Distribuição do Jogo

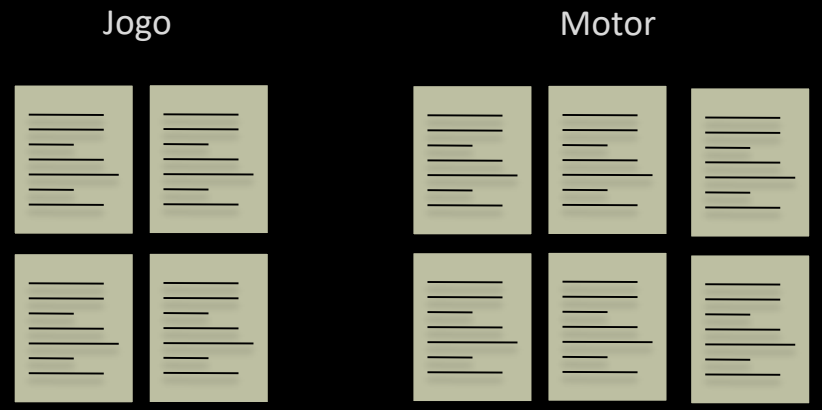
Programação de Jogos

Judson Santos Santiago

Introdução

- Os jogos são constituídos por vários arquivos
  - Código fonte (.cpp)
  - Inclusão (.h)
  - Recursos
    - Imagens (.png, .jpg)
    - Sons (.wav)
    - Scripts (.txt)
    - Ícone (.ico)
    - Cursor (.cur)
    - Etc.

Muitos desses arquivos contém o código do motor

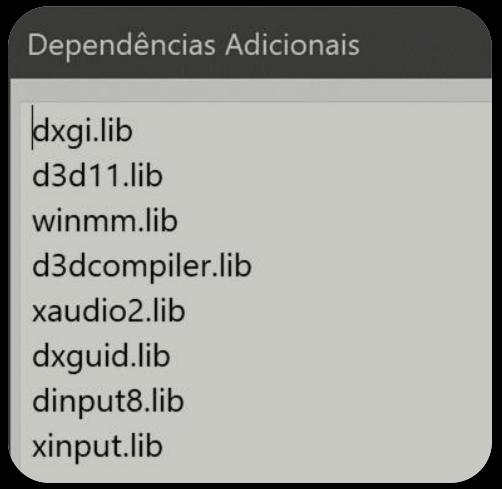


Introdução

- O motor pode ser mantido de forma independente
  - Em um projeto diferente do jogo
  - Encapsulado em uma biblioteca

    Ex.: `engine.lib`, `engine.dll`

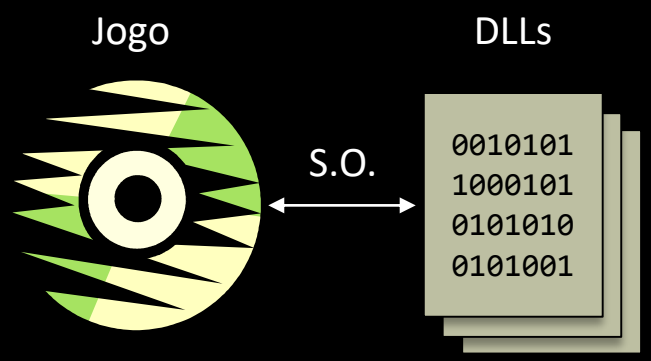
Da mesma forma que o motor usa as bibliotecas do DirectX, o jogo poderia usar as bibliotecas do motor

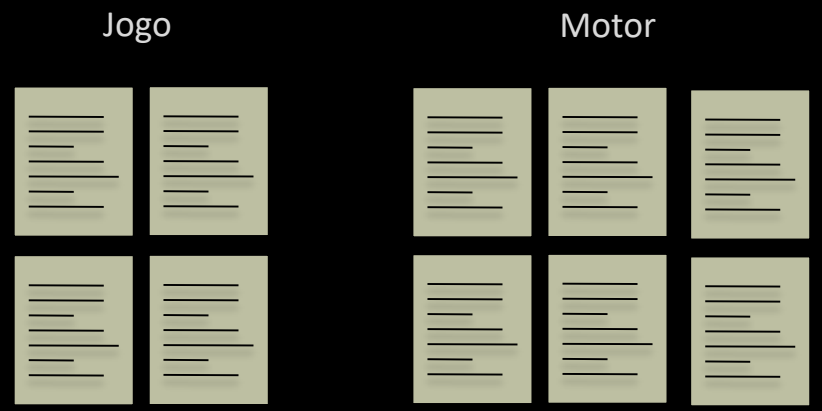


Configuração do Ligador

Introdução

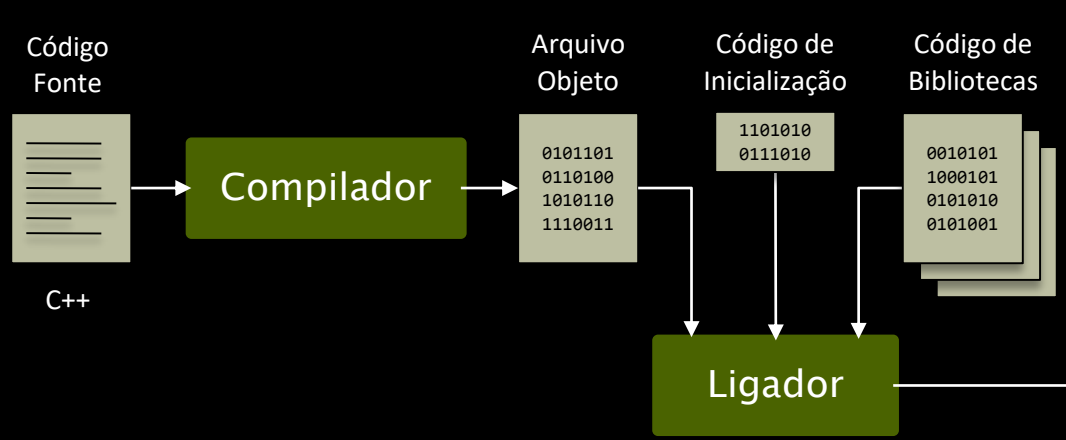
- Manter o motor em uma biblioteca possui benefícios
  - Impede o acesso e a alteração do código fonte
  - Reduz o tamanho do executável do jogo
  - Facilita o processo de atualização



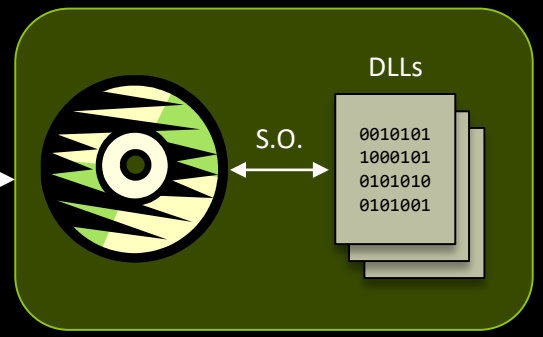


Criando uma Biblioteca

- Dinamic Link Library (DLL) é a implementação da Microsoft do conceito de biblioteca dinâmica



As DLLs são ligadas ao código pelo S.O. em tempo de execução



Programa Executável

Criando uma Biblioteca

- Um programa C++ típico é composto por:
  - Arquivo principal (.cpp): contém a função main
  - Arquivos de cabeçalho (.h): contém declarações
  - Arquivos fonte (.cpp): contém implementações

principal.cpp

```
#include <iostream>
#include "calc.h"
using namespace std;
int main()
{
    float a;
    a = media(8,10);
    cout << a << endl;
    return 0;
}
```

calc.h

```
float media(float, float);
float soma(float, float);
```

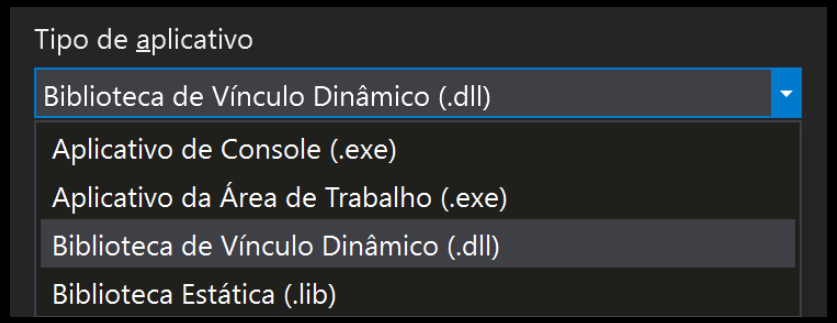
calc.cpp

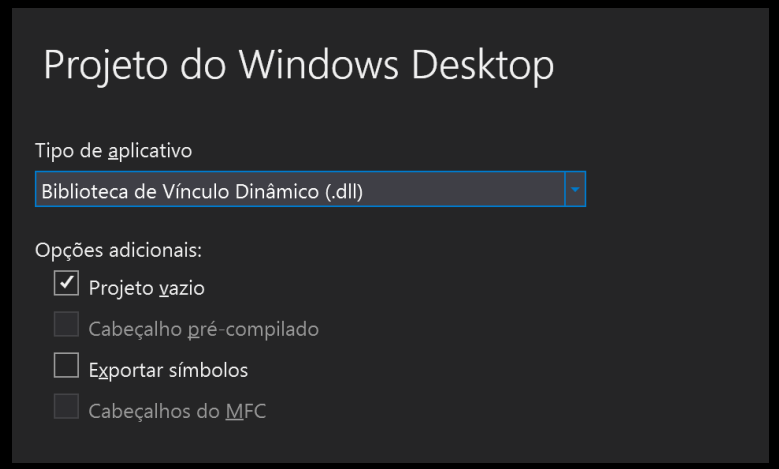
```
float media(float x, float y)
{
    return (x+y)/2;
}

float soma(float x, float y)
{
    return x+y;
}
```

Criando uma Biblioteca

- Para transformar um programa C++ em uma DLL:
  - Crie um novo projeto no Visual Studio

```
Assistente do Windows Desktop >
Biblioteca de Vínculo Dinâmico >
Projeto Vazio
```





Criando uma Biblioteca

- Para transformar um programa C++ em uma DLL:
  - Adicione apenas os arquivos que fazem parte da biblioteca (não adicione o arquivo principal)
  - Inclua no topo dos arquivos .h as seguintes linhas, substituindo <PROJETO> pelo nome do seu projeto:

```
#ifdef <PROJETO>_EXPORTS
#define DLL __declspec( dllexport )
#else
#define DLL __declspec( dllimport )
#endif
```

Criando uma Biblioteca

- Modifique as declarações das funções nos arquivos .h, acrescentando DLL antes do tipo das funções:

```
DLL float soma(float,float);
DLL float media(float,float);
```

- Modifique as declarações das classes nos arquivos .h, acrescentando DLL antes do nome da classe:

```
class DLL Engine
{
public:
    Engine();
    ~Engine();
};
```

Criando uma Biblioteca

- Em seguida, basta compilar o programa que a DLL será criada em uma das seguintes pastas:

```
"..\<PROJETO>\x86\Debug\"
"..\<PROJETO>\x86\Release\"
"..\<PROJETO>\x64\Debug\"
"..\<PROJETO>\x64\Release\"
```

A pasta depende da plataforma e da configuração utilizada na compilação

  - Juntamente com o arquivo .dll é criado também:
    - Um Arquivo de biblioteca (.lib)
      (liga um executável à DLL)
    - Arquivos de shaders (.cso)
      (pixel e vertex shaders do Direct3D)

Criando uma Biblioteca

- Para **distribuir a DLL** é necessário disponibilizar os arquivo .lib e .dll gerados na compilação, além dos arquivos de cabeçalho .h usados no projeto da DLL

```
+-- Bin\
|   +-- Debug\Engine.dll
|   +-- Release\Engine.dll
+-- Include\
|   +- Animation.h
|   +- Audio.h
|   +- ...
+-- Lib\
|   +- Engine.lib
+-- Shaders\
|   +- Pixel.cso
|   +- Vertex.cso
```

Tipicamente os desenvolvedores distribuem estes arquivos em uma estrutura de pastas

Usando uma DLL

- Ao criar um projeto do tipo DLL, o Visual Studio acrescenta a definição da constante `<PROJETO>_EXPORTS`
  - As declarações recebem a classe de armazenamento:

    ```
    __declspec ( dllexport )
    ```

  - Ao criar um projeto do tipo Aplicativo da Área de Trabalho, esta constante não é definida e portanto as classes e funções receberão a classe de armazenamento:

    ```
    __declspec ( dllimport )
    ```

Usando uma DLL

- Para **usar a DLL** basta criar um projeto do tipo usual da sua aplicação e indicar ao ambiente de programação onde procurar pelos arquivos .h, .lib e .dll
  - No Visual Studio vá em:
    - Projeto > Propriedades > Propriedades de Configuração >
      - Diretórios VC++
        - Edite a opção “Diretórios de Inclusão” para os .h
        - Edite a opção “Diretórios de Biblioteca” para os .lib
      - Depuração > Ambiente
        - Adicione `PATH=<Caminho para a DLL>`

As Bibliotecas do C++

- Por padrão, as funções da biblioteca do C++ não são incluídas no código executável, elas ficam em DLLs
  - As aplicações criadas no Visual Studio geram uma mensagem de erro indicando a falta de uma DLL quando executadas em máquinas que são tem o Visual Studio instalado
- Para contornar esse problemas existem 3 opções:
  - Copiar as DLLs do C++ e distribuir junto
  - Instalar o Microsoft Visual C++ Redistributable
  - Configurar o seu projeto para incluir as funções da biblioteca padrão do C++ no executável

As Bibliotecas do C++

- Para configurar o projeto da sua aplicação para incluir as bibliotecas do C++ no código executável
  - No Visual Studio vá em:
    - Projeto > Propriedades > Propriedades de Configuração > C/C++ > Geração de Código
  - Mude a opção:
    - Biblioteca em Tempo de Execução (Runtime Library):
      - De "Multi-threaded Debug DLL" para "Multi-threaded Debug" (Debug)
      - De "Multi-threaded DLL" para "Multi-threaded" (Release)

Distribuição do Jogo

- Uma vez gerado o executável do jogo, é preciso organizar uma pasta com todos os arquivos necessários:

```
+-- GeometryWars\
|   +-- Shaders\
|   |   +- Pixel.cso
|   |   +- Vertex.cso
|   +-- Resources\
|   |   +- Ico.ico
|   |   +- Cursor.cur
|   |   +- ...
|   +-- GeoWars.exe
|   +-- Engine.dll
```

A localização dos Shaders dependem da implementação do Renderer.cpp no motor do jogo:

```
D3DReadFileToBlob("Shaders/Vertex.cso", &vShader);
D3DReadFileToBlob("Shaders/Pixel.cso", &pShader);
```

Resumo

- Os jogos usam vários componentes dos motores
  - O motor de jogos normalmente é encapsulado em uma DLL e distribuído para os programadores do jogo
    - Impede o acesso e a alteração do código fonte
    - Reduz o tamanho do executável do jogo
    - Facilita o processo de atualização
  - O executável do jogo deve ser distribuído juntamente com a DLL do motor