Protótipo de um Ambiente 3D para Jogos, utilizando a *Engine* Crystal Space com DirectX e linguagem C++

RODRIGO PETER

Prof. Dalton Solano dos Reis
Orientador

FURB

Roteiro da Apresentação

- Introdução (Jogos Eletrônicos, Engines 3D)
- Objetivos do Trabalho
- Ambientes 3D
- Crystal Space (características e conceitos)
- DirectX
- Protótipo
- Conclusões e Extensões

Introdução - Jogos Eletrônicos(História)

- 1961 - Steve Russel - MIT - Space War
- 1966 - Aparelho de TV com jogo
- 1970 - Odissey
- 1972 - Atari Pong (versão “caseira” em 1974)
- 1977 - Atari Video Computer System
- 1985 - Master System/Nintendo
- 1989 - 16 bits (Super Nintendo)

FURB

Introdução - Jogos Eletrônicos(História)

- 1993 - Doom
- 1995 - Hoje
  - Nintendo: Nintendo 64, GameCube
  - Sega: Satum, DreamCast
  - Sony: PlayStation, PlayStation 2
  - Microsoft: Xbox

Introdução - Engines 3D

- API para visualização 3D em tempo real
- Visualização independente da API gráfica (OpenGL, DirectX)

Introdução - Engines 3D

- Implementa:
  - Visualização Eficiente
  - Detecção e tratamento de colisão
  - Outras características:
    - Primitivas matemáticas (vetores, matrizes)
    - Efeitos (iluminação, fog, lens flare)
    - Suporte a rede
    - Interface com o usuário

FURB

Objetivos do Trabalho

- Criação do Ambiente 3D
- Aplicação de “efeitos”
  - Iluminação Colorida
  - 6DOF
  - Transparência
  - Visualização em Primeira Pessoa

FURB

Ambientes 3D

- Três Dimensões (Sully)
  - *Object Space* (3D)
  - *World Space* (3D) (Tremblay)
  - *View Space* (3D)
  - *Screen Space* (2D)
- Elementos do Espaço 3D (O'Neill, superfícies verticais)

FURB

Crystal Space

- Conceito (pacote de componentes e bibliotecas)
- O projeto CS
- Divisão (Bibliotecas do CS)
  - *Geometry Library*
  - 3D engine
  - *Utility Library*
  - *Tool Library*

FURB

Crystal Space

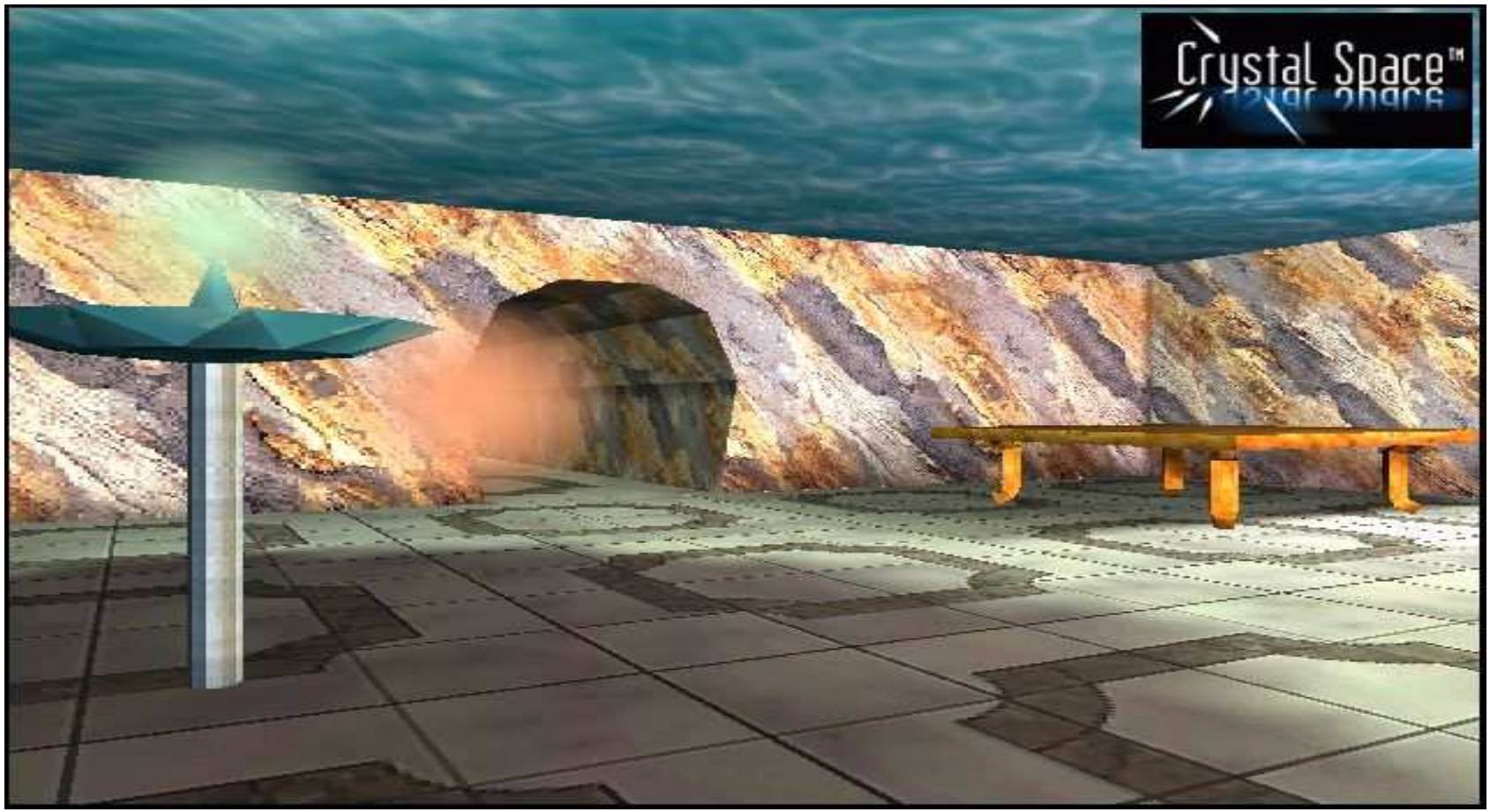
- Open Source (GNU LGPL)
- 6DOF
- Portais
- BSP Opcional (dentro de um setor)
- Renderizador de terrenos
- Objetos móveis
- Iluminação estática com sombras

FURB

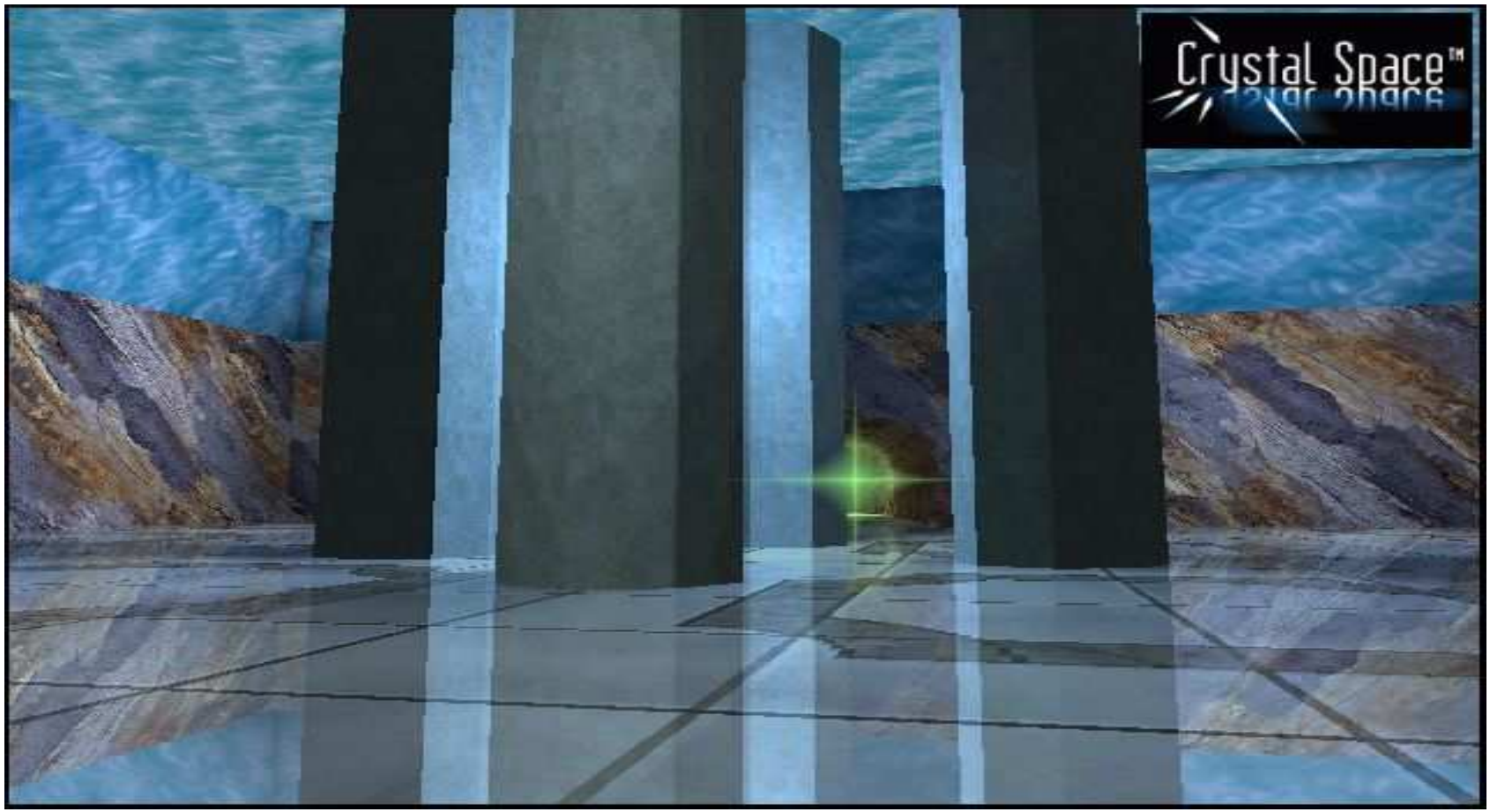
Crystal Space

- Outras características:
  - Transparência (alpha)
  - Espelhos opacos ou semi-transparentes
  - Esfera celeste dinâmica
  - Superfícies curvas (Bezier)
  - Estrutura de plugins
  - Portado para Windows, Linux, Unix, Macintosh, OS/2, NextStep

Crystal Space



Crystal Space



Crystal Space

Crystal Space - Aplicação CS

- SCF
- Contador de Referências
- Registro de Objetos
- Gerenciador de *plugin*
- Fila de Evento
- Relógio Virtual
- Linha de Comando

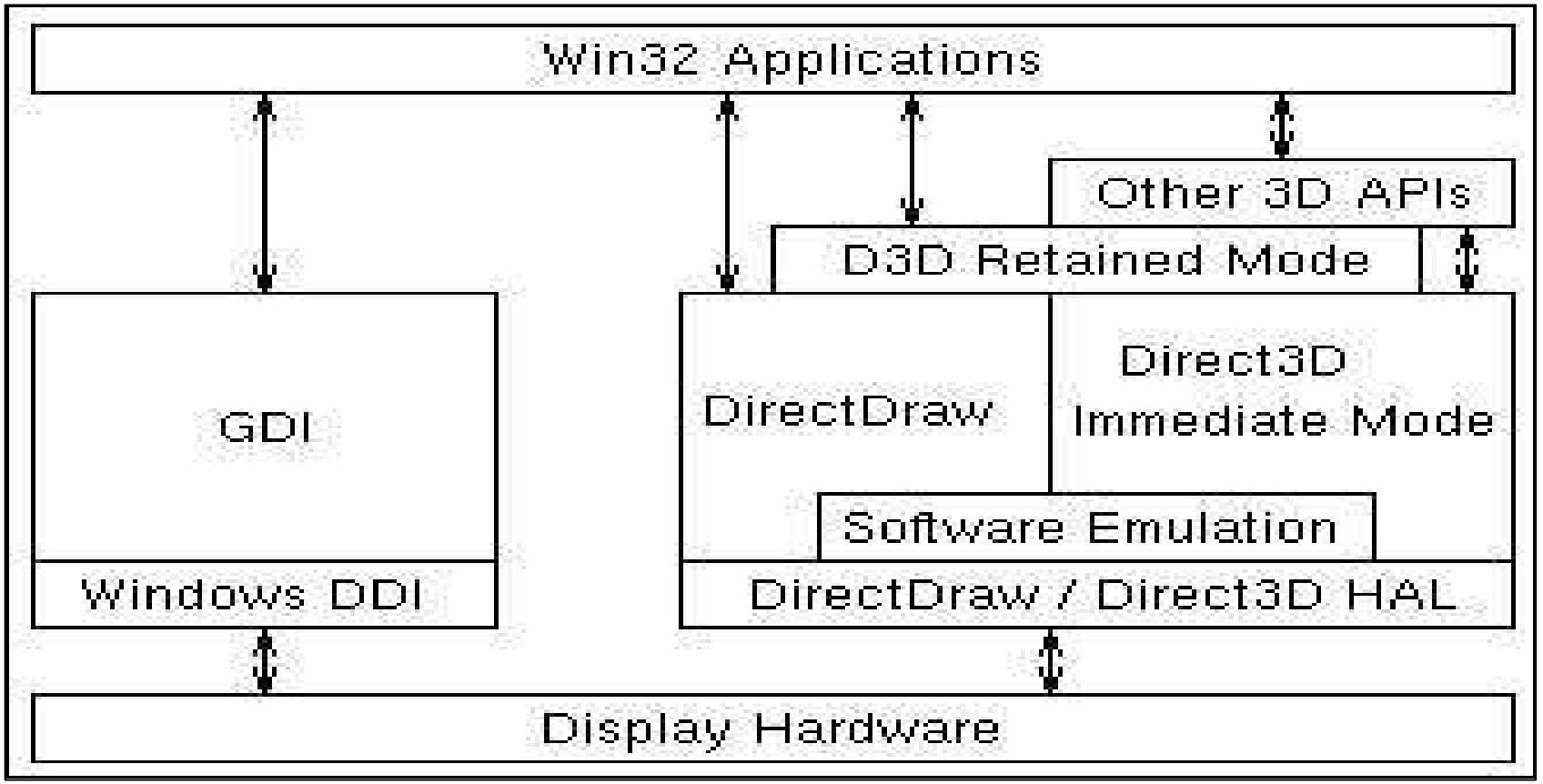
Crystal Space - Aplicação CS

- Gerenciador de Configuração
- *Drivers* de Entrada
- Classe csInitializer

DirectX

- Origem e Propósitos
- Componentes (versão 8.1)
  - DirectXGraphics
  - DirectXAudio
  - DirectInput
  - DirectPlay
  - DirectShow

DirectX



Fonte: http://www.scorpioncity.com/djdirectxtut.html

Protótipo

- Requisitos:
  - Sistema operacional suportado (Windows 98)
  - Microsoft Visual C++
  - pacote principal do Crystal Space
  - DirectX 6.1 ou superior
  - arquivos mapa
  - bibliotecas zlib, libpng, libjpeg

Protótipo

- Especificação e Implementação
  - Arquivo de Configuração (chaves, hierarquia)
  - Arquivo de Definições de Ambiente (arquivo mapa)
  - Classe Simple

Protótipo

```
; O padrão é Video.ScreenWidth = 640 e
;Video.ScreenHeight = 480
Video.ScreenWidth = 640
Video.ScreenHeight = 480
; Screen depth (currently supports 8,
15, ;16, and 32 bits, NO 24 bits!)
Video.ScreenDepth = 16
; Use full-screen mode if available
Video.FullScreen = no
```

video.cfg

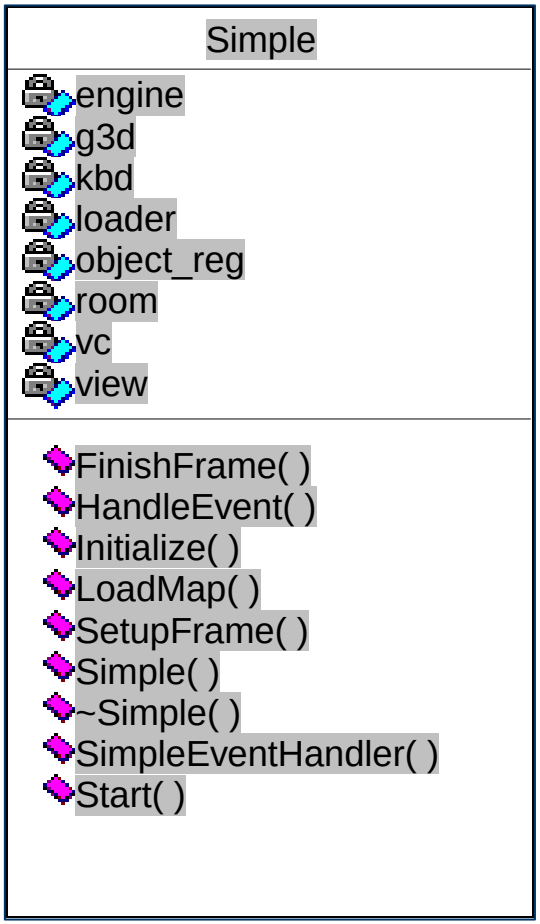
Protótipo - arquivo Mapa

```
WORLD {
  TEXTURES {
    TEXTURE 'muro_quarto_inicial' (FILE (/lib/stdtex/mosholes.png))
...
  }
  MATERIALS (
    MATERIAL 'muro_quarto_inicial' (TEXTURE ('muro_quarto_inicial'))
    ...
SECTOR 'room' (
    MESHOBJ 'walls' (
          ZFILL ()
          PLUGIN ('thing')
          PARAMS (
            VERTEX (-20,-1,-20) VERTEX (-20,-1,20)
            VERTEX (20,-1,20) VERTEX (20,-1,-20)
                ...
            MATERIAL ('muro_quarto_inicial')
            TEXLEN (5)
            POLYGON 'down1' (VERTICES (0,1,10,15) MATERIAL
('floors_1_d1n__128') COSFACT(0.8)
             TEXTURE (PLANE (floor)))
```

FURB

Protótipo

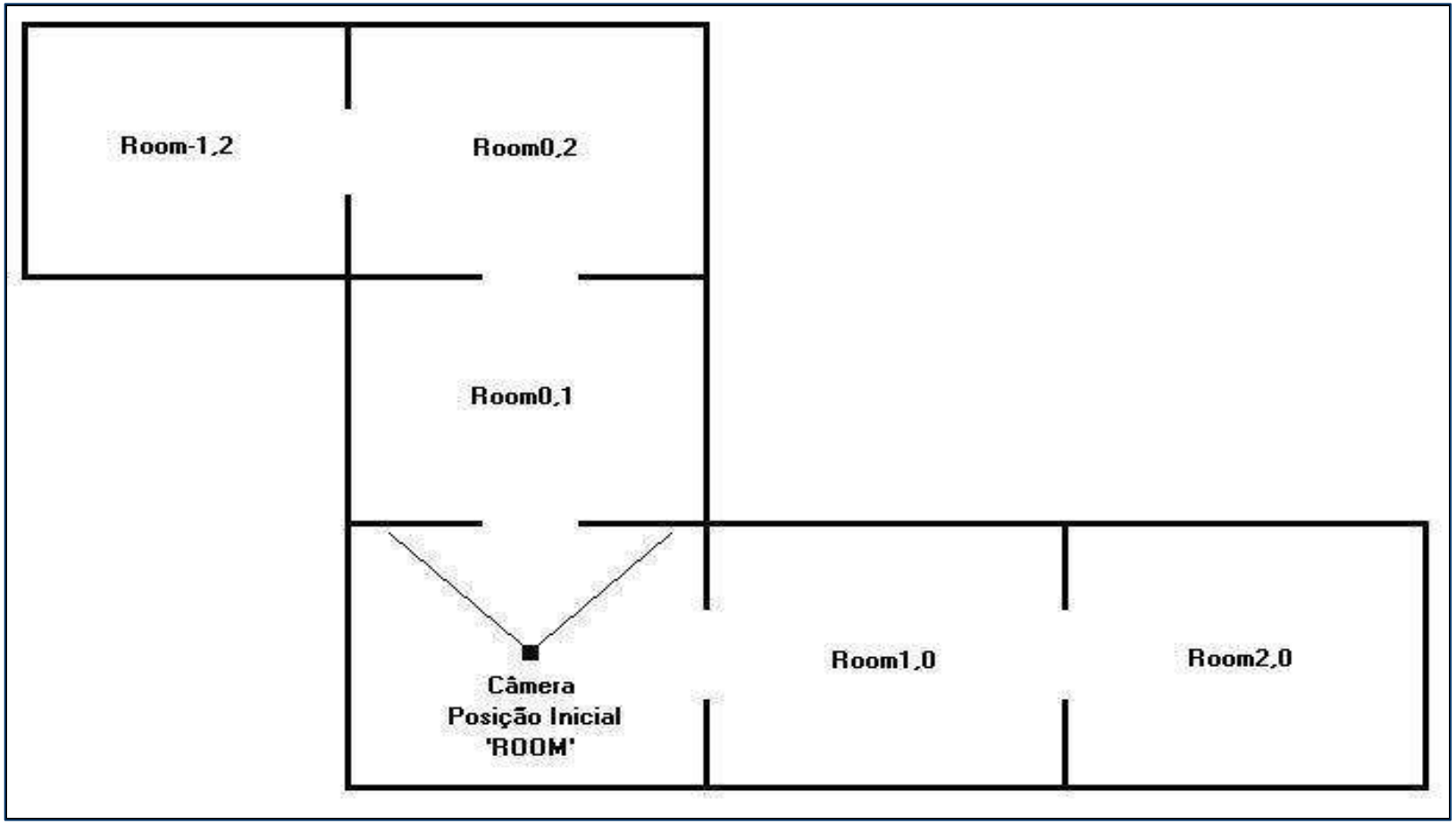
Atributos e Métodos da classe Simple



Protótipo

- Principais métodos de Simple:
  - Método ::Initialize() (::CreateEnvironment(), ::RequestPlugins(), ::OpenApplication())
  - Método ::Start() (loop)
  - Método ::HandleEvent() (eventos, ESC)
  - Método ::SetupFrame() (::GetTransform(), ::Move(), ::Draw())
  - Método ::LoadMap()

Protótipo - Planta Baixa



Protótipo

- Comandos de locomoção no Ambiente

Tecla	Efeito
Seta para Cima	Movimento para frente
Seta para Baixo	Movimento para trás
Seta para Direita	Movimenta para a direita
Seta para Esquerda	Movimenta para a esquerda
Page Up	Rotação para cima
Page Down	Rotação para baixo

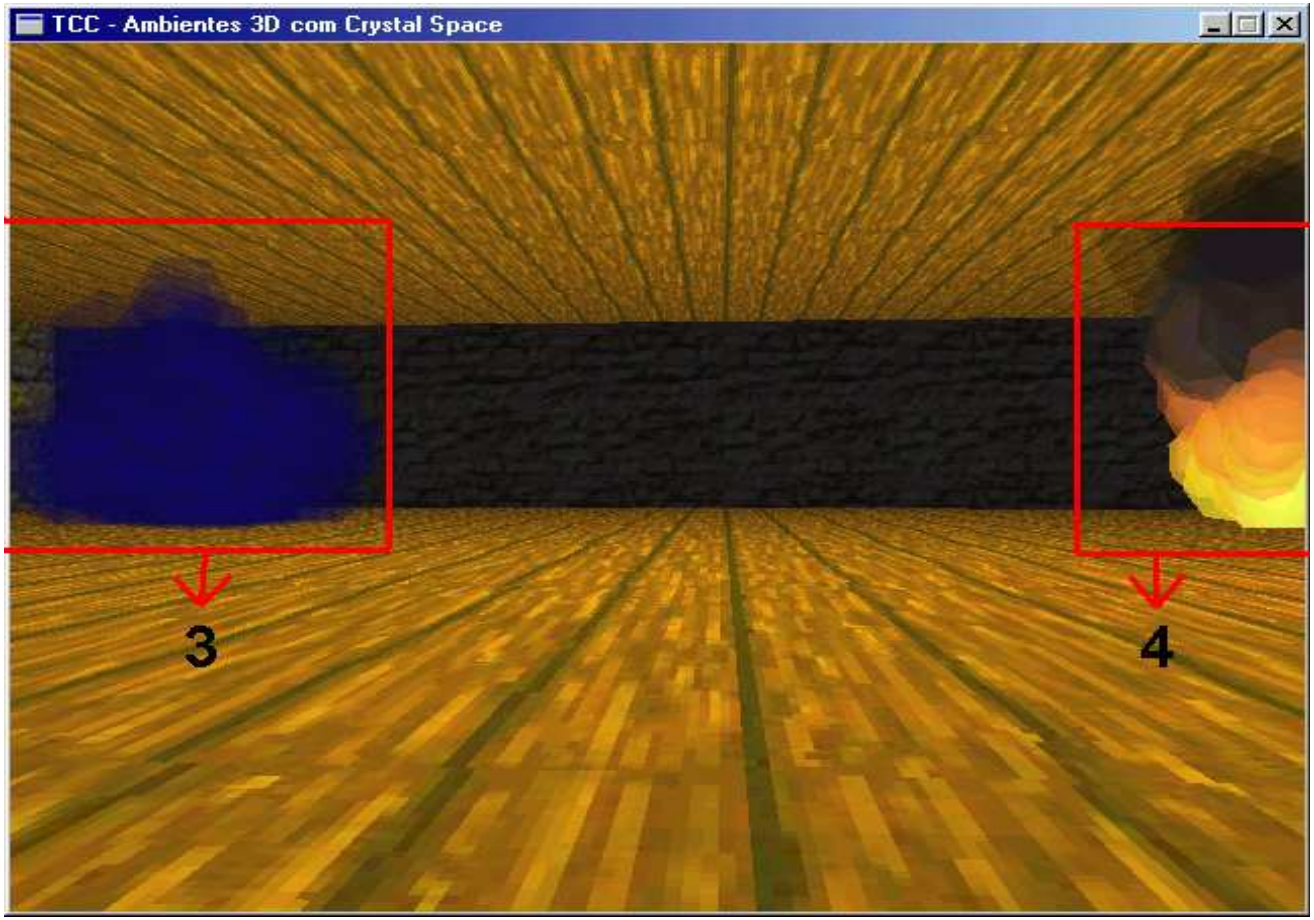
Protótipo

- Tela inicial
- 1-Chuva
- 2-Neve

Room

Protótipo

- 3-Fumaça
- 4-Fogo



Room -1,2

Conclusões

- Versões (garantias)
- Documentação (compilação, classes, tutoriais)
- DirectX
- Hardware de aceleração gráfica
- Aplicações em *background*
- CS

Extensões

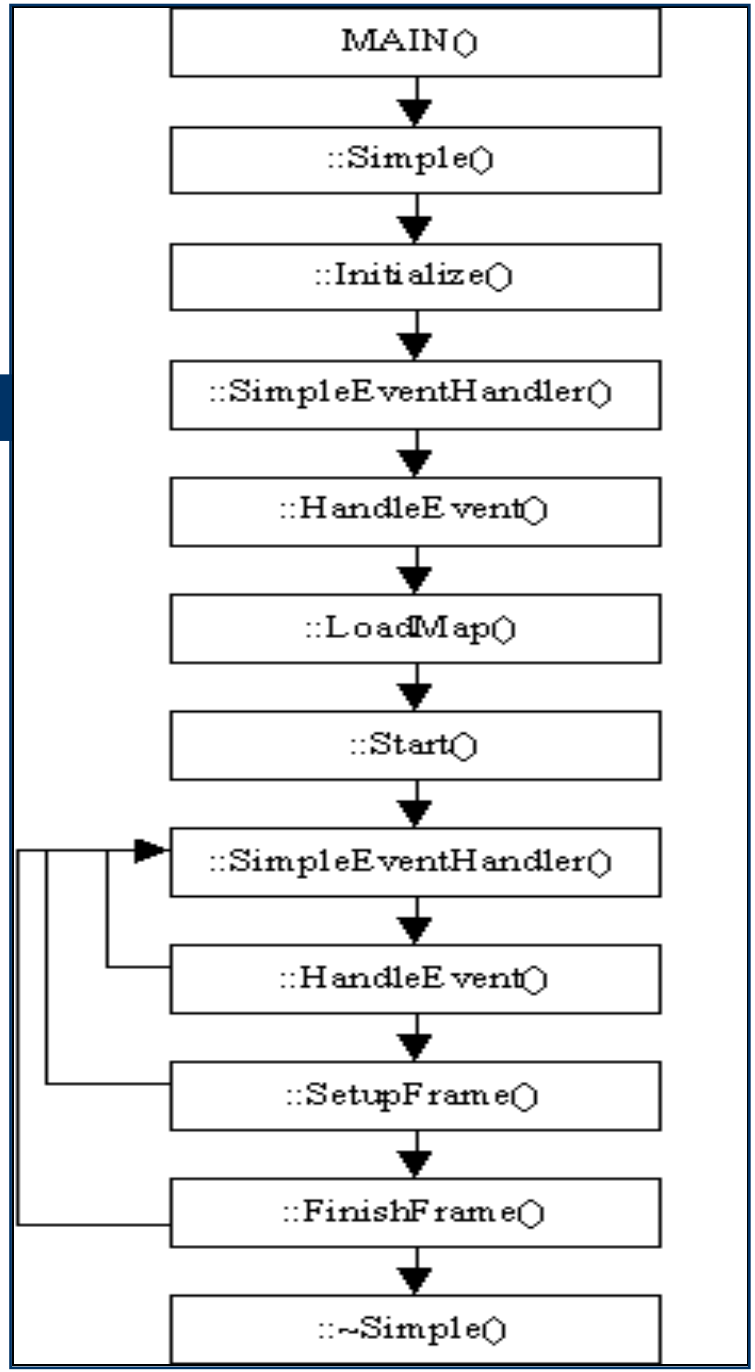
- Detecção de colisão
- Nova plataforma
- Outros comandos de teclado
- Multiusuário
- Gravidade

Obrigado !

Fim

Classe Simple

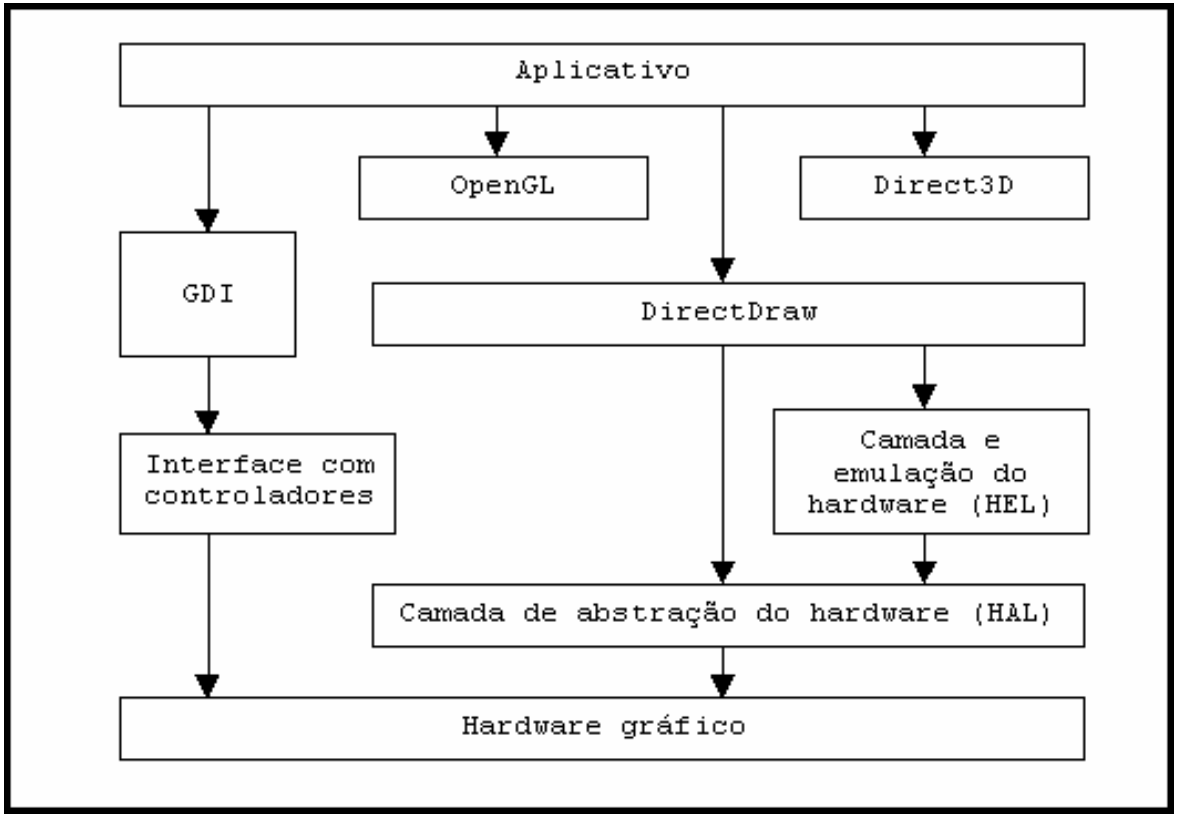
Seqüencia de execução dos métodos de Simple



FURB


FURB

DirectX


Aplicativo
OpenGL
Direct3D
GDI
DirectDraw
Interface com controladores
Camada e emulação do hardware (HEL)
Camada de abstração do hardware (HAL)
Hardware gráfico

::LoadMap()

```
Bool Simple::LoadMap ()
{
  // Seta o diretório atual VFS para o plano que deseja-se carregar.
  IVFS* VFS = CS_QUERY_REGISTRY (object_reg, iVFS);
  VFS->ChDir ("/lev/partsys");
  VFS->DecRef ();
// Carrega o arquivo do plano que é chamado de 'world' e encontra-se
// no diretório C:/CS/Data/PartSys.
  if (!loader->LoadMapFile ("world"))
  {
    csReport (object_reg, CS_REPORTER_SEVERITY_ERROR,
        "crystalspace.application.simpmap",
        "Couldn't load level!");
    return false;
  }
engine->Prepare ();
...
```

::SetupFrame()

```
void Simple::SetupFrame ()
{
// primeiramente, obtém o tempo percorrido do relógio virtual.
  CsTicks elapsed_time = vc->GetElapsedTicks ();
// agora, gira a câmera de acordo com o determinado no teclado (digitado)
  float speed = (elapsed_time / 1000.0) * (0.03 * 20);

  iCamera* c = view->GetCamera();
  if (kbd->GetKeyState (CSKEY_RIGHT))
    c->GetTransform ().RotateThis (CS_VEC_ROT_RIGHT, speed);
  if (kbd->GetKeyState (CSKEY_LEFT))
    c->GetTransform ().RotateThis (CS_VEC_ROT_LEFT, speed);
  if (kbd->GetKeyState (CSKEY_PGUP))
    c->GetTransform ().RotateThis (CS_VEC_TILT_UP, speed);
  if (kbd->GetKeyState (CSKEY_PGDN))
    c->GetTransform ().RotateThis (CS_VEC_TILT_DOWN, speed);
  if (kbd->GetKeyState (CSKEY_UP))
    c->Move (CS_VEC_FORWARD * 4 * speed);
  if (kbd->GetKeyState (CSKEY_DOWN))
    c->Move (CS_VEC_BACKWARD * 4 * speed);
...
```

::HandleEvent()

```
bool Simple::HandleEvent (iEvent& ev)
{
  if (ev.Type == csevBroadcast && ev.Command.Code == cscmdProcess)
  {
    simple->SetupFrame ();
    return true;
  }
  else if (ev.Type == csevBroadcast && ev.Command.Code == cscmdFinalProcess)
  {
    simple->FinishFrame ();
    return true;
  }
  else if (ev.Type == csevKeyDown && ev.Key.Code == CSKEY_ESC)
  {
    iEventQueue* q = CS_QUERY_REGISTRY (object_reg, iEventQueue);
    if (q)
    {
      q->GetEventOutlet()->Broadcast (cscmdQuit);
      q->DecRef ();
    }
    return true;
...
```

::Initialize()

```
bool Simple::Initialize (int argc, const char* const argv[])
{
  object_reg = csInitializer::CreateEnvironment (argc, argv);
  if (!object_reg) return false;
  csDebuggingGraph::SetupGraph (object_reg);

  if (!csInitializer::RequestPlugins (object_reg,
        CS_REQUEST_VFS,
        CS_REQUEST_SOFTWARE3D,
        CS_REQUEST_ENGINE,
        CS_REQUEST_FONTSERVER,
        CS_REQUEST_IMAGELOADER,
        CS_REQUEST_LEVELLOADER,
        CS_REQUEST_REPORTER,
        CS_REQUEST_REPORTERLISTENER,
        CS_REQUEST_END))

  {
...
```

Protótipo - Classe Simple (código)

```
class Simple
{
private:
  iObjectRegistry* object_reg;
  iEngine* engine;
  iLoader* loader;
  iGraphics3D* g3d;
  iKeyboardDriver* kbd;
  iVirtualClock* vc;
  iSector* room;
  iView* view;
  static bool SimpleEventHandler (iEvent& ev);
  bool HandleEvent (iEvent& ev);
  void SetupFrame ();
  void FinishFrame ();
  bool LoadMap ();
public:
  Simple ();
  ~Simple ();
  bool Initialize (int argc, const char* const argv[]);
  void Start ();
};
```