Controlador de Jogo

Programação de Jogos

Judson Santos Santiago

Introdução

- Um jogo é interativo por natureza
  - Sem interação ele seria apenas uma brincadeira, uma estória ou filme
- Os dispositivos de interação variam de acordo com o tipo de jogo e a plataforma para a qual ele foi desenvolvido
  - Os dispositivos mais comuns são:
    - Teclado
    - Mouse
    - Controle

Introdução

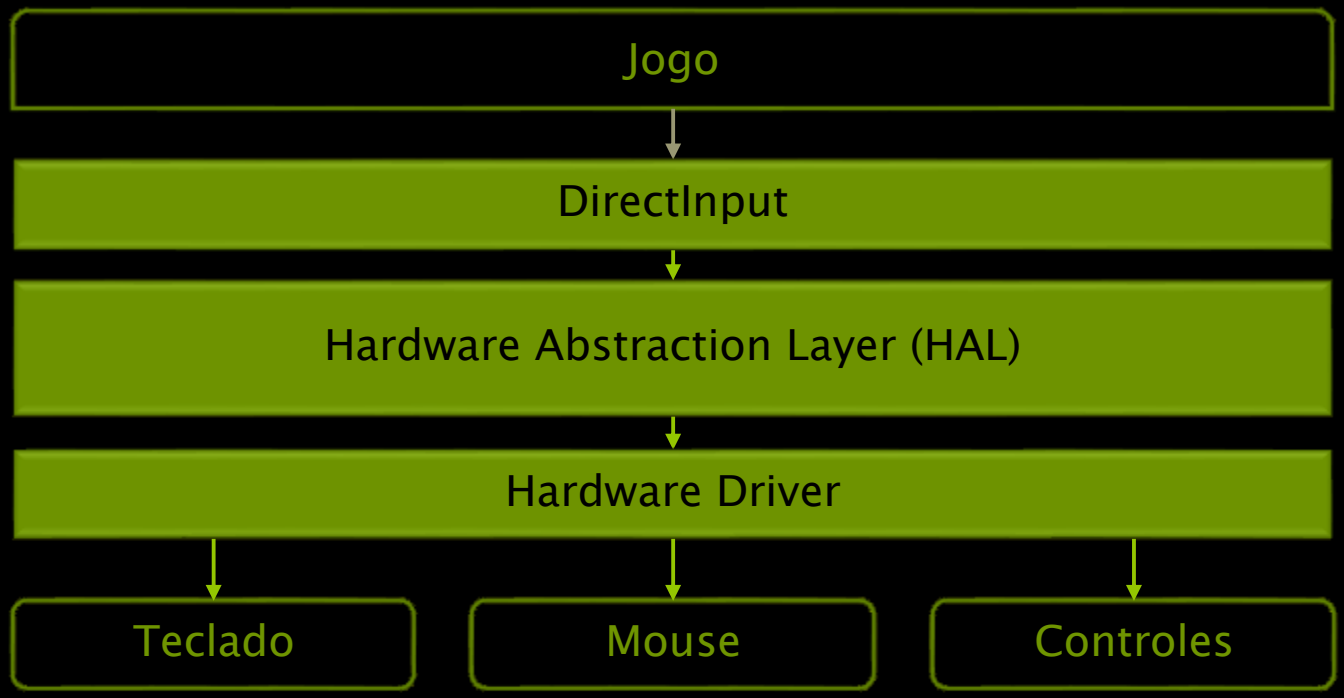
- A API do Windows fornece mecanismos para o tratamento do teclado e do mouse

```
// faz o tratamento das mensagens do windows
LRESULT CALLBACK WinProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch(msg)
    {
        case WM_KEYDOWN:                // tecla pressionada
        ...
```

- Para o tratamento do controle existem duas soluções:
  - DirectInput: trata todo tipo de dispositivo de entrada*
  - Xinput: trabalha com a família de controles do Xbox

DirectInput

- O DirectInput fornece uma interface única de acesso



DirectInput

- Simplifica a programação abstraindo os dispositivos
  - Do contrário, seria necessário suportar individualmente os dispositivos de cada fabricante
- Suporta qualquer dispositivo que possua um driver para Windows®
  - Teclados e Mouses
  - Controles
  - Volantes
  - Manches
  - Etc.

DirectInput

- Para o DirectInput todos os dispositivos são iguais
  - Todos possuem um certo número de objetos de interação

Manche

Controles

Volante

DirectInput

- Principais tipos de objetos de interação:
  - Eixos
    possuem muitas posições
    (valores contínuos)
    char ou short
  - Botões
    possuem apenas duas posições
    (pressionado ou liberado)
    booleano

DirectInput

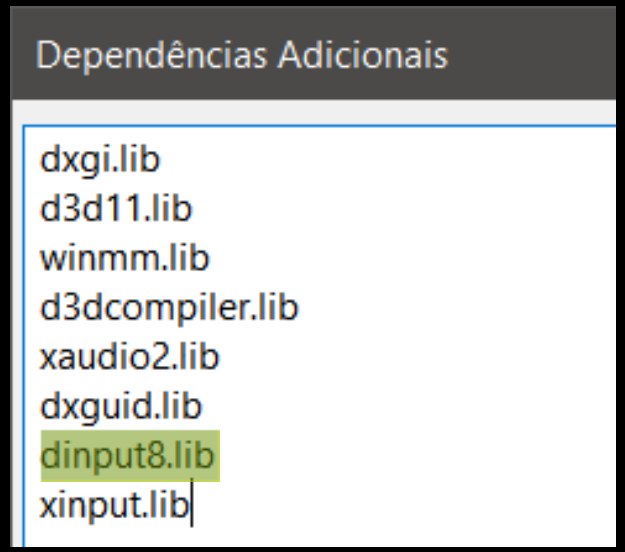
- Para ler um dispositivo é preciso:

1. Inicializar o DirectInput
2. Verificar os dispositivos conectados
3. Inicializar um dispositivo conectado
   i. Ajustar o nível de cooperação
   ii. Selecionar o formato de dados
   iii. Ajustar a faixa numérica dos eixos
   iv. Ajustar o tamanho da zona morta
4. Requisitar acesso ao dispositivo
5. Ler o estado do dispositivo

O estado do dispositivo é o valor de cada eixo e botão

Configuração

- A última versão da biblioteca é o DirectInput8
  - O sistema não sofre alterações há vários anos
  - A utilização da biblioteca requer a adição do arquivo dinput8.lib na lista de dependências do "vinculador" do Visual Studio



Configuração:

```
Projeto >
Propriedades >
Propriedades de Configuração >
Vinculador >
Entrada >
Dependências Adicionais
```

Inicialização

- É preciso incluir o arquivo de cabeçalho:

```
#include <dinput.h>
```

- E criar o objeto DirectInput:

```
LPDIRECTINPUT8 dinput = nullptr;

DirectInput8Create(
     GetModuleHandle(NULL),     // identificador da aplicação
     DIRECTINPUT_VERSION,       // número da versão do direct input
     IID_IDirectInput8,         // identificador da interface COM
     (void **) &dinput,         // objeto directinput
     NULL);                     // sempre nulo
```

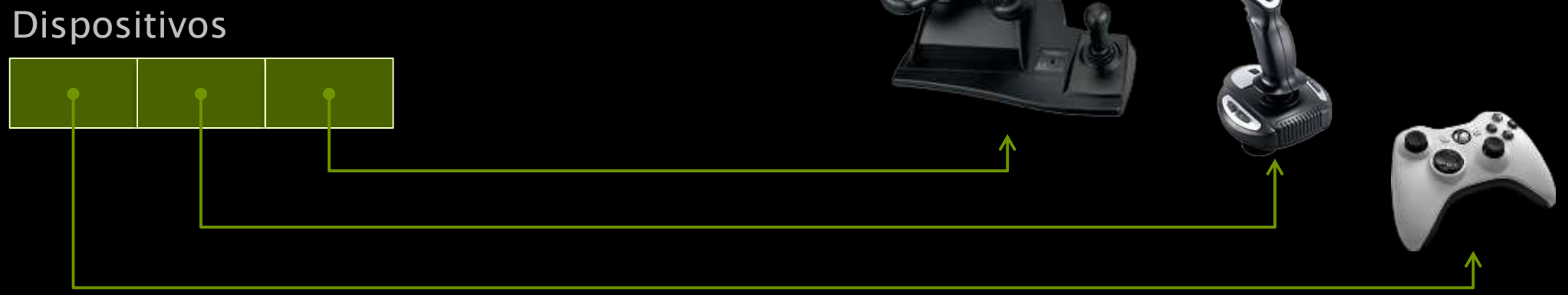
Enumeração de Dispositivos

- Cada dispositivo tem um identificador global único
  - GUID (Global Unique Identifier)
  - Teclado e mouse possuem identificadores genéricos (`GUID_SysKeyboard` e `GUID_SysMouse`)
  - Os demais dispositivos precisam ser consultados

Cada dispositivo tem um GUID

Enumeração de Dispositivos

- O DirectInput obtém os identificadores através da enumeração dos dispositivos
  - A enumeração usa uma função tipo CALLBACK:
    - É chamada por um programa externo
    - O DirectInput a chama cada vez que encontra um novo dispositivo



Enumeração de Dispositivos

- DirectInput8::EnumDevices() configura a enumeração

```
// configura enumeração de dispositivos
HRESULT EnumDevices(
     DWORD dwDevType,              // tipo de dispositivo a procurar
     LPDIENUMCALLBACK lpCallback,  // ponteiro para a função callback
     LPVOID pvRef,                 // valor de 32 bits
     DWORD dwFlags);               // tipo de busca
```

Tipos de Dispositivos	Descrição
DI8DEVCLASS_POINTER	Mouse ou Trackball
DI8DEVCLASS_KEYBOARD	Teclado
DI8DEVCLASS_GAMECTRL	Gamepad, Volante, Manche
DI8DEVCLASS_DEVICE	Diferente dos anteriores
DI8DEVCLASS_ALL	Todos

Enumeração de Dispositivos

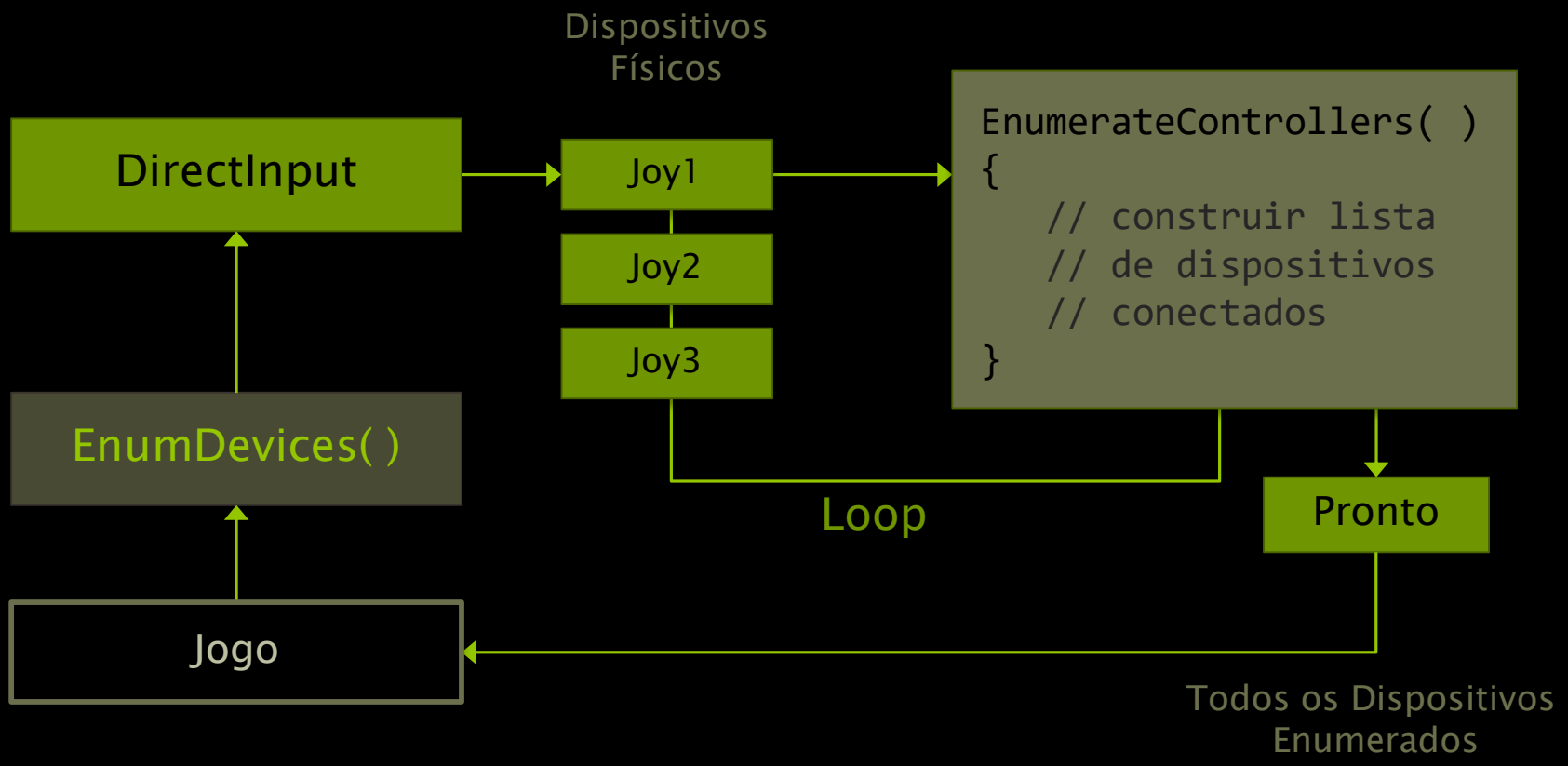
- O tipo de busca é controlado por uma constante:

Tipos de Busca	Descrição
DIEDFL_ALLDEVICES	Todos os dispositivos
DIEDFL_ATTACHEDONLY	Apenas os conectados
DIEDFL_FORCEFEEDBACK	Apenas ForceFeedback

```
// configura enumeração de dispositivos
dinput->EnumDevices(
     DI8DEVCLASS_GAMECTRL,    // procure apenas controles
     EnumerateControllers,    // função de enumeração
     &controllers,            // endereço da lista de controles
     DIEDFL_ATTACHEDONLY);    // apenas dispositivos conectados
```

Enumeração de Dispositivos

- A enumeração chama a função CALLBACK



Enumeração de Dispositivos

- A função CALLBACK a ser usada com EnumDevices deve ter o seguinte protótipo:

```
// função callback para enumeração
BOOL CALLBACK EnumerateControllers(
     LPDIDEVICEINSTANCE pdDDi,    // info do dispositivo encontrado
     LPVOID data);                // valor de 32 bits
```

- A função deve retornar um valor booleano indicando se o DirectInput deve ou não parar a busca

Constante	Descrição
DIENUM_CONTINUE	Continue a enumeração
DIENUM_STOP	Pare a enumeração

Enumeração de Dispositivos

- A função CALLBACK abaixo insere os dispositivos encontrados em uma lista:

```
BOOL CALLBACK EnumerateControllers(LPCDIDEVICEINSTANCE lpDDi, LPVOID data)
{
    // registro que guarda nome e GUID do controle
    JoyInfo joy;

    // copia o identificador e nome do dispositivo
    joy.guid = lpDDi->guidInstance;
    joy.name = lpDDi->tszInstanceName;

    // coloca controle na lista
    ((list<JoyInfo>*) data)->push_back(joy);

    // continua enumeração até o fim
    return DIENUM_CONTINUE;
}
```

Inicialização do Controle

- Com o identificador, podemos inicializar o controle:

```
LPDIRECTINPUTDEVICE8 joyDev = nullptr;

// cria o dispositivo
CreateDevice(joy.guid,                  // id do controle
             &joyDev,                   // objeto que representará o controle
             nullptr);                  // sempre nulo
```

- Ajustando o nível de cooperação

```
// ajusta o nível de cooperação para o controle
joyDev->SetCooperativeLevel(
       window->Id(),            // identificador da janela
       DISCL_BACKGROUND |       // pode ser consultado em background
       DISCL_NONEXCLUSIVE);     // acesso não exclusivo
```

Inicialização do Controle

- Selecionando o formato de dados:

```
// seleciona o formato de dados para o controle
joyDev->SetDataFormat(&c_dfDIJoystick);
```

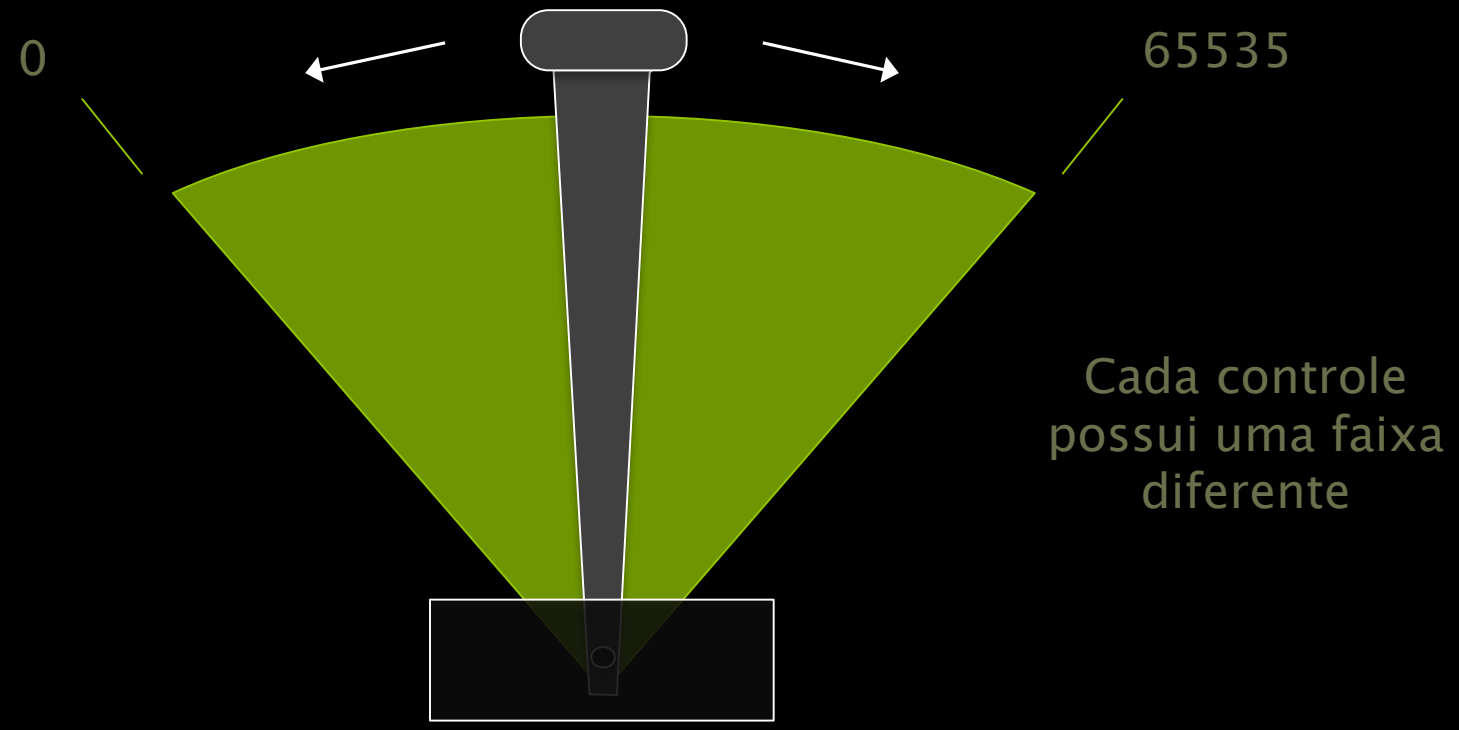
- O formato de dados escolhido é descrito pelo registro:

```
// registro descrevendo o estado do controle
struct DIJOYSTATE
{
    LONG lX, lY, lZ;     // eixos x, y e z do controle
    LONG lRx, lRy, lRz;  // eixos de rotação rx, ry e rz
    LONG rglSlider[2];   // controles tipo slider
    DWORD rgdwPOV[4];    // controles tipo Point Of View (D-Pad)
    BYTE rgbButtons[32]; // botões do controle

};
```

Ajustando as Propriedades

- Uma alavanca analógica gera um valor em uma faixa



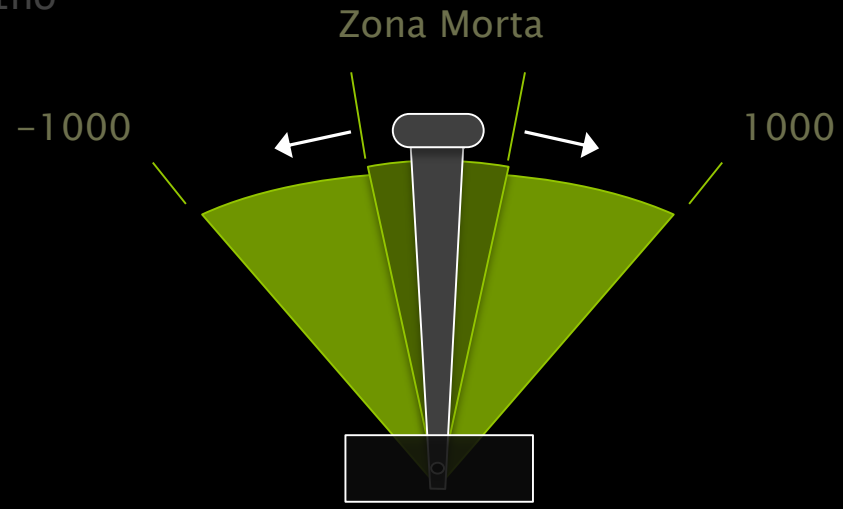
Cada controle possui uma faixa diferente

Ajustando as Propriedades

- Para que o jogo possa trabalhar com uma única faixa, independente do dispositivo, é preciso ajustar as propriedades do controle

```
HRESULT SetProperty(
    REFGUID rguidProp          // id da propriedade
    LPCDIPROPHEADER pdiph);    // endereço do cabeçalho
```

  - Principais Propriedades:
    - Faixa de movimento
    - Zona morta



Ajustando as Propriedades

- A faixa de movimento é ajustada através do registro DIPROPRANGE

```
struct DIPROPRANGE
{
    DIPROPHEADER diph;   // cabeçalho da propriedade
    LONG         lMin;   // valor mínimo da faixa
    LONG         lMax;   // valor máximo da faixa
};

struct DIPROPHEADER
{
    DWORD dwSize;        // tamanho da propriedade
    DWORD dwHeaderSize;  // tamanho do cabeçalho da propriedade
    DWORD dwObj;         // objeto cuja propriedade será modificada
    DWORD dwHow;         // como acessar o objeto
};
```

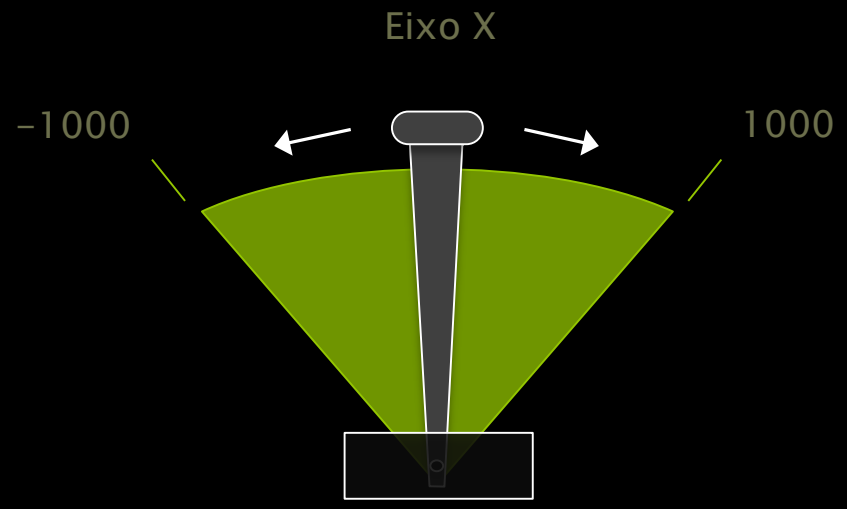
Ajustando as Propriedades

- O código abaixo configura o movimento do eixo para a faixa [-1000, 1000]

```
DIPROPRANGE axisRange;

axisRange.diph.dwSize       = sizeof(DIPROPRANGE);
axisRange.diph.dwHearderSize = sizeof(DIPROPHEADER);
axisRange.diph.dwHow        = DIPH_BYID;
axisRange.diph.dwObj        = GUID_XAxis;
axisRange.lMin              = -1000;
axisRange.lMax              = +1000;


joyDev->SetProperty(DIPROP_RANGE, &axisRange.diph);
```



Ajustando as Propriedades

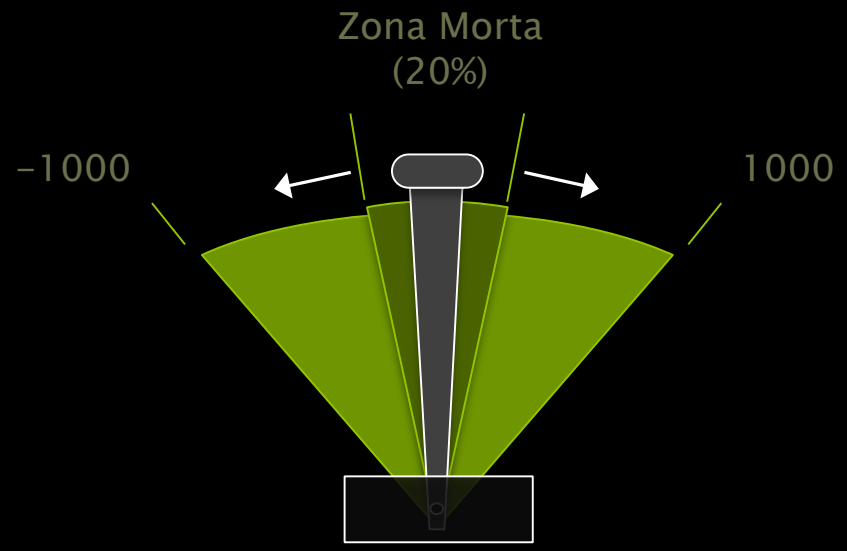
- A zona morta é configurada com DIPROPWORD

```
struct DIPROPDWORD {
    DIPROPHEADER diph;
    DWORD        dwData;
};
```

```
DIPROPWORD deadZone;

deadZone.diph.dwSize        = sizeof(DIPROPWORD);
deadZone.diph.dwHearderSize = sizeof(DIPROPHEADER);
deadZone.diph.dwHow         = DIPH_BYID;
deadZone.diph.dwObj         = GUID_XAxis;
deadZone.dwData             = 2000;

joyDev->SetProperty(DIPROP_DEADZONE, &deadZone.diph);
```



Lendo o Dispositivo

- Acessando e Liberando o dispositivo

```
// requisita acesso ao controle
joyDev->Acquire();

// libera acesso antes de encerrar o programa
joyDev->Unacquire();
```

- Lendo o estado do controle

```
// armazena estado do controle
DIJOYSTATE joyState;

// lê estado atual do controle
joyDev->GetDeviceState(sizeof(DIJOYSTATE), (LPVOID) &joyState);
```

Resumo

- O DirectInput é capaz de ler qualquer controlador de jogo reconhecido pelo Windows®
  - Todo controle é composto por:
    - Botões
    - Eixos
- O jogo deve verificar os dispositivos conectados
  - Usando uma função CALLBACK
- As faixas de valores e a zona morta dos eixos podem ser ajustadas individualmente para cada controle