UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE JOGOS

LABORATÓRIO 4

ARQUITETURA DE UM JOGO

EXERCÍCIOS DE APRENDIZAGEM

FAÇA OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. Considerando a arquitetura de um jogo apresentada na aula, utilize as funções GameInit, GameUpdate, GameDraw e GameFinalize para plotar pixels em posições aleatórias da janela.

   Para isso utilize a função SetPixel da biblioteca do Windows:

```
COLORREF SetPixel(HDC hdc, int X, int Y, COLORREF crColor);
```

   O dispositivo pode ser obtido com GetDC e liberado com ReleaseDC:

```
HDC hdc = GetDC(hwnd);
// utiliza dispositivo gráfico aqui
ReleaseDC(hwnd, hdc);
```

   Uma cor do tipo COLORREF pode ser obtida com a macro RGB:

```
COLORREF color = RGB(255,255,255);
As cores R(Red), G(Green) e B(Blue) variam de 0 a 255.
Uma cor RGB(0,0,0) é a cor preta e RGB(255,255,255) é a cor branca.
```

   A geração de números aleatórios pode ser feita com random:

```
#include <iostream>
#include <random>
using namespace std;

int main()
{
        random_device rd;
        mt19937 mt(rd());
        uniform_int_distribution<int> dist(0, 99);

        for (int i = 0; i < 10; ++i)
                cout << dist(mt) << " ";
        cout << endl;
        return 0;
}
```

2. Considerando a arquitetura de um jogo apresentada na aula, utilize as funções GameInit, GameUpdate, GameDraw e GameFinalize para desenhar e movimentar um bitmap dentro da janela.

Para isso utilize a função BitBlt da biblioteca do Windows:

```
BOOL BitBlt(
       HDC hdcDest,          // contexto do dispositivo de destino
       int nXDest,           // coordenada x do destino
       int nYDest,           // coordenada y do destino
       int nWidth,           // largura do bitmap
       int nHeight,          // altura do bitmap
       HDC hdcSrc,           // contexto do dispositivo fonte
       int nXSrc,            // coordenada x da fonte
       int nYSrc,            // coordenada y da fonte
       DWORD dwRop           // combinação das cores da fonte e destino
);
```

Carregue a imagem em um contexto de dispositivo próprio:

```
// variáveis
HBITMAP image;
BITMAP bm;
HDC hdc;
HDC hdcImg;

// carrega a imagem bitmap
image = (HBITMAP) LoadImage(
           NULL,                          // nulo para bitmaps
           "Resources\\CarKara.bmp",      // localização
           IMAGE_BITMAP,                  // tipo do recurso
           0,                             // largura da imagem
           0,                             // altura da imagem
           LR_LOADFROMFILE);              // tipo de carregamento

// lê as propriedades do bitmap
GetObject(image, sizeof(BITMAP), &bm);

// cria um contexto de dispositivo para o bitmap
hdc = GetDC(hwnd);
hdcImg = CreateCompatibleDC(hdc);
SelectObject(hdcImg, image);
```

Desenhe o bitmap:

```
// limpa a área cliente
RECT rect;
GetClientRect(hwnd, &rect);
FillRect(hdc, &rect, CreateSolidBrush(RGB(0,0,0)));

// desenha o bitmap
BitBlt(hdc, 0, 0, bm.bmWidth, bm.bmHeight, hdcImg, 0, 0, SRCCOPY);
```

Apague o bitmap e libere os contextos de dispositivo:

```
// apaga o contexto do dispositivo e o bitmap
DeleteDC(hdcImg);
DeleteObject((HBITMAP) image);
ReleaseDC(hwnd, hdc);
```