Документация по сбору метаданных изображений

Назначение программы

Программа анализирует графические файлы различных форматов (JPEG, BMP, PNG, GIF, TIFF, PCX) и извлекает информацию о размере, разрешении и глубине цвета. Результаты отображаются в таблице.

Используемые библиотеки

Программа использует PySide6 для интерфейса, Pillow для работы с изображениями и встроенные модули struct и threading для парсинга и многопоточности.

Способ извлечения метаданных

Программа читает файлы в двоичном режиме и ищет в них специальные маркеры, которые указывают на нужную информацию. Вместо загрузки всего файла в память, программа читает только необходимые байты из начала файла.

JPEG файлы

Для JPEG используется поиск маркеров в двоичных данных. Сначала проверяется сигнатура файла (первые два байта должны быть FF D8). Затем программа сканирует файл в поиске маркера FF C0, который содержит информацию о размере и количестве цветовых компонентов.

Из маркера FF C0 читаются:

- позиция +5:7 для высоты изображения
- позиция +7:9 для ширины
- позиция +9 для количества компонентов (обычно 3 для RGB)

Глубина цвета вычисляется как 8 бит (точность) умноженные на количество компонентов.

Разрешение DPI ищется в маркере FF E0 (JFIF заголовок). Если там указано, что единицы DPI присутствуют, то DPI читаются из позиций +12:14 и +14:16.

BMP файлы

BMP файлы имеют стандартизированную структуру заголовка, где все данные находятся в фиксированных позициях. После проверки сигнатуры

(первые два байта должны быть "BM") программа переходит на нужные позиции и читает значения:

- позиция 18-21 содержит ширину
- позиция 22-25 содержит высоту
- позиция 28-29 содержит биты на пиксель (8, 24, 32)

Разрешение DPI хранится в позициях 38-45, но оно выражено в пиксели на метр, поэтому для перевода в DPI используется коэффициент 39.3701.

PNG файлы

PNG состоит из последовательности блоков (чанков). Каждый блок содержит длину, тип и данные. Программа читает сигнатуру файла (89 50 4E 47 0D 0A 1A 0A), а затем в цикле обрабатывает чанки.

Нужный блок - IHDR (Image Header), который всегда первый. Из него считываются размеры и информация о цвете. Также ищется блок pHYs, который содержит разрешение в пиксели на метр.

Глубина цвета зависит от типа цвета: для RGB это битовая глубина умноженная на 3, для RGBA - умноженная на 4.

GIF файлы

GIF имеет более простую структуру. После проверки сигнатуры ("GIF87a" или "GIF89a") программа читает логический экран, где находятся размеры и информация о цветах. Глубина цвета определяется из упакованного поля в 10-й позиции файла.

GIF обычно не содержит информацию о разрешении, поэтому используется стандартное значение 72x72 DPI.

TIFF файлы

TIFF имеет более сложную структуру с таблицей дескрипторов (IFD). Файл начинается с информации о порядке байтов и магического числа 42. Затем указывается позиция первой IFD таблицы.

В таблице ищутся определённые теги: для размера используются теги 256 и 257, для глубины цвета - тег 258, для разрешения - теги 282 и 283.

PCX файлы

PCX имеет фиксированный заголовок из 128 байт. Размеры вычисляются из координат в заголовке, глубина цвета - из количества бит на пиксель и

количества плоскостей цвета. Разрешение уже указано в DPI и находится в позициях 8-11.

Архитектура программы

Программа состоит из нескольких частей. GUI на PySide6 позволяет пользователю выбрать папку с изображениями. Затем ImageScannerThread работает в отдельном потоке и рекурсивно сканирует папку, ища все поддерживаемые форматы.

Для обработки файлов используется ThreadPoolExecutor, который создаёт пул из 8 рабочих потоков. Это позволяет обрабатывать несколько файлов одновременно, заметно ускоряя процесс.

Каждый файл попадает в ImageMetadataReader, который определяет формат по сигнатуре и выбирает нужный парсер. Результаты упаковываются в объект ImageInfo и отправляются в GUI для отображения в таблице.

Особенности реализации

Программа использует бинарное чтение файлов вместо полной загрузки через PIL, что даёт значительное ускорение. Для разных форматов применяются разные подходы в зависимости от их структуры.

Многопоточность реализована через QThread и ThreadPoolExecutor, что позволяет обрабатывать большие папки без зависания интерфейса. Прогресс-бар показывает процент обработки в реальном времени.

Обработка ошибок реализована через try-except блоки с логированием. Если какой-то файл не удалось обработать, программа продолжает работу, но записывает ошибку в лог.