Direct3D

Programação de Jogos

Judson Santos Santiago

Introdução

- A API Win32 foi desenvolvida para a criação de aplicações
  - A GDI, componente responsável pelo desenho de pixels, linhas e imagens, é muito lenta para a programação de jogos

Para o desenvolvimento de jogos
a Microsoft desenvolveu um conjunto
de APIs conhecidas por

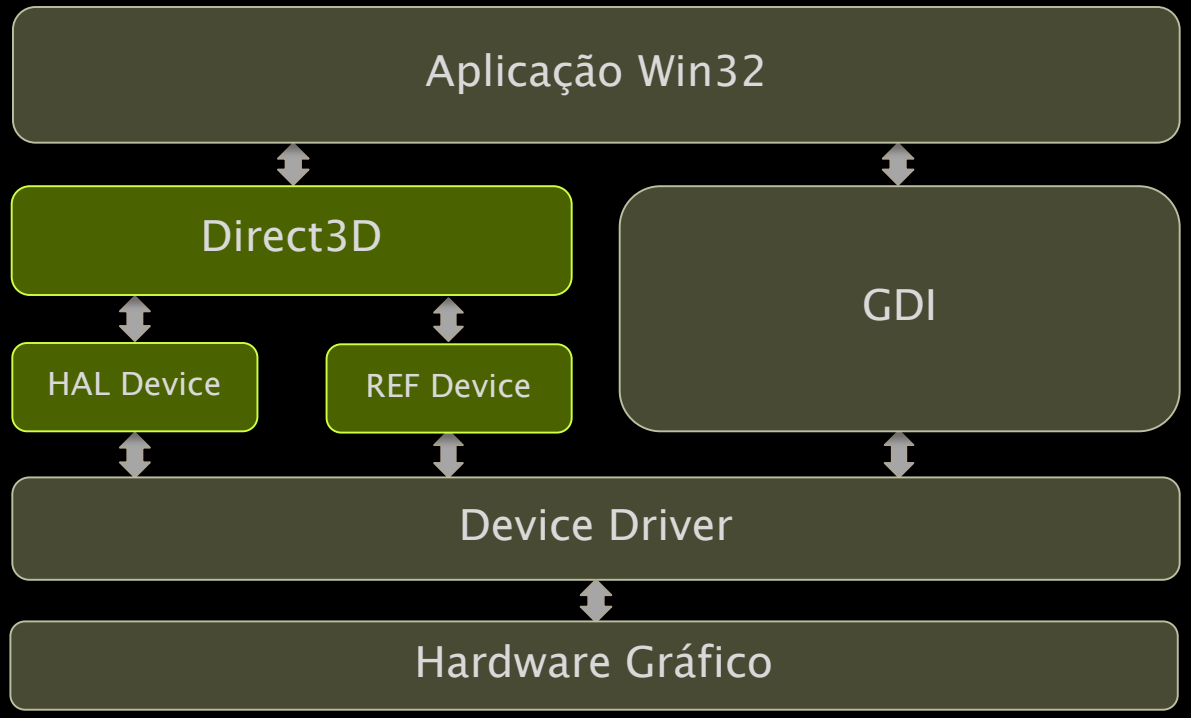
DirectX

DirectX

- O DirectX é um conjunto de APIs de baixo nível para criar jogos e outras aplicações multimídia de alto desempenho
  - Gráficos:
    - Direct3D 9
      Windows XP
    - Direct3D 10
      Windows Vista
    - Direct3D 11
      Windows 7
    - Direct3D 12
      Windows 10
  - Texto e Fontes:
    - DirectWrite
    - Direct2D
  - Áudio:
    - DirectSound
    - XAudio2
  - Entrada
    - DirectInput
    - Xinput

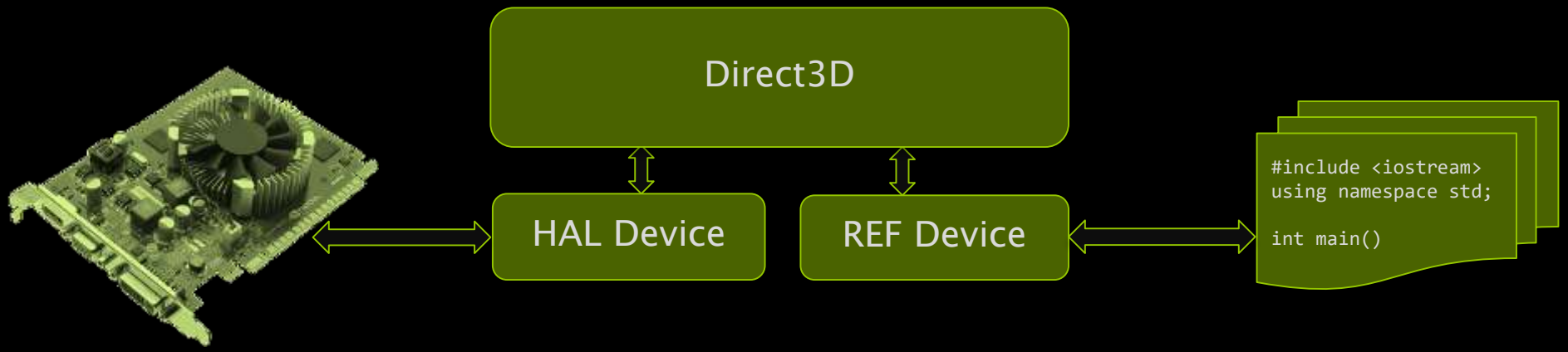
Direct3D

- Provê métodos para desenhar cenas eficientemente em uma tela usando o hardware gráfico disponível



Direct3D

- HAL (Hardware Abstracion Layer): é o dispositivo principal que utiliza as funções do hardware gráfico para acelerar o desenho



- REF (Reference): é um dispositivo de testes que implementa todas as funções do Direct3D em software

Instalação do DirectX

- DirectX SDK agora faz parte do Windows SDK

O último SDK do DirectX data de junho de 2010

  - No Windows 10:
    - Não é preciso instalar nada para usá-lo
    - Para desenvolver jogos é preciso instalar o Windows SDK
    - O Windows SDK já é instalado com o Visual Studio

Instalação do DirectX

- O SDK do Windows contém:
  - Documentação, exemplos e ferramentas
  - Arquivos para uso da API
    - Arquivos de cabeçalho (.h)
    - Arquivos de biblioteca (.lib)

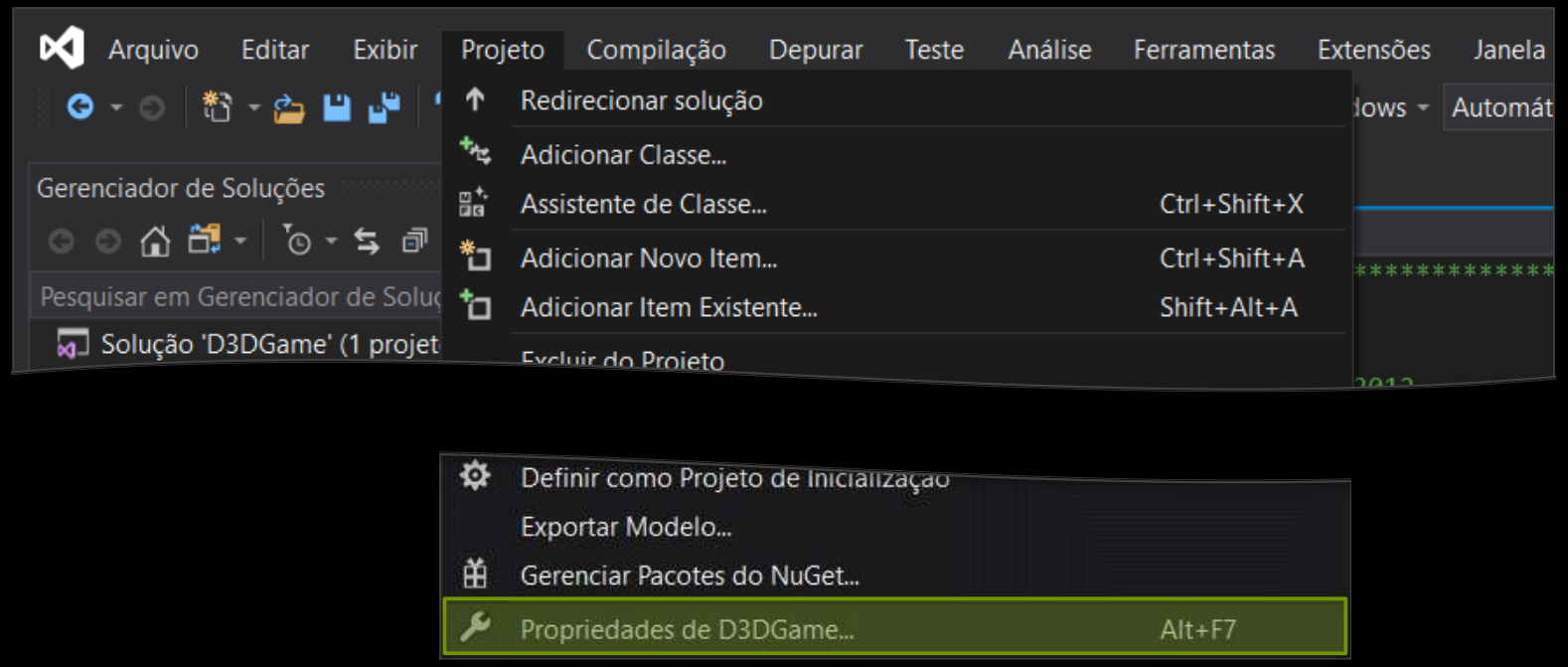
- Para usar as APIs do DirectX o programador precisa:
  - Incluir arquivos de cabeçalho

```
#include <dxgi.h>        // infraestrutura gráfica do DirectX
#include <d3d11.h>       // principais funções do Direct3D
```

  - Configurar o projeto para usar as bibliotecas

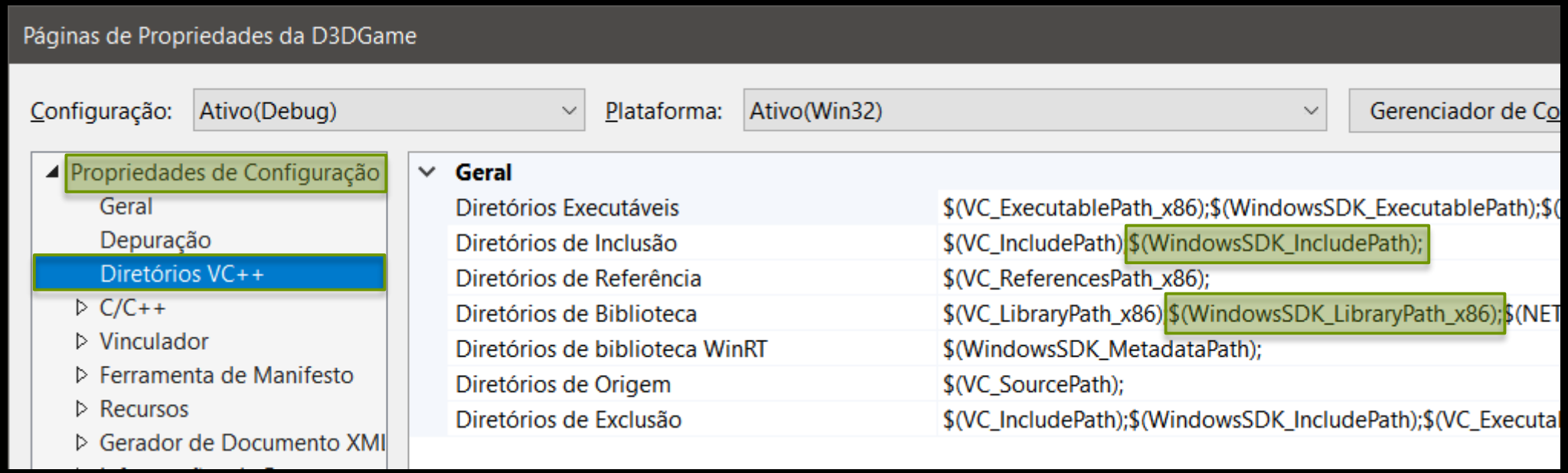
Configuração do Projeto

- Configurar o Projeto no Visual Studio
  Projeto > Propriedades de <NomeDoProjeto>...



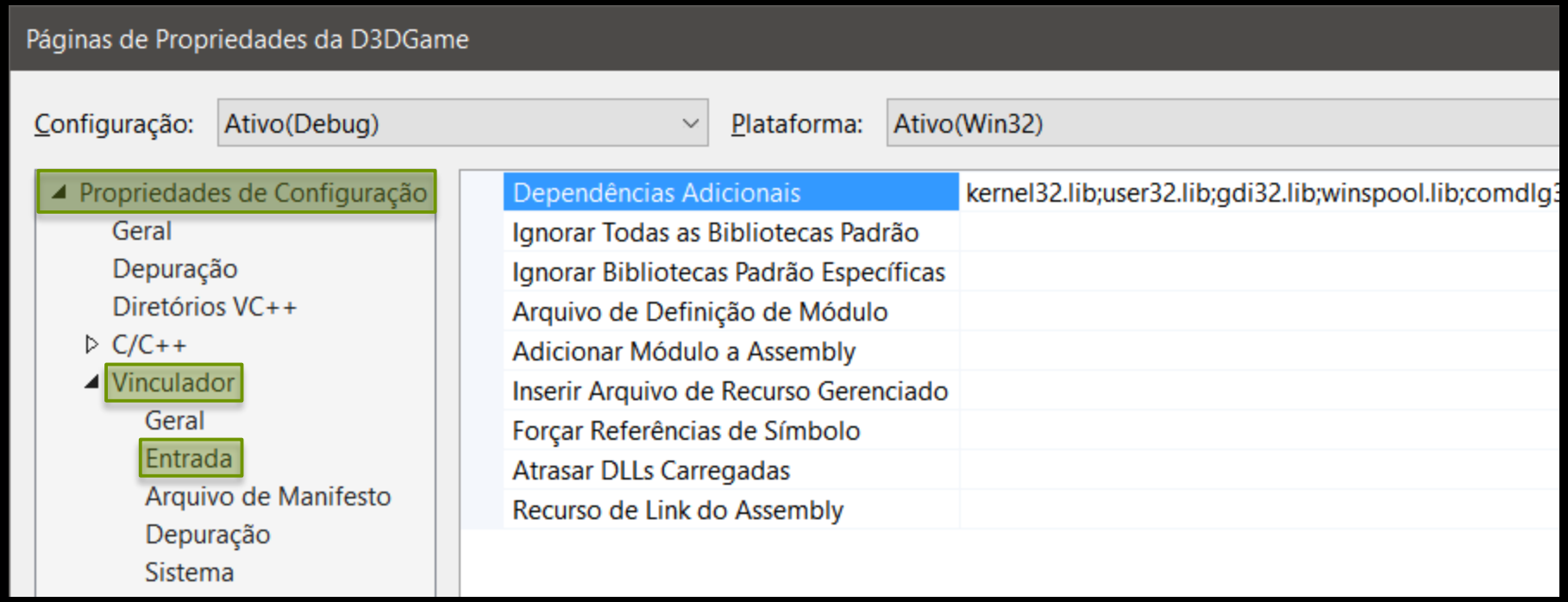
Configuração do Projeto

- Verificar o caminho de cabeçalhos e bibliotecas
  Propriedades de Configuração > Diretórios VC++



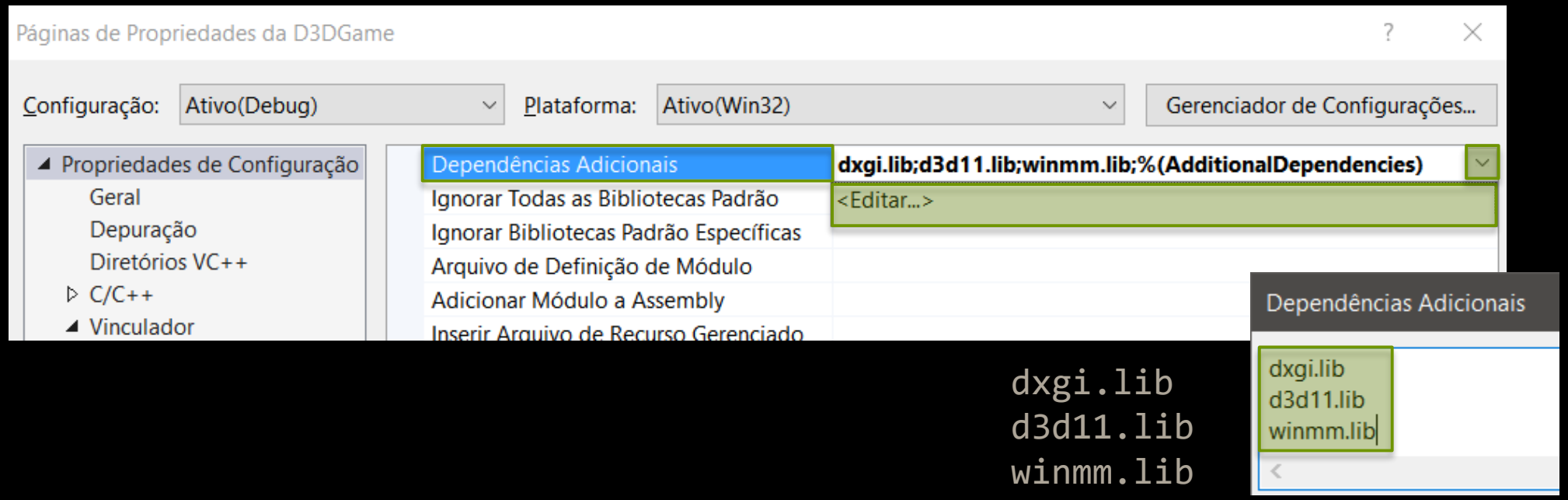
Configuração do Projeto

- Configurar o Ligador (Vinculador)
  Propriedades de Configuração > Vinculador > Entrada



Configuração do Projeto

- Adicionar dependências de bibliotecas
  Dependências Adicionais > Editar




```
dxgi.lib
d3d11.lib
winmm.lib
```

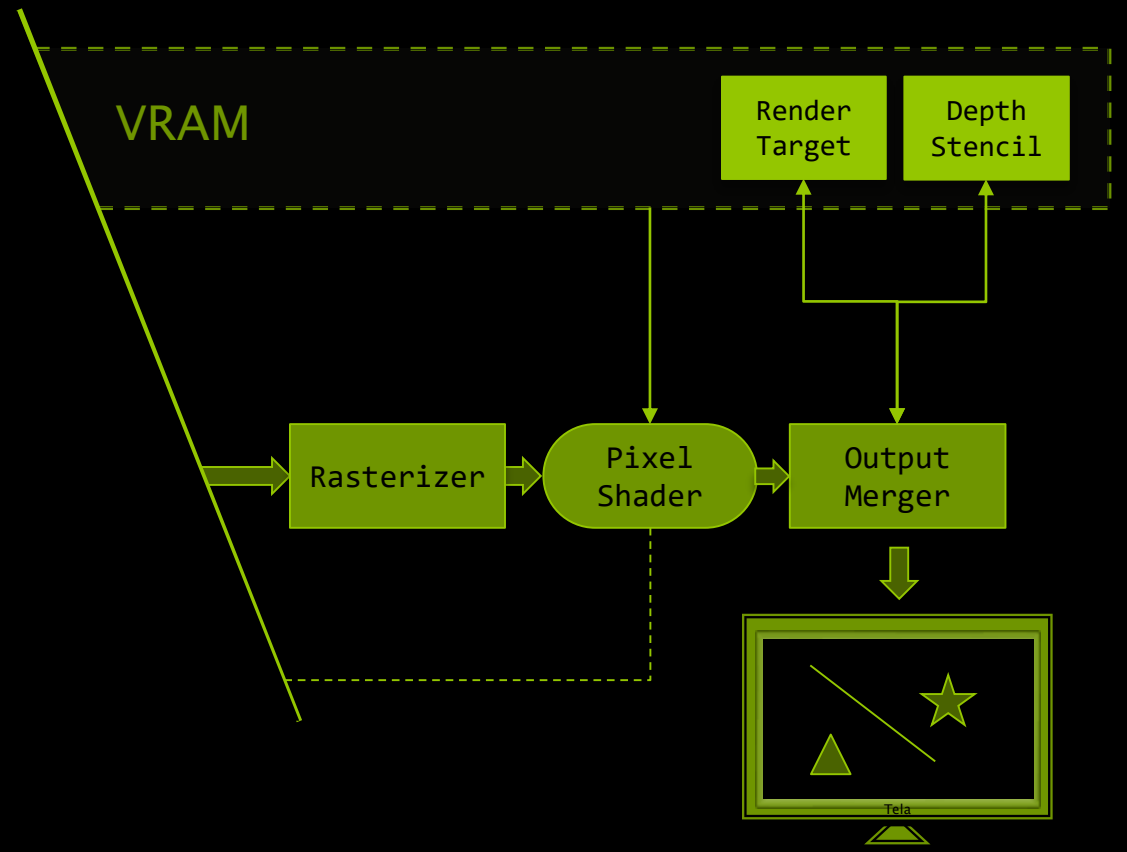
Inicialização do Direct3D

- O Direct3D é baseado em COM
  - Component Object Model (COM) é um padrão de interface binária para componentes de software introduzido pela Microsoft em 1993
    - Ele é usado para permitir a comunicação entre processos e a criação de objetos de forma independente da linguagem de programação
      - Utilizado em várias tecnologias:
        OLE, ActiveX, Windows Shell, DirectX, Windows Runtime, etc.

```
IDXGIDevice * dxgiDevice = nullptr;
d3dDev->QueryInterface(__uuidof(IDXGIDevice), (void**) &dxgiDevice);
...
dxgiDevice->Release();
```

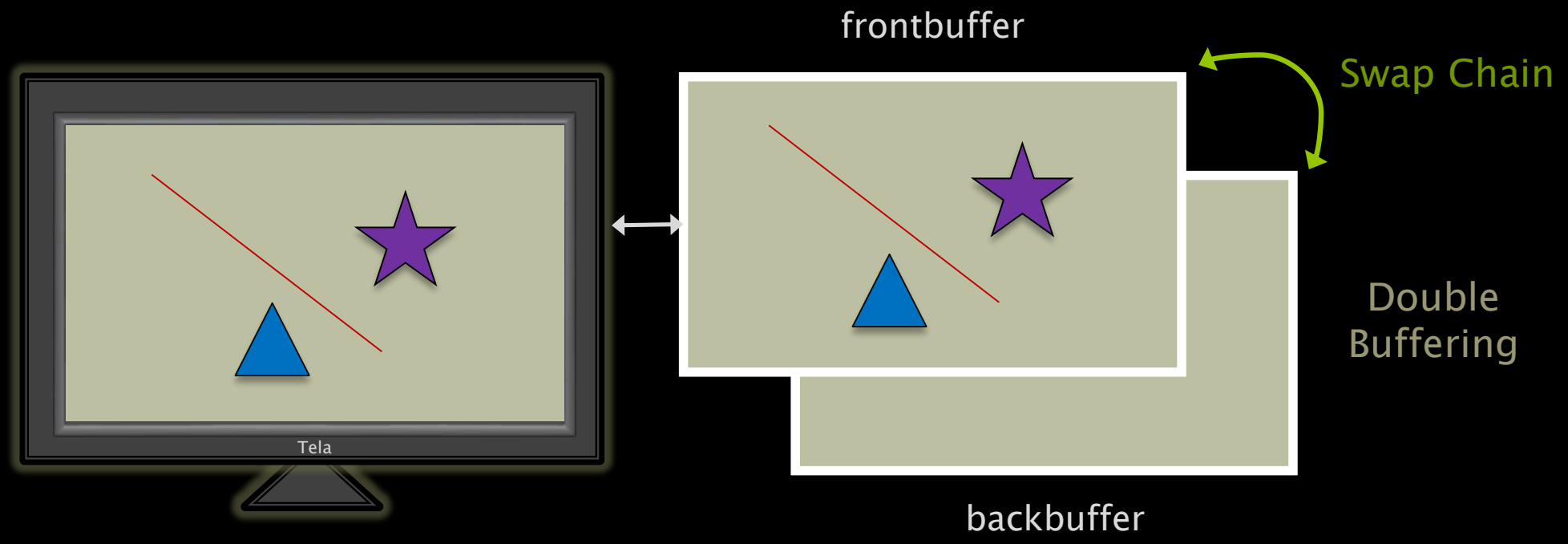
Inicialização do Direct3D

- Inicializar o Direct3D requer:
  - Criação de objetos:
    - Para o dispositivo Direct3D
    - Para uma Swap Chain
  - Configuração de:
    - Uma Render-Target View
    - Uma Viewport



Inicialização do Direct3D

- Os principais objetos são:
  - Dispositivo Direct3D: objeto com acesso ao dispositivo gráfico
  - Swap Chain: objeto que faz a troca entre superfícies de desenho



Inicialização do Direct3D

- Sem um **Buffer Duplo** obtém-se Screen Tearing

Quadro 1

Quadro 2

Inicialização do Direct3D

- Criação do dispositivo D3D

```
ID3D11Device        * d3dDev;         // dispositivo gráfico direct3D
ID3D11DeviceContext * d3dDevContext;  // contexto do dispositivo gráfico
D3D_FEATURE_LEVEL     featureLevel;   // nível de recursos D3D suportados pelo hardware

// cria objeto para o dispositivo gráfico
D3D11CreateDevice(
  NULL,                               // adaptador de vídeo (NULL = adaptador padrão)
  D3D_DRIVER_TYPE_HARDWARE,           // tipo de driver D3D (Hardware, Reference ou Software)
  NULL,                               // ponteiro para rasterizador em software
  D3D11_CREATE_DEVICE_DEBUG,          // modo de depuração ou modo normal
  NULL,                               // nível de recursos do D3D (NULL = maior suportado)
  0,                                  // tamanho do vetor de nível de recursos
  D3D11_SDK_VERSION,                  // versão do SDK do Direct3D
  &d3dDev,                            // guarda o dispositivo D3D criado
  &featureLevel,                      // nível de recursos do D3D utilizado
  &d3dDevContext);                    // contexto do dispositivo D3D
```

Inicialização do Direct3D

- Configuração da Swap Chain

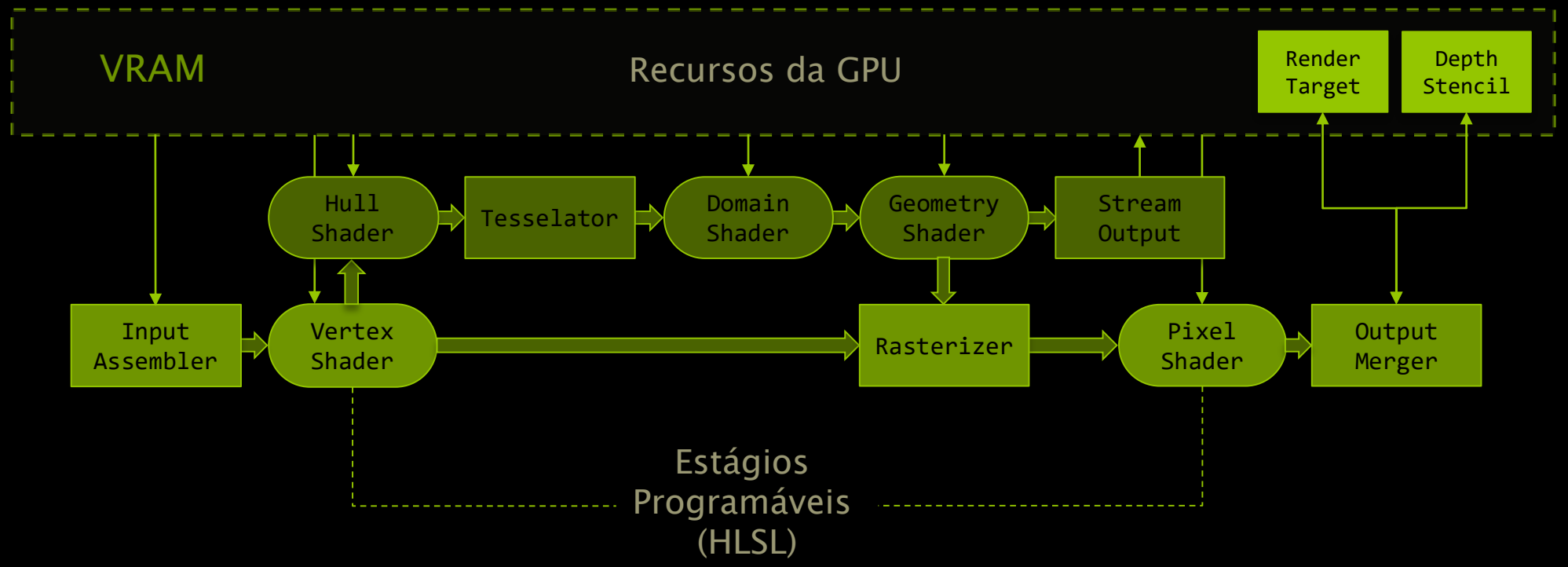
```
// descrição da swap chain
DXGI_SWAP_CHAIN_DESC swapDesc = {0};

swapDesc.BufferDesc.Width = window.Width();                 // largura do backbuffer
swapDesc.BufferDesc.Height = window.Height();               // altura do backbuffer
swapDesc.BufferDesc.RefreshRate.Numerator = 60;             // taxa de atualização em hertz
swapDesc.BufferDesc.RefreshRate.Denominator = 1;            // denominador da frequência
swapDesc.BufferDesc.Format = DXGI_FORMAT_R8G8B8A8_UNORM;    // formato de cores RGBA 32 bits
swapDesc.SampleDesc.Count = 1;                              // amostras por pixel (antialiasing)
swapDesc.SampleDesc.Quality = 0;                            // nível de qualidade da imagem
swapDesc.BufferUsage = DXGI_USAGE_RENDER_TARGET_OUTPUT;     // uso da superfície
swapDesc.BufferCount = 2;                                   // número de buffers (front + back)
swapDesc.OutputWindow = window.Id();                        // identificador da janela
swapDesc.Windowed = (window.Mode() != FULLSCREEN);          // modo em janela ou tela cheia
swapDesc.SwapEffect = DXGI_SWAP_EFFECT_FLIP_DISCARD;        // efeito da troca (descarte)
swapDesc.Flags = DXGI_SWAP_CHAIN_FLAG_ALLOW_MODE_SWITCH;    // tela cheia (backbuffer/desktop)
```

Inicialização do Direct3D

- Criação da Swap Chain

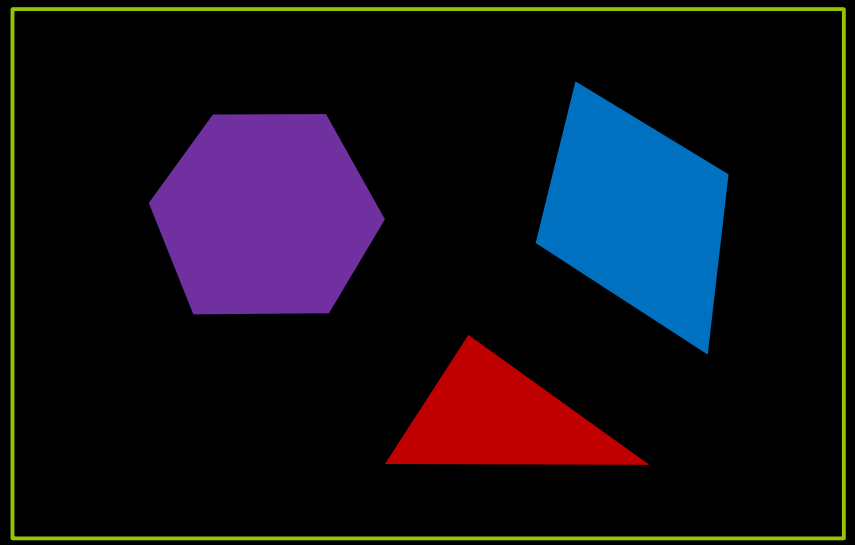
```
// pega um ponteiro para o dispositivo gráfico
IDXGIDevice * dxgiDevice = nullptr;
if FAILED(d3dDev->QueryInterface(__uuidof(IDXGIDevice), (void**) &dxgiDevice))
    return false;

// pega adaptador controlado pelo dispositivo gráfico
IDXGIAdapter * dxgiAdapter = nullptr;
if FAILED(dxgiDevice->GetParent(__uuidof(IDXGIAdapter), (void**) &dxgiAdapter))
    return false;

// pega um ponteiro para a DXGIFactory do adaptador
IDXGIFactory * dxgiFactory = nullptr;
if FAILED(dxgiAdapter->GetParent(__uuidof(IDXGIFactory), (void**) &dxgiFactory))
    return false;

// cria uma swap chain
if FAILED(dxgiFactory->CreateSwapChain(d3dDev, &swapDesc, &swapChain))
    return false;
```

Pipeline Direct3D



Render-Target

- Para que uma superfície possa ser alvo de desenhos no D3D deve-se criar uma Render-Target View
  - Uma View é o mecanismo usado para representar dados carregados na memória (recursos) que serão usados pelo hardware gráfico
  - Ela permite que os estágios do pipeline do D3D acessem apenas os dados necessários em cada tarefa



Superfície alvo

Render-Target

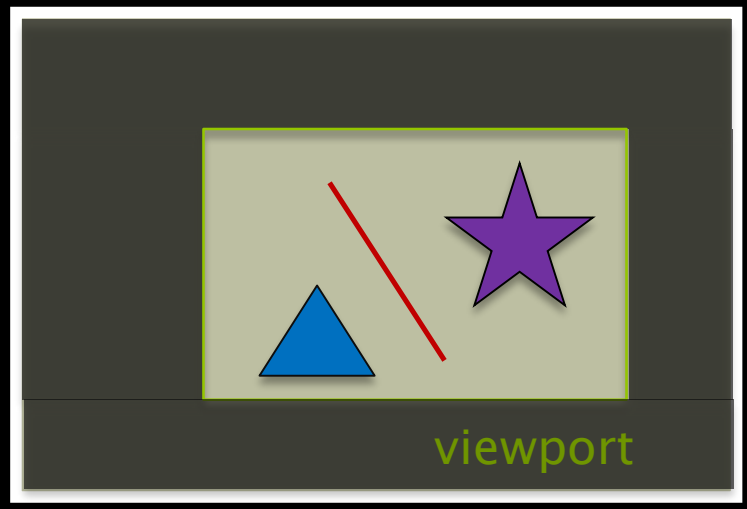
- Configuração da Render-Target View:

```
// pegando a superfície do backbuffer
ID3D11Texture2D * backBuffer;
swapChain->GetBuffer(
    0,                          // índice do buffer
    __uuidof(ID3D11Texture2D),  // tipo da interface usada para o buffer
    (void**) &backBuffer);      // superfície do backbuffer

// cria uma render-target view
ID3D11RenderTargetView * renderTargetView;
d3dDev->CreateRenderTargetView(
    backBuffer,                 // superfície a ser utilizada
    NULL,                       // acessa recursos em mipmap level 0
    &renderTargetView);         // render target view

// liga a render-target view ao estágio output-merger
d3dDev->OMSetRenderTargets(1, &renderTargetView, nullptr);
```

Viewport

- Definindo a exibição do conteúdo com uma viewport

```
// configura uma viewport
D3D11_VIEWPORT viewport;

viewport.TopLeftX = 0;
viewport.TopLeftY = 0;
viewport.Width    = window.Width();
viewport.Height   = window.Height();
viewport.MinDepth = 0.0f;
viewport.MaxDepth = 1.0f;
```



- Ligando a viewport ao estágio de rasterização

```
d3dDev->RSSetViewports(1, &viewport);
```

Finalização do Direct3D

- Liberando memória alocada:

```
if (renderTargetView) {
    renderTargetView->Release();          // libera a render-target view
    renderTargetView = nullptr;
}

if (swapChain) {
    swapChain->Release();                 // libera a swap chain
    swapChain = nullptr;
}

if (d3dDev) {
    d3dDev->Release();                    // libera o dispositivo gráfico
    d3dDev = nullptr;
}
```

Resumo

- O Direct3D é a API gráfica do DirectX voltada para o desenvolvimento de jogos
  - Um dispositivo Direct3D é usado com uma Swap Chain:
    - Frontbuffer (memória principal de exibição)
    - Backbuffer (memória auxiliar de desenho)
  - O backbuffer precisa ser conectado ao pipeline do D3D
    - Uma Render-Target define o backbuffer como “alvo de desenhos”
    - Uma Viewport delimita a área visível no backbuffer