

Применение кодировщиков изображений Win32 API

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается способ работы с кодировщиками изображений предоставляемых Win32 API. Для реализации применяются язык программирования C++ и библиотека GDI+. Результатом исследования станет разработанная программа перекодирования изображения из одного формата в другой.

Ключевые слова: Win32 API, GDI+, C++, обработка изображений.

Using the Win32 API Image Encoders

Vikhlyayev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses how to work with image encoders provided by the Win32 API. The C++ programming language and the GDI+ library are used for implementation. The result of the research will be a developed program for transcoding images from one format to another.

Keywords: Win32 API, GDI+, C++, image processing.

1 Введение

1.1 Актуальность

GDI+ (Graphics Device Interface Plus) является API от Microsoft, используемым для представления графики и изображения в Windows-приложениях. GDI+ включает в себя классы для создания, изменения и отображения изображений. Кодировщики изображений в GDI+ позволяют сохранять изображения в различных форматах, таких как BMP, JPEG, PNG, GIF и TIFF. Эти кодировщики являются частью GDI+ и встроены в библиотеку. Кодировщики (а также декодировщики) используются для преобразования изображения в определенный формат файла и обратно. Так как данная библиотека входит в состав Win32 API, её легко подключить и использовать. Нет необходимости устанавливать, какие либо зависимости или исходные файлы, достаточно только написать в коде информацию о подключении заголовочного файла и передать название библиотеки компилятору.

1.2 Обзор исследований

А.Е.Курочкин описал стандарт JPEG и кодирование изображений с потерями [1]. М.Е.Шахова, С.В.Букунов применили библиотеку WIN32 API для создания графических изображений в консольном приложении [2]. Е.К.Самаров провёл сравнительный анализ средств сжатия и обработки растровых изображений JPEG, JPEG 2000, ICER и PNG [3]. А.Ю.Поляков, В.А.Брусенцев провели исследование в программировании графики через GDI+ и DirectX [4]. Р.Хэй представил инструменты для редактирования графики [5].

1.3 Цель исследования

Цель исследования – описать метод работы с кодировщиками изображений Win32 API.

2 Материалы и методы

Для реализации используется язык программирования «C++» и библиотека GDI+ входящая в состав Win32 API. В приведённых примерах описаны только функции, предполагается, что с наружи используется пространство имён Gdiplus.

3 Результаты и обсуждения

Классы GDI+: GDI+ предоставляет классы, такие как Bitmap, Image, Graphics, которые используются для работы с изображениями. Класс Bitmap позволяет создавать и управлять изображениями, а также выполнять операции сохранения и загрузки.

Кодировщики и декодировщики: В GDI+ кодировщики и декодировщики изображений реализованы через ImageCodecInfo. Эта структура содержит информацию о кодеках, таких как MIME-тип, формат файла и CLSID.

Для каждого формата изображения существует уникальный CLSID (Class Identifier), который используется для идентификации соответствующего кодировщика. Функция GetImageEncoders используется для получения списка доступных кодировщиков.

Перед использованием любых функций GDI+, необходимо инициализировать библиотеку с помощью GdiplusStartup. Используется функция GetImageEncodersSize, GetImageDecodersSize, чтобы узнать количество доступных кодировщиков, декодировщиков и размер массива. Затем применяется функция GetImageEncoders, чтобы получить массив ImageCodecInfo, который содержит информацию обо всех доступных кодировщиках.

Затем используется функция GetImageDecoders, чтобы получить массив ImageCodecInfo, который содержит информацию обо всех доступных декодерах (рис.1,2).

```

void decoders_info()
{
    // Инициализация GDI+
    GdiplusStartupInput gdiplusStartupInput;
    ULONG_PTR gdiplusToken;
    GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

    UINT numDecoders = 0; // количество декодеров
    UINT sizeDecoders = 0; // размер массива декодеров

    GetImageDecodersSize(&numDecoders, &sizeDecoders);
    if (sizeDecoders == 0)
    {
        std::wcout << L"Нет доступных декодеров." << std::endl;
        return;
    }
    ImageCodecInfo* pImageCodecInfo = (ImageCodecInfo*)(malloc(sizeDecoders));
    if (pImageCodecInfo == NULL)
    {
        std::wcout << L"Ошибка выделения памяти." << std::endl;
        return;
    }
    Status res = GetImageDecoders(numDecoders, sizeDecoders, pImageCodecInfo);
    if (res != 0)
    {
        free(pImageCodecInfo);
        std::wcout << L"Не удалось получить декодеры" << std::endl;
        return;
    }
    std::cout << "decoders info list:\n\n";
    for (UINT j = 0; j < numDecoders; ++j)
    {
        std::wcout << L"Codec Name: " << pImageCodecInfo[j].CodecName << std::endl;
        std::wcout << L"Format Description: " << pImageCodecInfo[j].FormatDescription << std::endl;
        std::wcout << L"MimeType: " << pImageCodecInfo[j].MimeType << std::endl;
        std::wcout << L"Filename Extension: " << pImageCodecInfo[j].FilenameExtension << std::endl;
        std::wcout << L"Clsid: {"
            << pImageCodecInfo[j].Clsid.Data1 << L"-";
            << pImageCodecInfo[j].Clsid.Data2 << L"-";
            << pImageCodecInfo[j].Clsid.Data3 << L"-";

            for (int k = 0; k < 8; ++k)
            {
                std::wcout << std::hex << (int)pImageCodecInfo[j].Clsid.Data4[k];
            }
            std::wcout << L"}" << std::endl;
            std::wcout << L"-----" << std::endl;
        }
        free(pImageCodecInfo);
        // Завершение работы с GDI+
        GdiplusShutdown(gdiplusToken);
    }
}

```

Рис. 1. Метод вывода информации о декодировщиках

```
Codec Name: Built-in BMP Codec
Format Description: BMP
MimeType: image/bmp
Filename Extension: *.BMP;*.DIB;*.RLE
Clsid: {557cf400-1a04-11d3-9a7300f81ef32e}
-----
Codec Name: Built-in JPEG Codec
Format Description: JPEG
MimeType: image/jpeg
Filename Extension: *.JPG;*.JPEG;*.JPE;*.JFIF
Clsid: {557cf401-1a04-11d3-9a7300f81ef32e}
-----
Codec Name: Built-in GIF Codec
Format Description: GIF
MimeType: image/gif
Filename Extension: *.GIF
Clsid: {557cf402-1a04-11d3-9a7300f81ef32e}
-----
Codec Name: Built-in EMF Codec
Format Description: EMF
MimeType: image/x-emf
Filename Extension: *.EMF
Clsid: {557cf403-1a04-11d3-9a7300f81ef32e}
-----
Codec Name: Built-in WMF Codec
Format Description: WMF
MimeType: image/x-wmf
Filename Extension: *.WMF
Clsid: {557cf404-1a04-11d3-9a7300f81ef32e}
-----
Codec Name: Built-in TIFF Codec
Format Description: TIFF
MimeType: image/tiff
Filename Extension: *.TIF;*.TIFF
Clsid: {557cf405-1a04-11d3-9a7300f81ef32e}
-----
Codec Name: Built-in PNG Codec
Format Description: PNG
MimeType: image/png
Filename Extension: *.PNG
Clsid: {557cf406-1a04-11d3-9a7300f81ef32e}
-----
Codec Name: Built-in ICO Codec
Format Description: ICO
MimeType: image/x-icon
Filename Extension: *.ICO
Clsid: {557cf407-1a04-11d3-9a7300f81ef32e}
```

Рис. 2. Вывод доступных декодеров

Для использования кодера или декодера нужно знать его CLSID, это проще сделать через текстовую проверку формата изображения. Немного переделав метод можно реализовать возврат нужного компонента. Функция GetEncoderClsid сравнивает MIME-тип и возвращает CLSID кодировщика для нужного формата (рис.3).

```

int GetEncoderClsid(const WCHAR* format, CLSID* pClsid)
{
    UINT num = 0;           // number of image encoders
    UINT size = 0;          // size of the image encoder array in bytes

    ImageCodecInfo* pImageCodecInfo = NULL;

    GetImageEncodersSize(&num, &size);
    if (size == 0)
        return -1; // Failure

    pImageCodecInfo = (ImageCodecInfo*)(malloc(size));
    if (pImageCodecInfo == NULL)
        return -1; // Failure

    GetImageEncoders(num, size, pImageCodecInfo);

    for (UINT j = 0; j < num; ++j)
    {
        if (wcscmp(pImageCodecInfo[j].MimeType, format) == 0)
        {
            *pClsid = pImageCodecInfo[j].Clsid;
            free(pImageCodecInfo);
            return j; // Success
        }
    }

    free(pImageCodecInfo);
    return -1; // Failure
}

```

Рис. 3. Метод получения идентификатора кодировщика по MIME-типу

Далее изменения формата изображения. Перед любыми операциями с GDI+ необходимо инициализировать библиотеку. Класс `Gdiplus::Image` используется для загрузки изображения из файла. Конструктор класса принимает путь к изображению, которое нужно загрузить. Для получения списка доступных кодировщиков и поиска нужного CLSID по MIME-типу (например, `image/jpeg` для JPEG). После получения CLSID нужного кодировщика можно вызвать метод `Save` у объекта `Gdiplus::Image`, передав ему путь к файлу для сохранения и CLSID кодировщика. Это сохранит изображение в новом формате. После завершения операций нужно очистить ресурсы, выделенные для работы с GDI+. Это делается с помощью функции `GdiplusShutdown`, которая принимает токен, полученный при инициализации GDI+ (рис. 4).

```

void recode_img(const wchar_t* input_img, const wchar_t* output_img, const wchar_t* format)
{
    // Initialize GDI+.
    GdiplusStartupInput gdiplusStartupInput;
    ULONG_PTR gdiplusToken;
    GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

    CLSID encoderClsid;
    Status stat;
    Image* image = new Image(input_img);

    // Get the CLSID of the PNG encoder.
    GetEncoderClsid(format, &encoderClsid);

    stat = image->Save(output_img, &encoderClsid, NULL);

    if (stat != Ok)
    {
        printf("Failure: stat = %d\n", stat);
    }
    delete image;
    GdiplusShutdown(gdiplusToken);
}

```

Рис. 4. Функция изменения формата изображения

В результате работы функций, должно сгенерироваться новое аналогичное входному изображение, но другого формата. На рисунках 5,6

представлены файлы, содержащие исходное изображение в формате PNG и преобразованное в формат JPEG.

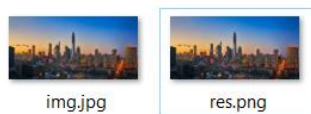


Рис. 5. Изображения разных форматов

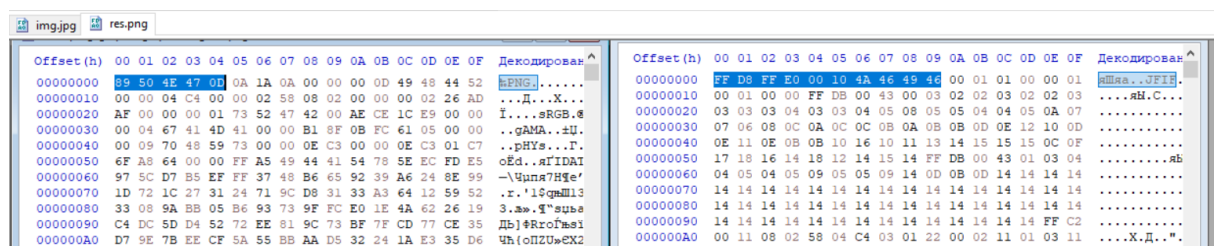


Рис. 6. Различия в сигнатуре файлов

4 Выводы

В результате исследования было рассмотрено создание функций работающими с кодировщиками Win32 API, представляемые через библиотеку GDI+. Данный метод является одним из самых простых способов обработки изображений в операционной системе Windows.

Библиографический список

1. Курочкин А.Е. Кодирование изображений с потерями. Стандарт JPEG. Минск, 2017.
2. Шахова М.Е., Букунов С.В. Применение библиотеки WIN32 API для создания графических изображений в консольном приложении // The Scientific Heritage. 2019. № 41-1 (41). С. 55-59.
3. Самаров Е.К. Сравнительный анализ средств сжатия и обработки растровых изображений JPEG, JPEG 2000, ICER и PNG // В сборнике: Перспективы, организационные формы и эффективность развития сотрудничества российских и зарубежных ВУЗов. Сборник материалов VII Ежегодной международной научной конференции. 2019. С. 416-421.
4. Поляков А.Ю., Брусенцев В.А. Программирование графики. GDI+ и DirectX: б-ка DI+. компонент DirectX, б-ки DirectX 9.0, приемы программирования в среде Visual C++ и C#, мат. аппарат преобразованный на плоскости и в пространстве. СПб.: БХВ-Петербург, 2005. 357 с.
5. Хэй Р. Инструменты для редактирования графики // Windows 2000 Magazine/Re. 2017. № 11. С. 40.
6. Самандаров И.Р., Маншуров Ш.Т., Душатов Н.Т., Миратоев З.М., Мустафин Р.Р. Обработка изображений в C++ с помощью библиотеки

OpenCV // Universum: технические науки. 2023. № 5-2 (110). С. 19-28.

7. Зубакин И.А., Фахми Ш.С. Обзор методов кодирования и декодирования изображений // Вопросы радиоэлектроники. Серия: Техника телевидения. 2008. № 1. С. 13-41.
8. Иванов А.В., Фахми Ш.С. Методы, алгоритмы и устройства кодирования и декодирования изображений. СПб., 2019.