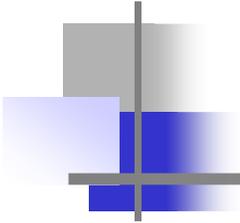


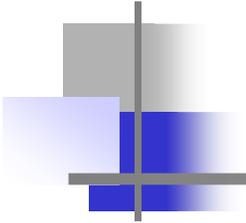
Visualização, Simulação e Games

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



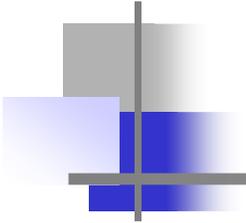
Parte 1 – Conceitos de Real Time Rendering

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



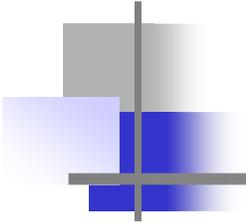
Parte 2 - Arquitetura de GPU's

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



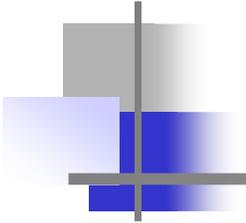
Parte 3 - Arquitetura de Game Engines

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



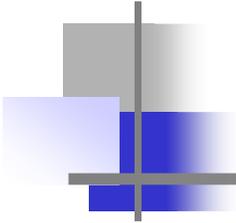
Parte 4 - Real-Time Physics

- 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.

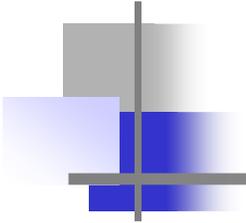
Finney, Kenneth C., **3D Game Programming All in One** Premiere Press, ISBN: 1-59200-136-X

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)

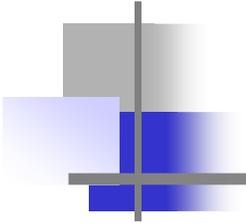
Resources::

Microsoft (www.microsoft.com/directx)

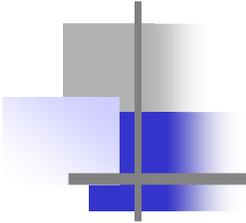
ATI (www.ati.com)

NVidia (www.nvidia.com)

XNA (www.xnatutorials.com)



Objetivos do Curso



Avaliação do Curso

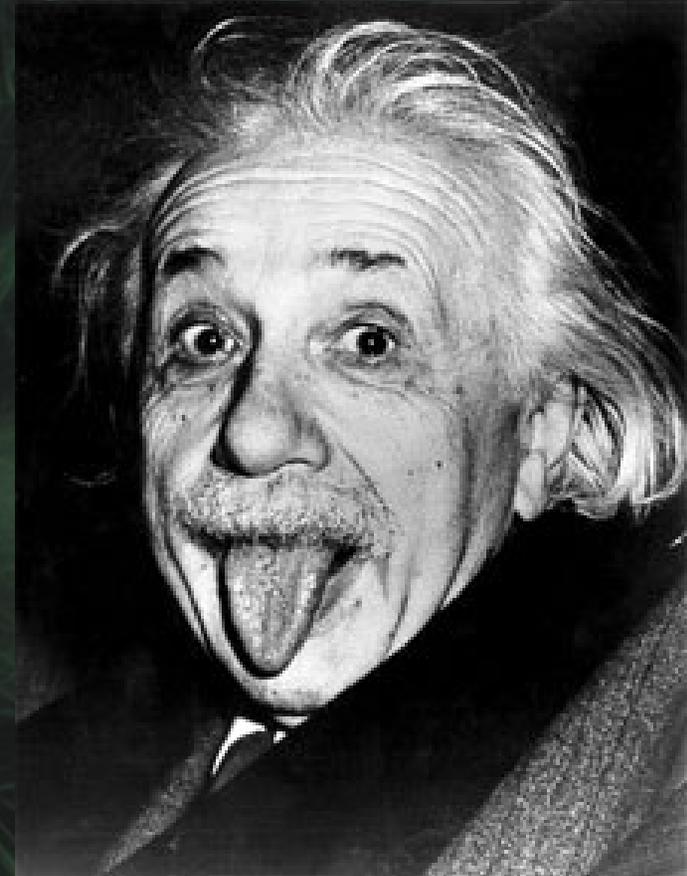
Parte 0 - Visão Geral

Para quê você quer fazer um game?

- Só por prazer... (porque tudo tem que ter razões econômicas\$?)
- 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
- Ilustrador
- 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

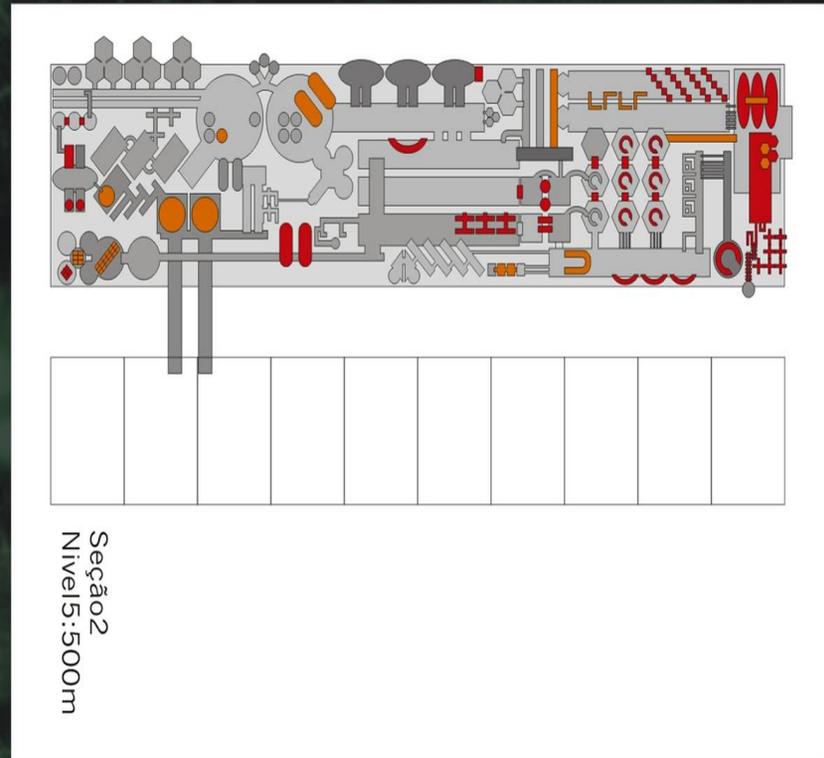
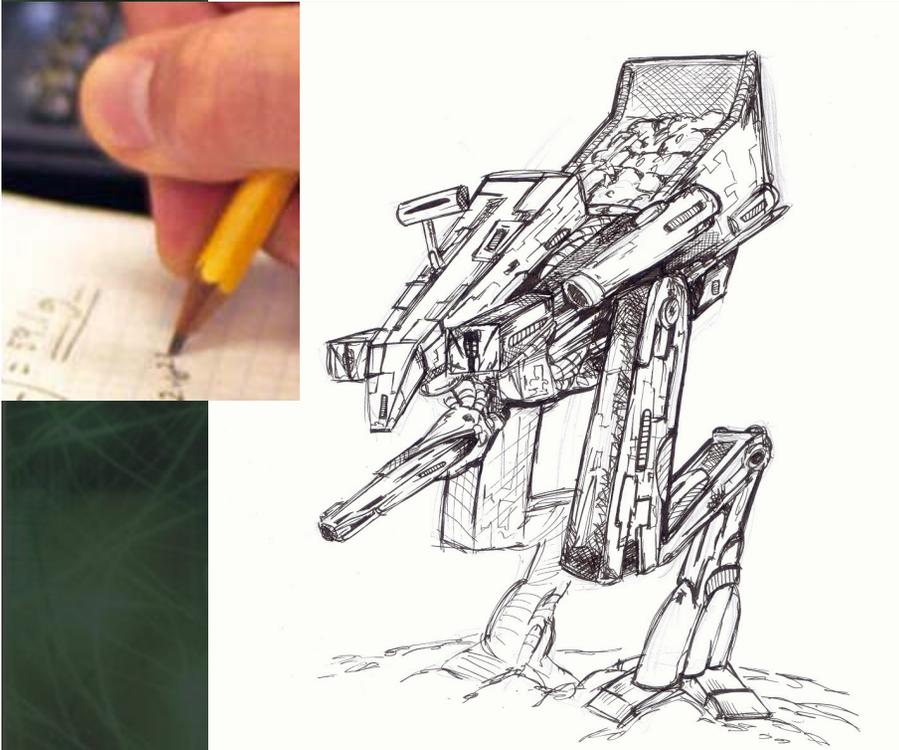


Computador sabe contar estórias?



3- Agora precisamos entender sua estória...

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



Ferramentas

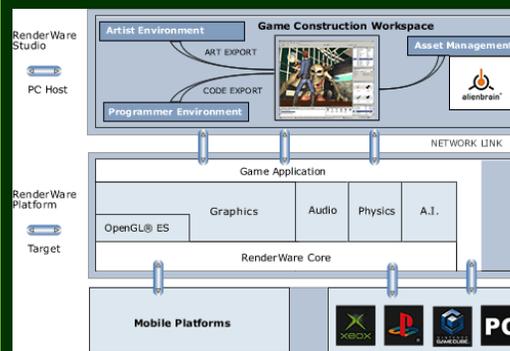
Quando se inventou o computador, criou-se uma máquina a mais, quando se criou o compilador, criou-se uma nova era tecnológica.

4- engine...

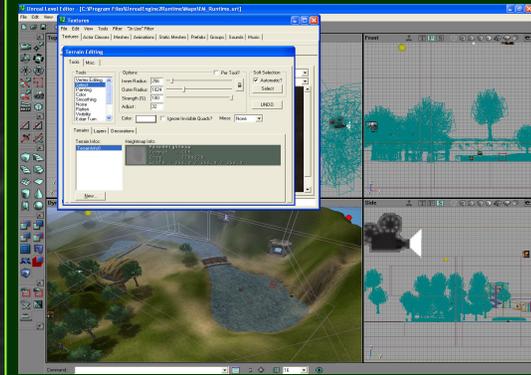
www.crytek.com



www.renderware.com



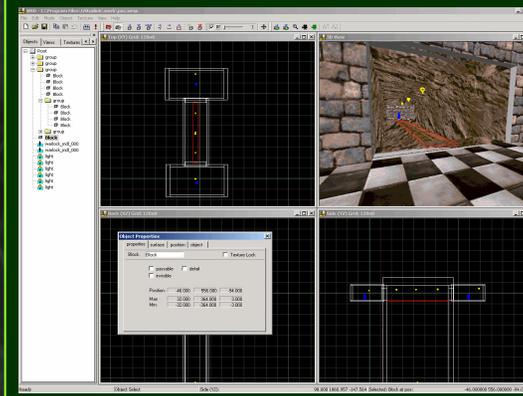
www.unrealengine.com



www.ogre3d.org



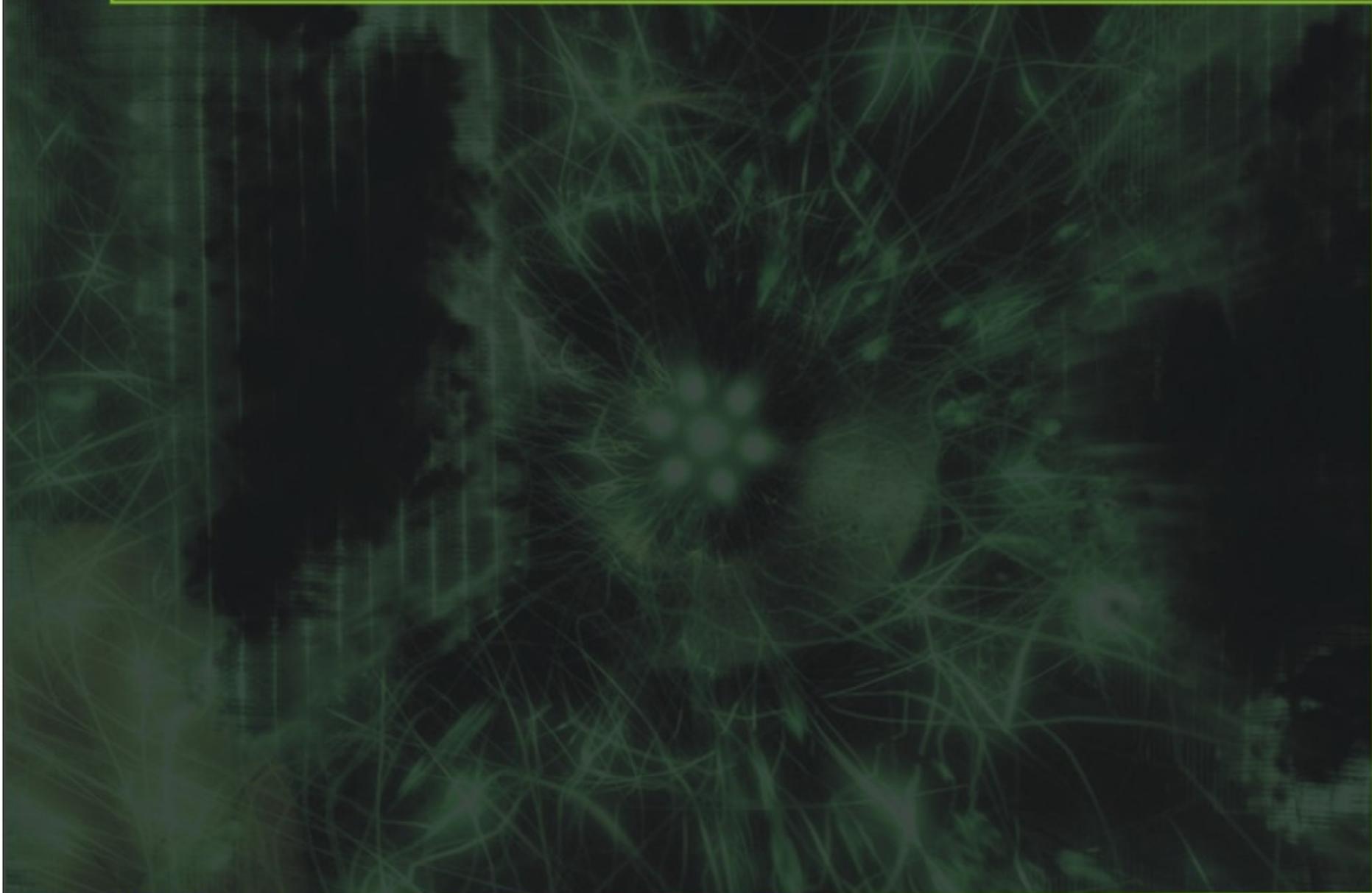
www.3dgamestudio.com



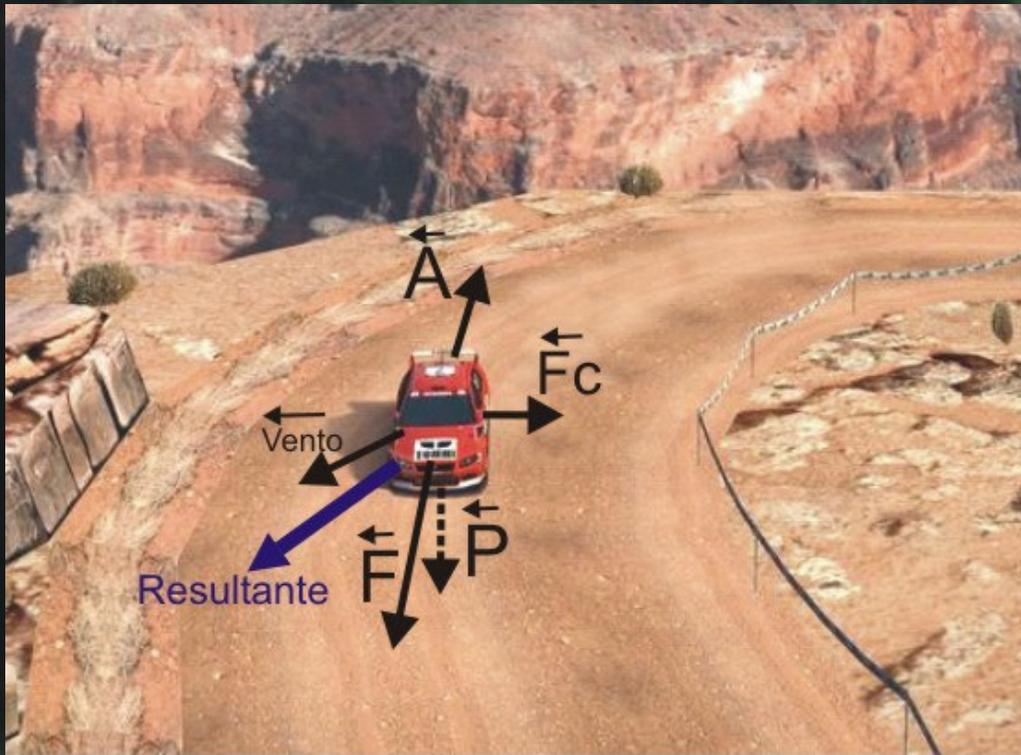
www.garagegames.com



4- engine...



Engine - Física



Física

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

Física



Home

About AGEIA

News

Technology

Partners

Developers

NovodeX home

downloads

support

tools & middleware

solution providers



Our Collision Detection
Makes Quite an Impact.



Download
Rocket Now



Download
NovodeX SDK



Image: Getty Images

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.

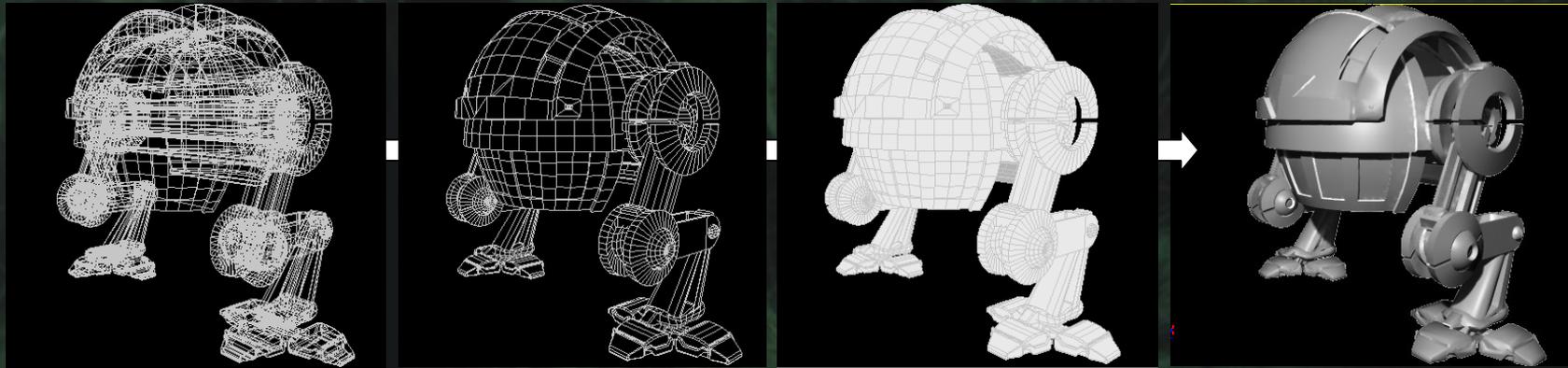
For non-game inquires for the NovodeX SDK please contact our business development department at bizdev@ageia.com

© 2005 AGEIA Inc. All Rights Reserved. [Contact AGEIA](#)

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

```
...
//! Background panorama class
//! This class implements a panorama background box surrounding the level
class panorama : public flyBspObject
{
public:
    int img[4]; //! array
    float size; //! size of
    flyVector color; //! color

    //! Default constructor
    panorama(int color)
    {
        type=TYPE_PANORAMA;
        img[0]=img[1]=img[2]=img[3]=img;
    }
    //! Default destructor
    virtual ~panorama()
    {
    }
    //! OpenGL drawing function, is called
    void draw();
    //! Parameter information retrieval
    int get_custom_parea_desc(int i:fly)
    {
    }
};

//! Background sky class
//! This class implements a sky represent
//! procedural textures (shaders) and a
class sky : public flyBspObject
{
public:
    flyMesh *objmesh; //! the object
    flyVector color;

    //! Default constructor
    sky(int color): objmesh(0), color(0)
    { type=TYPE_SKY; }
    //! Default destructor
    virtual ~sky()
    {
    }
    //! OpenGL drawing function, is called
    void draw();
    //! Parameter information retrieval
};

//! Description class for the robot class
class robot_desc : public flyClassDesc
{
public:
    //! Creates a new object of the described class
    flyBspObject *create() { return new robot; }
    //! Retrieves the name of the described class
    const char *get_name() { return "robot"; }
    //! Retrieves the integer value that represents the type of the described class
    int get_type() { return TYPE_ROBOT; }
};

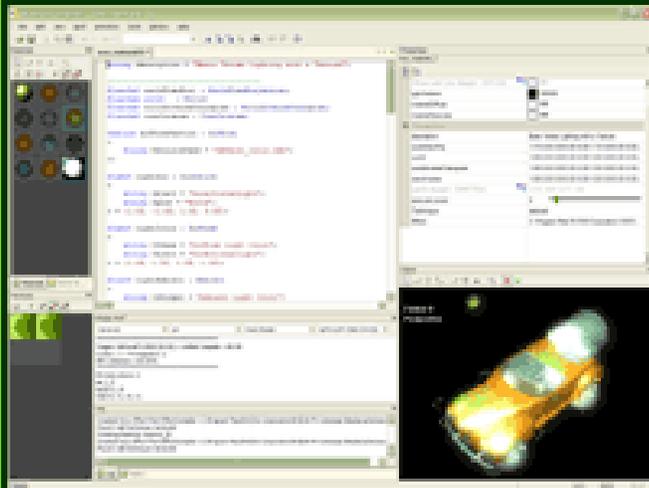
//! Description class for the generator class
class generator_desc : public flyClassDesc
{
public:
    //! Creates a new object of the described class
    flyBspObject *create() { return new generator; }
    //! Retrieves the name of the described class
    const char *get_name() { return "generator"; }
    //! Retrieves the integer value that represents the type of the described class
    int get_type() { return TYPE_GENERATOR; }
};

//! Description class for the door class
class door_desc : public flyClassDesc
{
public:
    //! Copy-constructor
    door(const door& in)
    : flyBspObject(in),
    objmesh(in.objmesh),
    animstate(in.animstate),
    ntriggers(in.ntriggers),
    animstate(in.animstate),
    state(in.state),
    nlights(in.nlights),
    bp(in.bp)
    {
    }
    //! Members initialisation function, is called upon activating the object
    void init();
    //! State update function, is called once per frame, animates the door
    int step(int dt);
    //! Returns the object's mesh in a flyMesh object
    flyMesh *get_mesh()
    { return objmesh; }
    //! OpenGL drawing function, is called whenever the engine's camera can see the object
    void draw();
    //! Message receiving function, is used for communication between objects
    int message(const flyVector p, float rad, int esp, int para, void *data);
    //! Parameter information retrieval function
    int get_custom_parea_desc(int i:flyClassDesc *pd);
    //! Creates another instance of the object similar to this one
    flyBspObject *clone()
    { return new door(*this); }
};
...

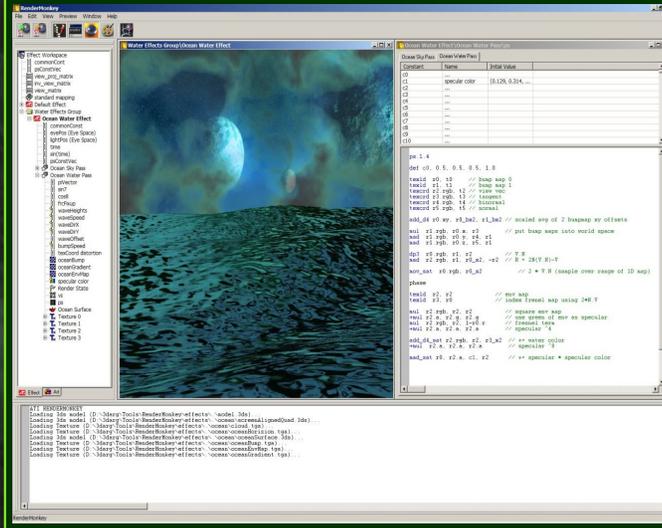
```

5- Ainda falando de programação

NVidia FX Composer

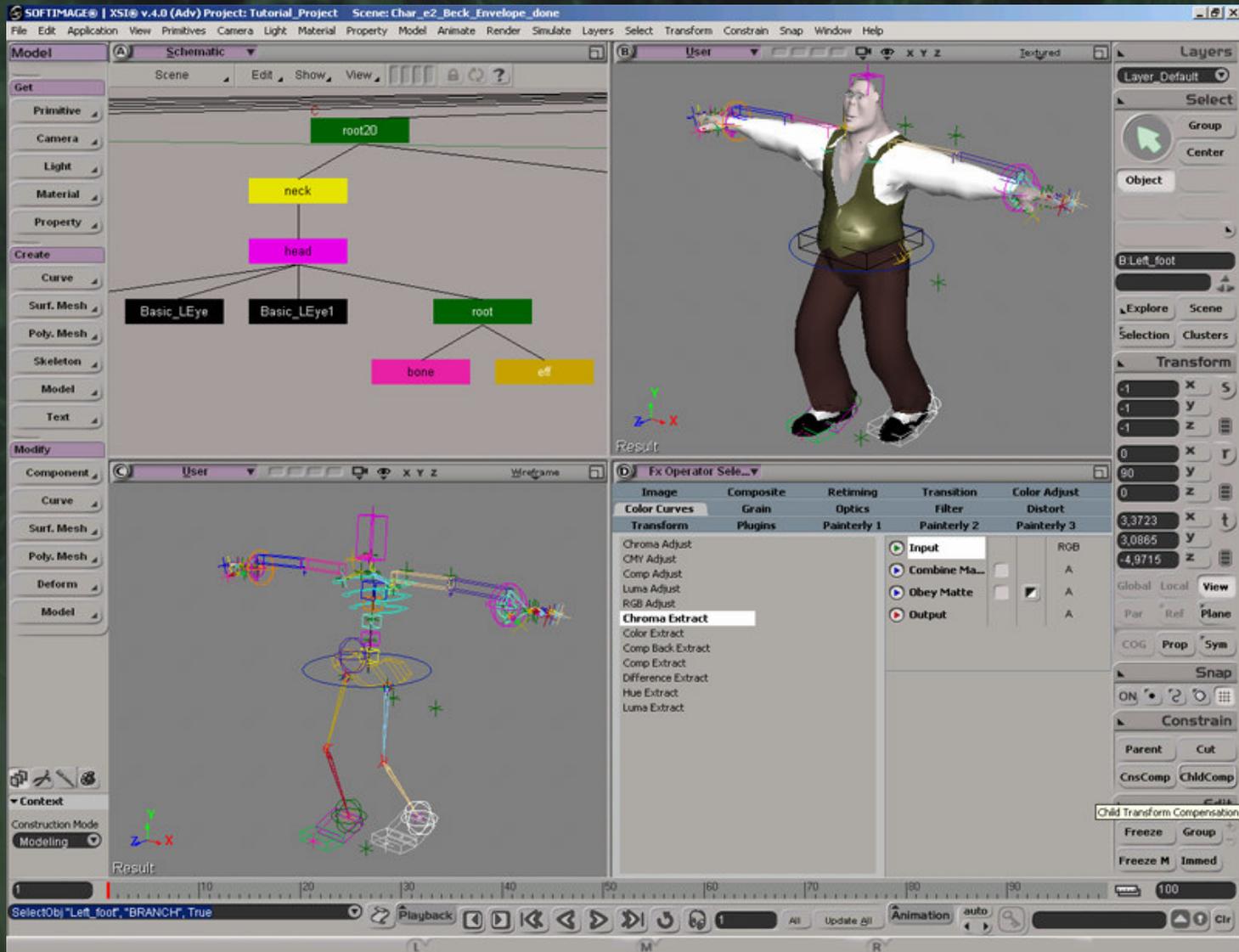


ATI Rendermonkey

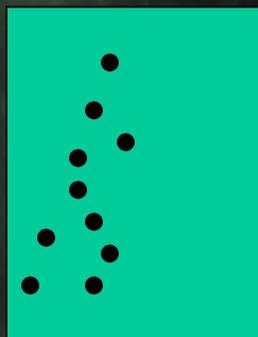


- Shaders: HLSL , CG, OpenGL Shader Language

6- Modelagem



6- Animação



Necessidade
Da animação

Eu acredito



Eu não acredito



Grau de fidelidade da modelagem