|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТИ |
| **Кафедра** | ПМИИ |

|  |  |
| --- | --- |
|  |  |
| **Дисциплина: «Системное программирование»**  **Отчет по лабораторной работе №7**  **«Битовые карты и метафайлы»**  **Выполнил: студент группы А-13а-19**  **Кокляева М. О.**  **Преподаватель: Меньшикова К. Г.**  **Москва, 2024** | |

**1. Краткое описание функций, параметров и используемых событий.**

1) Работа с битовыми картами в WinAPI.

* **Дескриптор HBITMAP**

Дескриптор растрового изображения.

* **CreateCompatibleBitmap**

**HBITMAP CreateCompatibleBitmap (HDC** *hdc***, int** *nWidth***, int** *nHeight***);**

Функция **CreateCompatibleBitmap** создает точечный рисунок, совместимый с устройством, которое связано с заданным контекстом устройства.

Параметры: *hdc* – дескриптор контекста устройства; *nWidth* – определяет ширину точечного рисунка, в пикселях; *nHeight* – определяет высоту точечного рисунка, в пикселях.

Если функция завершается успешно, возвращаемое значение - дескриптор совместимого точечного рисунка (аппаратно-зависимая точечная картинка (DDB)). Если функция завершается ошибкой, возвращаемое значение - ПУСТО (NULL).

* **CreateCompatibleDC**

**HDC CreateCompatibleDC(HDC***hdc***);**

Функция CreateCompatibleDC создает контекст устройства в памяти (DC), совместимый с заданным устройством.

Параметр: *hdc* – дескриптор существующего контекста устройства (DC). Если этот дескриптор равен ПУСТО (NULL), функция создает контекст устройства в памяти, совместимый с текущим экраном приложения.

Если функция завершается успешно, возвращаемое значение - дескриптор контекста устройства в памяти. Если функция завершается ошибкой, возвращаемое значение - ПУСТО (NULL).

* **BitBlt**

**BOOL BitBlt (HDC***hdcDest***, int***nXDest***, int***nYDest***, int***nWidth***, int***nHeight***, HDC***hdcSrc***, int***nXSrc***, int***nYSrc***, DWORD***dwRop***);**

Функция BitBlt выполняет передачу битовых блоков данных о цвете, соответствующих прямоугольнику пикселей из заданного исходного контекста устройства в целевой контекст устройства.

Параметры: *hdcDest* – дескриптор целевого контекста устройства; *nXDest*, *nYDest* – устанавливают x- и y-координату, в логических единицах измерения, левого верхнего угла целевого прямоугольника; *nWidth*, *nHeight* – устанавливают ширину и высоту, в логических единицах измерения, исходного и целевого прямоугольников; *hdcSrc* – дескриптор источникового контекста устройства; *nXSrc*, *nYSrc* – устанавливают x- и y-координату, в логических единицах измерения, левого верхнего угла исходного прямоугольника; *dwRop* – устанавливает код растровой операции. Эти коды определяют, как данные о цвете для исходного прямоугольника должны быть объединены с данными о цвете для целевого прямоугольника, чтобы достигнуть конечного цвета. Список ниже показывает некоторые общие коды растровых операций.

|  |  |
| --- | --- |
| **Значение** | **Описание** |
| **BLACKNESS** | Заполняет целевой прямоугольник, используя цвет, связанный с индексом 0 (черный) в физической палитре. |
| **DSTINVERT** | Инвертирует целевой прямоугольник. |
| **MERGECOPY** | Объединяет цвета исходного прямоугольника с кистью в текущий момент выбранной в hdcDest, при помощи использования булева оператора AND. |
| **MERGEPAINT** | Объединяет цвета инвертированного исходного прямоугольника с цветами целевого прямоугольника при помощи использования булева оператора OR. |
| **NOTSRCCOPY** | Копирует инвертированный исходный прямоугольник в целевой. |
| **NOTSRCERASE** | Комбинирует цвета исходных и целевых прямоугольников при помощи использования булева оператора OR и затем инвертирует получающийся в результате цвет. |
| **PATCOPY** | Копирует кисть, в текущий момент выбранную в hdcDest, в целевой точечный рисунок. |
| **PATINVERT** | Комбинирует цвета кисти, в текущий момент выбранной в hdcDest, с цветами целевого прямоугольника при помощи использования булева оператора XOR. |
| **PATPAINT** | Комбинирует цвета кисти, в текущий момент выбранной в hdcDest, с цветами инвертированного исходного прямоугольника при помощи использования булева оператора OR. Результаты этой операции объединяются с цветами целевого прямоугольника при помощи использования булева оператора OR. |
| **SRCAND** | Комбинирует цвета исходных и целевых прямоугольников при помощи использования булева оператора AND. |
| **SRCCOPY** | Копирует исходный прямоугольник непосредственно в целевой прямоугольник. |
| **SRCERASE** | Комбинирует инвертированные цвета целевого прямоугольника с цветами исходного прямоугольника при помощи использования булева оператора AND. |
| **SRCINVERT** | Комбинирует цвета источникового и целевого прямоугольников при помощи использования булева оператора XOR. |
| **SRCPAINT** | Комбинирует цвета источникового и целевого прямоугольников при помощи использования булева оператора OR. |
| **WHITENESS** | Заполняет целевой прямоугольник, используя цвет, связанный с индексом 1 (белый) в физической палитре. |

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается с ошибкой, величина возвращаемого значения - нуль.

* **DeleteDC**

**BOOL DeleteDC(HDC***hdc***);**

Функция DeleteDC удаляет заданный контекст устройства (DC).

Параметр: *hdc* – дескриптор контекста устройства (DC).

Если функция завершается успешно, величина возвращаемого значения - не нуль. Если функция завершается с ошибкой, величина возвращаемого значения - нуль.

* **LoadBitmap**

**HBITMAP LoadBitmap(HINSTANCE***hInstance***, LPCTSTR***lpBitmapName***);**

Функция LoadBitmap загружает заданный ресурс растрового изображения из модуля исполняемого файла.

Параметры: *hInstance* – дескриптор экземпляра модуля, исполняемый файл которого содержит загоужаемый точечный рисунок; *lpBitmapName* – указатель на символьную строку с нулем в конце, которая содержит название ресурса загружаемого растрового изображения.

Если функция завершается успешно, возвращаемое значение - дескриптор устанавливаемого точечного рисунка. Если функция завершается с ошибкой, возвращаемое значение - ПУСТО (NULL).

2) Работа с битовыми картами на C#.

Класс ***System.Drawing.Bitmap.***

|  |  |
| --- | --- |
| ***Конструкторы*** | |
| **Bitmap(Image)** | Инициализирует новый экземпляр класса Bitmap из указанного существующего изображения. |
| **Bitmap(Image, Int32, Int32)** | Инициализирует новый экземпляр класса Bitmap из указанного существующего изображения, масштабированного до заданного размера. |
| **Bitmap(Int32, Int32)** | Инициализирует новый экземпляр класса Bitmap с заданным размером. |
| **Bitmap(Int32, Int32, Graphics)** | Инициализирует новый экземпляр класса Bitmap с заданным размером и с разрешением указанного объекта Graphics. |
| ***Методы*** | |
| **DrawImage (<Bitmap>, x, y,Width, Height)** | Рисует заданный объект Image в заданном месте, используя указанный размер. |
| **DrawImage(<Bitmap>,<Rect>,<Rect>, GraphicsUnit.Pixel)** | Рисует заданную часть указанного объекта Image в заданном месте, используя заданный размер. |
| **GetPixel** | Возвращает цвет указанного пикселя в этом изображении Bitmap. |
| **SetPixel** | Задает цвет указанного пикселя в этом объекте Bitmap. |
| **MakeTransparent()** | Делает прозрачным прозрачный цвет по умолчанию для этого элемента Bitmap. |
| **MakeTransparent(Color)** | Делает заданный цвет прозрачным для данного изображения Bitmap. |
| **Dispose** | Выполняет определяемые приложением задачи, связанные с удалением, высвобождением или сбросом неуправляемых ресурсов |
| **Save(<имя файла>)** | Сохраняет объект Image в указанный файл или поток |

3) WinAPI. Работа с метафайлами.

* **Дескриптор HMETAFILE**

Microsoft Windows Metafile Format (WMF) используется для хранения векторных и растровых изображений и графических данных в памяти. Векторные данные, хранимые WMF-файлом, описывают команды Microsoft Windows Graphics Device Interface (GDI).

* **CreateMetaFile(NULL)**

**HDC CreateMetaFile(LPCSTR** *pszFile***);**

Функция CreateMetaFile создает контекст устройства для метафайла.

Параметр: *pszFile* – указывает на имя файла для создаваемого метафайла в формате Windows.

Если функция выполняется успешно, возвращаемое значение является дескриптором контекста устройства для метафайла в формате Windows. Если функция завершается неудачно, возвращаемое значение равно NULL.

* **CloseMetaFile**

**HDC CloseMetaFile(HDC** *hdc***);**

Функция CloseMetaFile закрывает контекст устройства метафайла.

Параметр: *hdc* – дескриптор контекста устройства для метафайла в формате Windows.

Если функция выполняется успешно, возвращаемое значение является дескриптором контекста устройства для метафайла в формате Windows. Если функция завершается неудачно, возвращаемое значение равно NULL.

* **PlayMetaFile**

**HDC PlayMetaFile(HDC** *hdc*, **HMETAFILE** *hmf***);**

Функция PlayMetaFile отображает изображение, сохраненное в данном метафайле формата Windows на указанном устройстве.

Параметры: *hdc* – дескриптор контекста устройства для метафайла в формате Windows; *hmf* – дескриптор метафайла в формате Windows.

Если функция выполняется успешно, возвращаемое значение – не нуль. Если функция завершается неудачно, возвращаемое значение равно нулю.

* **CopyMetaFile**

**HDC CopyMetaFile(HMETAFILE** *hmf*, **LPCSTR** *lpszFile***);**

Функция CopyMetaFile копирует содержимое метафайла формата Windows в указанный файл.

Параметры: *hmf* – дескриптор исходного метафайла в формате Windows; *lpszFile* – указатель на имя файла назначения. Если этот параметр NULL, исходный метафайл копируется в память.

Если функция выполняется успешно, возвращаемое значение является дескриптором контекста устройства для метафайла в формате Windows. Если функция завершается неудачно, возвращаемое значение равно нулю.

* **DeleteMetaFile**

**BOOL DeleteMetaFile(HMETAFILE** *hmf***);**

Функция CopyMetaFile копирует содержимое метафайла формата Windows в указанный файл.

Параметр: *hmf* – дескриптор исходного метафайла в формате Windows.

Если функция выполняется успешно, возвращаемое значение – не нуль. Если функция завершается неудачно, возвращаемое значение равно нулю.

* **GetMetaFile**

**HMETAFILE GetMetaFile(LPCSTR** *lpName***);**

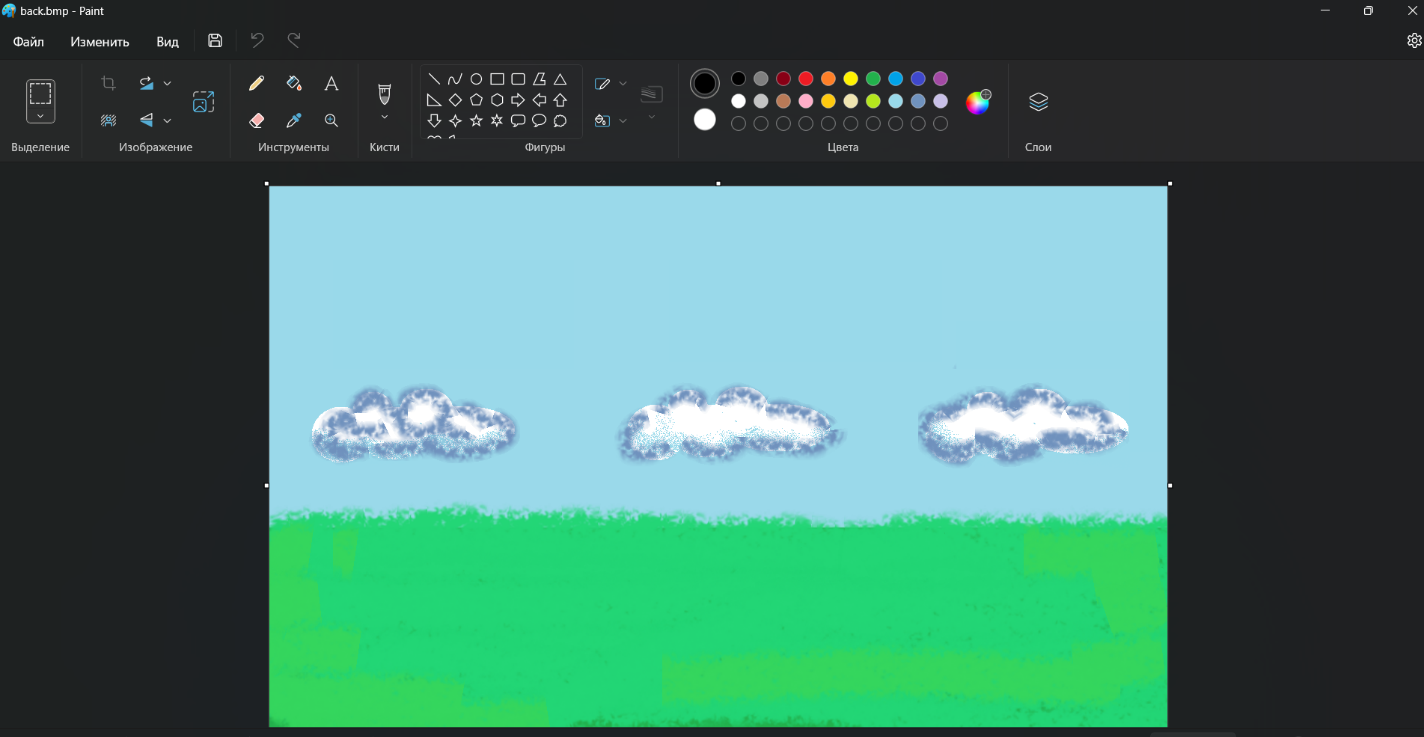
Функция CopyMetaFile копирует содержимое метафайла формата Windows в указанный файл.

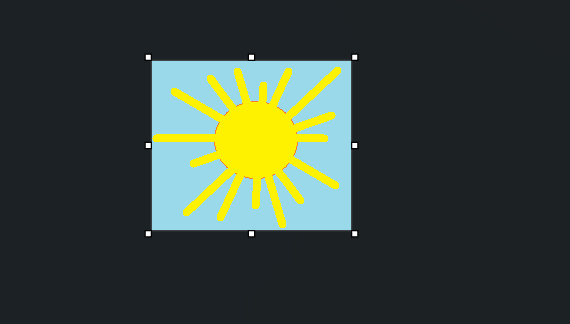
Параметр: *lpName* – указатель на строку, заканчивающуюся нулем, которая указывает имя метафайла.

Если функция выполняется успешно, возвращаемое значение является дескриптором метафайла в формате Windows. Если функция завершается неудачно, возвращаемое значение равно нулю.

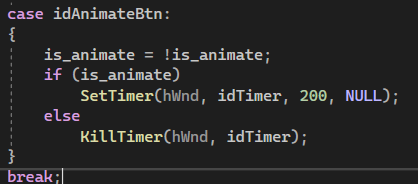
**2. Выполнение работы.**

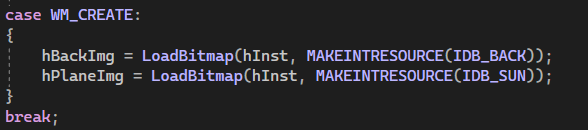
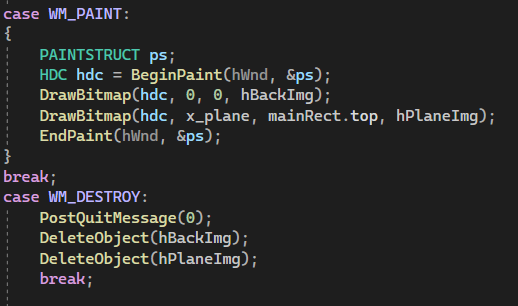
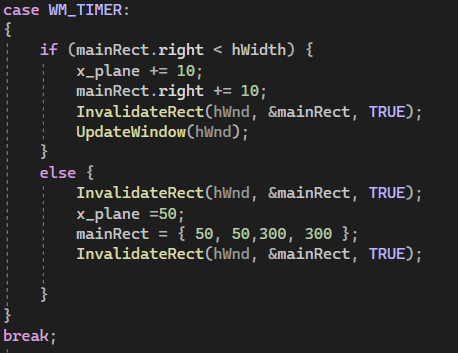
Перед началом работы приложения Win API создать две картинки (битовые карты) и включить их в ресурсы программы. Программный код должен обеспечивать вывод на рабочую поверхность окна большой картинки, а вторая (небольшая) картинка должна медленно перемещаться по ее поверхности (анимация). Картинки должны составлять единый сюжет











2.В приложении C# создать слайд-презентацию. На слайдах необходимо демонстрировать (двигать, приближать, удалять, меркнуть, проявлять частями и т.д. ) различные битовые карты (фото, графики, диаграммы, объекты и т.д.). Организовать демонстрацию слайдов

1). по времени,

2). по щелчкам пользователя.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinFormsApp1

{

public partial class Form1 : Form

{

private System.Windows.Forms.Timer timer;

private int animationStep;

private int Step = 0;

// Дополнительные переменные для хранения состояний анимации

private int currentAnimation = 0;

private Graphics GB;

private Font font;

private System.Drawing.Bitmap bitmap;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void OnTimerTick(object sender, EventArgs e)

{

GB.Clear(Color.White);

switch (currentAnimation)

{

case 0:

// Первая анимация

GB.DrawImage(bitmap, bitmap.Width - animationStep \* 90, bitmap.Height - animationStep \* 70, bitmap.Width, bitmap.Height);

if (++animationStep > 10)

{

GB.DrawString("Кошечки бывают разные", font, Brushes.Black, 150, 150);

animationStep = 0;

currentAnimation++;

bitmap.Dispose();

timer.Interval = 2000; // Пауза перед следующей анимацией

}

break;

case 1:

// Вторая анимация

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\Main\_kun.jpg");

timer.Interval = 100; // Установите новый интервал

GB.DrawImage(bitmap, 0, 0, bitmap.Width + 90 - 90 \* (10 - animationStep), bitmap.Height + 40 - 70 \* (10 - animationStep));

if (++animationStep > 10)

{

GB.DrawString("Мейн кун", font, Brushes.Lime, 150, 150);

currentAnimation++;

animationStep = 0;

bitmap.Dispose();

timer.Interval = 2000; // Пауза перед следующей анимацией

}

break;

case 2:

// Третья анимация

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\britain.jpeg");

timer.Interval = 150; // Установите новый интервал

if (animationStep < 10)

{

GB.DrawImage(bitmap, 0, 0, bitmap.Width - (bitmap.Width / 10) \* animationStep, bitmap.Height - (bitmap.Height / 10) \* animationStep);

// GB.DrawString("Британская", font, Brushes.Black, 150, 150);

animationStep++;

}

else

{

animationStep = 0;

currentAnimation++;

bitmap.Dispose();

timer.Interval = 1000; // Пауза перед следующей анимацией

}

break;

case 3:

// Четвёртая анимация

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\bengalskaya.jpg");

timer.Interval = 150; // Установите новый интервал

GB.DrawImage(bitmap, 0, 0, bitmap.Width / (12 - animationStep), bitmap.Height / (12 - animationStep));

if (++animationStep > 11)

{

GB.DrawString("Бенгальская кошка", font, Brushes.Orange, 150, 150);

animationStep = 0;

bitmap.Dispose();

timer.Stop();

}

break;

}

}

private void Run\_Click(object sender, EventArgs e)

{

GB = pictureBox1.CreateGraphics();

font = new Font("Comic Sans MS", 24, FontStyle.Bold);

timer = new System.Windows.Forms.Timer();

timer.Interval = 50; // укажите интервал в миллисекундах

timer.Tick += OnTimerTick;

animationStep = 0;

currentAnimation = 0;

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\class.jpg");

timer.Start();

}

private void PainClick()

{

GB.Clear(Color.White);

switch (Step)

{

case 0:

// Первая анимация

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\class.jpg");

for (float i = 0; i <= 10; i++)

{

GB.DrawImage(bitmap, bitmap.Width - i \* 90, bitmap.Height - i \* 70, bitmap.Width, bitmap.Height);

System.Threading.Thread.Sleep(50);

}

GB.DrawString("Кошечки бывают разные", font, Brushes.Black, 150, 150);

bitmap.Dispose();

break;

case 1:

// Вторая анимация

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\Main\_kun.jpg");

GB.DrawImage(bitmap, 0, 0, 90, 40);

System.Threading.Thread.Sleep(150);

for (int i = 10; i >= 0; --i)

{

GB.DrawImage(bitmap, 0, 0, bitmap.Width + 90 - 90 \* i, bitmap.Height + 40 - 70 \* i);

System.Threading.Thread.Sleep(50);

}

GB.DrawString("Мэйн Кун", font, Brushes.Lime, 140, 200);

bitmap.Dispose();

break;

case 2:

// Третья анимация

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\britain.jpeg");

for (int i = 0; i < 10; i++)

{

GB.DrawImage(bitmap, 0, 0, bitmap.Width - (bitmap.Width / 10) \* i, bitmap.Height - (bitmap.Height / 10) \* i);

//GB.DrawString("Британская кошка", font, Brushes.Orange, 10, 3);

System.Threading.Thread.Sleep(100);

GB.Clear(Color.White);

System.Threading.Thread.Sleep(50);

}

bitmap.Dispose();

break;

case 3:

// Четвёртая анимация

bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP7\WinFormsApp1\bengalskaya.jpg");

for (int i = 12; i > 0; --i)

{

GB.DrawImage(bitmap, 0, 0, bitmap.Width / i, bitmap.Height / i);

System.Threading.Thread.Sleep(50);

}

GB.DrawString("Бенгальская кошка", font, Brushes.Orange, 10, 3);

bitmap.Dispose();

break;

}

}

private void button1\_Click(object sender, EventArgs e)

{

GB = pictureBox1.CreateGraphics();

font = new Font("Comic Sans MS", 24, FontStyle.Bold);

PainClick();

Step++;

if (Step >= 4)

{

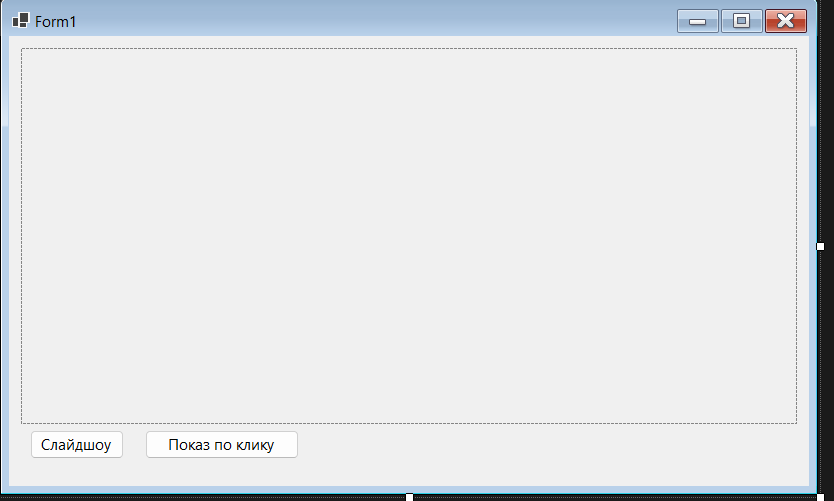
Step = 0;

}

}

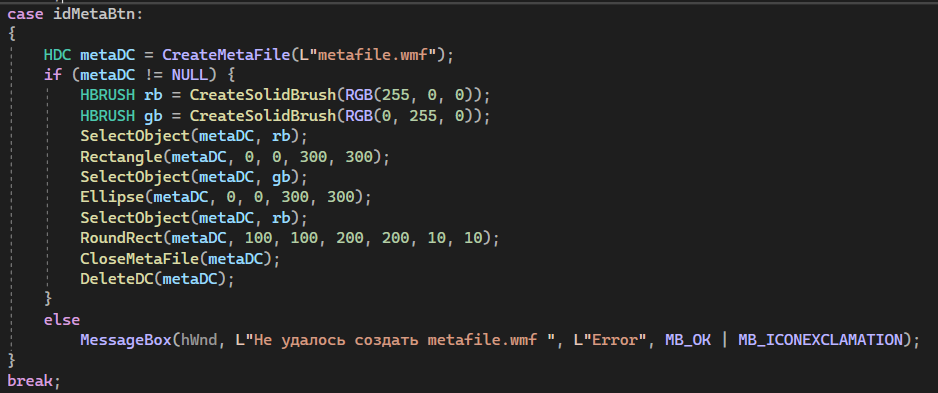
}

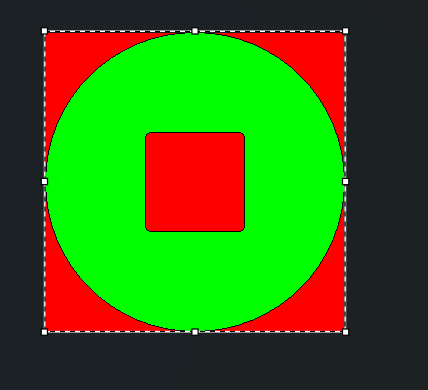
}

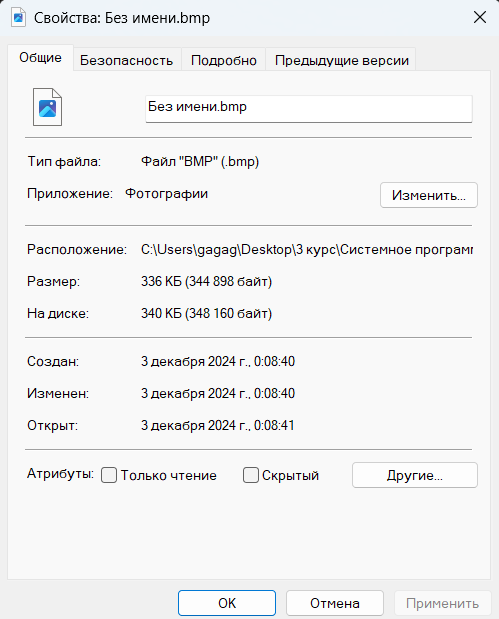


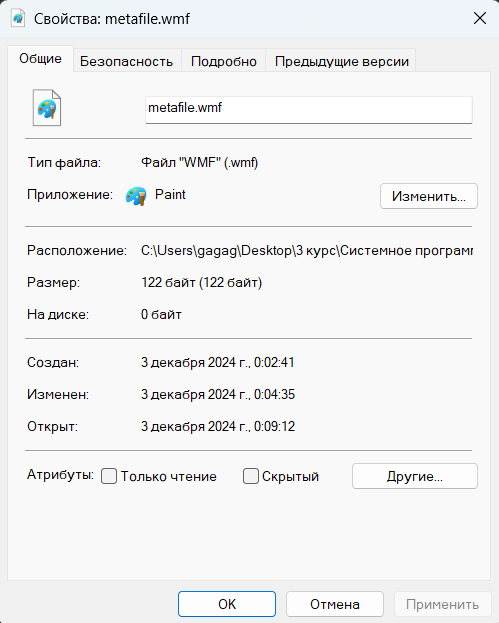
3.В приложении Win API создать метафайл и нарисовать в нем несколько геометрических фигур (сохранить файл). Создать в любом приложении такую же картинку и сохранить в формате .bmp. Сравнить файлы по размеру.











Разница более чем в 2800 раз.

