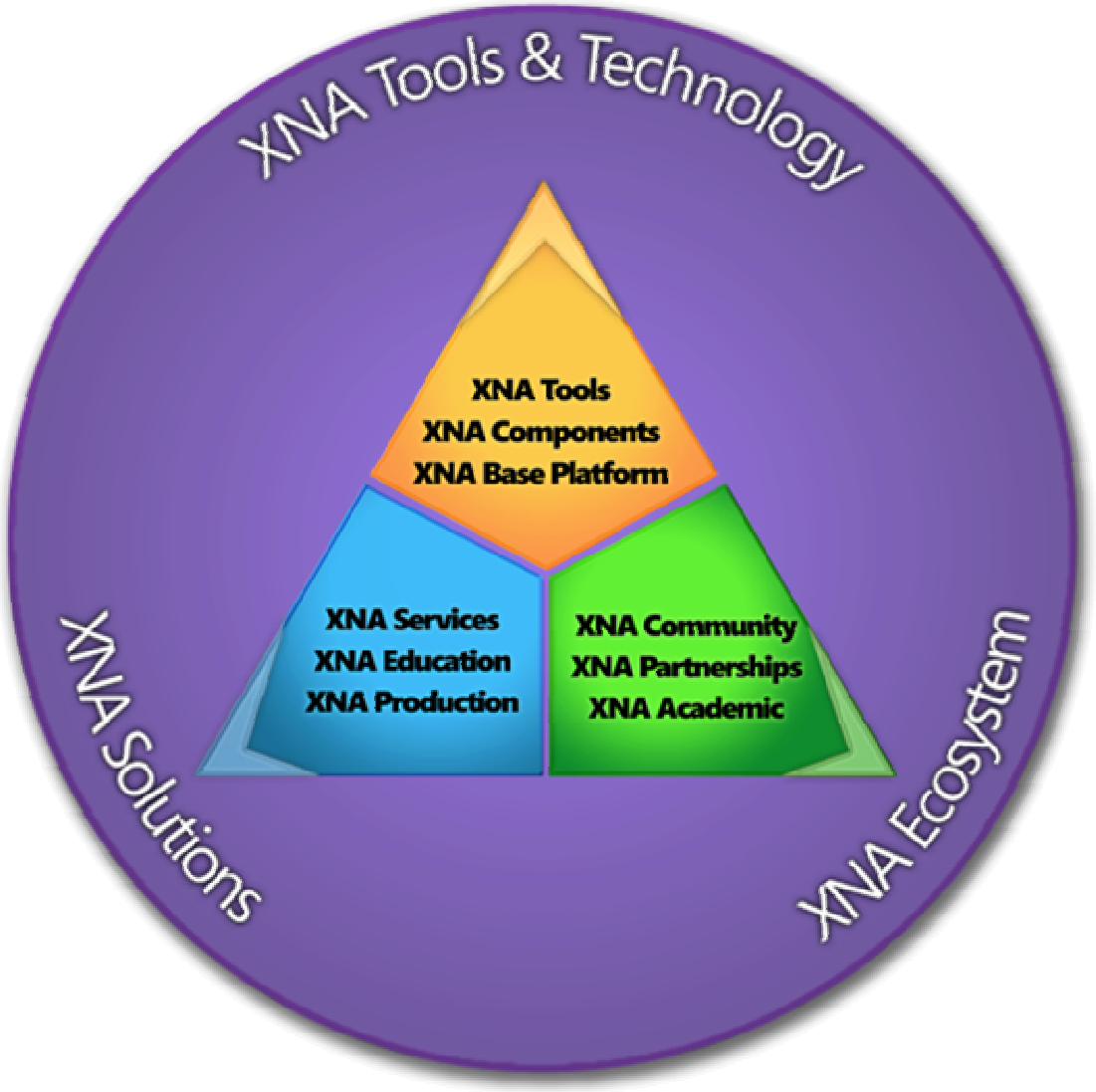
g. XNA



XNA

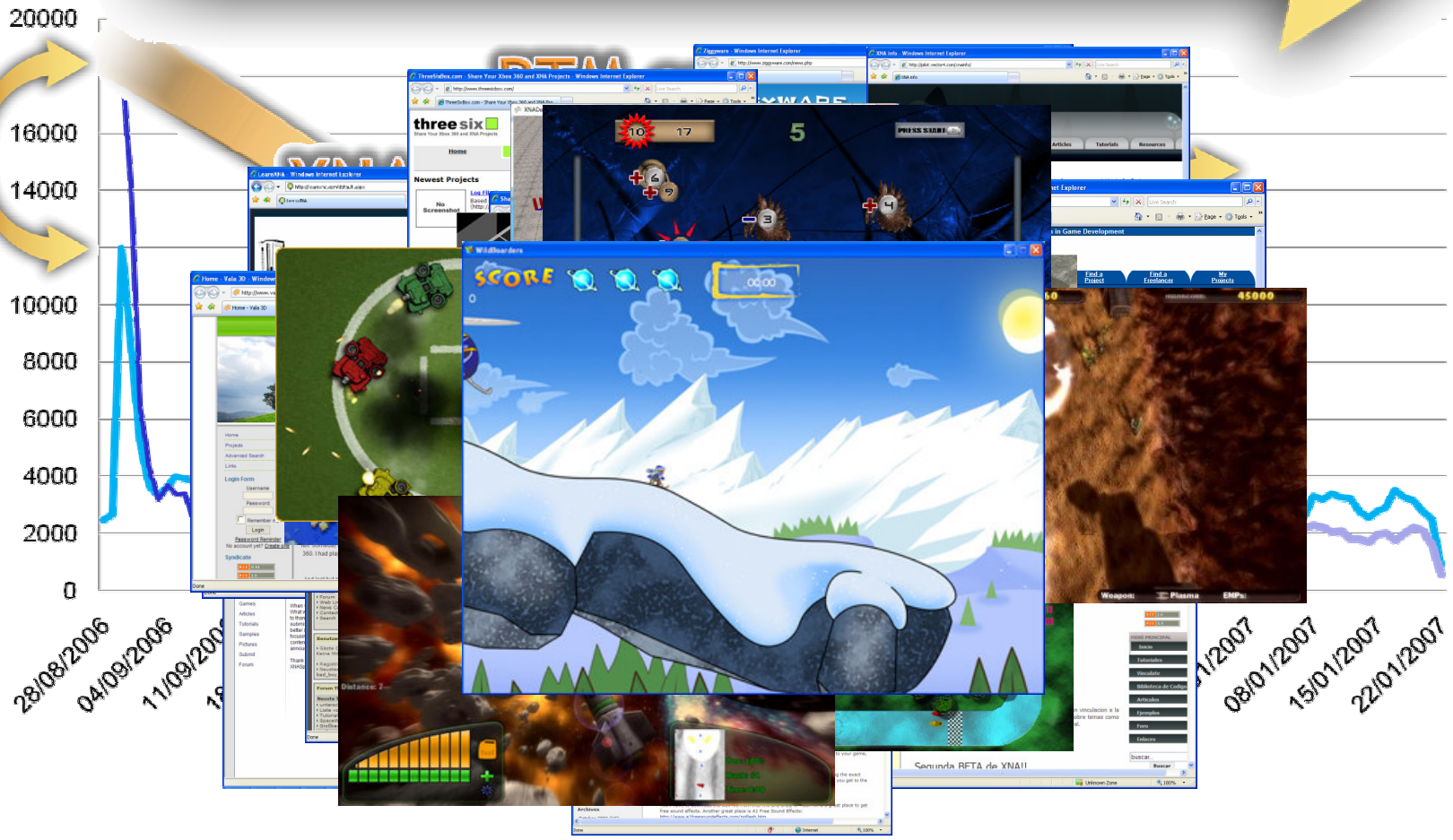
- **O que significa?**
  - **XNA = Xna Não é um Acrônimo ☺**
- **O que é?**
  - **Nova iniciativa (plataforma + comunidade) da Microsoft para desenvolver jogos**
- **Para rodar onde?**
  - **Tanto em** Windows **como** Xbox 360 **(no futuro, também em dispositivos móveis!)**
- **Qual o custo disso?**
  - **Nenhum! (tudo através de C# e Game Studio Express)**

XNA


XNA Tools & Technology
XNA Tools
XNA Components
XNA Base Platform
XNA Services
XNA Education
XNA Production
XNA Community
XNA Partnerships
XNA Academic
XNA Solutions
XNA Ecosystem

Os resultados até então

Mais de 50 sites de comunidades ativas (8 idiomas)



XNA - Elementos

XNA Game Studio Express

Estende o C# Express com suporte ao XNA Framework, suporta conteúdo multimídia e integração com o Xbox 360

XNA Framework

Framework para desenvolvimento de jogos multi-plataforma

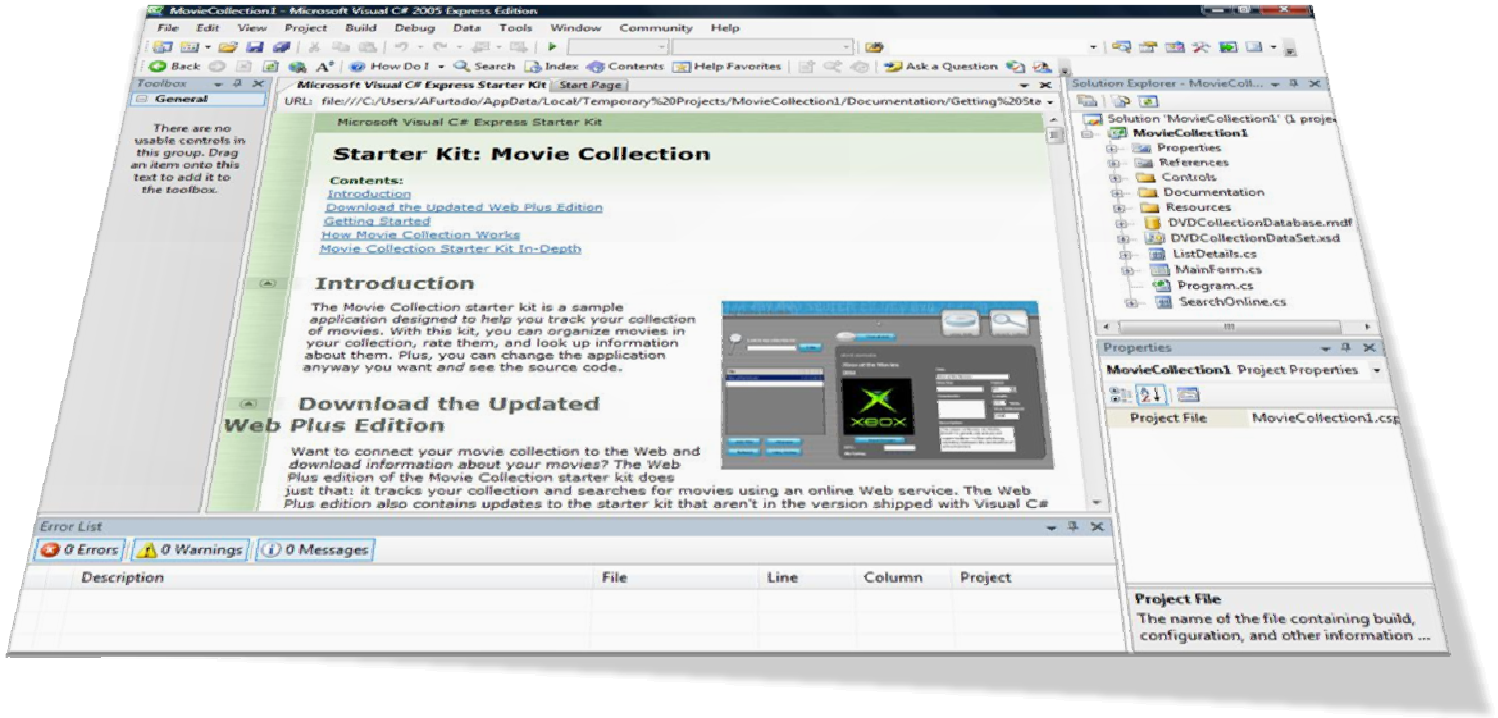
.NET Framework para o 360

Versão customizada do .NET Compact Framework

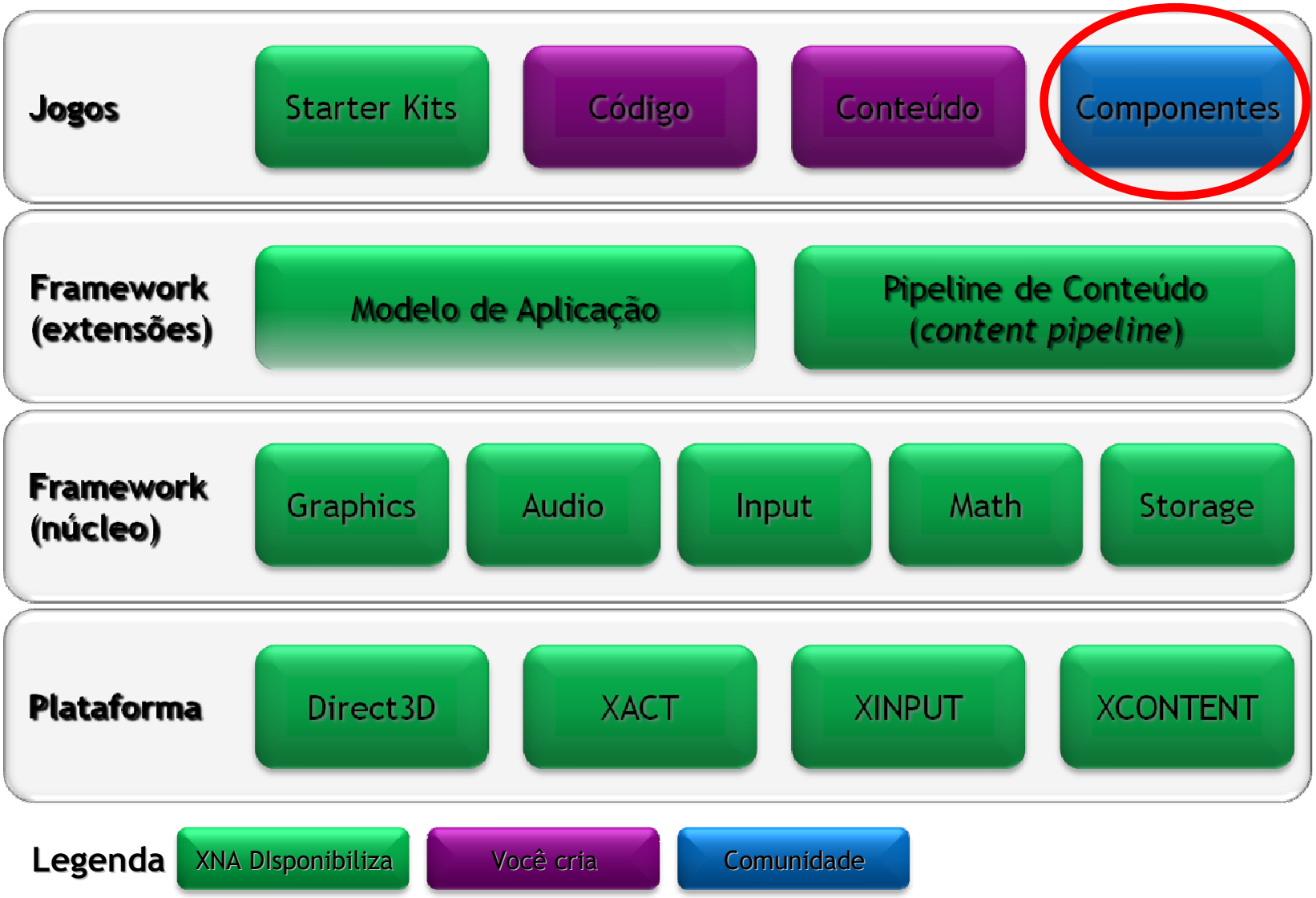
XNA Framework - Camadas

XNA Starter Kits

- Mini-jogos e aplicações prontos para uso/extensão
- Podem ser usados como destino final ou ponto de partida
- Novos kits sempre vão surgindo
- Cada kit possui um tópico em foco



XNA Framework - Componentes

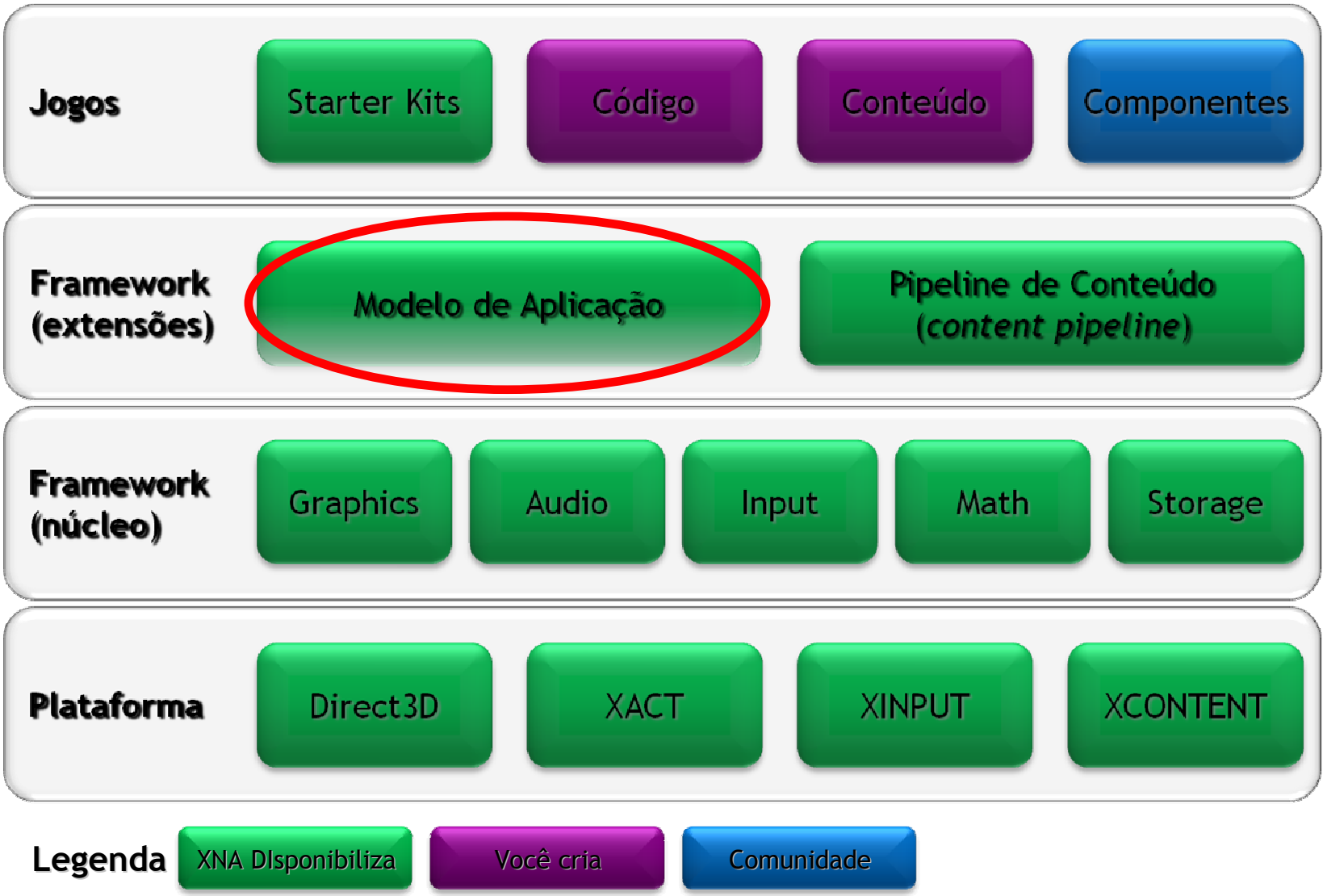


XNA Framework - Componentes

- Componentes reusáveis e plugáveis a qualquer jogo XNA
- Exemplos de componentes:
  - Câmeras
  - Terrenos
  - Elementos gráficos
  - Modificadores de comportamento
  - Utilitários (ex.: contadores de frames por segundo)

- Cenário:
  - Você adquiriu um contador de FPS (frames por segundo) de um terceiro.
  - Como reusá-lo e configurá-lo em seus jogos?

XNA Framework – Modelos de Aplicação



XNA Framework – Modelos de Aplicação

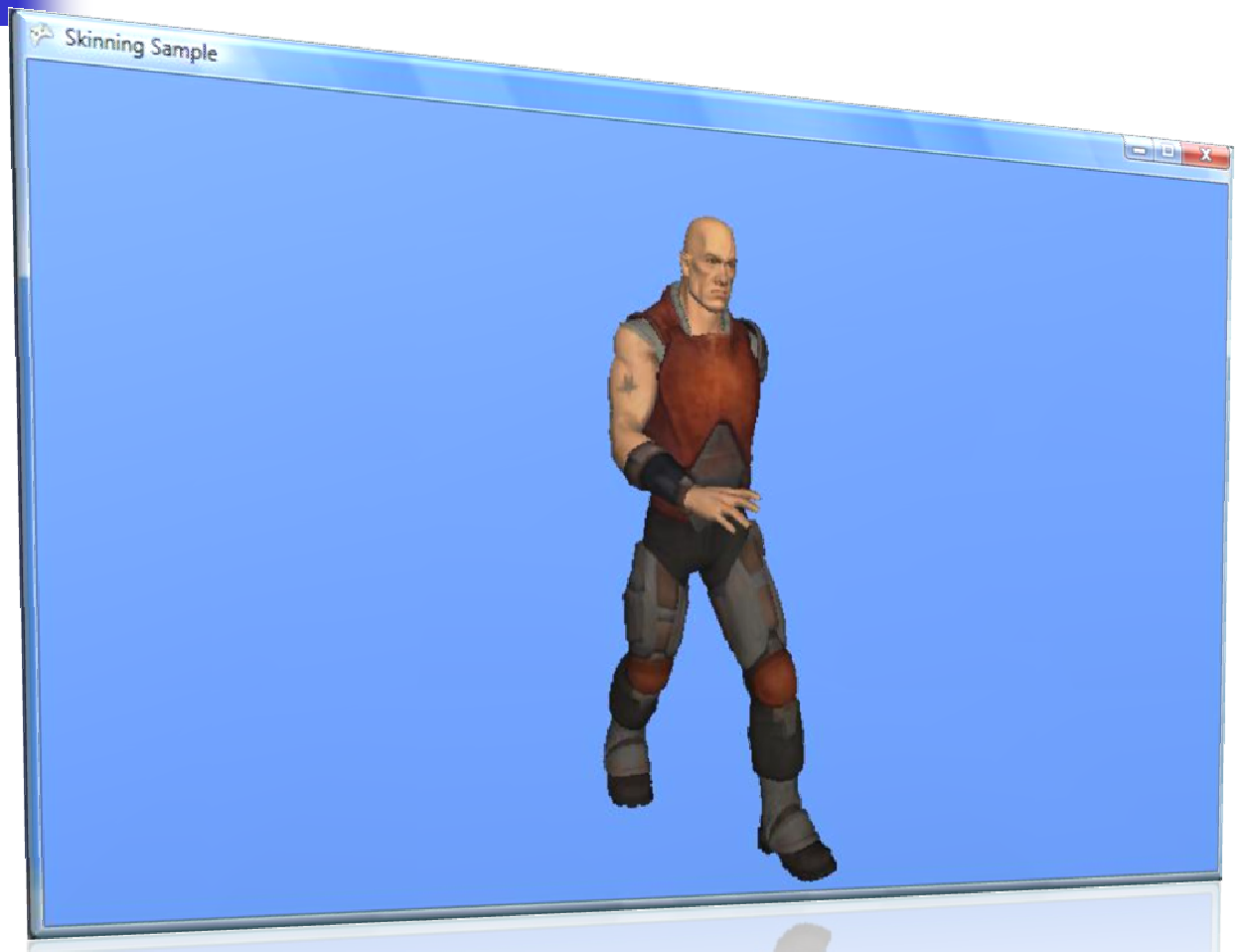
- Desenvolvimento acelerado
  - Primeiras linhas de código do jogo já estão lá
- Abstração de plataforma
  - Qual a melhor maneira de criar o loop do jogo?
  - Como criar e gerenciar a janela do jogo?
  - Quais são e como eu sigo as boas práticas da plataforma?
- Extensível
  - Provê componentes-base facilmente extensíveis.
  - Interfaces para “plugar” bibliotecas já existentes

XNA Framework – Graphics

XNA Framework – Graphics

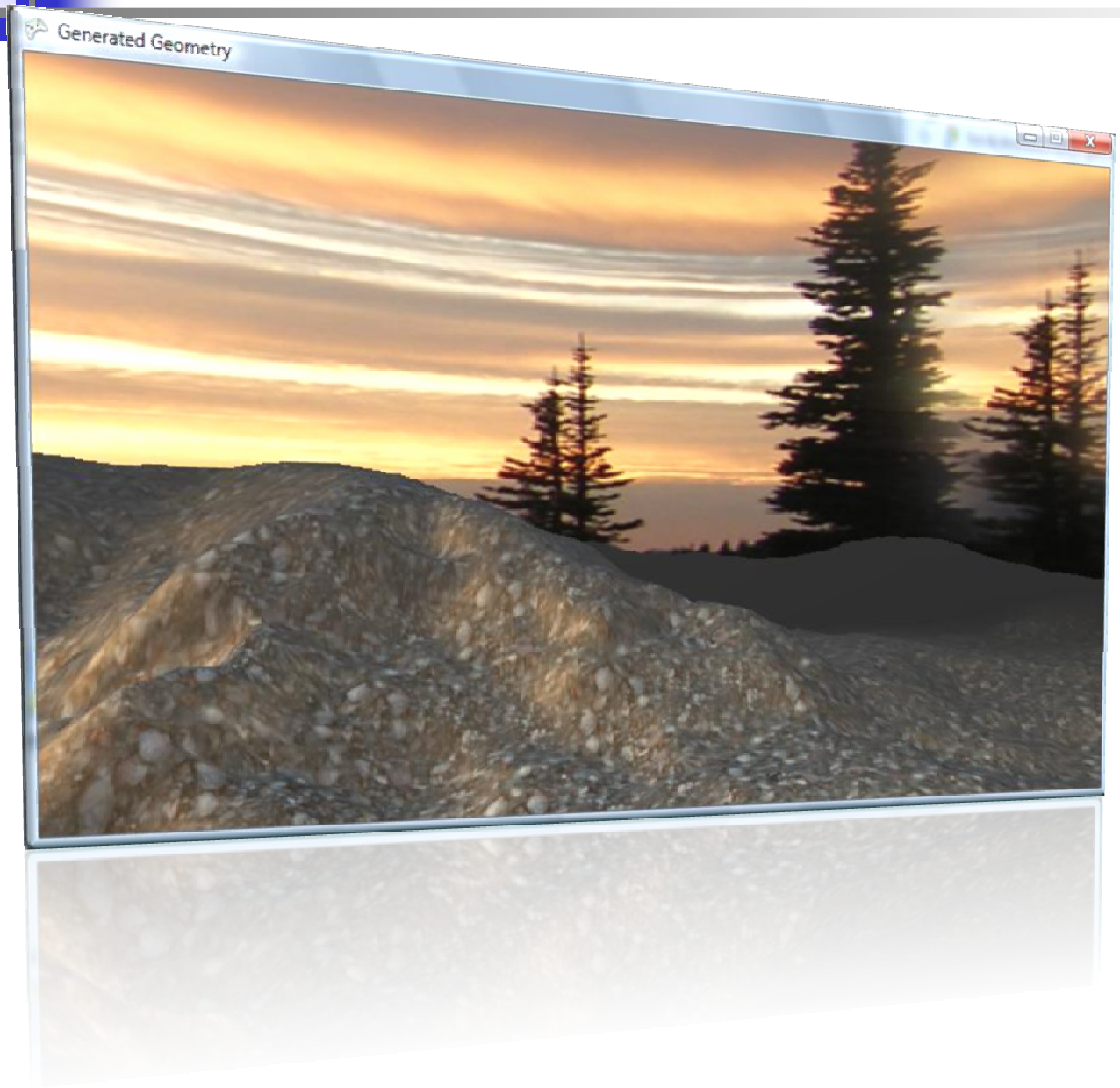
- Provê capacidade para renderização de baixo-nível
- Construído em cima do Direct3D 9
- Pipeline (“fluxo”) programável para conteúdo multimídia
  - BasicEffect acelera o desenvolvimento
  - SpriteBatch para 2D e partículas
- Recursos disponibilizados
  - Model (modelos), Texture (texturas), Effects (efeitos), e Shaders

Graphics: Skining & Animation

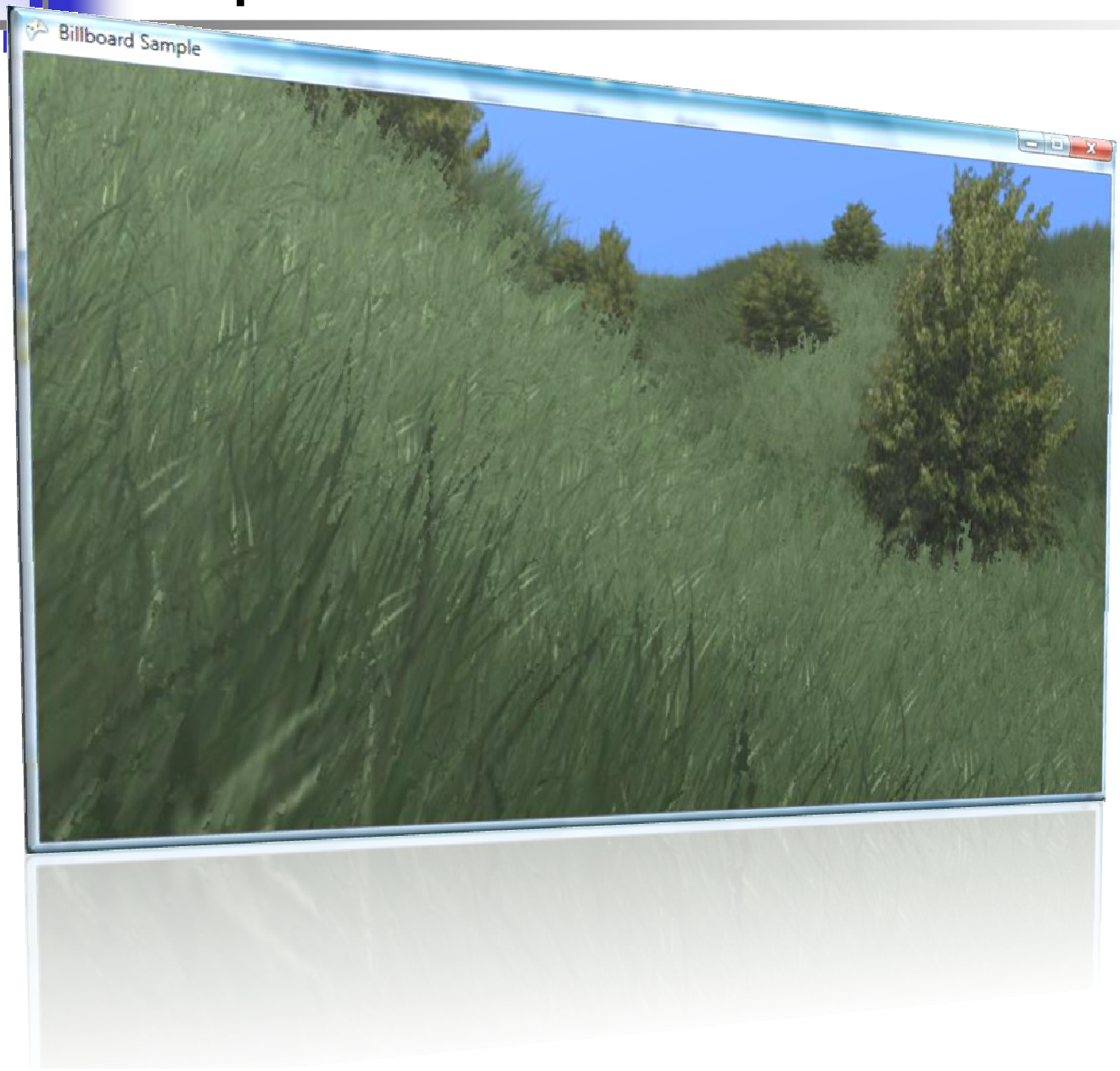


Graphics:: Sprite Effect

Graphics:: Generated Geometry

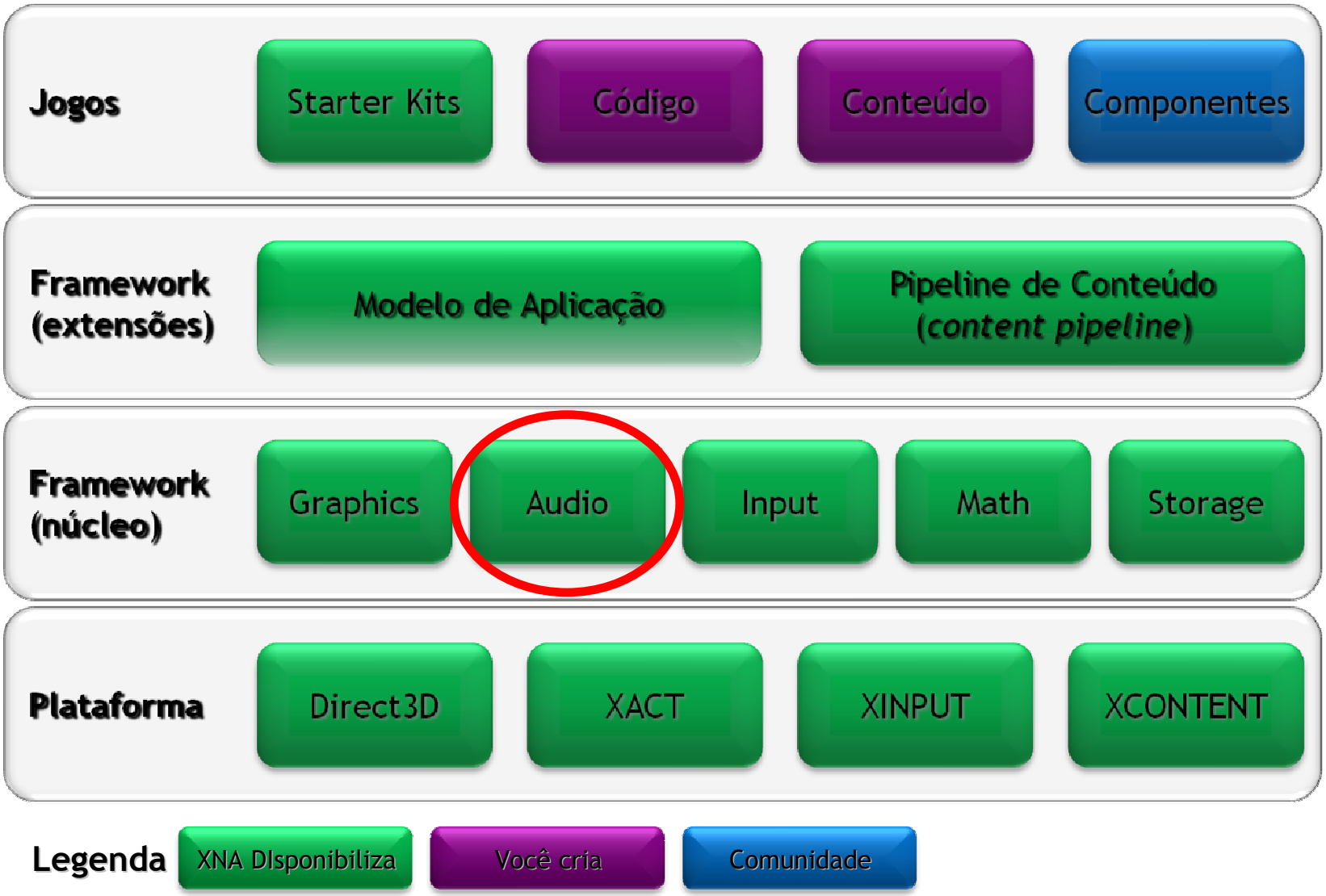


Graphics:: Billboard



Graphics:: BoxCollider

XNA Áudio

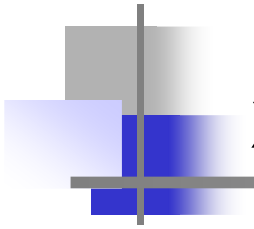

Jogos
Starter Kits
Código
Conteúdo
Componentes
Framework (extensões)
Modelo de Aplicação
Pipeline de Conteúdo (content pipeline)
Framework (núcleo)
Graphics
Audio
Input
Math
Storage
Plataforma
Direct3D
XACT
XINPUT
XCONTENT
Legenda
XNA DIsponibiliza
Você cria
Comunidade

XNA Áudio

- **Baseado no XACT (Cross-Platform Audio Creation Tool)**
  - **Permite que projetistas e programadores de som trabalhem mais naturalmente**
- **Construção de recursos de som através do XACT Tool**
- **Integração fácil para desenvolvedores**
  - **Acesso a sons através de seu nome lógico**
  - **Looping, streaming, e gerenciamento de memória**
  - **Não há gerenciamento de buffer em baixo nível**

XNA Input


Jogos
Starter Kits
Código
Conteúdo
Componentes
Framework (extensões)
Modelo de Aplicação
Pipeline de Conteúdo (content pipeline)
Framework (núcleo)
Graphics
Audio
Input
Math
Storage
Plataforma
Direct3D
XACT
XINPUT
XCONTENT
Legenda
XNA DIsponibiliza
Você cria
Comunidade



XNA Input

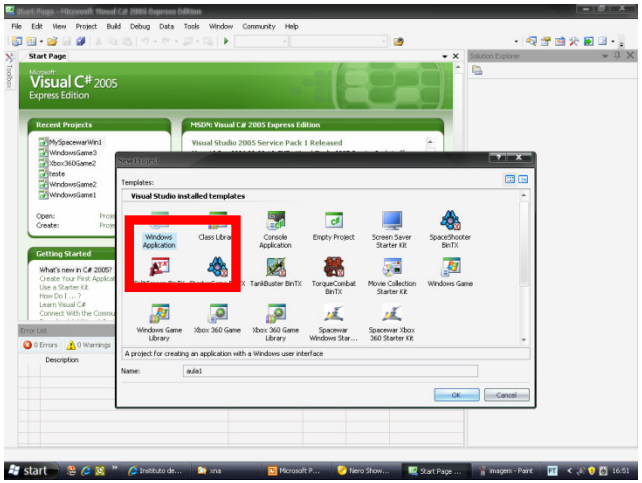
- Torna a obtenção do input do usuário extremamente fácil
- Modelo de programação imediato
  - Nenhuma inicialização
  - Nenhum gerenciamento de estado
- Suporte a
  - Xbox 360 controller
  - Keyboard
  - Mouse (apenas para Windows)

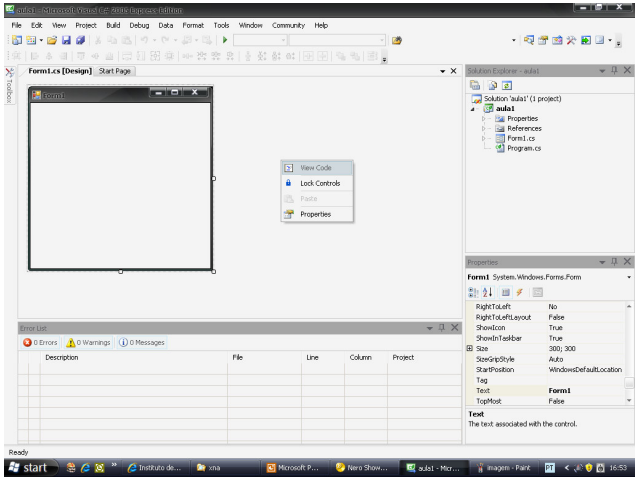
XNA Math e Storage

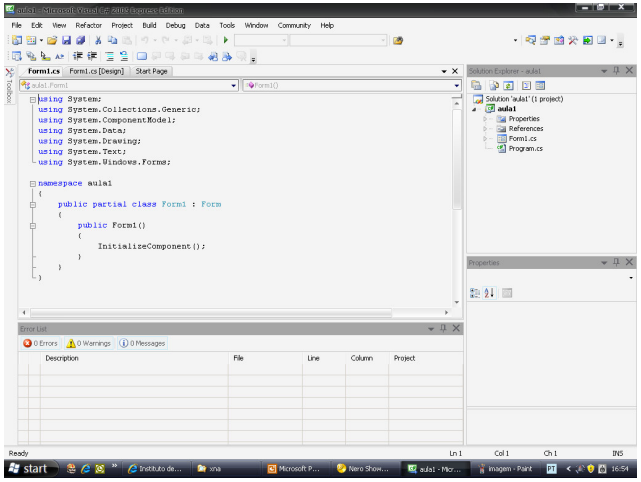
XNA Math e Storage

- **Math**
  - **Vector, Matrix, Quaternion, Plane, AABB, Shpere, Ray, Frustum, Curve**
  - **Lado direito (destro) por padrão**
  - **"*Helpers*" para interseção e movimentação**
- **Storage (armazenamento)**
  - **Provê uma maneira fácil de ler/escrever dados e salvamentos de jogos**
  - **Arquivos armazenados no local correto em cada plataforma**

XNA – C#

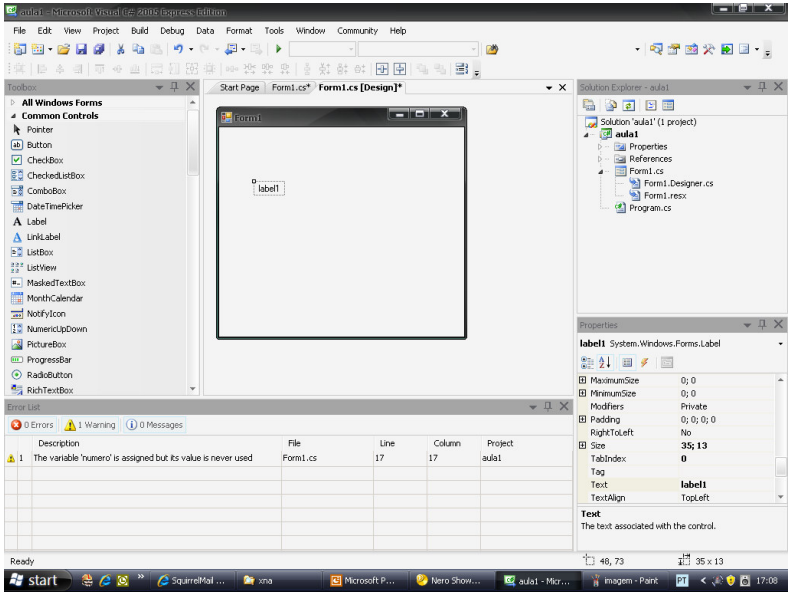

Visual C# 2005
Express Edition
Visual Studio installed templates
Windows Application
Console Application
Empty Project
Windows Game
A project for creating an application with a Windows user interface
Name:
aula1
OK
Cancel


Form1.cs [Design]
Start Page
View Code
Lock Controls
Properties
Solution Explorer - aula1
Solution 'aula1' (1 project)
References
Program.cs
Error List
Description
File
Line
Column
Project
Ready


Form1.cs
Form1.cs [Design]
Start Page
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace aula1
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}
}
}
Solution Explorer - aula1
Error List
Description
File
Line
Column
Project
Ready

C#

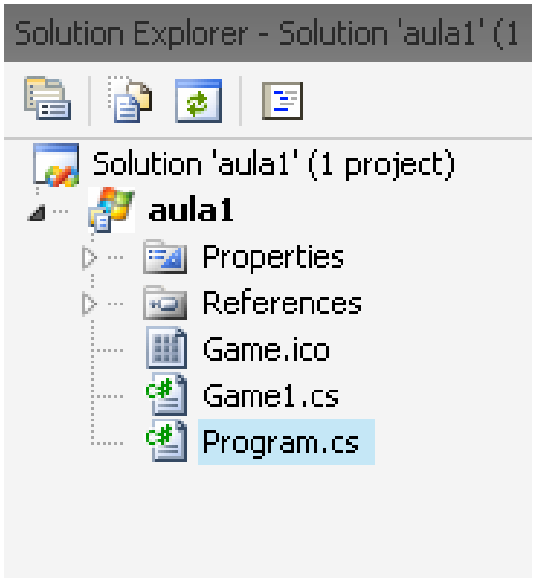
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace aula1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            string mensagem = "Primeiro programa em c#";
            label1.Text = mensagem;
            label1.Visible = true;

        }
    }
}
```



Arquitetura XNA

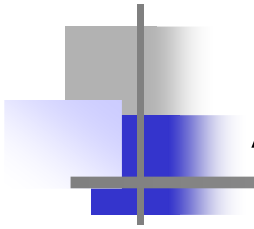
Projeto:

-Program.cs
-Game1.cs

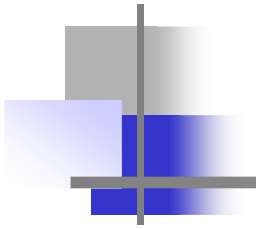


Arquitetura XNA

```
namespace Jogo
{
   public class Game1 : Microsoft.Xna.Framework.Game
    {

       public Game1()
       protected override void Initialize()
       protected override void LoadGraphicsContent (bool loadAllContent)
       protected override void UnloadGraphicsContent (bool unloadAllContent)
       protected override void Update(GameTime gameTime)
       protected override void Draw(GameTime gameTime)
    }
}
```



Alguns Exemplos



PONG

Elementos.cs

```
using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Input;
```

PONG

Elementos.cs

```
class c_jogador
 {
    public Point posicao;
    public int velocidade;

    public c_jogador(int x, int y)
    {
       posicao = new Point(x, y);
       velocidade = 3;
    }

   public int EntradaTeclado()
   {
      KeyboardState currentState;
      currentState = Keyboard.GetState();
      Keys[] currentKeys = currentState.GetPressedKeys();
      int erro = 1;
      foreach (Keys key in currentKeys)
      {
         if (key == Keys.Left)
            posicao.X -= velocidade;
         if (key == Keys.Right)
            posicao.X += velocidade;
         if (key == Keys.Escape)
            erro = -1;
      }
      return erro;
   }
}
```

PONG

Elementos.cs

```
class c_bolinha
 {
    public Point posicao;
    public int velocidadeH;
    public int velocidadeV;

    public c_bolinha(int x, int y)
    {
       posicao = new Point(x, y);
       velocidadeH = 3;
       velocidadeV = 3;
    }

    public void mover_bola(int Width, int Height)
    {
       posicao.X += velocidadeH;
       posicao.Y += velocidadeV;
       if ((posicao.X < 0) || (posicao.X > 800 - Width))  velocidadeH *= -1;
       if (posicao.Y < 0)   velocidadeV *= -1;
       if (posicao.Y > (600 - Height))
       {
          posicao.X = 200;
          posicao.Y = 50;
          velocidadeH = 3;
          velocidadeV = 3;
       }
    }
}
```

PONG

Elementos.cs

```
class c_bolinha
 {
    public Point posicao;
    public int velocidadeH;
    public int velocidadeV;

    public c_bolinha(int x, int y)
    {
       posicao = new Point(x, y);
       velocidadeH = 3;
       velocidadeV = 3;
    }

    public void mover_bola(int Width, int Height)
    {
       posicao.X += velocidadeH;
       posicao.Y += velocidadeV;
       if ((posicao.X < 0) || (posicao.X > 800 - Width))  velocidadeH *= -1;
       if (posicao.Y < 0)   velocidadeV *= -1;
       if (posicao.Y > (600 - Height))
       {
          posicao.X = 200;
          posicao.Y = 50;
          velocidadeH = 3;
          velocidadeV = 3;
       }
    }
}
```

PONG

Game1.cs

```
public class Game1 : Microsoft.Xna.Framework.Game
  {
     GraphicsDeviceManager graphics;
     ContentManager content;



     Texture2D jogador, bola;
     SpriteBatch sprite;

     c_jogador obj_jogador;
     c_bolinha obj_bolinha;

     int acertos = 0;
```

PONG

Game1.cs

```
void iniciar_classes()
      {
         if (obj_jogador == null)
         {
            obj_jogador = new c_jogador(400, 550);
            obj_bolinha = new c_bolinha(200, 50);
         }
         else
         {
            obj_jogador.posicao = new Point(400, 550);
            obj_bolinha.posicao = new Point(200, 50);
         }
         acertos = 0;
         obj_bolinha.velocidadeH = 3;
         obj_bolinha.velocidadeV = 3;
      }
```

PONG

Game1.cs

```
protected override void Initialize()
      {

          base.Initialize();

          jogador = content.Load<Texture2D>("jogador_icon");
          bola = content.Load<Texture2D>("bola_icon");
          sprite = new SpriteBatch(graphics.GraphicsDevice);

          iniciar_classes();
      }
```

PONG

Game1.cs

```
protected override void Update(GameTime gameTime)
      {
         if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
            this.Exit();

         if (obj_jogador.EntradaTeclado() == -1) this.Exit();
         if (obj_jogador.posicao.X < 5)
            obj_jogador.posicao.X = 5;
         if (obj_jogador.posicao.X > 700)
            obj_jogador.posicao.X = 700;

         obj_bolinha.mover_bola(bola.Width, bola.Height);

         AcertouBola();

         base.Update(gameTime);
      }
```

PONG

Game1.cs

```
void AcertouBola()
      {
         if (obj_bolinha.posicao.Y >= (obj_jogador.posicao.Y - bola.Height))

            if ((obj_bolinha.posicao.X >= (obj_jogador.posicao.X - bola.Width)) && (obj_bolinha.posicao.X <=
obj_jogador.posicao.X + jogador.Width))
            {
               obj_bolinha.velocidadeV *= -1;
               acertos++;

               if (acertos > 6)
               {
                  if (obj_bolinha.velocidadeV > 0)
                     obj_bolinha.velocidadeV++;
                  else
                     obj_bolinha.velocidadeV--;

                  if (obj_bolinha.velocidadeH > 0)
                     obj_bolinha.velocidadeH++;
                  else
                     obj_bolinha.velocidadeH--;
               }
            }
      }
```

PONG

Game1.cs

```
protected override void Draw(GameTime gameTime)
        {
            graphics.GraphicsDevice.Clear(Color.Black);

            sprite.Begin(SpriteBlendMode.AlphaBlend);
            sprite.Draw(jogador, new Rectangle(obj_jogador.posicao.X, obj_jogador.posicao.Y, jogador.Width,
jogador.Height), Color.White);
            sprite.Draw(bola, new Rectangle(obj_bolinha.posicao.X, obj_bolinha.posicao.Y, bola.Width, bola.Height),
Color.White);

            sprite.End();

            base.Draw(gameTime);
        }
```

Game1.cs

```
// Posicao e Rotacao do modelo no mundo
        Vector3 modelposition = Vector3.Zero;
        float modelrotation = 0.0f;

// Posicao da Camera no mundo
        Vector3 camposition = new Vector3(0.0f, 50.0f, 5000.0f);

//Aspect ratio da projection matrix...
        float aspectRatio = 640.0f / 480.0f;

// Modelo 3D
        Model myModel;
```

Game1.cs

```
protected override void LoadGraphicsContent(bool loadAllContent)
    {
        if (loadAllContent)
        {
            myModel = content.Load<Model>("Content\\Modelos\\p1_wedge");
        }

    }
```

Game1.cs

```
protected override void Draw(GameTime gameTime)
        {
            graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

            // Modelos podem ser compostos por diversas malhas...
            foreach (ModelMesh mesh in myModel.Meshes)
            {
                // Cada malha possui efeitos (materiais...)
                foreach (BasicEffect effect in mesh.Effects)
                {
                    effect.EnableDefaultLighting();
                    effect.World = Matrix.CreateRotationY(modelrotation) * mesh.ParentBone.Transform
                        * Matrix.CreateTranslation(modelposition);
                    effect.View = Matrix.CreateLookAt(camposition, Vector3.Zero, Vector3.Up);
                    effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.ToRadians(45.0f),
                        aspectRatio, 1.0f, 10000.0f);
                }
                // Desenha a malha
                mesh.Draw();
            }

            base.Draw(gameTime);
        }
```

Game1.cs

```
protected override void Update(GameTime gameTime)
      {
          if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
              this.Exit();

          modelrotation += gameTime.ElapsedGameTime.Milliseconds * MathHelper.ToRadians(0.1f);

          base.Update(gameTime);
      }
```

Scrolling de Background

Game1.cs

```
Texture2D[] Texturas_fundo = new Texture2D[5];

int fundo1 = 0;
int fundo2 = 0;
int fundo3 = 0;
int fundo4 = 0;
int fundo5 = 0;

SpriteBatch sprite;
```

Scrolling de Background

Game1.cs

```
protected void update_background()
{
        fundo1 -= 3;
        if (fundo1 <= -this.Window.ClientBounds.Width)
            fundo1 = this.Window.ClientBounds.Width;

        fundo2 -= 3;
        if (fundo2 <= -this.Window.ClientBounds.Width)
            fundo2 = this.Window.ClientBounds.Width;

        fundo3 -= 3;
        if (fundo3 <= -this.Window.ClientBounds.Width)
            fundo3 = this.Window.ClientBounds.Width;

        fundo4 -= 3;
        if (fundo4 <= -this.Window.ClientBounds.Width)
            fundo4 = this.Window.ClientBounds.Width;

        fundo5 -= 3;
        if (fundo5 <= -this.Window.ClientBounds.Width)
            fundo5 = this.Window.ClientBounds.Width;
}
```

Scrolling de Background

Game1.cs

```
protected override void LoadGraphicsContent(bool loadAllContent)
    {
        if (loadAllContent)
        {

        //Carregar fundo na matriz de imagens
        Texturas_fundo[0] = content.Load<Texture2D>("Background1") as Texture2D;
        Texturas_fundo[1] = content.Load<Texture2D>("Background2") as Texture2D;
        Texturas_fundo[2] = content.Load<Texture2D>("Background3") as Texture2D;
        Texturas_fundo[3] = content.Load<Texture2D>("Background4") as Texture2D;
        Texturas_fundo[4] = content.Load<Texture2D>("Background5") as Texture2D;

        //posicao X de cada imagem de fundo
        fundo1 = 0;
        fundo2 = this.Window.ClientBounds.Width;
        fundo3 = this.Window.ClientBounds.Width * 2;
        fundo4 = this.Window.ClientBounds.Width * 3;
        fundo5 = this.Window.ClientBounds.Width * 4;

        sprite = new SpriteBatch(graphics.GraphicsDevice);
        }
    }
```

Scrolling de Background

Game1.cs

```
protected override void Update(GameTime gameTime)
      {

          if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
              this.Exit();

          update_background();

          base.Update(gameTime);
      }
```

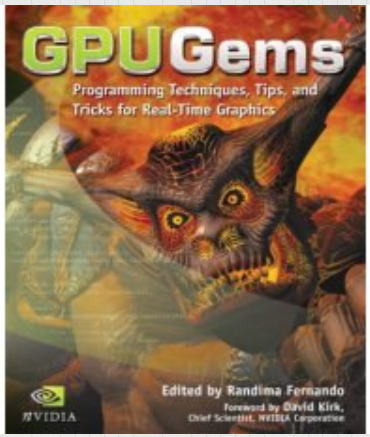
Scrolling de Background

Game1.cs

```
protected override void Draw(GameTime gameTime)
        {
            graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

            sprite.Begin(SpriteBlendMode.AlphaBlend);

            sprite.Draw(Texturas_fundo[0], new Rectangle(fundo1, 0,
this.Window.ClientBounds.Width, this.Window.ClientBounds.Height), Color.White);
            sprite.Draw(Texturas_fundo[1], new Rectangle(fundo2, 0,
this.Window.ClientBounds.Width, this.Window.ClientBounds.Height), Color.White);
            sprite.Draw(Texturas_fundo[2], new Rectangle(fundo3, 0,
this.Window.ClientBounds.Width, this.Window.ClientBounds.Height), Color.White);
            sprite.Draw(Texturas_fundo[3], new Rectangle(fundo4, 0,
this.Window.ClientBounds.Width, this.Window.ClientBounds.Height), Color.White);
            sprite.Draw(Texturas_fundo[4], new Rectangle(fundo5, 0,
this.Window.ClientBounds.Width, this.Window.ClientBounds.Height), Color.White);

            sprite.End();

            base.Draw(gameTime);
        }
```

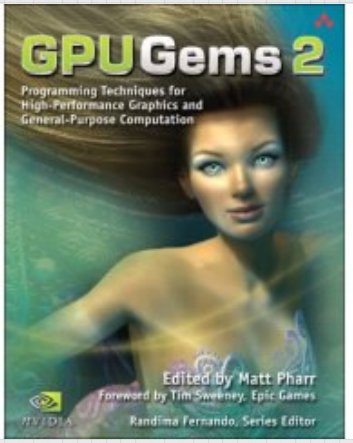
Parte 2 - Arquitetura de GPU´s

referência

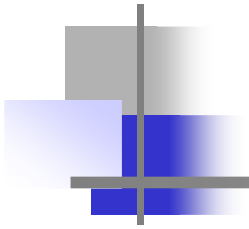


GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics
Randima Fernando

referência

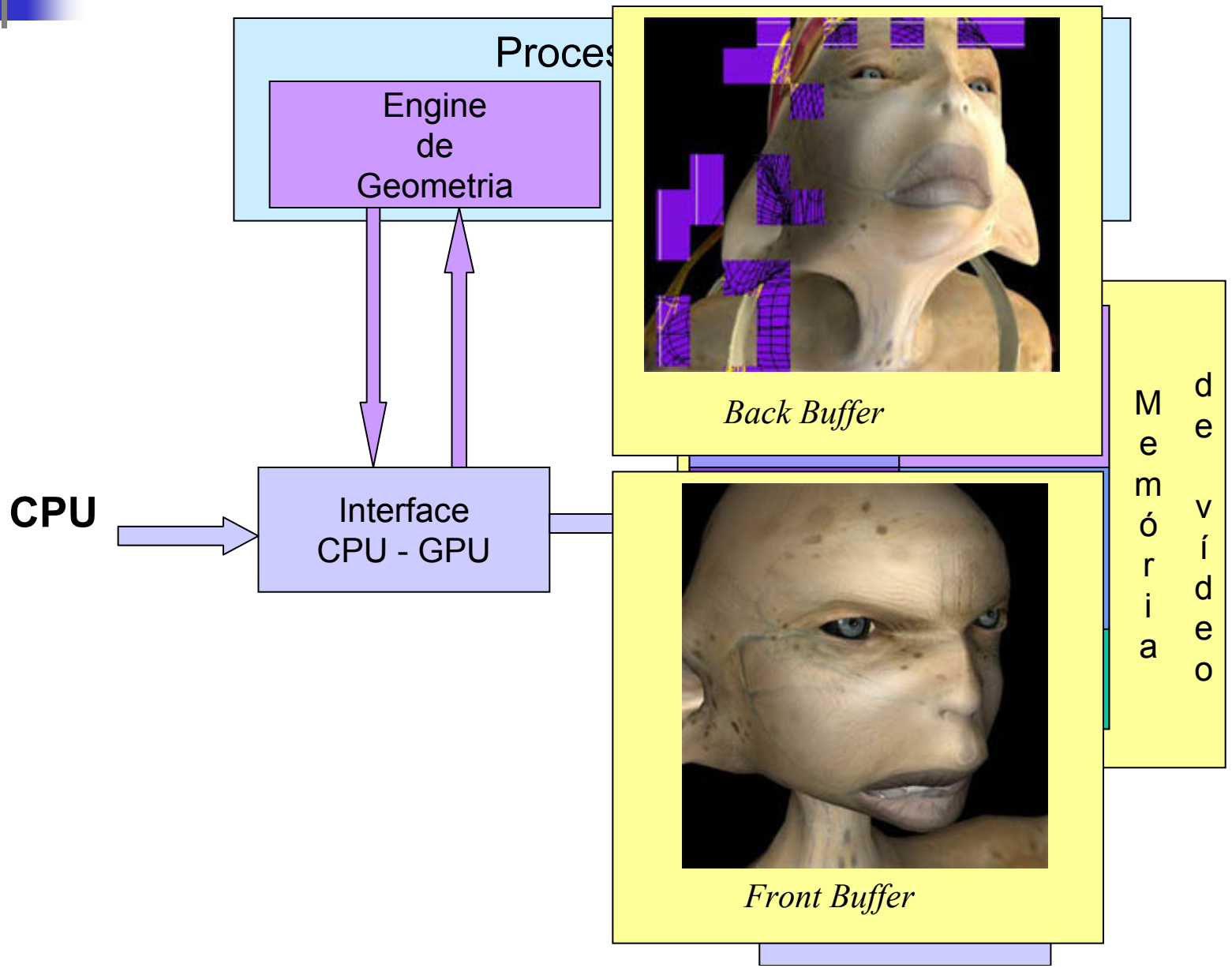


GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation
Matt Pharr, Randima Fernando

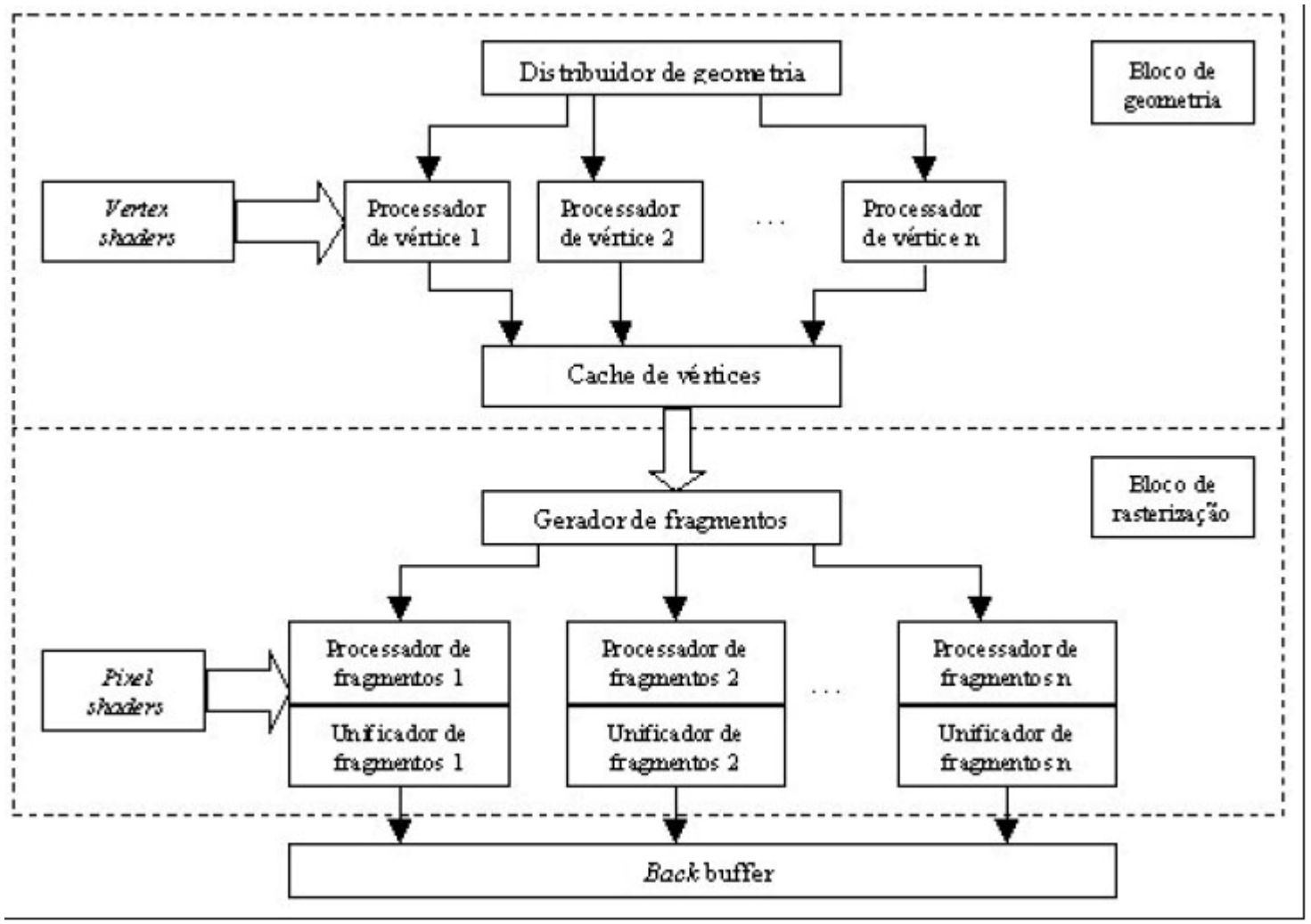


a. Arquitetura de Hardware

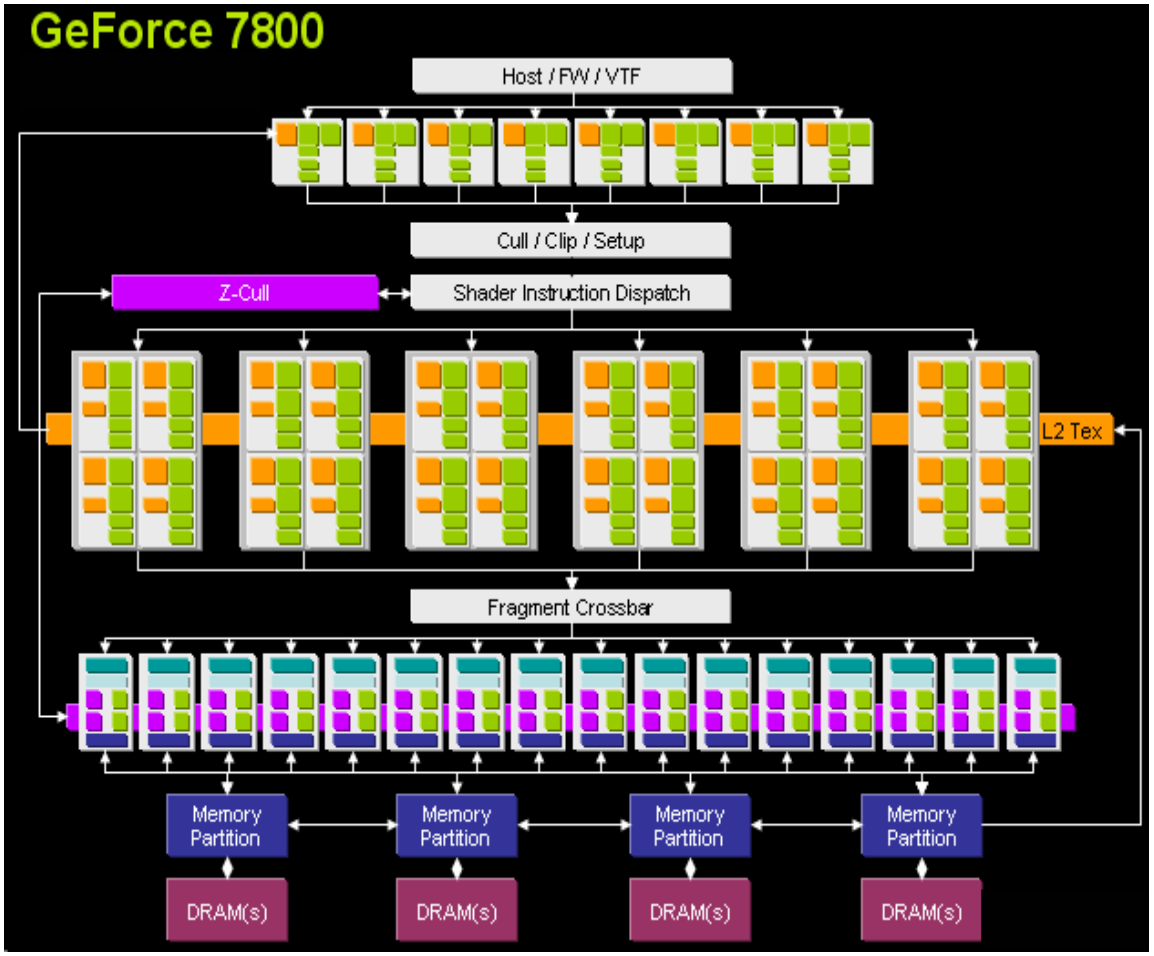
Arquitetura de GPUs

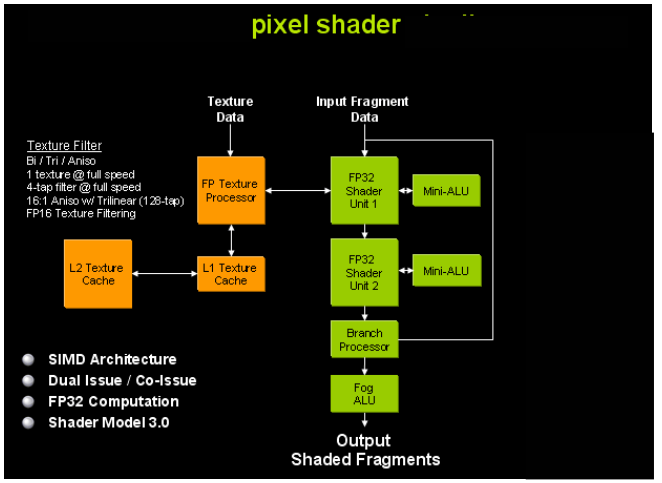


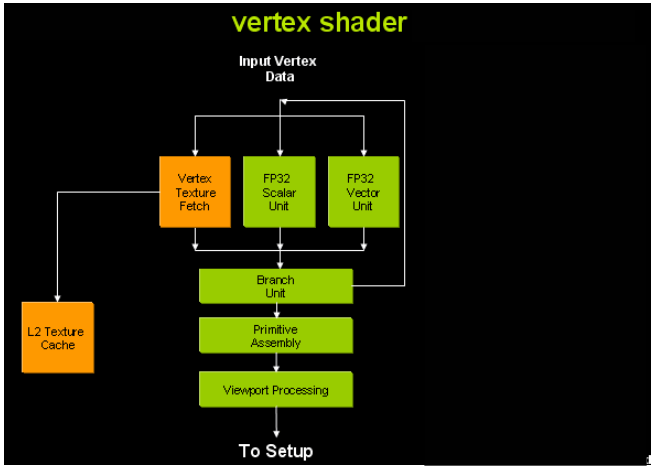
Arquitetura de GPUs

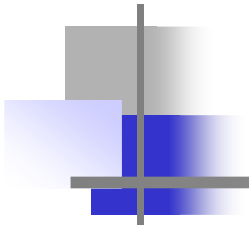


Arquitetura de GPUs





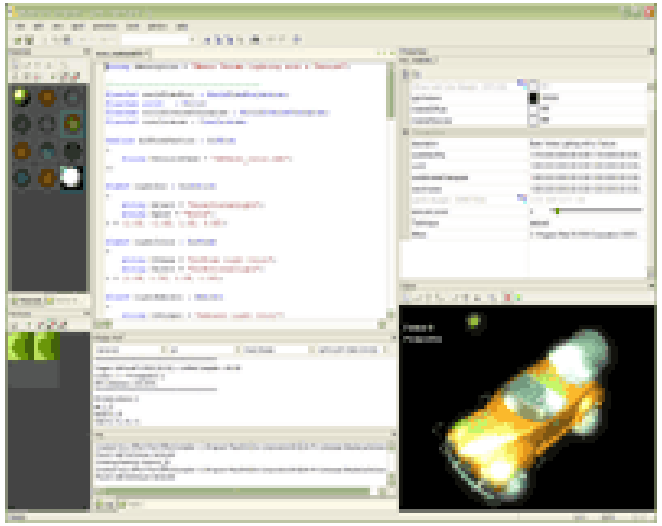




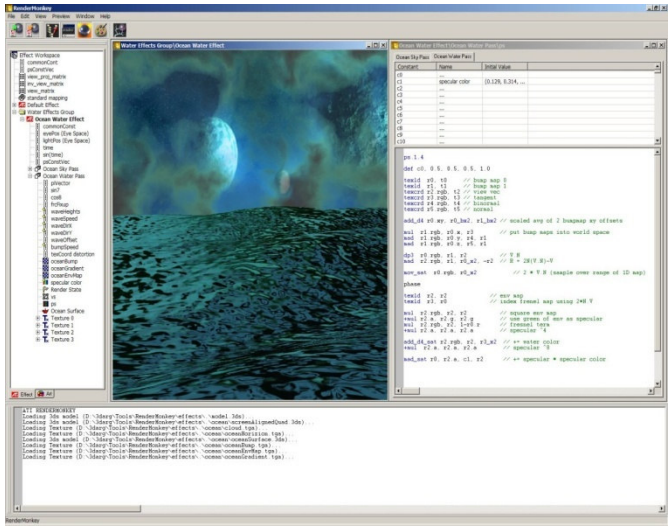
b. Programação de GPUs

Programação de GPUs

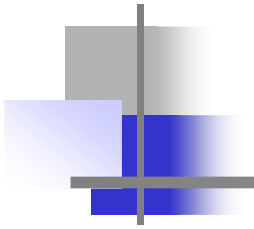
- Linguagens: Cg, HLSL, OpenGL Shader Language
- Ambientes de Desenvolvimento



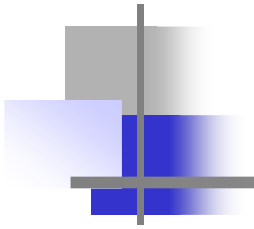
NVídia FX Composer



ATI Rendermonkey



c. Vertex Programming



c. Vertex Programming

Operações nesta etapa:

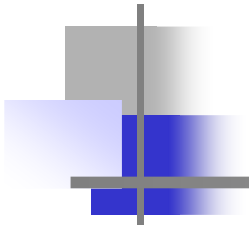
- Transformação da posição do vértice
- Geração de coordenadas de textura para a posição do vértice
- Iluminação sobre o vértice
- Operações para determinar o material a ser aplicado ao vértice

Efeitos:

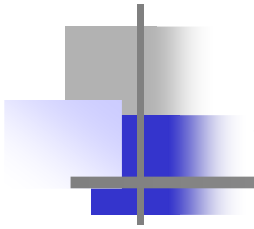
- Geração de texturas procedurais
- Efeitos de iluminação per-vertex
- Animação procedural em vértices
- Displacement mapping

c. Vertex Programming

```
uniform vec3 LightPosition;
const float CEspecular = 0.25;
const float CDifusa = 1.0 - CEspecular;
varying float IntensidadeLuz;
varying vec2 PosicaoMC;

void main(void)
{
        vec3 PosicaoCC = vec3(gl_ModelViewMatrix * gl_Vertex);
        vec3 normT = normalize(gl_NormalMatrix * gl_Normal);

        vec3 vetorLuz = normalize(LightPosition - PosicaoCC);
        vec3 VetorReflexo = reflect(-VetorLuz, normT);

        vec3 vetorCamera = normalize(-PosicaoCC);
        float difuso = max(dot(VetorLuz, normT), 0.0);
        float coef = 20;
        float especular = max(dot(VetorReflexo, VetorCamera), 0.0);
        especular = pow(especular, coef);
        IntensidadeLuz = CDifusa * difuso + CEspecular * especular;
        PosicaoMC = gl_Vertex;
        gl_Position = ftransform();
}
```



d. Pixel Programming



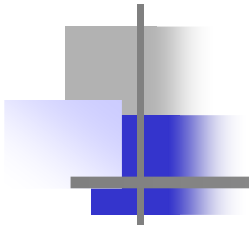
d. Pixel Programming

Operações nesta etapa:

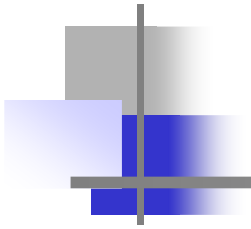
- Computar a cor de um fragmento
- Alterar iluminação “per-pixel”
- gl_FragCoord

d. Pixel Programming

```
// Apenas uma variável será definida, contendo a cor do material a ser aplicado.

uniform vec3 material = (0.0, 0.0, 1.0);

// As duas variáveis que provêm do vertex shader são a posição e a cor resultante da
// iluminação, ambas interpoladas.
varying vec2 PosicaoMC;
varying float IntensidadeLuz;

// Início do pixel shader.
void main(void)
{
        // A variável cor acumulará o resultado da iluminação com o material.
        vec3 cor = material * IntensidadeLuz;

        // A variável pré-definida gl_FragColor deve receber no final a cor com que o
        // fragmento será pintado.
        gl_FragColor = vec4(cor, 1.0);
}
```



e. General Pourpose GPUs



Requisitos dos Motores

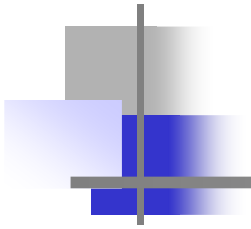
Encapsulamento

Integração

Independência de Plataforma

Otimização em Hardware

Gerenciamento de Projeto



Arquitetura

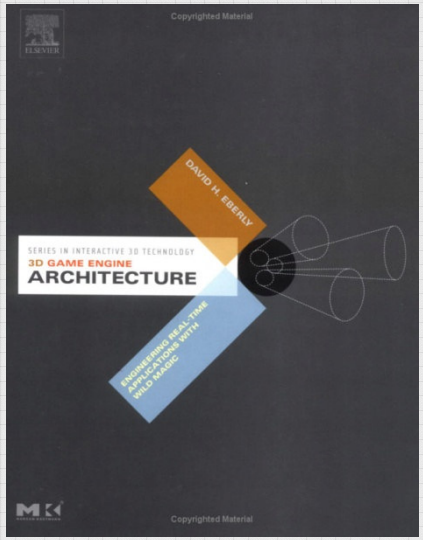
Níveis de Abstração:

SDK

Ferramental

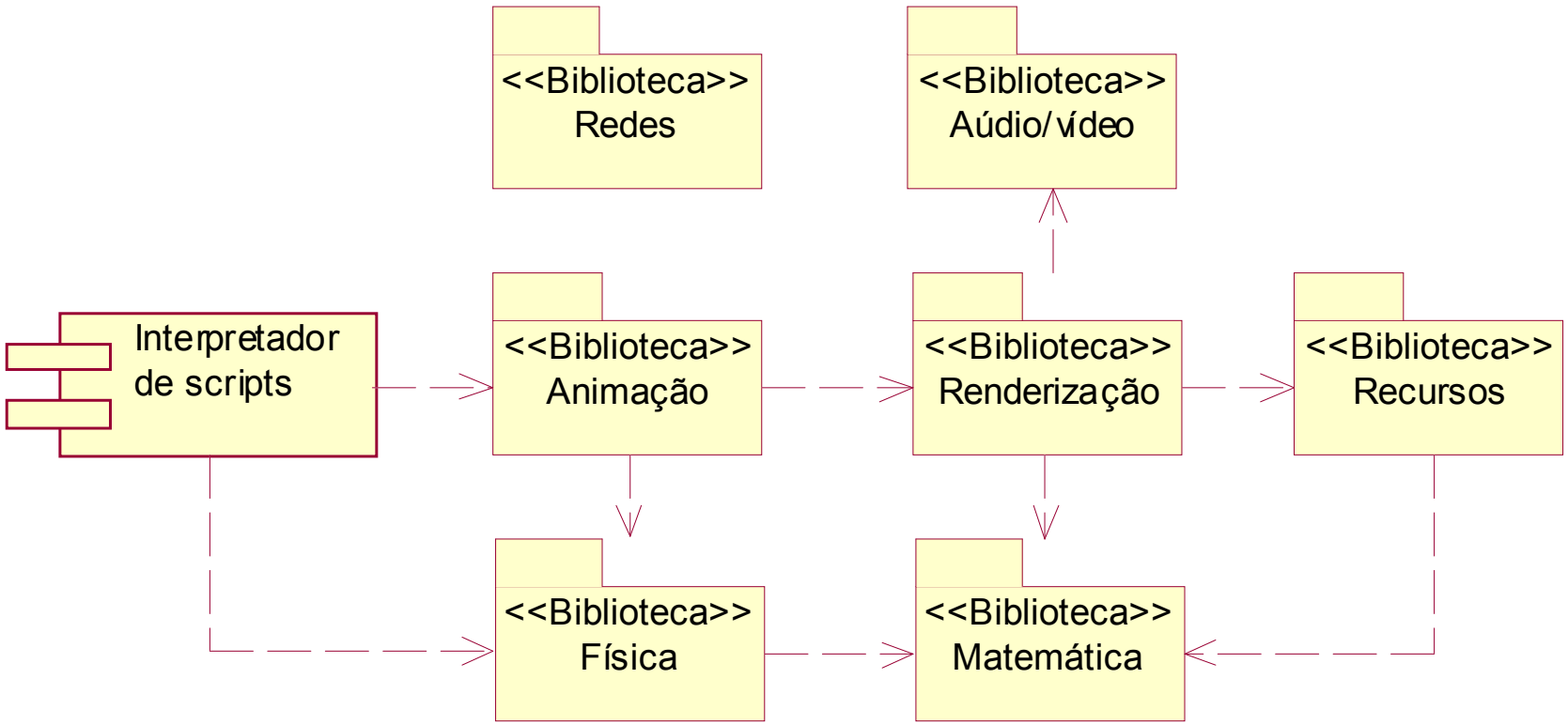
Arquitetura SDK

referência



3D Game Engine Architecture : Engineering Real-Time Applications with Wild Magic
David H. Eberly

Arquitetura SDK



Biblioteca Matemática

Operações de Vetores

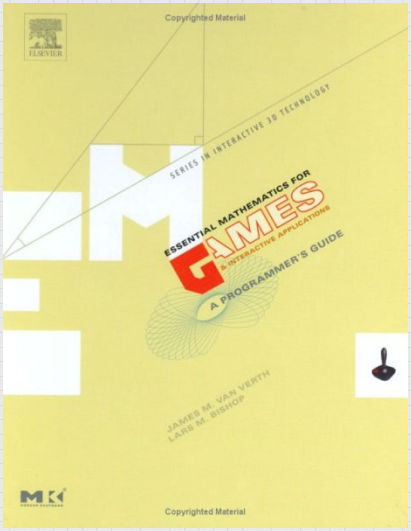
Operações de Matrizes

Operações de Quaternions

Operações de Interceção

Biblioteca baseada em GPUs

referência



Essential Mathematics for Games and Interactive Applications : A Programmer's Guide
James M. Van Verth, Lars M. Bishop

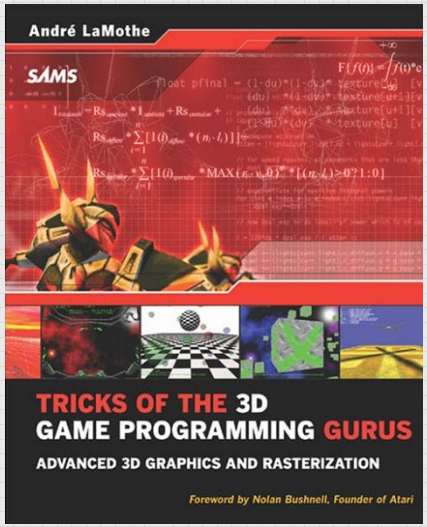
Biblioteca de Rendering

Abstração de APIs

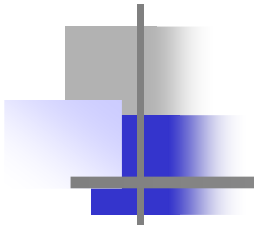
Implementação do Pipeline

Leitor de shaders

referência



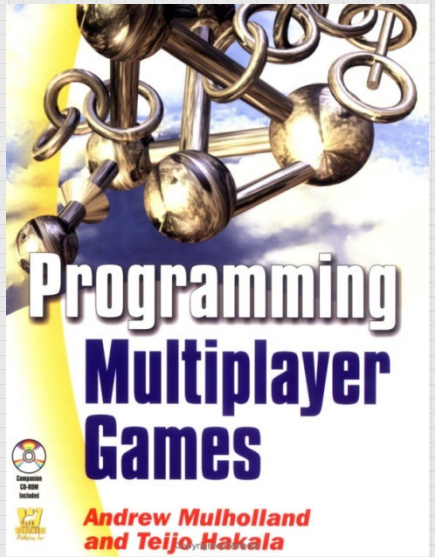
Tricks of the 3D Game Programming Gurus-Advanced 3D Graphics and Rasterization
André LaMothe



Biblioteca de Física

Biblioteca de Rede

referência



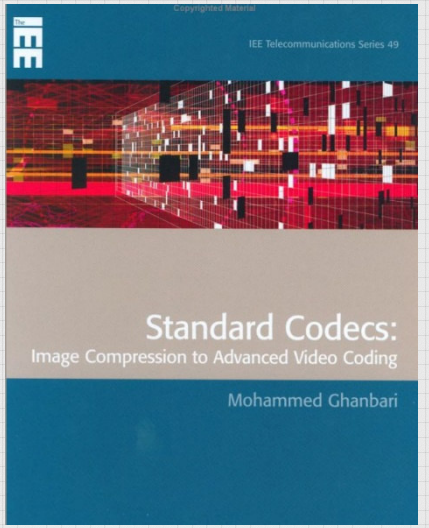
Programming Multiplayer Games
Andrew Mulholland

Biblioteca de Recursos

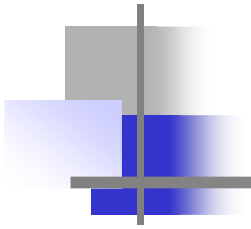
referência

www.gametutorials.com

referência



Standard Codecs: Image Compression to Advanced Video Coding
M. Ghanbari



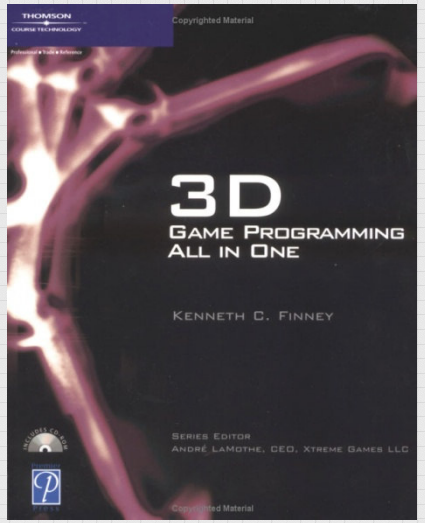
Parser de Scripts

Mapeamento com atributos dos objetos dinâmicos de um cenário

Mapeamento com algumas funções do SDK

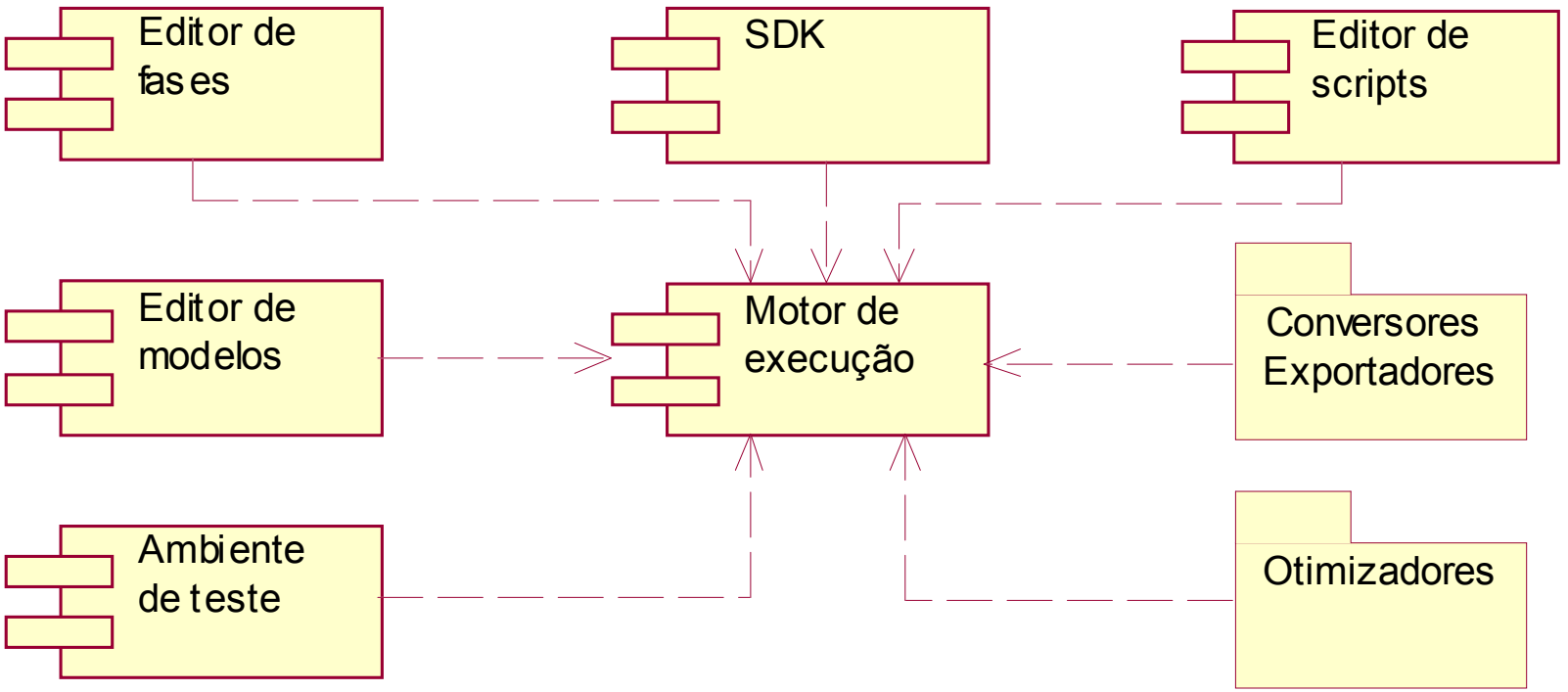
Arquitetura Ferramental

referência

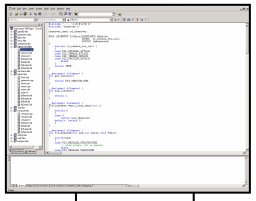


3D Game Programming All in One
Kenneth C Finney

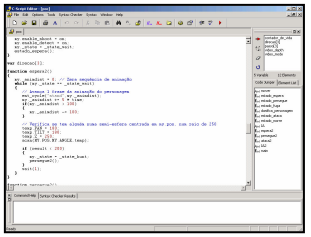
Arquitetura Ferramental



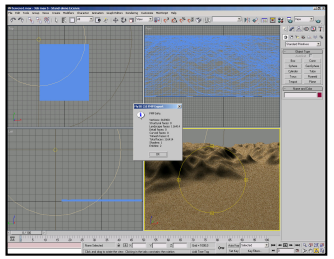
Arquitetura Ferramental



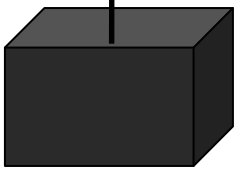
SDK



Script editors

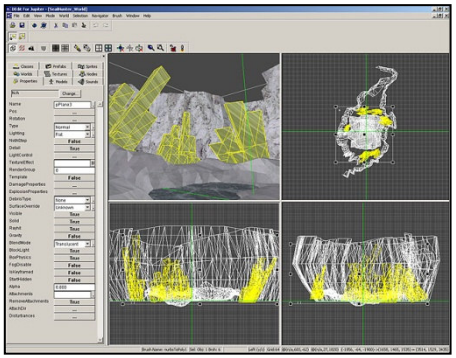


Conversores / Exportadores



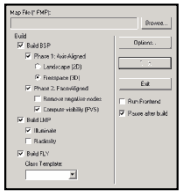
Engine Core

Front End



Level Editors

EXEMPLO



Builders

Arquitetura Ferramental