МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Графический редактор BMP-изображений

Студент гр. 0381 _____ Самойлов З.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Самойлов З.А.

Группа 0381

Тема работы: Графический редактор BMP-изображений

Исходные данные:

Программа должна реализовывать весь следующий функционал по обработке bmp-файла:

1. Фильтр rgb-компонент.
2. Рисование квадрата.
3. Поменять местами 4 куска области.
4. Найти доминирующий цвет и заменить его на заданный цвет.

Содержание пояснительной записки:

Аннотация, содержание, описание классов, проверка работы программы, заключение, список источников, приложение.

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 05.04.2021

Дата сдачи реферата: 29.05.2021

Дата защиты реферата: 01.06.2021

Студент _____ Самойлов З.А.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

Курсовая работа представляет собой создание графического редактора для BMP-изображений. При разработке в качестве основного инструмента был использован C++. Для создания GUI использован фреймвок Qt. Для сборки и компиляции использовались qmake и g++. В результате была получена готовая к работе и распространению программа, выполняющая функции графического редактора для BMP-изображений с глубиной 24 бита.

SUMMARY

Coursework is the creation of a graphical editor for BMP-images. During the development process, C++ was used as the main development tool. Qt framework was used to create the GUI. qmake and g++ were used for building and compilation. As a result, a program ready for use and distribution was made that performs functions of a graphical editor for BMP-images with depth of 24 bits.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Написать графический редактор. Программа должна реализовывать весь следующий функционал по обработке bmp-файла:

1. Фильтр rgb-компонент.
2. Рисование квадрата.
3. Поменять местами 4 куска области.
4. Найти доминирующий цвет и заменить его на заданный цвет.

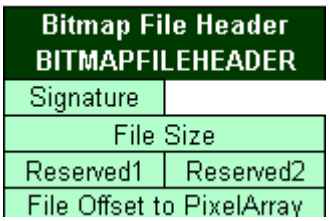
1. КЛАСС IMAGE

Класс Image содержит в себе информацию открытого .bmp файла и инструментал, предназначенный для различных манипуляций с файлом.

1.1. Описание полей класса

fileHeader — массив, содержащий в себе заголовок файла.

Побайтовое разделение выполнено в соответствии с данной схемой, где:



Signature — сигнатура файла («BM»),

File Size — размер файла,

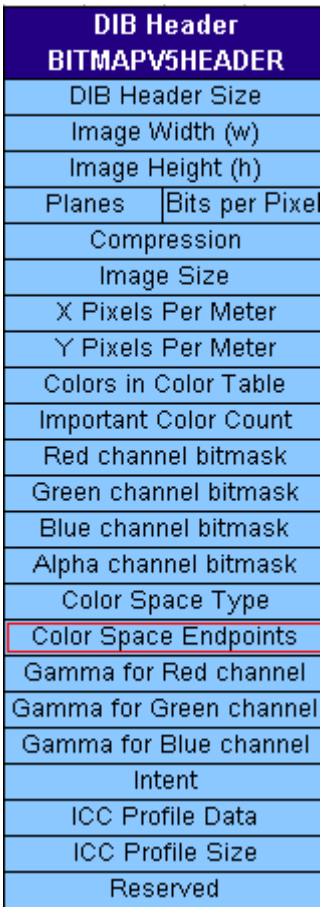
Reserved 1-2 — слоты, опционально используемые программами,

File Offset to PixelArray — адресс байта, с которого начинается массив пикселей изображения.

fileHeaderSize — размер массива *fileHeader*.

infoHeader — массив, содержащий в себе заголовок изображения.

Побайтовое разделение выполнено в соответствии с данной схемой, где:



DIB Header Size — размер данного заголовка,

Image Width — ширина изображения,

Image Height — высота изображения,

Bits per Pixel — глубина изображения,

Image Size — размер изображения.

Описание остальных полей массива не имеет значения, так как в работе использованы только перечисленные, другие же считываются и сохраняются без изменений.

InfoHeaderSize — размер *infoHeader*, считывается напрямую из первого поля заголовка.

Прежде, чем переходить к следющему полю, необходимо обозначить структуру *Color*. Данная структура содержит в себе компоненты *r*, *g* и *b*, отвечающие за каналы цвета. Их значения находятся в диапазоне от 0 до 1. С помощью *Color* программа записывает каждый пиксель изображения.

m_colors — массив пикселей изображения.

fpath — путь к открытому программой файлу.

paddingAmount — пэддинг изображения для корректной работы с пикселями.

pixmap — указатель, используемый для создания и изменения представления изображения в специальном классе приложения.

m_width — ширина изображения. Значение можно взять из второго заголовка класса. Переменная создана только для удобного доступа.

m_height — высота изображения. Аналогично *m_width*.

lastRead — индикатор результата последней попытки открытия файла.

1.2. Описание методов ввода/вывода изображения

Read — метод открытия нового файла. Совершает попытку открыть файл по указанному пути. В случае успеха, начинает считывание (при всех неудачных проверках функция завершается и *lastRead* устанавливается в значение *false*). В два буферных массива записываются полностью заголовок файла и частично заголовок изображения. В заголовке файла выполняется проверка на корректность сигнатуры, в заголовке изображения — на глубину изображения в 24 бит. Если проверки пройдены, буферные заголовки копируются в заголовки класса, заголовок изображения дочитывается до конца. Далее вычисляются значения *m_width, m_height u paddingAmount*. После этого, матрица пикселей полностью считывается в *m_colors*, по *pixmap*

создается новое изображение, в *fpath* копируется принятое функцией значение, и *lastRead* устанавливается в значение *true*.

Export — метод сохранения изображения в файл. Имеет две вариации: сохранение в файл с новым именем и сохранение в файл с тем же именем. Совершает попытку открыть файл по указанному пути. В случае успеха, начинает запись заголовков в файл. Затем, в файл записывается матрица пикселей *m_colors* с добавлением паддинга.

1.3. Описание методов изменения изображения

SetRGB — метод изменения компоненты пикселей всего изображения. Проходит по *m_colors* и изменяет в каждом пикселе заданную компоненту на заданное значение.

MajorColorChange — находит доминирующий цвет и заменяет на указанный. Создаёт ассоциативный контейнер, проходит по *m_colors* и добавляет в контейнер новые пиксели, либо, если пиксель там уже есть, увеличивает количество появлений пикселя. Затем в контейнере находится самый частый цвет и заменяется в матрице на указанный.

RectangleDraw — считывает координаты прямоугольника (левый верхний и правый нижний), цвет и толщину линий. В цикле толщины проходит по горизонтальным и вертикальным границам прямоугольника, заполняет их заданным цветом, так же заполняет внутренние соседние к границам ряды пикселей до необходимой толщины.

RectangleFill — считывает координаты прямоугольника (левый верхний и правый нижний) и цвет. В цикле заменяет каждый пиксель области на заданный цвет.

RectangleShuffle — считывает координаты прямоугольника (левый верхний и правый нижний) и флаг. Высчитывает размер половины области как по оси X, так и по оси Y. При флаге 0 проходит по области и меняет местами

диагональные четверти. При флаге 1 меняет местами 2 с 3 и 1 с 4 четверти, затем 1 с 3 четверти, образуя перемещение по часовой стрелке.

Fill — копирует участок в вектор, затем вставляет участок сеткой во всё изображение.

RectangleFillOutside — выполняет функционал, аналогичный *RectangleFill*, но заполняет цветом область ВНЕ прямоугольника.

Split — выполняет деление изображения на N * M частей. Имеет две вариации. Первая добавляет в изображение сетку с заданной толщиной. Вторая создаёт новые файлы для каждой ячейки после деления изображения, именуя их от 1.bmp до (N*M).bmp, и сохраняет их в указанную директорию.

UpdatePixmap — предназначен для оперативного отображения изменений в изображении и создания резервной копии изображения для возможности отмены действия.

2. КЛАСС IMAGEVIEWER

Класс ImageViewer предоставляет виджет окна приложения, предназначенный для отображения изображения и изменений в нём. Наследуется от класса Qlabel, содержит переопределенные методы для работы мышью.

2.1. Описание полей класса

flag — предназначен для определения текущего требуемого функционала класса.

Start — координаты точки начала выбора участка изображения.

End — координаты точки конца выбора участка изображения.

ReleasedAct — связующий триггер для отслеживания отпускания кнопки мыши.

Offset — текущая точка нажатия. Отличается от *Start* инверсией отсчета пикселей по оси Y.

RubberBand — инструмент Qt для выделения области или текста.

2.2. Описание методов слежения за мышью

mousePressEvent — отслеживает нажатие клавиши мыши. Устанавливает координаты *Offset*, при *flag* = 1 создает новый r*ubberBand* и устанавливает координаты начала выделения *start*.

mouseMoveEvent — отслеживает перемещение мыши. При *flag* = 0 проверяет зажатие левой кнопки мыши и перемещает изображение внутри окна приложения. При *flag* = 1 изменяет размеры r*ubberBand om start* до текущих координат мыши.

mouseReleaseEvent — отслеживает отпускание клавиши мыши. При *flag* = 1 убирает r*ubberBand,* устанавливает значение *flag* = 0, устанавливает текущие координаты мыши в end и запускает триггер *ReleasedAct*.

3. КЛАСС MAINWINDOW

Класс MainWindow предоставляет основное окно приложения. Наследуется от QMainWindow, содержит виджет ImageViewer, изображение Image и основной интерфейс пользователя.

3.1. Описание полей класса

image — указатель на объект класса Image, изображение.

scaleFactor — коэффициент текущего уровня приближения/отдаления изображения.

color — содержит в себе цвет, выбранный пользователем в одной из функций. Используется многократно.

imageViewer — указатель на виджет ImageViewer внутри MainWindow.

ReleasedAct — триггер, принимающий значение одноименного триггера в ImageViewer.

help — указатель на окно помощи.

ScrollArea — указатель на виджет вертикального и горизонтального ползунков.

fileMenu, viewMenu, editMenu — указатели на меню верхней панели управления приложением.

В соответствие с каждым действием в меню назначены триггеры.

3.2. Описание методов класса

Open — метод открытия нового файла. Вызывает файловый менеджер для выбора файла, ограничивает его выбор форматом .bmp. Если файл выбран, вызывает метод *Read* класса *Image*. При неудачном чтении или отличии файла от стандарата сообщает об ошибке.

Save — метод сохранения открытого файла. Заменяет первичный файл на измененный.

SaveAs - метод сохранения открытого файла. Сохраняет файл по пользовательскому пути с новым именем.

About — вызывает окно информации о приложении и помощи.

Exit — завершает работу приложения.

FitToWindow — устанавливает размер *ImageViewer* равным либо свободному пространству в *MainWindow* , либо размеру изображения.

ZoomIn, ZoomOut — функции зумирования, вызывают метод *Scale* с коэффициентом 1.25 и 0.75 соответственно.

NormalSize — устанавливает размер *ImageViewer* равным размеру изображения.

SetImage — устанавливает изображение в *ImageViewer*, обновляет доступные пункты меню.

RgbChange — запрашивает компоненту для изменения и и её новое значение, вызывает метод *setRGB* класса *Image*.

Selection — устанавливает *flag* класса *ImageViewer* в 1 для выделения. Повторное нажатие изменяет значение на 0.

Rectangle — предоставляет выбор между рисованием прямоугольника, заполнением его или области вне него цветом, перемещением четвертей области по одному из двух алгоритмов и заполнением всего

изображения сеткой из ячеек скопированной области. Вызывает соответствующие методы класса *Image*.

MajorColorChange — принимает цвет замены от пользователя и вызывает одноименный метод класса *Image*.

Split — принимает от пользователя толщину сетки, количество вертикальных и горизонтальных частей. Если толщина равна 0, принимает от пользователя путь для сохранения и создает новые изображения для каждой ячейки сетки. Иначе, рисует сетку на изображении. Стоит обратить внимание, что размеры изображения не всегда можно разделить нацело на принятое значение, поэтому сетка может иметь остаточные области справа и сверху.

Backup — выполняет откат последнего действия.

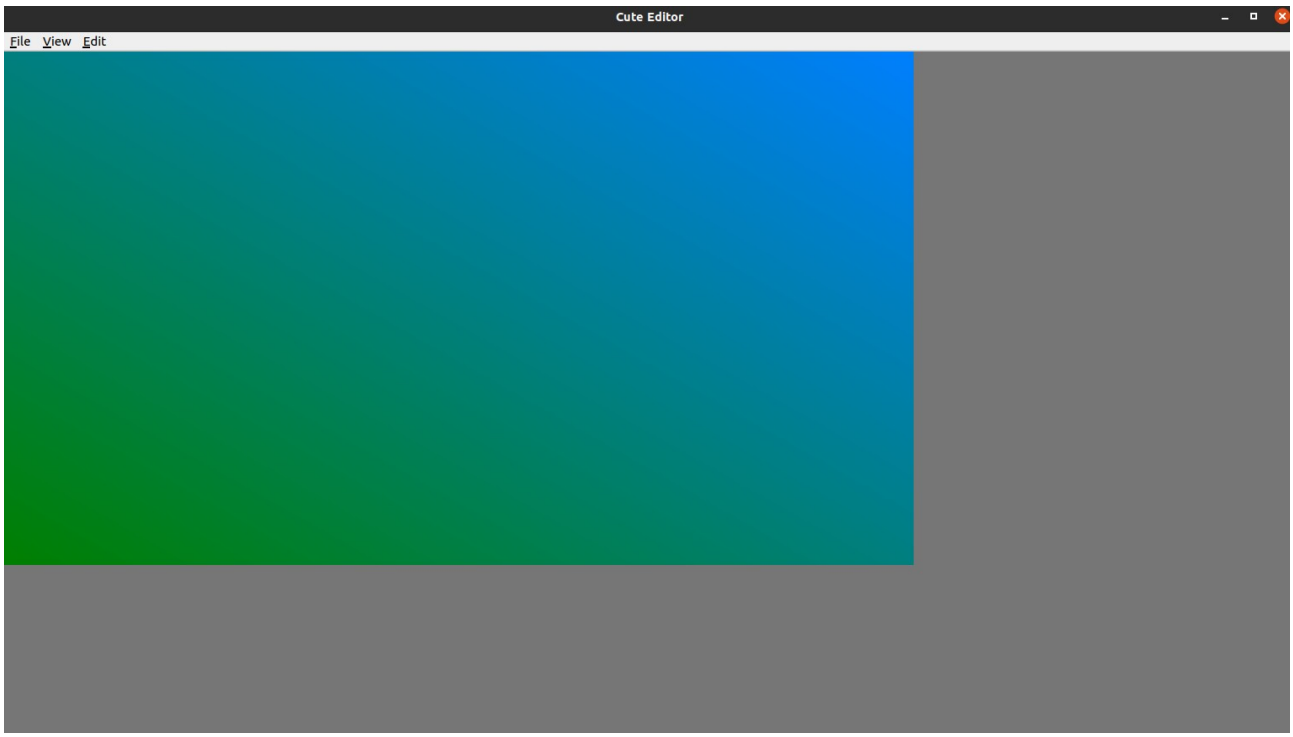
CreateActions, UpdateActions, CreateMenus — предназначены для создания меню и связи действий в нём с методами класса.

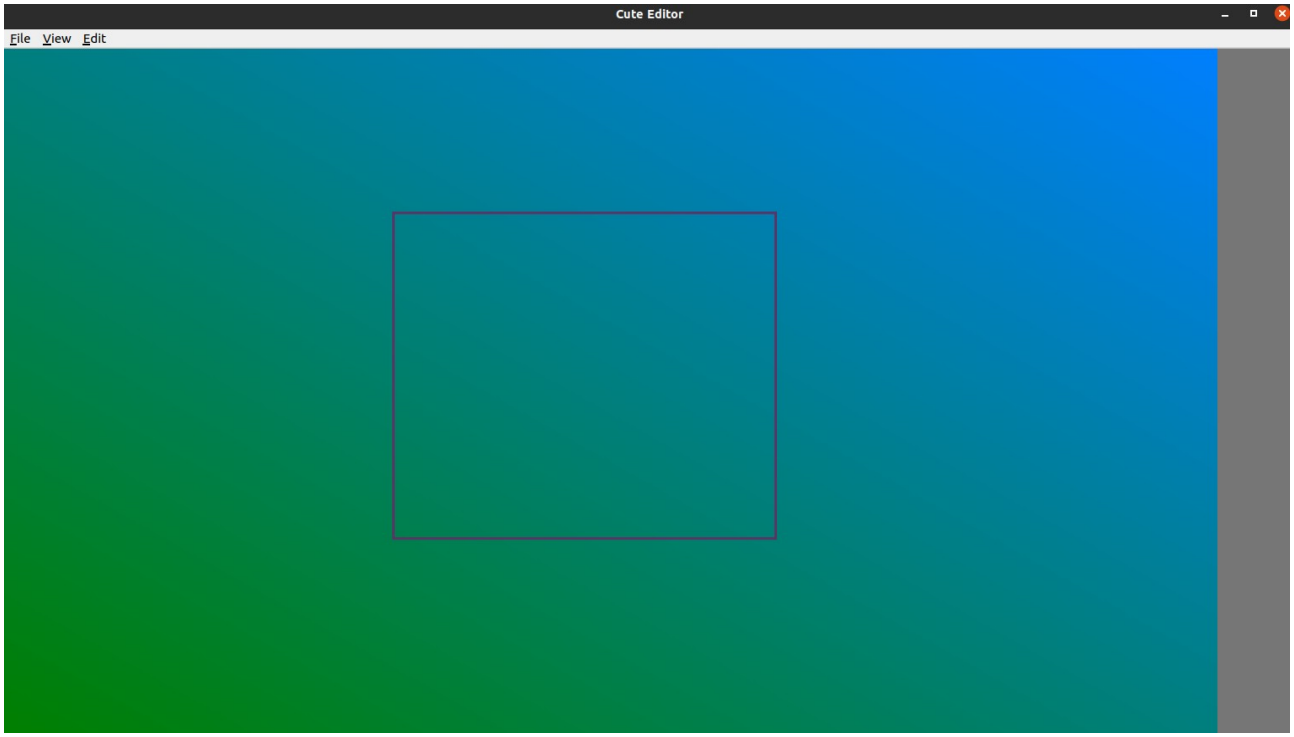
Scale — изменяет размер *ImageViewer.* Максимальное приближение/отдаление - в 4 раза.

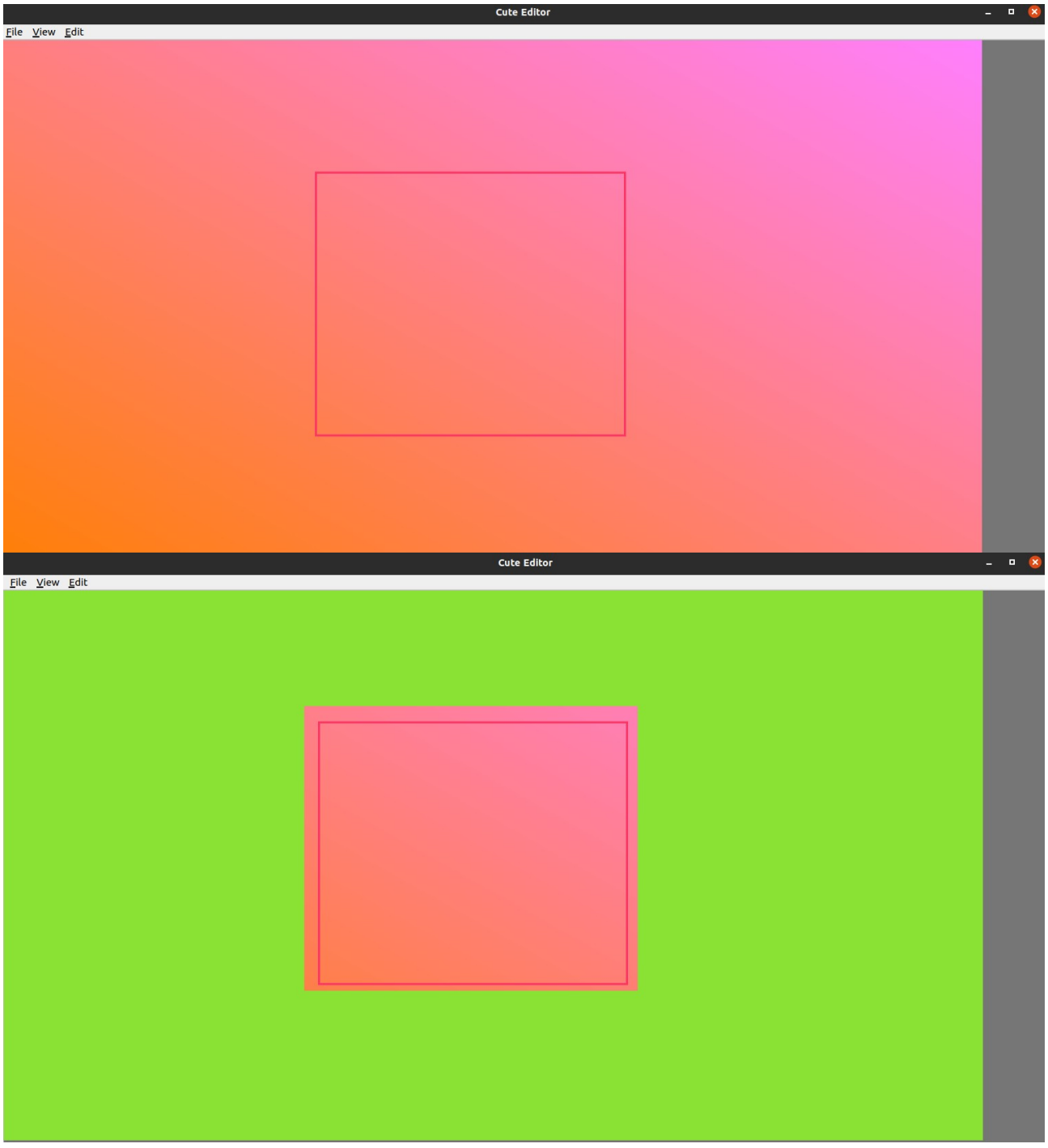
AdjustScrollBar — изменяет размер виджета ползунков в соответсвие с размером *ImageViewer*.

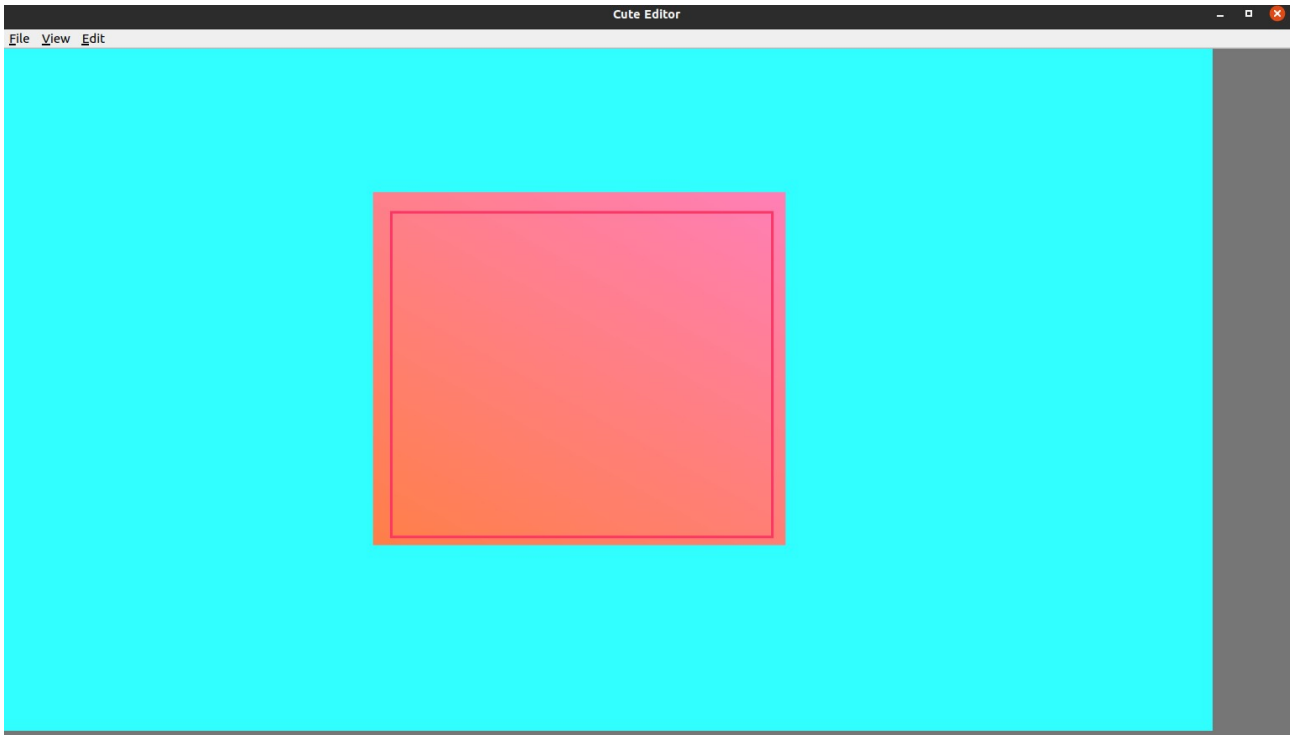
4. ТЕСТИРОВАНИЕ

Рисование квадрата, замена компоненты, заполнение вне области, замена самого частого цвета:

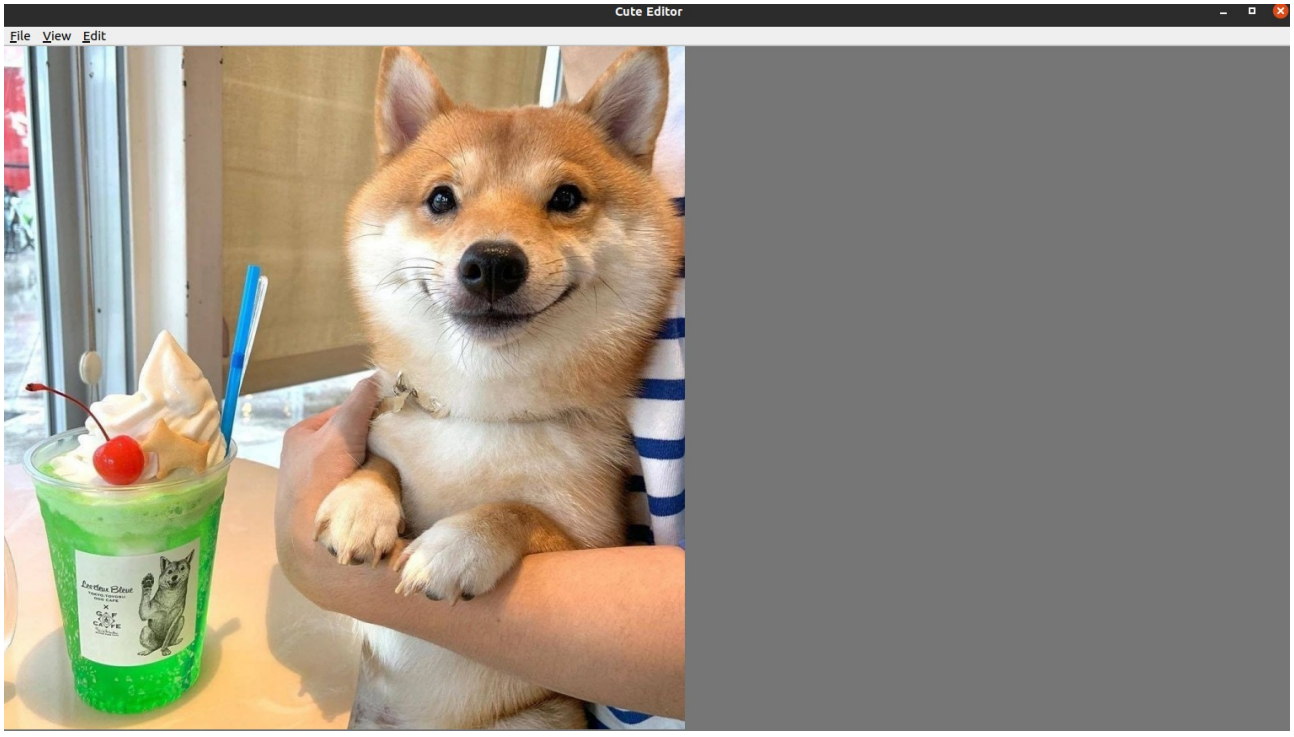


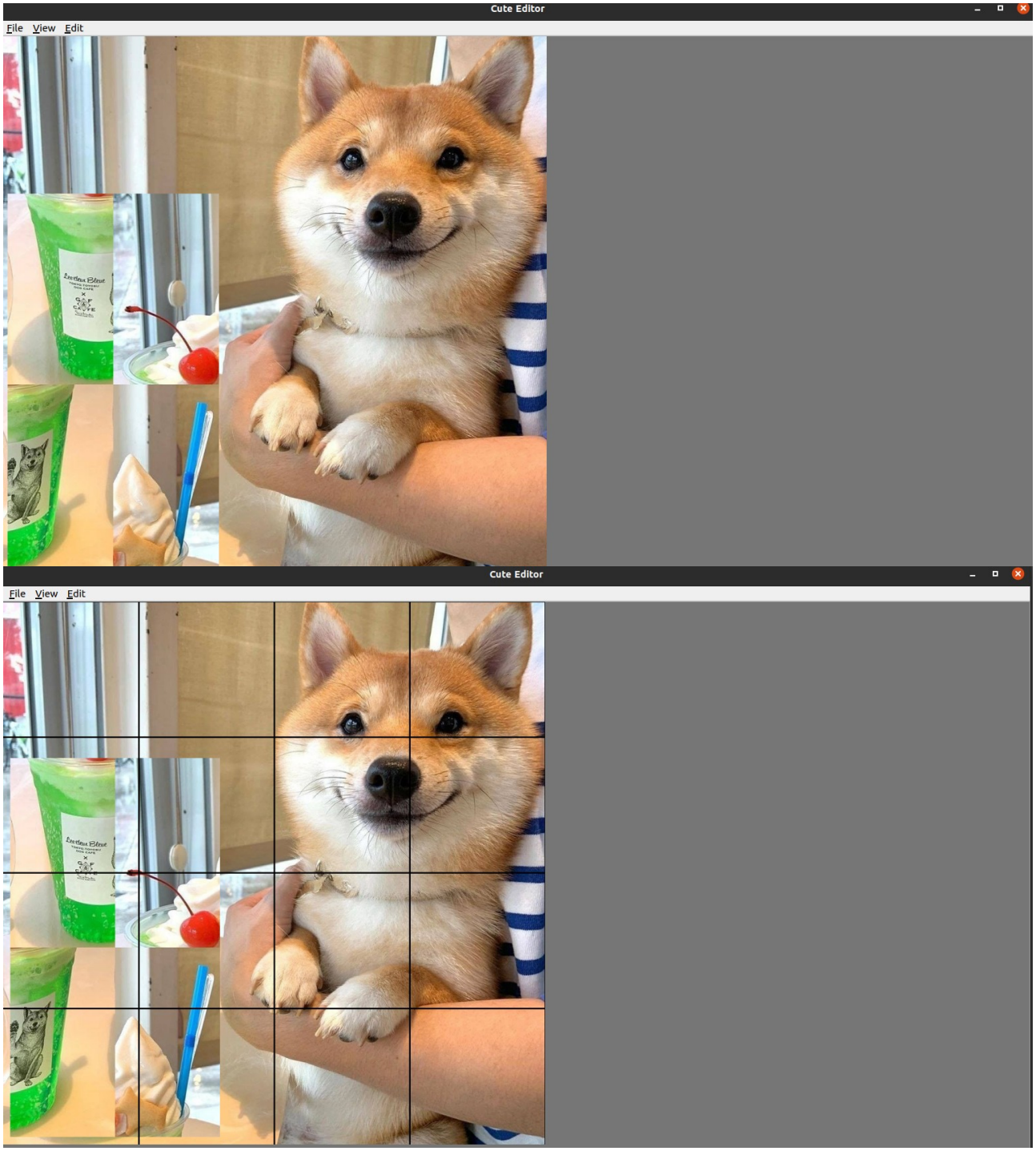


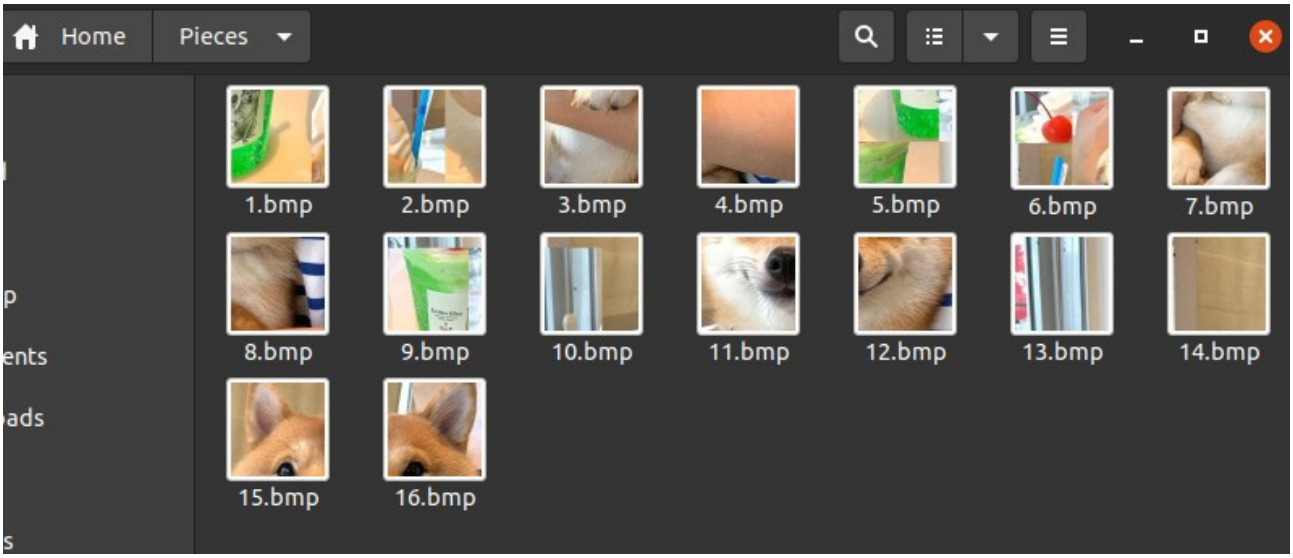

Cute Editor
File View Edit
Cute Editor
File View Edit



Перемещение четвертей, рисование сетки и сохранение ячеек в папку:




Cute Editor
File View Edit
Cute Editor
File View Edit



ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы были получены навыки работы с C++, Qt и Bit Map изображениями. Был написан простой графический редактор, выполняющий функции обработки изображения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Creating a Bitmap Image (.bmp) using C++. https://youtu.be/vqT5j38bWGg
2. Reading a Bitmap Image (.bmp) using C++. https://youtu.be/NcEE5xmpgQ0
3. Qt Documentation. https://doc.qt.io/
4. https://en.cppreference.com/w/