Лабораторная работа №7

Битовые карты и метафайлы

А-13а-19 Кутдусов Р.К.

Подготовка к лабораторной работе

1. Работа с битовыми картами в **Win API**.

- дескриптор **HBITMAP**
  дескриптор совместимого точечного рисунка (аппаратно-зависимая точечная картинка (**DDB**))
- **CreateCompatibleBitmap**
  создает точечный рисунок, совместимый с устройством, которое связано с заданным контекстом устройства

```
HBITMAP CreateCompatibleBitmap(
  HDC hdc,         // дескриптор DC
  int nWidth,      // ширина рисунка, в пикселях
  int nHeight      // высота рисунка, в пикселях
);
```

- **CreateCompatibleDC**
  создает контекст устройства в памяти (**DC**), совместимый с заданным устройством

```
HDC CreateCompatibleDC(
  HDC hdc   // дескриптор DC
);
```

- **BitBlt** (режимы копирования: SRCCOPY, SRCINVERT, SRCPAINT и т.д.)
  выполняет передачу битовых блоков данных о цвете, соответствующих прямоугольнику пикселей из заданного исходного контекста устройства в целевой контекст устройства

```
BOOL BitBlt(
  HDC hdcDest, // дескриптор целевого DC
  int nXDest,  // x-коорд. левого верхнего угла целевого прямоугольника
  int nYDest,  // y-коорд. левого верхнего угла целевого прямоугольника
  int nWidth,  // ширина целевого прямоугольника
  int nHeight, // высота целевого прямоугольника
  HDC hdcSrc,  // дескриптор исходного DC
  int nXSrc,   // x-коорд. левого верхнего угла исходного прямоугольника
  int nYSrc,   // y-коорд. левого верхнего угла исходного прямоугольника
  DWORD dwRop  // код растровой операции
);
```

Таблица ниже показывает некоторые общие **коды растровых операций**.

Значение	Описание
BLACKNESS	Заполняет целевой прямоугольник, используя цвет, связанный с индексом **0** в физической палитре. (Этот цвет является черным для заданной по умолчанию физической палитры.)
CAPTUREBLT	**Windows 98/Me, Windows 2000/XP:** включает любые окна, которые наслоены поверх вашего окна в результирующем изображении. По умолчанию, изображение содержит только ваше окно.
DSTINVERT	Инвертирует целевой прямоугольник.
MERGECOPY	Объединяет цвета исходного прямоугольника с кистью в текущий момент выбранной в ***hdcDest***, при помощи использования булева оператора **И** (**AND**).
MERGEPAINT	Объединяет цвета инвертированного исходного прямоугольника с цветами целевого прямоугольника при помощи использования булева оператора **ИЛИ** (**OR**).

NOMIRRORBITMAP	**Windows 98/Me, Windows 2000/XP:** Препятствует точечному рисунку быть зеркалируемым.
NOTSRCCOPY	Копирует инвертированный исходный прямоугольник в целевой.
NOTSRCERASE	Комбинирует цвета исходных и целевых прямоугольников при помощи использования булева оператора **ИЛИ** (**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 исключающее **ИЛИ** (**XOR**).
PATPAINT	Комбинирует цвета кисти, в текущий момент выбранной в ***hdcDest***, с цветами инвертированного исходного прямоугольника при помощи использования булева оператора **ИЛИ** (**OR**). Результаты этой операции объединяются с цветами целевого прямоугольника при помощи использования булева оператора **ИЛИ** (**OR**).
SRCAND	Комбинирует цвета исходных и целевых прямоугольников при помощи использования булева оператора **И** (**AND**).
SRCCOPY	Копирует исходный прямоугольник непосредственно в целевой прямоугольник.
SRCERASE	Комбинирует инвертированные цвета целевого прямоугольника с цветами исходного прямоугольника при помощи использования булева оператора **И** (**AND**).
SRCINVERT	Комбинирует цвета источникового и целевого прямоугольников при помощи использования булева оператора исключающее **ИЛИ** (**XOR**).
SRCPAINT	Комбинирует цвета источникового и целевого прямоугольников при помощи использования булева оператора **ИЛИ** (**OR**).
WHITENESS	Заполняет целевой прямоугольник, используя цвет, связанный с индексом ***1*** в физической палитре. (Этот цвет является белым для заданной по умолчанию физической палитры.)

- **DeleteDC**

  удаляет заданный контекст устройства (**DC**)

```
BOOL DeleteDC(
  HDC hdc   // дескриптор DC
);
```

- **LoadBitmap**(hInstance, MAKEINTRESOURCE(IDB_BITMAP))

  загружает заданный ресурс растрового изображения из модуля исполняемого файла

```
HBITMAP LoadBitmap(
  HINSTANCE hInstance,  // дескриптор экземпляра приложения
  LPCTSTR lpBitmapName  // имя ресурса рисунка
);
```

2. Работа с битовыми картами на **C#**.

Класс System.Drawing.Bitmap

позволяет считывать и сохранять файлы различных графических форматов

- 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.

- метод **DrawImage (<Bitmap>, x, y,Width, Height)** рисует заданный объект в заданном месте, используя указанный размер и **DrawImage(<Bitmap>,<Rect>,<Rect>, GraphicsUnit.Pixel)** рисует заданную часть указанного объект в заданном месте, используя заданный размер.
- **GetPixel** возвращает цвет указанного пикселя в этом изображении Bitmap и **SetPixel** задает цвет указанного пикселя в этом объекте Bitmap.
- **MakeTransparent**

Делает прозрачным прозрачный цвет по умолчанию для этого элемента Bitmap.
Делает заданный цвет прозрачным для данного изображения Bitmap.

- **Dispose**

освобождает все ресурсы, используемые данным объектом Graphics

- **Save(<имя файла>)**

Сохраняет данное изображение в указанный поток в указанном формате.

3. Работа с метафайлами в **Win API**.

- Дескриптор **HMETAFILE**

Дескриптор расширенного метафайла, который будет выведен в статическом элементе управления.

- **CreateMetaFile(NULL)**

```
HDC WINAPI CreateMetaFile(LPCSTR lpszFileName);
```

Создание контекста метефайла. Параметр lpszFileName должен указывать на строку, содержащую путь к имени файла, в который будут записаны команды GDI, или NULL. В последнем случае создается метафайл в оперативной памяти.

- **CloseMetaFile**

```
HMETAFILE WINAPI CloseMetaFile(HDC hdc);
```

После выполнения рисования в контексте метафайла следует закрыть метафайл. Эта функция закрывает метафайл для контекста hdc и возвращает идентификатор метафайла. Идентификатор закрытого метафайла использовать нельзя, так как он не содержит никакой полезной информации.

- **PlayMetaFile**

Можно проиграть метафайл в контексте отображения или контексте устройства, вызвав функцию PlayMetaFile:

```
BOOL WINAPI PlayMetaFile(HDC hdc, HMETAFILE hmf);
```

- **CopyMetaFile**(<метафайл>,<имя файла.wmf>)

```
HMETAFILE WINAPI CopyMetaFile(HMETAFILE hmf,
  LPCSTR lpszFileName);
```

Можно скопировать метафайл в обычный дисковый файл.

- **DeleteMetaFile**

```
BOOL WINAPI DeleteMetaFile(HMETAFILE hmf);
```

Удаление метафайла с помощью функции DeleteMetaFile делает недействительным идентификатор метафайла hmf и освобождает оперативную память, занятую метафайлом. Если метафайл был создан как обычный дисковый файл, функция DeleteMetaFile не удаляет его с диска.

- **GetMetaFile (**<имя файла.wmf>)

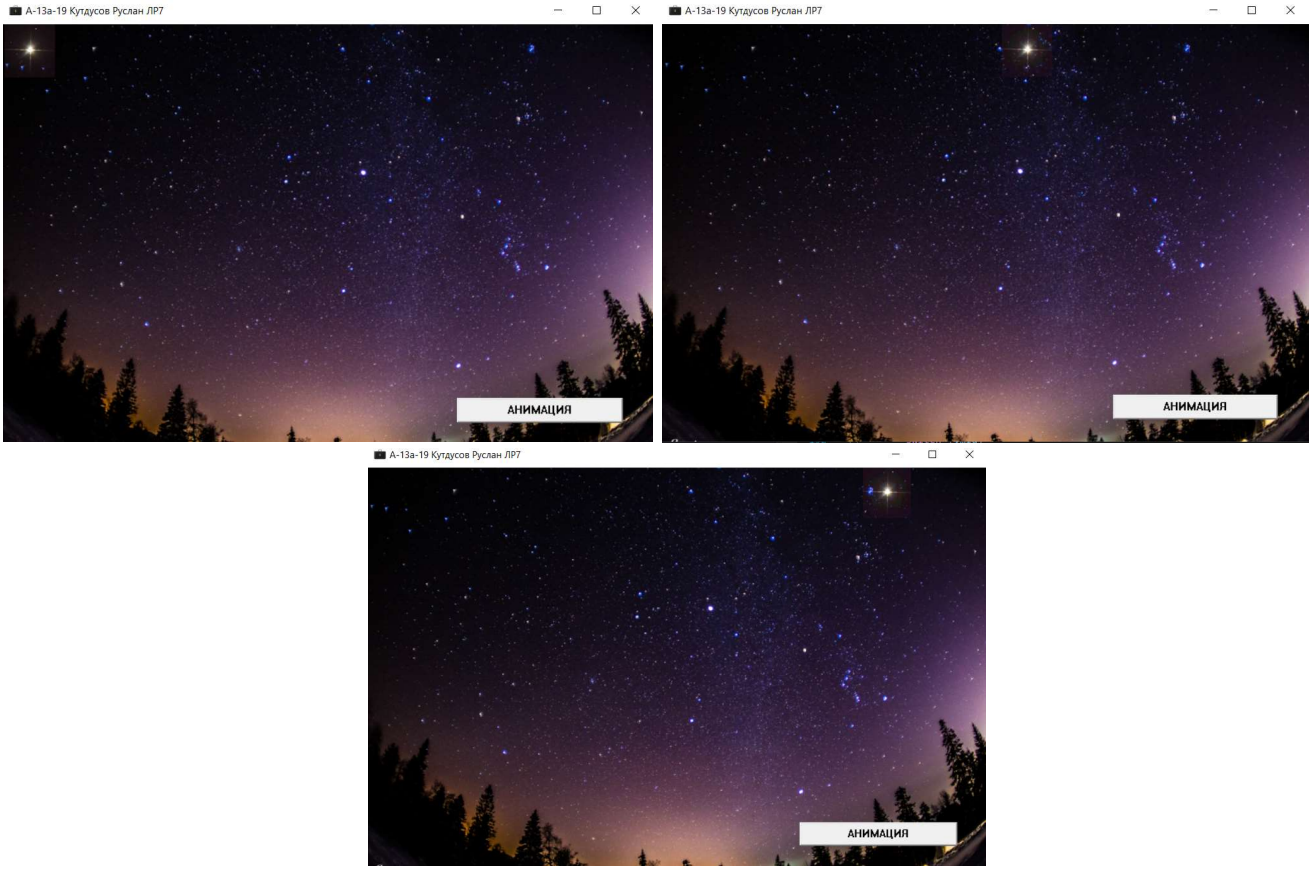
```
HMETAFILE WINAPI GetMetaFile(LPCSTR lpszFileName);
```

Для того чтобы воспользоваться метафайлом, хранящимся в виде дискового файла, его следует загрузить при помощи функции GetMetaFile, указав ей в качестве единственного параметра путь к соответствующему файлу.

Ход работы

1. 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.

Движение крупной звезды по звёздному небу:



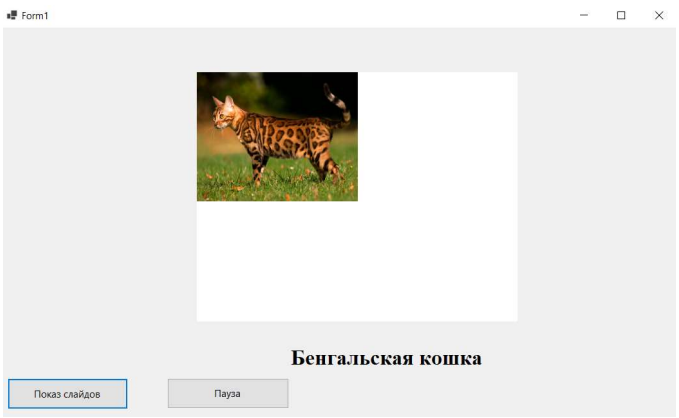

```
HBITMAP hSky, hStar;

bool anim = false;

HDC h_dc, h_comp_dc;

int x_0 = 0;
int y_0 = 0;
int x_h = -5;

hSky = LoadBitmap(hInstance, MAKEINTRESOURCE(IDB_BITMAP2)); // из ресурсов
hStar = LoadBitmap(hInstance, MAKEINTRESOURCE(IDB_BITMAP1));
```

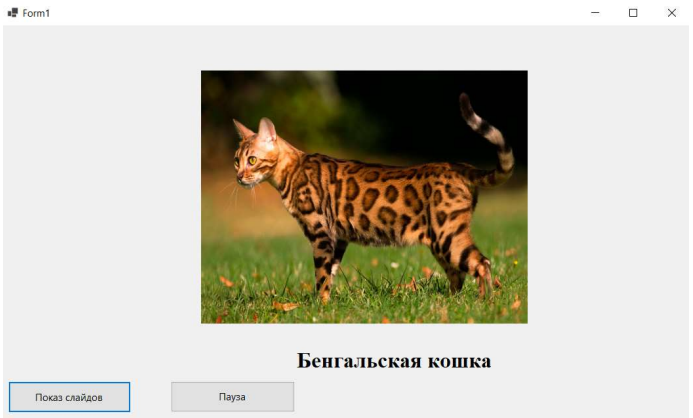
```
case WM_TIMER:
		if (x_0 <= -750) x_h = -x_h;
		SelectObject(h_comp_dc, hSky); // выбираем небо
		BitBlt(h_dc, 0, 0, 810, 550, h_comp_dc, 0, 0, SRCCOPY);
		SelectObject(h_comp_dc, hStar); // выбираем звезду
		BitBlt(h_dc, 0, 0, 810, 550, h_comp_dc, x_0, y_0, SRCPAINT);
		x_0 += x_h;
		if (x_0 == 0)  x_h = -x_h;
		break;
	case WM_COMMAND:
		switch (vmId)
		{
		case ID_ANIM:
			anim = !anim;
			if (anim) {
				h_dc = GetDC(hWnd);
				h_comp_dc = CreateCompatibleDC(h_dc);
				SetTimer(hWnd, TIMER_1, 200, NULL);
			}
			else {
				KillTimer(hWnd, TIMER_1);
				DeleteDC(h_comp_dc);
				ReleaseDC(hWnd, h_dc);
			}
			break;
		default:
			return DefWindowProc(hWnd, message, wParam, lParam);
		}
		break;
```

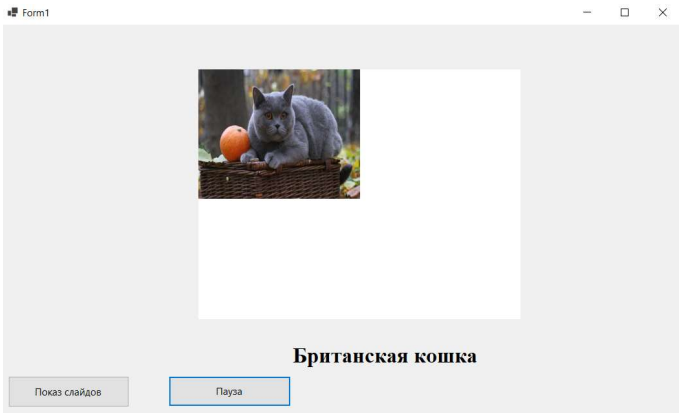
Звезда будет перемещаться наверху от левой стенки, до правой, до левой и т. д., можно остановить, запустить снова. Звезда была на черном фоне. Режим `SRCPAINT` создаёт эффект прозрачности, поэтому более мелкие звёзды на заднем плане видны.

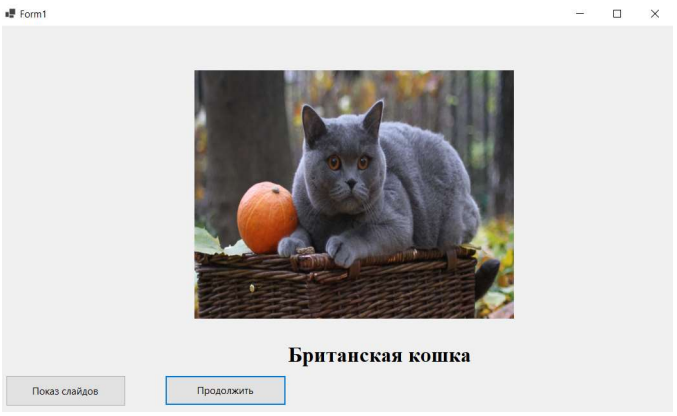
2. В приложении C# создать слайд-презентацию. На слайде должны постепенно проявляться (двигаться, приближаться, удаляться и т. д.) различные картинки – битовые карты (графики, диаграммы, объекты и т. д.), а также выводиться соответствующий текст.

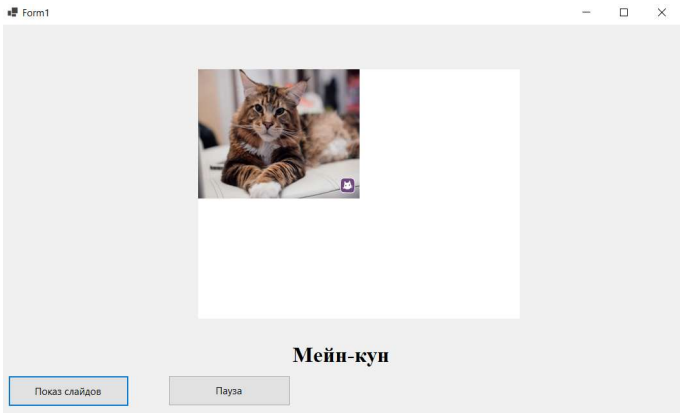
Породы кошек.

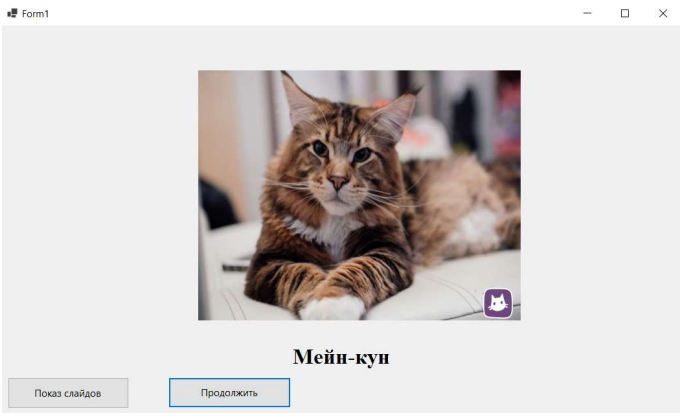












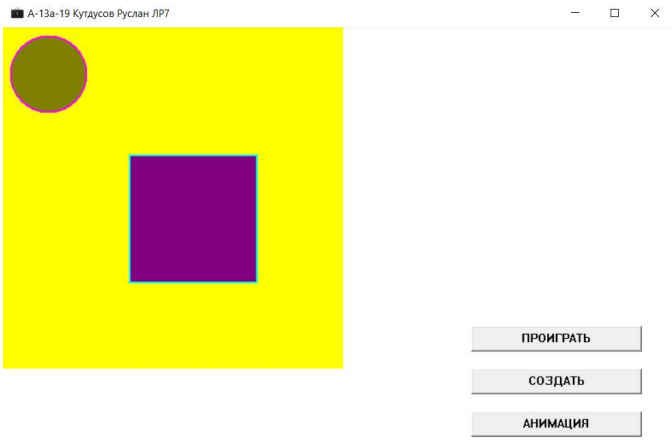

```
private bool paused = false; // можно остановить интересующий слайд

private void button1_Click(object sender, EventArgs e)
        {
            Thread thread = new Thread(new ThreadStart(show));
            thread.Start();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (!paused) button2.Text = "Продолжить";
            else button2.Text = "Пауза";
            paused = !paused;
        }

        void pause()
        {
            while (paused) ;
            return;
        }

        void show()
        {
                Graphics dc = pictureBox1.CreateGraphics();
                dc.Clear(Color.White);
                Bitmap bm = new
Bitmap("C:\\Users\\user\\source\\repos\\lab_7_sp\\lab_7_sp_winforms\\lab_7_sp_winforms\\bengalskaja-
koshka-1.jpg"); // любой формат картинок допустим
                if (paused) pause();
                // label1.Text = "Бенгальская кошка";
                label1.BeginInvoke((MethodInvoker)(() => label1.Text = "Бенгальская кошка"));
                for (int i = 10; i > 0; --i)
                {
                    dc.DrawImage(bm, 0, 0, 500 / i, 400 / i);
                    Thread.Sleep(50);
                }
                bm.Dispose();
                if (paused) pause();
                Thread.Sleep(3000);
                if (paused) pause();
                dc.Clear(Color.White);
```

```
                bm = new
Bitmap("C:\\Users\\user\\source\\repos\\lab_7_sp\\lab_7_sp_winforms\\lab_7_sp_winforms\\british-2.jpg");
                // label1.Text = "Британская кошка";
                label1.BeginInvoke((MethodInvoker)(() => label1.Text = "Британская кошка"));
                for (int i = 10; i > 0; --i)
                {
                    dc.DrawImage(bm, 0, 0, 500 / i, 400 / i);
                    Thread.Sleep(50);
                }
                if (paused) pause();
                Thread.Sleep(3000);
                if (paused) pause();
                dc.Clear(Color.White);
                bm = new
Bitmap("C:\\Users\\user\\source\\repos\\lab_7_sp\\lab_7_sp_winforms\\lab_7_sp_winforms\\coon-3.png");
                // label1.Text = "Мейн-кун";
                label1.BeginInvoke((MethodInvoker)(() => label1.Text = "Мейн-кун"));
                for (int i = 10; i > 0; --i)
                {
                    dc.DrawImage(bm, 0, 0, 500 / i, 400 / i);
                    Thread.Sleep(50);
                }
                bm.Dispose();
                if (paused) pause();
                Thread.Sleep(3000);
                bm.Dispose();
                if (paused) pause();
        }
```

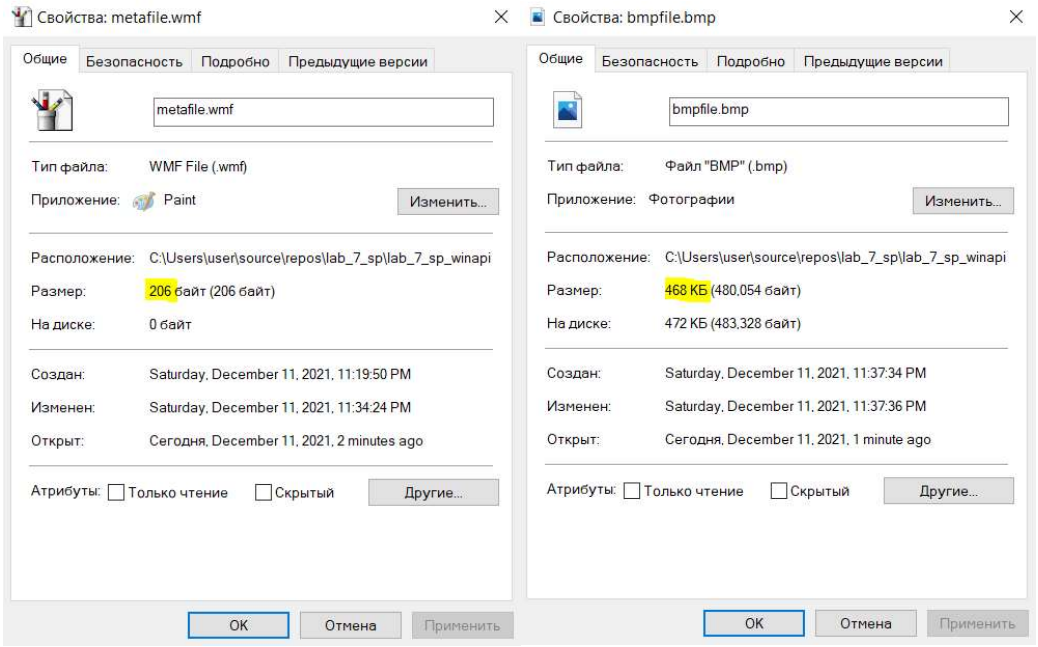
3. В приложении Win API создать метафайл и нарисовать в нем несколько геометрических фигур (сохранить файл). Создать в любом приложении такую же картинку и сохранить в формате .bmp. Сравнить файлы по размеру.

```
HDC meta_dc;

case ID_CREATE:
                    meta_dc = CreateMetaFile(L"metafile.wmf");
                    if (meta_dc != NULL) {
                            rect = {0, 0, 400, 400};
                            FillRect(meta_dc, &rect, CreateSolidBrush(RGB(255, 255, 0)));
                            SelectObject(meta_dc, CreateSolidBrush(RGB(128, 128, 0)));
                            SelectObject(meta_dc, CreatePen(PS_SOLID, 2, RGB(255, 0, 255)));
                            Ellipse(meta_dc, 10, 10, 100, 100);
                            SelectObject(meta_dc, CreateSolidBrush(RGB(128, 0, 128)));
                            SelectObject(meta_dc, CreatePen(PS_SOLID, 2, RGB(0, 255, 255)));
                            Rectangle(meta_dc, 150, 150, 300, 300);
                            CloseMetaFile(meta_dc);
                    }
                    break;
            case ID_PLAY:
                    PlayMetaFile(GetDC(hWnd), GetMetaFile(L"metafile.wmf"));
                    break;
```

Получился рисунок:



Теперь преобразуем .wmf формат в .bmp формат в Paint-е. Посмотрим на размеры и сравним:



Размер .bmp намного превышает размер метафайла.