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

Курсовая работа
по дисциплине «Программирование»
Тема: работа с bmp файлами

Студент гр. 8303 _____ Бородкин Ю.В.

Преподаватель _____ Чайка К.В

Санкт-Петербург
2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Бородкин Ю.В.

Группа 8303

Тема работы: работа с bmp файлами

Исходные данные:

Написать программу для обработки изображений форматат bmp , создать собственный класс для загрузки и хранения изображения и реализовать следующие методы: фильтр rgb-компонент, отрисовка квадрата, смена 4 равных частей в прямоугольной области, смена часто встречаемого пикселя на заданный.

Содержание пояснительной записки:

- Содержание
- Введение
- Описание функций
- Тестирование
- Исходный код
- Использованные источники

Предполагаемый объем пояснительной записки:

Не менее 65 страниц.

Дата выдачи задания: 27.02.2019

Дата защиты реферата:

Студент _____ Бородкин Ю.В.

Преподаватель _____ Чайка К. В.

АННОТАЦИЯ

В данной работе была создана программа, для работы с файлами-изображениями формата bmp. Был разработан GUI для загрузки/сохранения файлов, инверсия цвета, преобразования области изображения в чёрно-белое, изменения размера изображения, отрисовки отрезка, отображения полной информации об изображении и файле, а также справкой, поясняющей работу программы. Был представлен исходный код и предоставлено тестирование программы.

SUMMARY

In this paper, a program was created to work with image files in bmp format. A GUI was developed for loading / saving files, inverting color, converting an image area into black and white, resizing an image, drawing a segment, displaying full information about the image and file, as well as help explaining the operation of the program. The source code was submitted and the testing program provided.

.

Оглавление

Введение.

Приложение написано на языке C++ с использованием фреймворка Qt. Для работы с с файлом bmp был создан класс Image, алгоритмы для обработки bmp-изображения были написаны на базе использования средств Qt.

Цель и условие работы.

Вариант 3

Общие сведения

- 24 бита на цвет
- без сжатия
- файл всегда соответствует формату BMP (но стоит помнить, что версий у формата несколько)
- обратите внимание на выравнивание; мусорные данные, если их необходимо дописать в файл для выравнивания, должны быть нулями.
- обратите внимание на порядок записи пикселей
- все поля стандартных BMP заголовков в выходном файле должны иметь те же значения что и во входном (разумеется кроме тех, которые должны быть изменены).

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

1. **Фильтр rgb-компонент.** Этот инструмент должен позволять для всего изображения либо установить в 0 либо установить в 255 значение заданной компоненты. Функционал определяется
    - Какую компоненту требуется изменить
    - В какой значение ее требуется изменить
2. **Рисование квадрата.** Квадрат определяется:
    - Координатами левого верхнего угла
    - Размером стороны
    - Толщиной линий
    - Цветом линий
    - Может быть залит или нет
    - Цветом которым он залит, если пользователем выбран залитый
3. **Поменять местами 4 куска области.** Выбранная пользователем прямоугольная область делится на 4 части и эти части меняются местами. Функционал определяется:
    - Координатами левого верхнего угла области
    - Координатами правого нижнего угла области
    - Способом обмена частей: “по кругу”, по диагонали

4. Находит самый часто встречаемый цвет и заменяет его на другой заданный цвет.

Функционал определяется

- Цветом, в который надо перекрасить самый часто встречаемый цвет

1. Функция Main().

В функции main() создается экземпляр класса MainWindow. Далее запускается метод exec().

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

2. Функции класса MainWindow

2.1 Конструктор класса MainWindow()

MainWindow::MainWindow(QWidget *parent);

В конструкторе класса MainWindow инициализируются указатели на экземпляры классов Image, MyGraphicView, RGB_window, Reference, Info. Также выводится в status bar сообщение о черном цвете и обрабатывается сигнал selection() экземпляра класса MyGraphicView и запускается слот selection().

```
MainWindow::MainWindow(QWidget *parent) :
QMainWindow(parent),
ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    img = new Image();
    picture = new MyGraphicView();
    reference = new Reference();
    info = new Info();
    rgb_window = new RGB_window();
    statusBar()->showMessage("rgb(0; 0; 0)");
    bool succ = connect(picture, SIGNAL(selection()), this, SLOT(selection()));
    Q_ASSERT(succ);
}
```

2.2 Слот обработки загрузки изображения on_open_triggered()

void MainWindow::on_open_triggered()

В слоте вызывается функция QFileDialog::getOpenFileName(), которая отображает окно выбора пути для загрузки изображения. После этого запускается функция loadImage() класса Image, которая возвращает номер ошибки. Если в ходе выполнения функции loadImage() возникают ошибки, то запускается функция QMessageBox::critical(), которая выводит окно с ошибкой. Если ошибку в ходе выполнения функций не обнаружилось, то происходит дальнейшая проверка на версию bmp файла, в случае успешной проверки - изображение функцией update() класса MyGraphicView загружается на сцену и функцией addWidget() добавляется к виджету. В конце работы функция заполняет структуру information класса Info сведениями об изображении.

```
 void MainWindow::on_open_triggered()
{
    QString str = QFileDialog::getOpenFileName(nullptr, "Выберите файл для открытия",
"/home/user", "*.bmp");

    if (str == nullptr) return;

    button_pressed = 0;
    picture->button_pressed = 0;
    switch (img->loadImage(str.toLocal8Bit().constData())) {
        case ALL_OK:
            picture->height = img->bih.height;
            picture->width = img->bih.width;
            info->info.name = QFileInfo(str).baseName();
            info->info.path = QFileInfo(str).path() + "/" + info->info.name + ".bmp";
            switch(img->bih.size){
                case CORE:
                    img->bih.height = 0;
                    img->bih.width = 0;
                    QMessageBox::critical(this, "Ошибка", "Версия CORE bmp изображения не
поддерживается.");
                    return;
                case V3:
                    info->info.version = 3;
                    break;
                case V4:
                    info->info.version = 4;
                    break;
                case V5:
                    info->info.version = 5;
                    break;
                default:
                    img->bih.height = 0;
                    img->bih.width = 0;
                    QMessageBox::critical(this, "Ошибка", "Некорректный заголовок у bmp файла");
                    return;
            }
            info->info.size = img->bfh.size;
```

```
            info->info.width = img->bih.width;
            info->info.height = img->bih.height;
            info->info.isReadable = QFileInfo(str).isReadable();
            info->info.isWriteable = QFileInfo(str).isWritable();
            picture->update(img->getPixmap());
            ui->gridLayout->addWidget(picture);
            return;
        case LOAD_ERR:
            img->bih.height = 0;
            img->bih.width = 0;
            QMessageBox::critical(this, "Ошибка", "Возникла ошибка при открытии файла");
            return;
        case BIG_IMG:
            img->bih.height = 0;
            img->bih.width = 0;
            QMessageBox::critical(this, "Ошибка", "Размер изображения не должен превышать
10000x10000 пикселей");
            return;
    }
}
```

2.3 Слот обработки сохранения изображения on_save_triggered()

void MainWindow::on_save_triggered()

В слоте вызывается функция QFileDialog::getSaveFileName(), которая отображает окно выбора пути для сохранения изображения. После этого запускается функция saveImage() класса Image, которая возвращает номер ошибки. Если во время сохранения изображения происходит ошибка, то запускается функция QMessageBox::critical(), которая выводит окно с ошибкой.

```
void MainWindow::on_save_triggered()
{
    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Невозможно сохранить пустое изображение");
        return;
    }
    button_pressed = 0;
    picture->button_pressed = 0;

    QString str = QFileDialog::getSaveFileName(nullptr, "Выберите папку для сохранения",
"/home/user", "*.bmp");

    if (str == nullptr) return;

    switch (img->saveImage((str + ".bmp").toLocal8Bit().constData())) {
        case ALL_OK:
            return;
        case SAVE_ERR:
            QMessageBox::critical(this, "Ошибка", "Возникла ошибка при сохранении файла");
            return;
    }
}
```

2.4 Слот обработки изменения размера изображения on_rgbcomp_clicked()

void MainWindow::on_rgbcomp_clicked()

В слоте вызывается метод exec у экземпляра класса RGB_window, в ходе вызова метода появляется окно с выбором значения компонента и самого компонента, получив значения, передаём в функцию rgb_comp экземпляра класса Image, после обновляем сцену с помощью метода update класса MyGraphicView и помещается на виджет функцией addWidget.

```
void MainWindow::on_rgbcomp_clicked() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }

    button_pressed = 0;
    picture->button_pressed = 0;

    rgb_window->exec();

    if (rgb_window->_close) return;
    rgb_window->_close = 1;

    int comp = rgb_window->comp();
    int value = rgb_window->get_value();

    img->rgb_comp(comp, value);
    picture->update(img->getPixmap());
}
```

2.5 Слот selection()

void MainWindow::selection();

Если во время вызова слота была нажата кнопка вызова пипетки, получаем координаты относительно положения сцены с помощью перегруженного метода mousePressEvent. После этого получаем по координатам значение пикселя и выводим в statusBar. Если же при вызове слота были нажаты кнопки отрисовки квадрата или смены 4 частей в выделенной области, то вызываются функции drawSquare() или chParts() класса Image соответственно. После чего сцена класса MyGraphicView также обновляется функцией update() и помещается на виджет функцией addWidget().
Если во время выполнения слота возникает ошибка, связанная с некорректно введенными координатами, то выполняется функция QMessageBox::critical(), которая выводит в окне пользователю характер ошибки.

```
void MainWindow::selection() {
   switch (button_pressed) {

      case PIPETTE: {
         QPoint coord(picture->coord.x, picture->coord.y);
         if (coord.x() < 0 || coord.y() >= img->bih.height || coord.y() < 0 || coord.x() >= img-
>bih.width) {
            QMessageBox::critical(this, "Ошибка", "Точка находится вне изображения");
            return;
         }
         int y = img->bih.height - 1 - coord.y();
         int x = coord.x();
         statusBar()->showMessage("rgb(" + QString::number(img->rgb[y][x].red) + "; " +
QString::number(img->rgb[y][x].green) + "; " + QString::number(img->rgb[y][x].blue) + ")");
         return;
      }
      case SQUARE: {
         if (picture->coord.x < 0 || picture->coord.y < 0 || picture->coord.x >= img->bih.width  ||
picture->coord.y >= img->bih.height || picture->c_end.x < 0 || picture->c_end.y < 0 || picture-
>c_end.x >= img->bih.width  || picture->c_end.y >= img->bih.height){
            QMessageBox::critical(this, "Ошибка", "Выделенная область выходит за пределы
рисунка, попробуйте снова");
            button_pressed = 0;
            return;
         }
         int thick = ui->thick->value();
         if (picture->coord.x - thick + 1 < 0 || picture->coord.y - thick + 1 < 0 || picture->coord.x +
thick - 1 > img->bih.width || picture->coord.y + thick - 1 > img->bih.height || picture->c_end.x - thick
+ 1 < 0 || picture->c_end.y - thick + 1 < 0 || picture->c_end.x + thick - 1 > img->bih.width || picture-
>c_end.y + thick - 1 > img->bih.height) {
            QMessageBox::critical(this, "Ошибка", "Выделенная область выходит за пределы
рисунка, попробуйте снова");
            button_pressed = 0;
            return;
         }
         QPoint point1(picture->coord.x, picture->coord.y);
         QPoint point2(picture->c_end.x, picture->c_end.y);
         img->drawSquare(point1.x() > point2.x() ? point2.x() : point1.x(), point1.y() > point2.y() ?
point2.y() : point1.y(), abs(point2.x() - point1.x()), thick, color, check, fcolor);
         picture->update(img->getPixmap());
         check = false;
         button_pressed = 0;
         return;
      }
      case CHPARTS: {
         if (picture->coord.x < 0 || picture->coord.y < 0 || picture->coord.x >= img->bih.width  ||
picture->coord.y >= img->bih.height || picture->c_end.x < 0 || picture->c_end.y < 0 || picture-
>c_end.x >= img->bih.width  || picture->c_end.y >= img->bih.height){
            QMessageBox::critical(this, "Ошибка", "Выделенная область выходит за пределы
рисунка, попробуйте снова");
            button_pressed = 0;
            return;
         }
         QPoint point1(picture->coord.x, picture->coord.y);
         QPoint point2(picture->c_end.x, picture->c_end.y);
         int x1 = point1.x() > point2.x() ? point2.x() : point1.x();
         int y1 = point1.y() > point2.y() ? point2.y() : point1.y();
```

```
            int x2 = point1.x() < point2.x() ? point2.x() : point1.x();
            int y2 = point1.y() < point2.y() ? point2.y() : point1.y();
            img->chParts(x1, y1, x2, y2, ui->diagonally->isChecked());
            picture->update(img->getPixmap());
            button_pressed = 0;
            return;
        }
    }
}
```

2.6 Слот обработки нажатия на кнопку пипетки on_pipette_clicked()

void MainWindow::on_pipette_clicked();

Если на момент нажатия на кнопку изображение не было загружено, то выполняется функция QMessageBox::critical(), которая выводит пользователю сообщение об ошибке. Далее переменным button_pressed классов MainWindow и MyGraphicView присваивается значение макроса PIPETTE.

```
void MainWindow::on_pipette_clicked() {
    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }
    button_pressed = PIPETTE;
    picture->button_pressed = PIPETTE;
}
```

2.7 Слот обработки нажатия на кнопку отрисовки квадрата on_draw_Square_clicked()

void MainWindow::on_draw_Square_clicked();

Если на момент нажатия на кнопку изображение не было загружено, то выполняется функция QMessageBox::critical(), которая выводит пользователю сообщение об ошибке. Далее идет проверка на то, необходима ли заливка изображения и вызывается функция QColorDialog::getColor, для получения цвета контура и заливки, затем переменным button_pressed классов MainWindow и MyGraphicView присваивается значение макроса SQUARE.

```
void MainWindow::on_draw_Square_clicked() {
    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }
```

```
    button_pressed = 0;
    picture->button_pressed = 0;
    color = QColorDialog::getColor(Qt::white, this, "Выберите цвет контура");
    if (!color.isValid()) return;
    check = ui->isFill->isChecked();
    if (check) {
        fcolor = QColorDialog::getColor(Qt::white, this, "Выберите цвет заливки");
        if (!fcolor.isValid()) return;
    }
    picture->color = color;
    picture->thick = ui->thick->value();
    button_pressed = SQUARE;
    picture->button_pressed = SQUARE;
}
```

2.8 Слот обработки нажатия на кнопку смены наиболее встречаемого пикселя on_ch_of_pix_clicked()

void MainWindow::on_ch_of_pix_clicked();

Если на момент нажатия на кнопку изображение не было загружено, то выполняется функция QMessageBox::critical(), которая выводит пользователю сообщение об ошибке. Далее с помощью функции QColorDialog::getColor() переменной color присваивается значение цвета, выбранного пользователем в окне. Далее запускается метод экземпляра img класса Image edit_often_color и обновляется сцена с помощью метода update.

```
void MainWindow::on_ch_of_pix_clicked() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }

    button_pressed = 0;
    picture->button_pressed = 0;

    color = QColorDialog::getColor(Qt::white, this, "Выберите цвет пикселя");
    if (!color.isValid()) return;

    img->edit_often_color(color);
    picture->update(img->getPixmap());
}
```

2.9 Слот обработки вывода информации on_info_triggered()

void MainWindow::on_info_triggered()

Если на момент нажатия на кнопку изображение не было загружено, то выполняется функция QMessageBox::critical(), которая выводит пользователю сообщение об ошибке.После этого вызывается функция setInfo класса Info.

```
void MainWindow::on_info_triggered() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Чтобы просмотреть информацию об изображении
- загрузите его");
        return;
    }

    button_pressed = 0;
    picture->button_pressed = 0;
    info->setInfo();
    info->exec();
}
```

2.10 Слот обработки нажатия на кнопку смены частей в заданной области on_chParts_clicked()

void MainWindow::on_chParts_clicked();

Если на момент нажатия на кнопку изображение не было загружено, то выполняется функция QMessageBox::critical(), которая выводит пользователю сообщение об ошибке, затем переменным button_pressed классов MainWindow и MyGraphicView присваивается значение макроса CHPARTS.

```
void MainWindow::on_chParts_clicked() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }

    button_pressed = CHPARTS;
    picture->button_pressed = CHPARTS;
}
```

2.11 Слот обработки вывода справки void on_reference_triggered()

void MainWindow::on_reference_triggered();

Если на момент нажатия на кнопку изображение не было загружено, то выполняется функция QMessageBox::critical(), которая выводит пользователю сообщение об ошибке, затем переменным button_pressed классов MainWindow и MyGraphicView присваивается значение макроса CHPARTS.

```
void MainWindow::on_reference_triggered() {
    button_pressed = 0;
    picture->button_pressed = 0;
    reference->show();
```

```
}
```

3. Функции класса Image

3.1 Функция загрузки изображения loadImage()

```
int Image::loadImage(const char *file);
```

Функция принимает указатель на строку, являющуюся путем к файлу, который нужно открыть. Функцией fopen() открывает файл для считывания, а функцией fread() осуществляется считывание структур BITMAPFILEHEADER и BITMAPINFO. Далее под массив rgb выделяется память функцией new RGB*[]. С помощью вложенного цикла for и функции fread() считывается каждый пиксель изображения и помещается в массив rgb. Возвращаемым значением является -1 или -2, в случае возникновения ошибки при считывании или 0, если ошибок не было.

```
int Image::loadImage(const char *file) {
    FILE* f = fopen(file, "rb");
    if (!f)
        return -1;
    fread(&bfh, sizeof(bfh), 1, f);
    fread(&bih, sizeof(bih), 1, f);
    if (bih.height > 10000 || bih.width > 10000)
        return -2;
    size_t padding = 0;
    if ((bih.width*3) % 4)
        padding = 4 - (bih.width*3) % 4;
    rgb = new RGB* [bih.height];
    for (int i = 0; i < bih.height; i++){
        rgb[i] = new RGB[bih.width + 1];
    }
    fseek(f, long(bfh.bfOffBits), SEEK_SET);
    for (int i = 0; i <  bih.height; i++) {
        int j;
        for (j = 0;  j < bih.width; j++){
            fread(&rgb[i][j], sizeof(RGB), 1, f);
        }
        if (padding)
            fread(&rgb[i][j], padding, 1, f);
    }
    fclose(f);
    return 0;
}
```

3.2 Функция сохранения изображения saveImage()

```
int Image::saveImage(const char* file);
```

Функция принимает указатель на строку, являющуюся путем к файлу для сохранения. Функцией fopen() открывает файл для записи, а функцией fwrite() осуществляется запись в файл структур BITMAPFILEHEADER и BITMAPINFO. С помощью вложенного цикла for и функции fwrite() каждый пиксель изображения из массива rgb записывается в файл. Возвращаемым значением является -1, в случае возникновения ошибки при считывании или 0, если ошибок не было.

```
int Image::saveImage(const char* file) {
    FILE* f = fopen(file, "wb");
    if (!f)
        return -1;
    fwrite(&bfh, sizeof(bfh), 1, f);
    fwrite(&bih, sizeof(bih), 1, f);
    size_t padding = 0;
    if ((bih.width*3) % 4)
        padding = 4 - (bih.width*3) % 4;
    fseek(f, long(bfh.bfOffBits), SEEK_SET);
    for (int i = 0; i < bih.height; i++) {
        int j;
        for (j = 0; j < bih.width; j++){
            fwrite(&rgb[i][j], sizeof(RGB), 1, f);
        }
        if (padding)
            fwrite(&rgb[i][j], padding, 1, f);
    }
    fclose(f);
    return 0;
}
```

3.3 Функция получения пиксельной карты getPixmap()

QPixmap Image::getPixmap();

Создается указатель на экземпляр класса QImage *image и объявляется переменная pixel типа QColor. С помощью вложенного цикла for определяются цветовые компоненты пикселя pixel и функцией setPixel() устанавливаются в переменную *image. Далее функция getPixmap() возвращает пиксельную карту изображения, используя функцию fromImage().

```
 QPixmap Image::getPixmap(){
      QImage *image = new QImage(bih.biWidth, bih.biHeight, QImage::Format_RGB16);
      QColor pixel;
       for (int i = bih.biHeight - 1; i >= 0; i--) {
              for (int j = 0; j < bih.biWidth; j++) {
              pixel.setRed(rgb[i][j].red);
              pixel.setGreen(rgb[i][j].green);
              pixel.setBlue(rgb[i][j].blue);
              image->setPixel(j, bih.biHeight - i - 1, pixel.rgb());
              }
       }
   return QPixmap::fromImage(*image);
}
```

3.4 Функция rgb-компонента rgb_comp()

int Image::rgb_comp(int comp, int value);

Функция принимает значение компоненты, которую необходимо изменить, и саму компоненту. Затем устанавливает значение данной компоненты в 0 или 255 для всего изображения.

```
int Image::rgb_comp(int comp, int value) {

  if (comp == R) {
     for (int i = 0; i <  bih.height; i++)
        for (int j = 0;  j < bih.width; j++)
           rgb[i][j].red = uchar(value);
  } else if (comp == G) {
     for (int i = 0; i <  bih.height; i++)
        for (int j = 0;  j < bih.width; j++)
           rgb[i][j].green = uchar(value);
  } else if (comp == B) {
         for (int i = 0; i <  bih.height; i++)
            for (int j = 0;  j < bih.width; j++)
               rgb[i][j].blue = uchar(value);
  }

  return 0;
}
```

3.5 Функция отрисовки квадрата drawSquare()

int Image::drawSquare(int x, int y, int len, int thick, QColor color, bool fill, QColor fcolor);

Функция принимает координаты левого верхнего угла, толщину линий, цвет, флаг на заливку и цвет заливки. С помощью вложенных циклов for отрисовываются сначала горизонтальные линии, затем вертикальные. После проверяется: установлен ли флаг на заливку, если да, то квадрат заливается цветом fcolor.

```
int Image::drawSquare(int x, int y, int len, int thick, QColor color, bool fill, QColor fcolor) {

      for (int i = x - thick + 1; i <= x + len + thick - 1; i++)
         for (int j = -thick + 1; j <= 0; j++) {
            rgb[bih.height - 1 - y - j][i].red = uchar(color.red());
            rgb[bih.height - 1 - y - j][i].green = uchar(color.green());
            rgb[bih.height - 1 - y - j][i].blue = uchar(color.blue());
                                                    /* horizontal lines */
            rgb[bih.height - 1 - y - len + j][i].red = uchar(color.red());
            rgb[bih.height - 1 - y - len + j][i].green = uchar(color.green());
            rgb[bih.height - 1 - y - len + j][i].blue = uchar(color.blue());
         }
```

```
    for (int j = bih.height - 1 - y + thick - 1; j >= bih.height - 1 - y - len - thick + 1; j--)
        for (int i = -thick + 1; i <= 0; i++) {
            rgb[j][x + i].red = uchar(color.red());
            rgb[j][x + i].green = uchar(color.green());
            rgb[j][x + i].blue = uchar(color.blue());
                                                    /* vertical lines */
            rgb[j][x + len - i].red = uchar(color.red());
            rgb[j][x + len - i].green = uchar(color.green());
            rgb[j][x + len - i].blue = uchar(color.blue());
        }

    /* fill */

    if (fill) {
        for (int j = bih.height - y - 2; j >= bih.height - y - len; j--)
            for (int i = x + 1; i < x + len; i++) {
                rgb[j][i].red = uchar(fcolor.red());
                rgb[j][i].green = uchar(fcolor.green());
                rgb[j][i].blue = uchar(fcolor.blue());
            }
    }
    return 0;
}
```

3.6 Функция замены заданного цвета на другой replace()

int Image::replace(QColor rgb1, QColor rgb2);

Функция цвет который требуется заменить и цвет, на который требуется заменить. С помощью вложенных циклов проходим массив пикселей и каждый пиксель проверяем на совпадение каждой из компонент RGB с цветом rgb1. В случае выполнения условия заменяем компоненты того пикселя на компоненты rgb2.

```
int Image::replace(QColor rgb1, QColor rgb2) {

  for (int i = 0; i <  bih.height; i++)
    for (int j = 0;  j < bih.width; j++)
      if (rgb[i][j].red == rgb1.red() && rgb[i][j].blue == rgb1.blue() && rgb[i][j].green == rgb1.green()) {
          rgb[i][j].red = uchar(rgb2.red());
          rgb[i][j].blue = uchar(rgb2.blue());
          rgb[i][j].green = uchar(rgb2.green());
       }

    return 0;
}
```

3.7 Функция замены часто встречаемого пикселя edit_often_color()

int Image::edit_often_color(QColor color);

Функция принимает цвет, на который необходимо заменить часто встречаемый пиксель. В ходе выполнения задания был создан словарь, ключом которого является цвет, а значение, количество данных пикселей в изображении. После заполнения словаря с помощью функции find_max находит наиболее часто встречаемый цвет и заменяет его с помощьью метода класса Imgae replace в изображении.

```
int Image::edit_often_color(QColor color) {

    long size = 1000000;
    long pos = 1;
    QColor temp(rgb[0][0].red, rgb[0][0].green, rgb[0][0].blue);
    dict* d = new dict[size];

    add(d, temp);

    for (int i = 0; i <  bih.height; i++)
       for (int j = 1;  j < bih.width; j++) {

          temp.setRgb(rgb[i][j].red, rgb[i][j].green, rgb[i][j].blue);
          long check = in(d, temp, pos);

          if (pos == size) {
             size *= 2;
             dict* tmp = d;
             d = new dict[size];
             memcpy(d, tmp, ulong(pos) * sizeof(dict));
             delete [] tmp;
          }

          if (check == -1) {
             add(d + pos++, temp);
          } else {
             d[check].amount++;
          }
       }

    QColor of_pix = find_max(d, pos);

    Image::replace(of_pix, color);

    delete [] d;

    return 0;
}
```

3.8 Функция смены частей в заданной области chParts()

int Image::chParts(int x1, int y1, int x2, int y2, bool isDiagonally);

Функция принимает координаты прямоугольной области и флаг isDiaginally, который определяет каким образом будем двигать части в области, в случае

значения true флага, будет произведена смена частей по диагонали, иначе по часовой стрелке.

```
int Image::chParts(int x1, int y1, int x2, int y2, bool isDiagonally) {

    if (isDiagonally) {
        for (int i = y1; i < y1 + (y2 - y1) / 2; i++)
            for (int j = x1; j < x1 + (x2 - x1) / 2; j++) {
                swap(rgb[bih.height - i - 1][j].red, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 - x1) / 2].red);
                swap(rgb[bih.height - i - 1][j].green, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 - x1) /
2].green);
                swap(rgb[bih.height - i - 1][j].blue, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 - x1) / 2].blue);

                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].red, rgb[bih.height - i - 1][j + (x2 - x1) / 2].red);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].green, rgb[bih.height - i - 1][j + (x2 - x1) /
2].green);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].blue, rgb[bih.height - i - 1][j + (x2 - x1) / 2].blue);
            }

    } else {
        for (int i = y1; i < y1 + (y2 - y1) / 2; i++)
            for (int j = x1; j < x1 + (x2 - x1) / 2; j++) {
                swap(rgb[bih.height - i - 1][j].red, rgb[bih.height - i - 1][j + (x2 - x1) / 2].red);
                swap(rgb[bih.height - i - 1][j].green, rgb[bih.height - i - 1][j + (x2 - x1) / 2].green);
                swap(rgb[bih.height - i - 1][j].blue, rgb[bih.height - i - 1][j + (x2 - x1) / 2].blue);

                swap(rgb[bih.height - i - 1][j].red, rgb[bih.height - i - 1 - (y2 - y1) / 2][j].red);
                swap(rgb[bih.height - i - 1][j].green, rgb[bih.height - i - 1 - (y2 - y1) / 2][j].green);
                swap(rgb[bih.height - i - 1][j].blue, rgb[bih.height - i - 1 - (y2 - y1) / 2][j].blue);

                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].red, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 -
x1) / 2].red);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].green, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2
- x1) / 2].green);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].blue, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 -
x1) / 2].blue);
            }
    }

    return 0;
}
```

4. Функции класса MyGraphicView

4.1 Конструктор класса MyGraphicView()

MyGraphicView::MyGraphicView(QWidget* parent):QGraphicsView (parent);

В конструкторе класса MainWindoe устанавливается положение сцены в верхний левый угол и инициализируется группа group класса QGraphicsItemGroup.

```
MyGraphicView::MyGraphicView(QWidget* parent):QGraphicsView (parent)
{
```

```
    this->setAlignment(Qt::AlignLeft | Qt::AlignTop);
      group = new QGraphicsItemGroup();

}
```

4.2 Слот нажатия на кнопку мыши mousePressEvent(QMouseEvent* event)

void MyGraphicView::mousePressEvent(QMouseEvent* event);

В слоте устанавливаются координаты мыши относительно начала сцены, получаются координаты мыши при ее нажатии и записываются в структуру coord. Переменной mouse_press присваивается значение true, в случае если не нажали на кнопку пипетки.

```
void MyGraphicView::mousePressEvent(QMouseEvent* event) {
    QPoint pos = mapToScene(event->pos()).toPoint();
    coord.x = pos.x();
    coord.y = pos.y();
    if (button_pressed != PIPETTE)
        mouse_press = true;
}
```

4.3 Слот прекращения нажатия кнопки мыши mouseReleaseEvent(QMouseEvent* event)

void MyGraphicView::mouseReleaseEvent(QMouseEvent* event);

В слоте устанавливаются координаты мыши относительно начала сцены, получаются координаты мыши при ее прекращении нажатия и записываются в структуру c_end, в случае если не нажали на кнопку “нарисовать квадрат”. Переменной mouse_pressed присваивается значение false. Со сцены удаляются все группы deleteItemsFromGroup() и испускается сигнал selection().

```
void MyGraphicView::mouseReleaseEvent(QMouseEvent* event) {
    if (mouse_press && button_pressed != SQUARE) {
        QPoint pos = mapToScene(event->pos()).toPoint();
        c_end.x = pos.x();
        c_end.y = pos.y();
        if (c_end.x < 0)
            c_end.x = 0;
        if (c_end.y < 0)
            c_end.y = 0;
    }
    mouse_press = false;
    this->deleteItemsFromGroup(group);

    if (button_pressed != PIPETTE) button_pressed = 0;

    emit selection();
}
```

4.4 Слот движения мыши mouseMoveEvent(QMouseEvent *event)

void MyGraphicView::mouseMoveEvent(QMouseEvent *event);

В слоте устанавливаются координаты мыши относительно начала сцены. Если была нажата кнопка отрисовки квадрата, то получаются координаты мыши при ее перемещении относительно нажатия кнопки, удаляются все группы элементов. Создается новая группа с набором параллельных отрезков и добавляется на сцену. В случае, если нажата кнопка смены частей в заданной области: создается новая группа, на которой отрисовывается прямоугольник и добавляется на сцену.

```
void MyGraphicView::mouseMoveEvent(QMouseEvent *event) {
    if (mouse_press && button_pressed == SQUARE) {
        QPoint pos = mapToScene(event->pos()).toPoint();
        int x = pos.x();
        int y = pos.y();
        this->deleteItemsFromGroup(group);
        group = new QGraphicsItemGroup();
        QPen penColor(color);
        if (x < 0)
            x = 0;
        y = ((y >= coord.y) ? coord.y + abs(x - coord.x) : coord.y - abs(x - coord.x));
        if (y < 0) {
            y = 0;
        }
        if (y == 0) {
            x = (x >= coord.x) ? coord.x + coord.y : coord.x - coord.y;
        }
        group->addToGroup(scene->addLine(coord.x, coord.y, x, coord.y, penColor));
        group->addToGroup(scene->addLine(x, coord.y, x, y, penColor));
        group->addToGroup(scene->addLine(x, y, coord.x, y, penColor));
        group->addToGroup(scene->addLine(coord.x, y, coord.x, coord.y, penColor));
        scene->addItem(group);
        c_end.x = x;
        c_end.y = y;
    } else if (mouse_press && button_pressed == CHPARTS) {
        QPoint pos = mapToScene(event->pos()).toPoint();
        int x = pos.x();
        int y = pos.y();
        this->deleteItemsFromGroup(group);
        group = new QGraphicsItemGroup();
        QPen penBlack(Qt::black);
        if (x<0)
            x=0;
        if (y<0)
            y=0;
        group->addToGroup(scene->addLine(coord.x, coord.y, x, coord.y, penBlack));
        group->addToGroup(scene->addLine(x, coord.y, x, y, penBlack));
        group->addToGroup(scene->addLine(x, y, coord.x, y, penBlack));
        group->addToGroup(scene->addLine(coord.x, y, coord.x, coord.y, penBlack));
        scene->addItem(group);
    }
```

```
}
```

4.5 Функция обновления сцены update(QPixmap pixmap)

void MyGraphicView::update(QPixmap pixmap);

Функция принимает пиксельную карту изображения, пиксельная карта добавляется к сцене функцией addPixmap() и загружается на виджет.

```
void MyGraphicView::update(QPixmap pixmap){
    scene = new QGraphicsScene();
    scene->addPixmap(pixmap);
    this->setScene(scene);
}
```

4.6 Функция удаления элементов из группы deleteItemsFromGroup(QGraphicsItemGroup *group)

void MyGraphicView::deleteItemsFromGroup(QGraphicsItemGroup *group);

Функция принимает указатель на группу, перебирает элементы группы оператором foreach и удаляет их.

```
void MyGraphicView::deleteItemsFromGroup(QGraphicsItemGroup *group) {
    foreach(QGraphicsItem *item, scene->items())
        if(item->group() == group)
            delete item;
}
```

5. Функции класса Info

5.1 Функция вывода информации об изображении setInfo()

void Info::setInfo();
Функция загружает в виджет подробную информарцию об изображении, размер изображения, его название, полный путь, размер, версию bmp, а также права на чтение и запись, хранящуюся в структуре info.

```
void Info::setInfo() {
    ui->_name->setText(info.name);
    ui->_path->setText(info.path);
    ui->_version->setText(QString::number(info.version));
    ui->_size->setText(QString::number(info.size) + " байт");
    ui->_width->setText(QString::number(info.width) + " пикселей");
    ui->_height->setText(QString::number(info.height) + " пикселей");
    ui->_readable->setText(info.isReadable ? "Есть" : "Нет");
    ui->_writable->setText(info.isWriteable ? "Есть" : "Нет");
```

```
}
```

6. Функции класса RGB_comp

6.1 Функция обработки слота нажатия на кнопку «ок» on_ok_clicked()

void RGB_window::on_ok_clicked()

Функция устанавливает публичную перемнную класса _close в 0, для того, чтобы при закрытии окна rgb_window, не сработала функция.

```
void RGB_window::on_ok_clicked()
{
    this->close();
    _close = 0;
}
```

6.2 Функция получения компонента требуемого для изменения comp()

int RGB_window::comp();

Функция возвращает значения 1, 2 или 3, для нажатых кнопка R, G или B соответственно.

```
int RGB_window::comp() {
    if (ui->comp_r->isChecked()) {
        return 1; /* red */
    }
    if (ui->comp_g->isChecked()) {
        return 2; /* green */
    }

    return 3; /* blue */
}
```

6.3 Функции получения значения get_value()

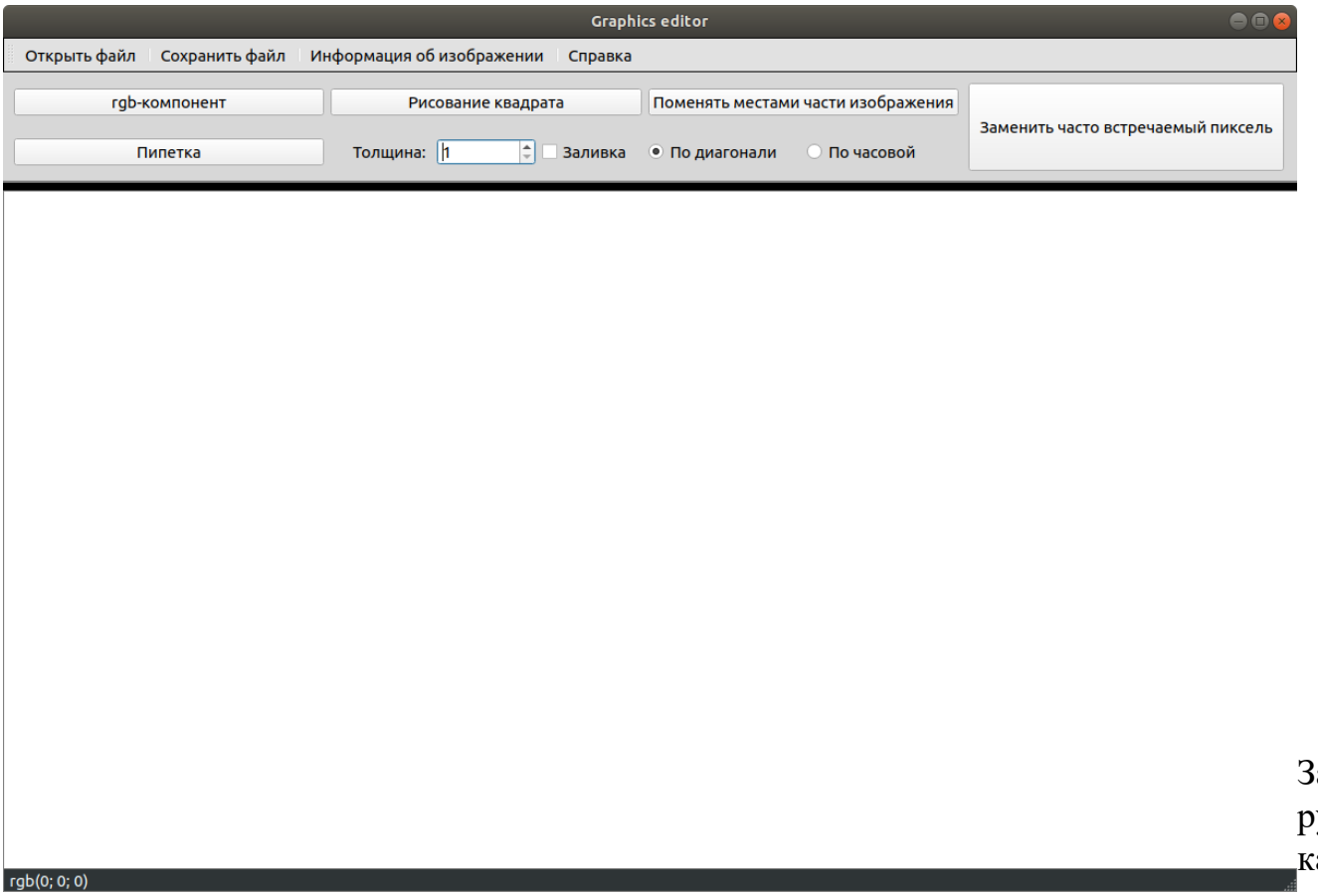
int RGB_window::get_value();

Функция возвращает 0 или 255, для нажатых кнопок «0» или «255» соответственно.

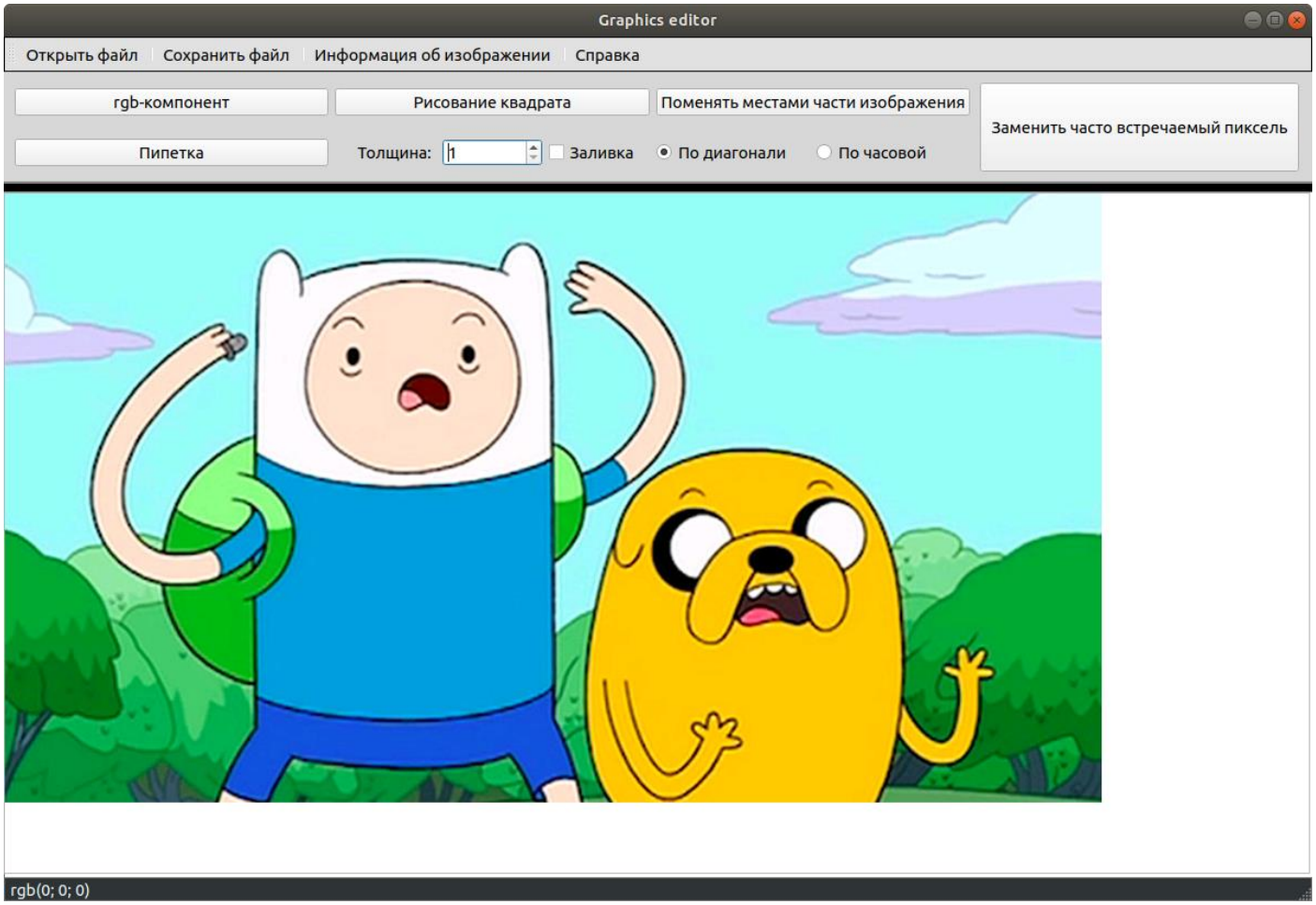
```
int RGB_window::get_value() {
    if (ui->min->isChecked())
        return ui->min->text().toInt(); /* return zero */
    return ui->max->text().toInt(); /* return 255 */
}
```

Тестирование программы.

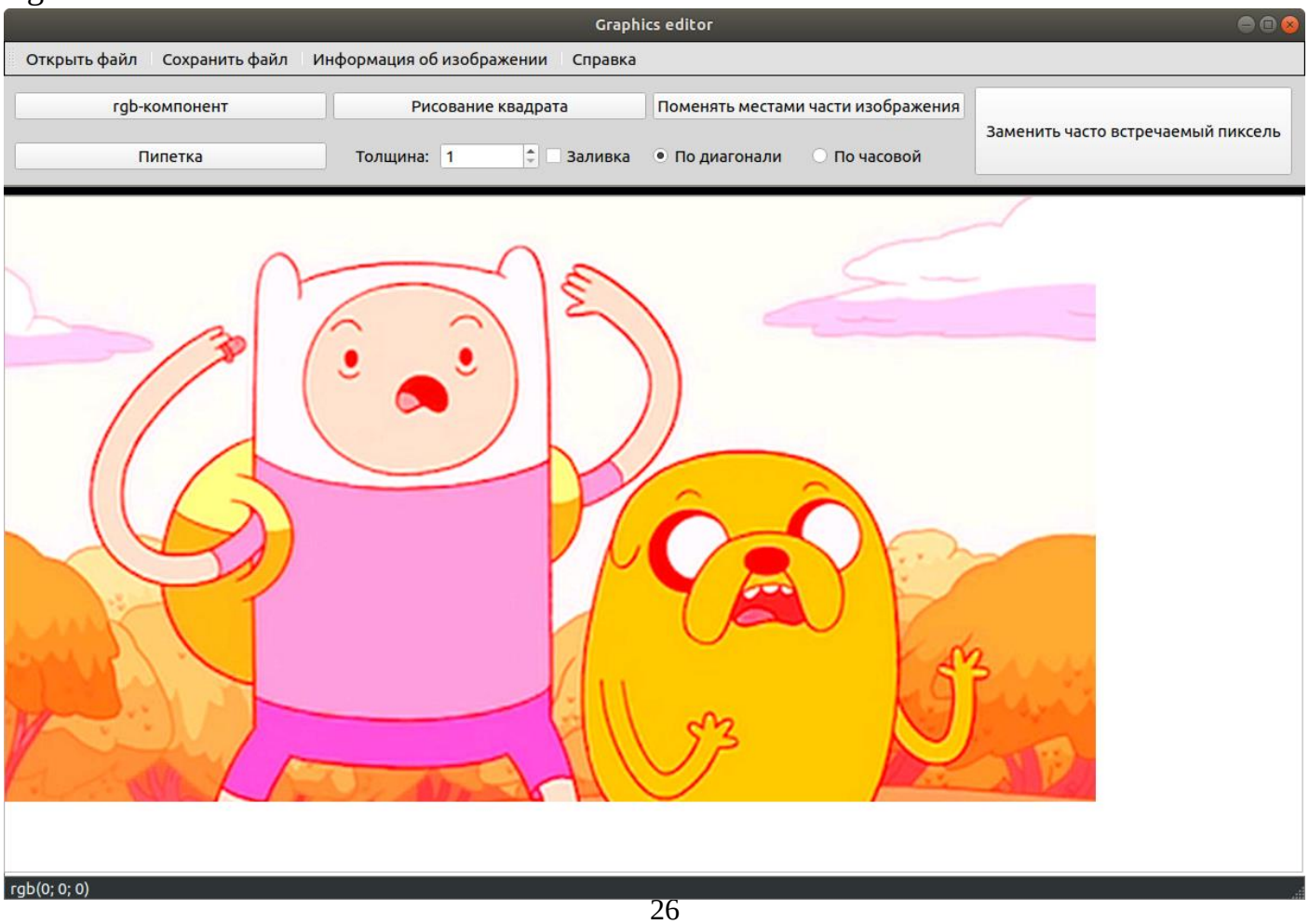
Запуск программы:



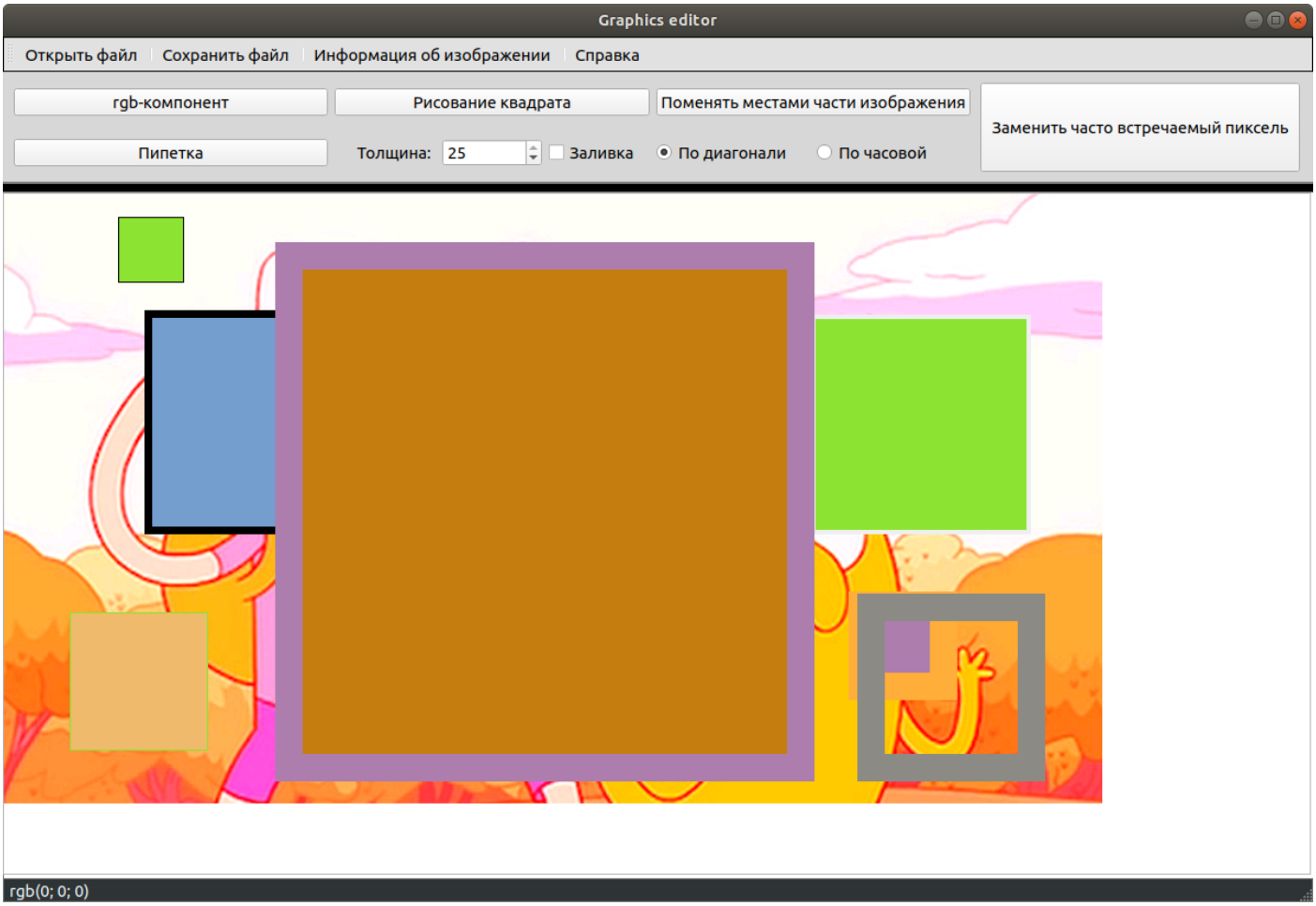
Загрузка изображения:



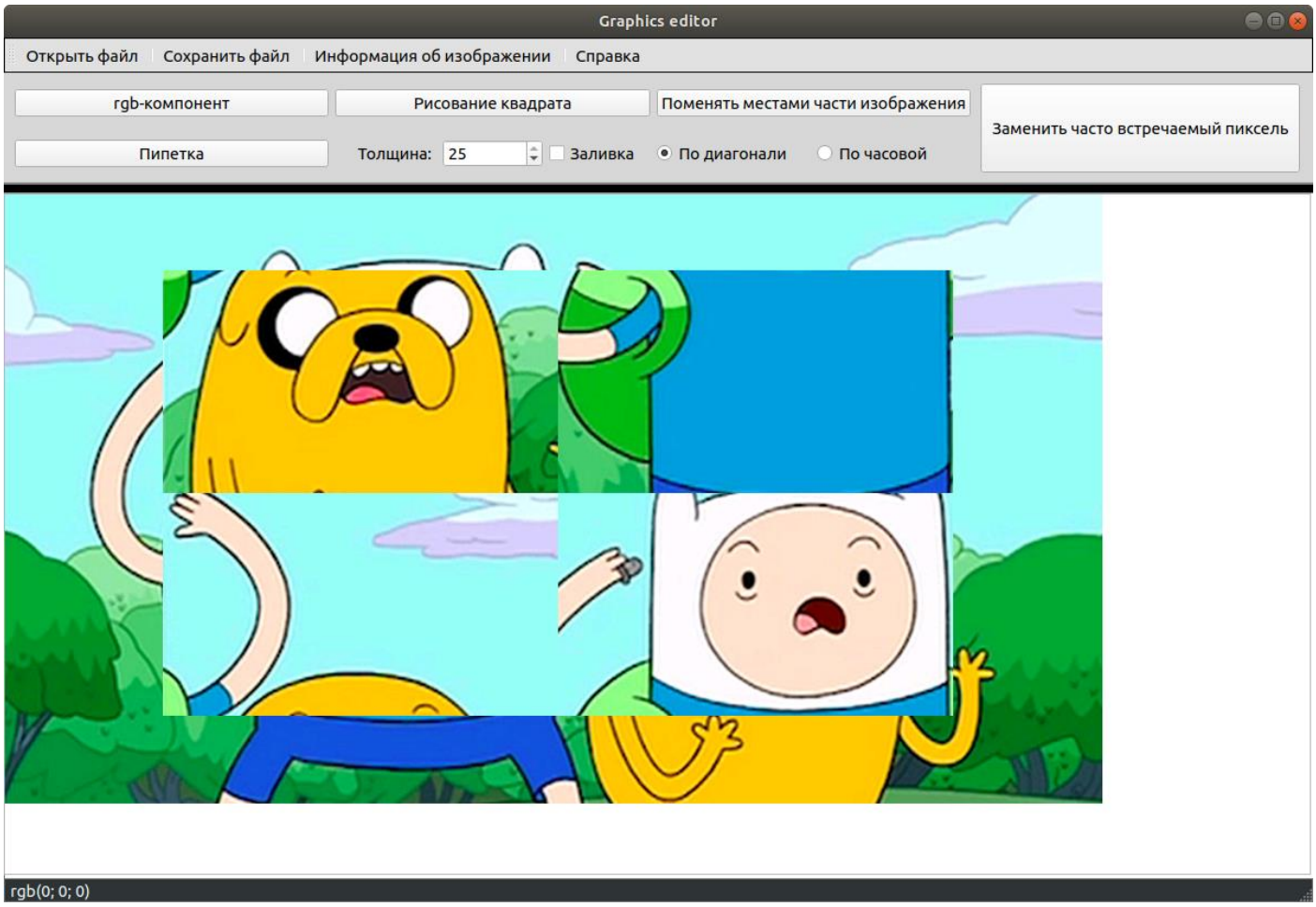
rgb-компонент:



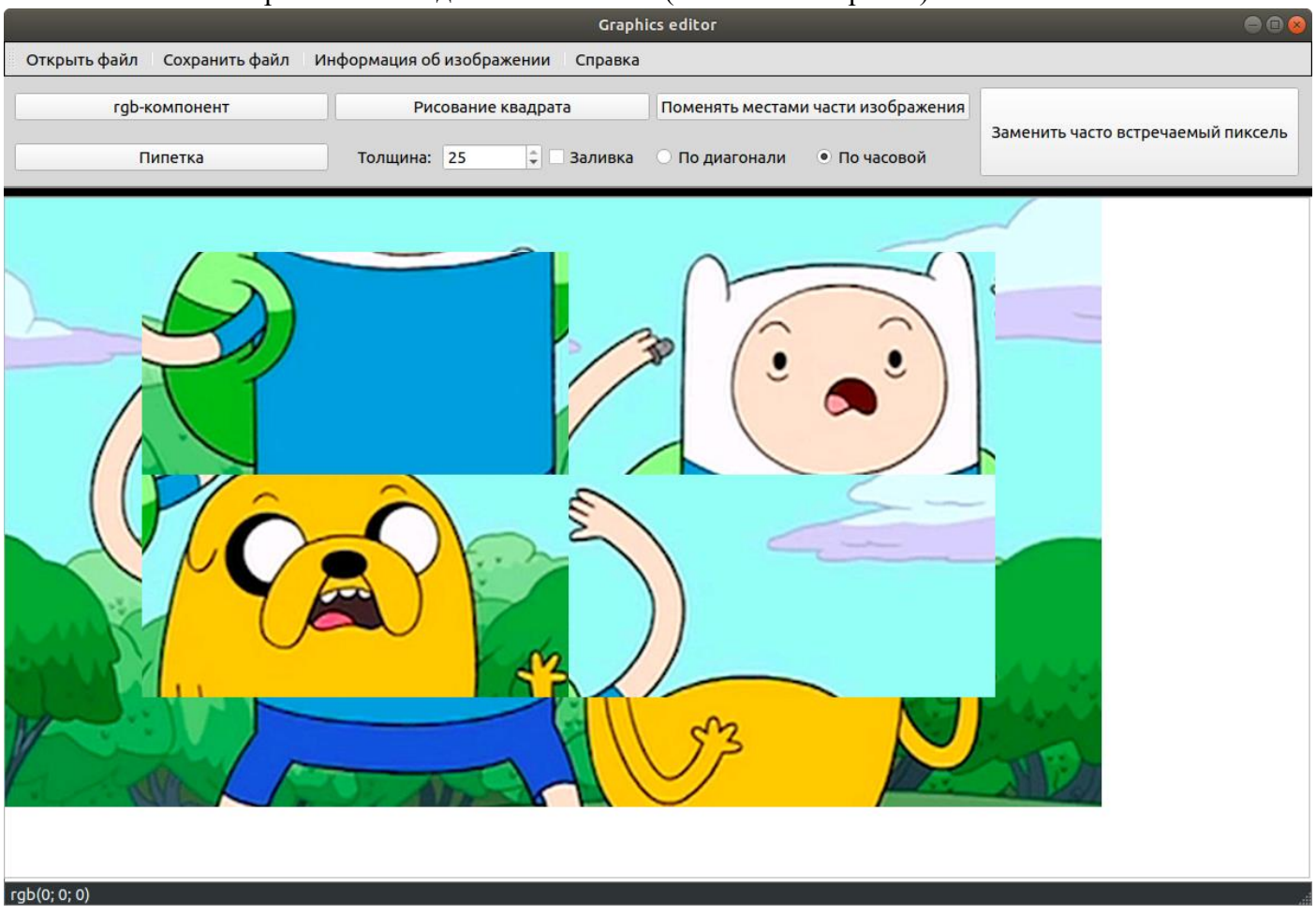
Отрисовка квадрата:



Смена частей изображения в выделенной области(по диагонали):

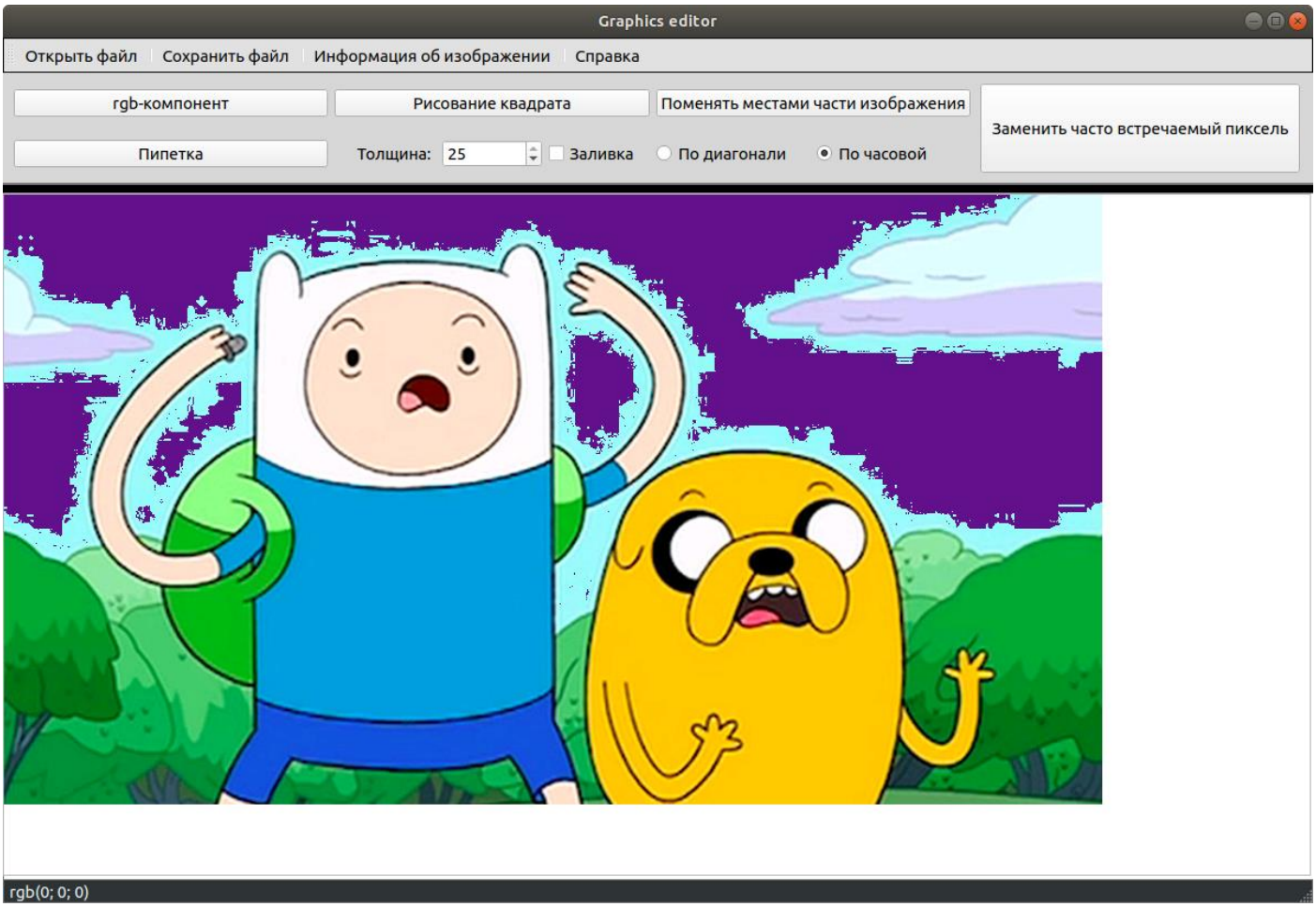


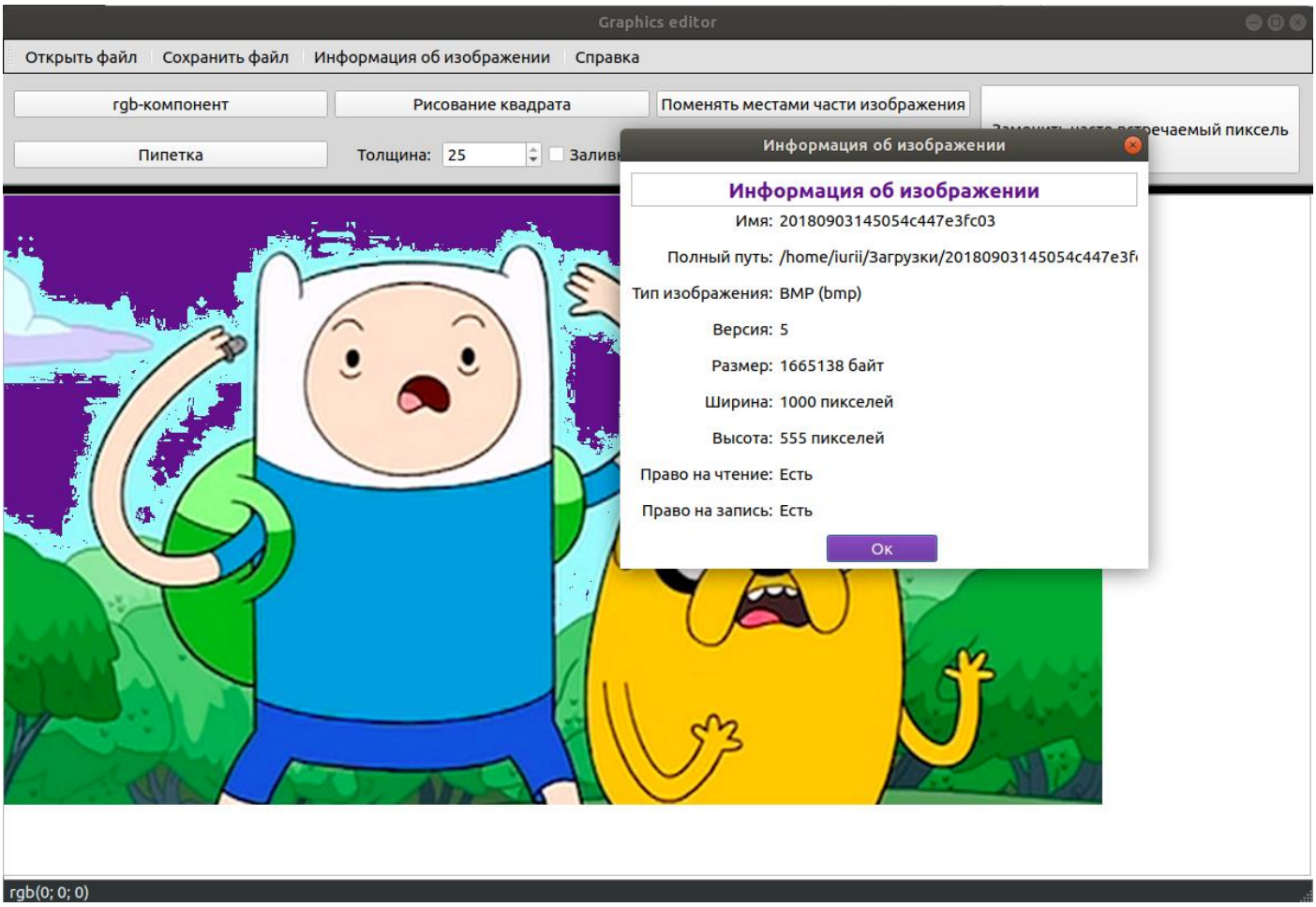
Смена частей изображения в выделенной области(по часовой стрелке):



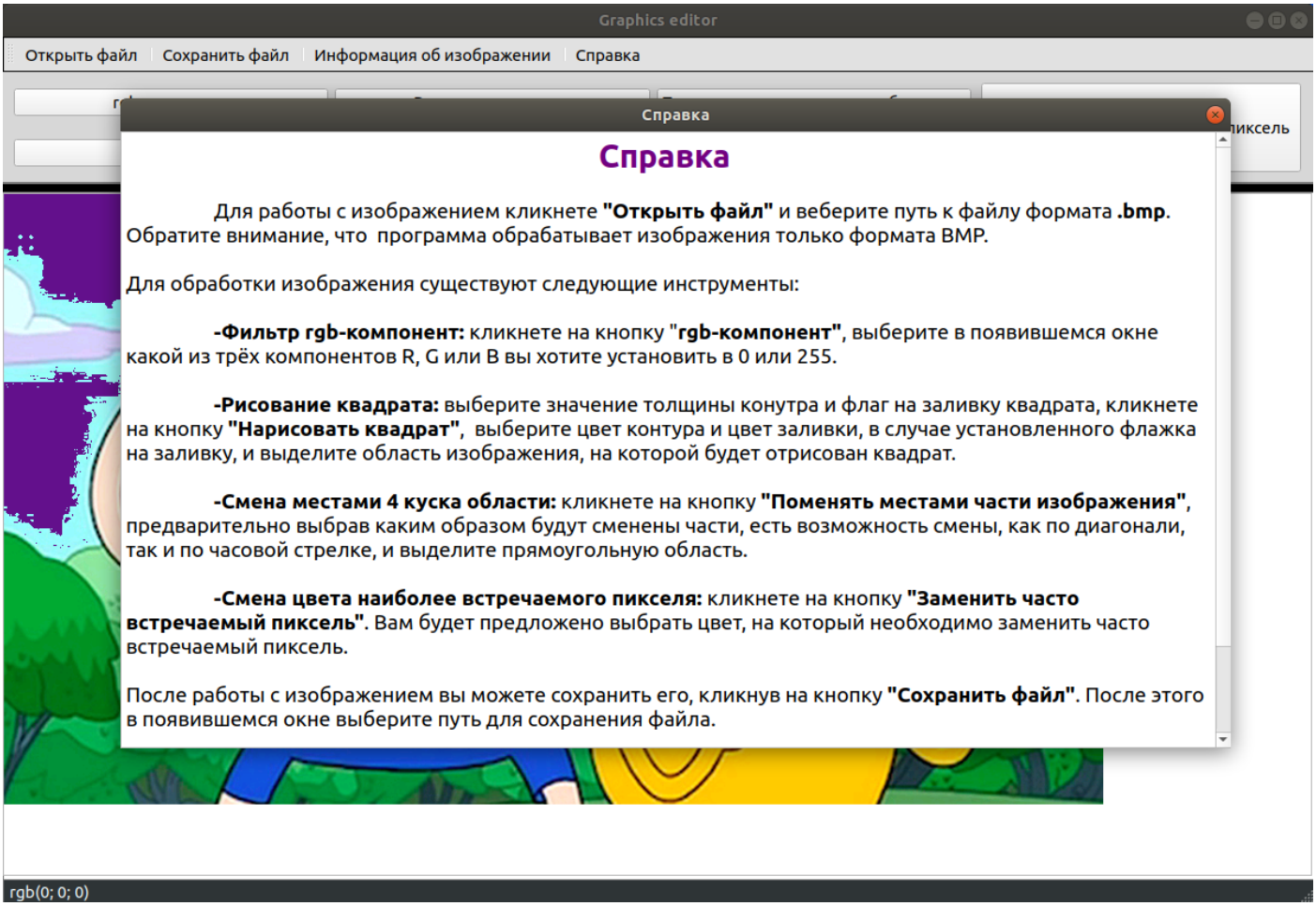
Смена часто встречаемого пикселя в изображении:Вывод информации об изображении:

Смена часто встречаемого пикселя в изображении:Вывод информации об изображении:
Вывод информации об изображении:





Вывод справки по работе с приложением:



Заключение.

В ходе выполнения работы было создано desktop-приложение с GUI для обработки файлов-изображений в формате bmp. Для работы программы были созданы и описаны все необходимые классы и структуры. Программа была протестирована, результат работы программы соответствует заданным условиям.

Список использованных источников

1. https://ru.wikipedia.org/wiki/BMP
2. http://math.ivanovo.ac.ru/dalgebra/Khashin/gr/bmp/bmp.html
3. https://prog-cpp.ru/
4. https://doc.qt.io/

Исходный код программы

image.h

```
#ifndef IMAGE_H
#define IMAGE_H 1

#include <QMessageBox>
#include <QImage>
#include <cstdlib>
#include <algorithm>
#define R 1
#define G 2
#define B 3

class Image{
#pragma pack(push, 1)
typedef struct BITMAPFILEHEADER {
    int16_t type;
    int32_t size;
    int16_t reserved1;
    int16_t reserved2;
    int32_t bfOffBits;
} BITMAPFILEHEADER;


typedef struct BITMAPINFOHEADER {
  int32_t size;
  int32_t width;
  int32_t height;
  int16_t planes;
  int16_t bitCount;
  int32_t compression;
  int32_t sizeImage;
  int32_t xPixPerMeter;
  int32_t yPixPerMeter;
  int32_t clrUsed;
  int32_t clrImportant;
} BITMAPINFOHEADER;


typedef struct RGB {
    unsigned char blue;
    unsigned char green;
    unsigned char red;
} RGB;
#pragma pack(pop)



public:
    QImage *image;
    BITMAPFILEHEADER bfh;
    BITMAPINFOHEADER bih;
    RGB** rgb;
    QPixmap getPixmap();
```

```
    int loadImage(const char*);
    int saveImage(const char*);
    int rgb_comp(int, int);
    int drawSquare(int, int, int, int, QColor, bool, QColor);
    int replace(QColor, QColor);
    int edit_often_color(QColor);
    int chParts(int, int, int, int, bool);
};
#endif // IMAGE_H
```

info.h

```
#ifndef INFO_H
#define INFO_H

#include <QDialog>
#include "image.h"

namespace Ui {
class Info;
}

class Info : public QDialog
{
    Q_OBJECT

public:
    explicit Info(QWidget *parent = nullptr);
    struct Information {
        QString name;
        QString path;
        int version;
        long size;
        int width;
        int height;
        bool isReadable;
        bool isWriteable;
    }info;
    void setInfo();
    ~Info();

private:
    Ui::Info *ui;
};

#endif // INFO_H
```

mainwindow.h

```
#include "rgb_window.h"
#include "mygraphicview.h"
#include "reference.h"
#include "info.h"
```

```
#define BIG_IMG -2
#define SAVE_ERR -1
#define LOAD_ERR -1
#define ALL_OK 0
#define PIPETTE 1
#define SQUARE 2
#define RGBCMP 3
#define CHPARTS 4
#define CORE 12
#define V3 40
#define V4 108
#define V5 124


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
   void selection();

   void on_open_triggered();

   void on_save_triggered();

   void on_rgbcomp_clicked();

   void on_pipette_clicked();

   void on_draw_Square_clicked();

   void on_ch_of_pix_clicked();

   void on_chParts_clicked();

   void on_reference_triggered();

   void on_info_triggered();

private:
    bool check;
    QColor fcolor;
    QColor color;
    uchar button_pressed;
```

```
    Ui::MainWindow *ui;
    RGB_window* rgb_window;
    Reference* reference;
    Info* info;
    MyGraphicView* picture;
    Image *img;
};

#endif // MAINWINDOW_H
```

mygraphicview.h

```
#ifndef MYGRAPHICVIEW_H
#define MYGRAPHICVIEW_H

#include <QGraphicsView>
#include <QGraphicsScene>
#include <QGraphicsItemGroup>
#include <QImage>
#include <QObject>
#include <QMouseEvent>

#define PIPETTE 1
#define SQUARE 2
#define RGBCMP 3
#define CHPARTS 4


class MyGraphicView : public QGraphicsView
{
    Q_OBJECT


public:
    explicit MyGraphicView(QWidget* parent = nullptr);
    ~MyGraphicView();
    int button_pressed = 0;
    int thick;
    QColor color;
    struct Coordinate {
        int x;
        int y;
    }coord, c_end;
    void update(QPixmap pixmap);
    int height;
    int width;

signals:
    void selection();
```

```
private slots:
    void mouseReleaseEvent(QMouseEvent *event);
    void mousePressEvent(QMouseEvent *event);
    void mouseMoveEvent(QMouseEvent *event);


private:
    QGraphicsItemGroup* group;
    QPixmap pixmap;
    QGraphicsScene* scene;
    bool mouse_press = false;

private:
    void deleteItemsFromGroup(QGraphicsItemGroup*);

};
#endif //MYGRAPHICVIEW_H
```

rgb_window.h

```
#ifndef RGB_WINDOW_H
#define RGB_WINDOW_H
#include <QDialog>

namespace Ui {
class RGB_window;
}

class RGB_window : public QDialog
{
    Q_OBJECT

public:
    explicit RGB_window(QWidget *parent = nullptr);
    ~RGB_window();
    int comp();
    int get_value();
    int _close = 1;

private slots:
    void on_ok_clicked();

private:
    Ui::RGB_window *ui;
};

#endif // RGB_WINDOW_H
```

reference.h

```
#ifndef REFERENCE_H
#define REFERENCE_H
```

```
#include <QDialog>

namespace Ui {
class reference;
}

class Reference : public QDialog
{
    Q_OBJECT

public:
    explicit Reference(QWidget *parent = nullptr);
    ~Reference();

private:
    Ui::reference *ui;
};

#endif // REFERENCE_H
```

image.cpp

```
#include "image.h"

using namespace std;

int Image::loadImage(const char *file) {
    FILE* f = fopen(file, "rb");
    if (!f)
        return -1;
    fread(&bfh, sizeof(bfh), 1, f);
    fread(&bih, sizeof(bih), 1, f);
    if (bih.height > 10000 || bih.width > 10000)
        return -2;
    size_t padding = 0;
    if ((bih.width*3) % 4)
        padding = 4 - (bih.width*3) % 4;
    rgb = new RGB* [bih.height];
    for (int i = 0; i < bih.height; i++){
        rgb[i] = new RGB[bih.width + 1];
    }
    fseek(f, long(bfh.bfOffBits), SEEK_SET);
    for (int i = 0; i <  bih.height; i++) {
        int j;
        for (j = 0;  j < bih.width; j++){
            fread(&rgb[i][j], sizeof(RGB), 1, f);
        }
        if (padding)
            fread(&rgb[i][j], padding, 1, f);
    }
    fclose(f);
    return 0;
}
```

```
int Image::saveImage(const char* file) {
    FILE* f = fopen(file, "wb");
    if (!f)
        return -1;
    fwrite(&bfh, sizeof(bfh), 1, f);
    fwrite(&bih, sizeof(bih), 1, f);
    size_t padding = 0;
    if ((bih.width*3) % 4)
        padding = 4 - (bih.width*3) % 4;
    fseek(f, long(bfh.bfOffBits), SEEK_SET);
    for (int i = 0; i < bih.height; i++) {
        int j;
        for (j = 0; j < bih.width; j++){
            fwrite(&rgb[i][j], sizeof(RGB), 1, f);
        }
        if (padding)
            fwrite(&rgb[i][j], padding, 1, f);
    }
    fclose(f);
    return 0;
}

QPixmap Image::getPixmap() {
    image = new QImage(bih.width, bih.height, QImage::Format_RGB16);
    QColor pixel;
    for (int i = bih.height - 1; i >= 0; i--) {
        for (int j = 0; j < bih.width; j++) {
            pixel.setRed(rgb[i][j].red);
            pixel.setGreen(rgb[i][j].green);
            pixel.setBlue(rgb[i][j].blue);
            image->setPixel(j, bih.height - i - 1, pixel.rgb());
        }
    }
    return QPixmap::fromImage(*image);
}

int Image::rgb_comp(int comp, int value) {

    if (comp == R) {
        for (int i = 0; i <  bih.height; i++)
            for (int j = 0;  j < bih.width; j++)
                rgb[i][j].red = uchar(value);
    } else if (comp == G) {
        for (int i = 0; i <  bih.height; i++)
            for (int j = 0;  j < bih.width; j++)
                rgb[i][j].green = uchar(value);
    } else if (comp == B) {
        for (int i = 0; i <  bih.height; i++)
            for (int j = 0;  j < bih.width; j++)
                rgb[i][j].blue = uchar(value);
    }
```

```
    return 0;
}

int Image::drawSquare(int x, int y, int len, int thick, QColor color, bool fill, QColor fcolor) {

    for (int i = x - thick + 1; i <= x + len + thick - 1; i++)
        for (int j = -thick + 1; j <= 0; j++) {
            rgb[bih.height - 1 - y - j][i].red = uchar(color.red());
            rgb[bih.height - 1 - y - j][i].green = uchar(color.green());
            rgb[bih.height - 1 - y - j][i].blue = uchar(color.blue());
                                                    /* horizontal lines */
            rgb[bih.height - 1 - y - len + j][i].red = uchar(color.red());
            rgb[bih.height - 1 - y - len + j][i].green = uchar(color.green());
            rgb[bih.height - 1 - y - len + j][i].blue = uchar(color.blue());
        }

    for (int j = bih.height - 1 - y + thick - 1; j >= bih.height - 1 - y - len - thick + 1; j--)
        for (int i = -thick + 1; i <= 0; i++) {
            rgb[j][x + i].red = uchar(color.red());
            rgb[j][x + i].green = uchar(color.green());
            rgb[j][x + i].blue = uchar(color.blue());
                                                    /* vertical lines */
            rgb[j][x + len - i].red = uchar(color.red());
            rgb[j][x + len - i].green = uchar(color.green());
            rgb[j][x + len - i].blue = uchar(color.blue());
        }

    /* fill */

    if (fill) {
        for (int j = bih.height - y - 2; j >= bih.height - y - len; j--)
            for (int i = x + 1; i < x + len; i++) {
                rgb[j][i].red = uchar(fcolor.red());
                rgb[j][i].green = uchar(fcolor.green());
                rgb[j][i].blue = uchar(fcolor.blue());
            }
    }
    return 0;
}

typedef struct my_Dict {
    QColor color;
    long amount;
}dict;

void add(dict* d, QColor color) {
    d->amount = 1;
    d->color = color;
}

long in(dict* d, QColor color, long size) {
    for (long i = 0; i < size; i++)
        if (d[i].color == color)
```

```
            return i;
    return -1;
}

QColor find_max(dict* d, long size) {

    long max = d->amount;
    long max_i = 0;

    for (int i = 1; i < size; i++)
        if (d[i].amount > max) {
            max = d[i].amount;
            max_i = i;
        }

    return d[max_i].color;
}

int Image::replace(QColor rgb1, QColor rgb2) {

    for (int i = 0; i <  bih.height; i++)
        for (int j = 0;  j < bih.width; j++)
            if (rgb[i][j].red == rgb1.red() && rgb[i][j].blue == rgb1.blue() && rgb[i][j].green ==
rgb1.green()) {
                rgb[i][j].red = uchar(rgb2.red());
                rgb[i][j].blue = uchar(rgb2.blue());
                rgb[i][j].green = uchar(rgb2.green());
            }

    return 0;
}

int Image::edit_often_color(QColor color) {

    long size = 1000000;
    long pos = 1;
    QColor temp(rgb[0][0].red, rgb[0][0].green, rgb[0][0].blue);
    dict* d = new dict[size];

    add(d, temp);

    for (int i = 0; i <  bih.height; i++)
        for (int j = 1;  j < bih.width; j++) {

            temp.setRgb(rgb[i][j].red, rgb[i][j].green, rgb[i][j].blue);
            long check = in(d, temp, pos);

            if (pos == size) {
                size *= 2;
                dict* tmp = d;
                d = new dict[size];
                memcpy(d, tmp, ulong(pos) * sizeof(dict));
                delete [] tmp;
```

```
            }

            if (check == -1) {
                add(d + pos++, temp);
            } else {
                d[check].amount++;
            }
        }

    QColor of_pix = find_max(d, pos);

    Image::replace(of_pix, color);

    delete [] d;

    return 0;
}

int Image::chParts(int x1, int y1, int x2, int y2, bool isDiagonally) {

    if (isDiagonally) {
        for (int i = y1; i < y1 + (y2 - y1) / 2; i++)
            for (int j = x1; j < x1 + (x2 - x1) / 2; j++) {
                swap(rgb[bih.height - i - 1][j].red, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 - x1) / 2].red);
                swap(rgb[bih.height - i - 1][j].green, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 - x1) /
2].green);
                swap(rgb[bih.height - i - 1][j].blue, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 - x1) /
2].blue);

                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].red, rgb[bih.height - i - 1][j + (x2 - x1) / 2].red);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].green, rgb[bih.height - i - 1][j + (x2 - x1) /
2].green);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].blue, rgb[bih.height - i - 1][j + (x2 - x1) /
2].blue);
            }

    } else {
        for (int i = y1; i < y1 + (y2 - y1) / 2; i++)
            for (int j = x1; j < x1 + (x2 - x1) / 2; j++) {
                swap(rgb[bih.height - i - 1][j].red, rgb[bih.height - i - 1][j + (x2 - x1) / 2].red);
                swap(rgb[bih.height - i - 1][j].green, rgb[bih.height - i - 1][j + (x2 - x1) / 2].green);
                swap(rgb[bih.height - i - 1][j].blue, rgb[bih.height - i - 1][j + (x2 - x1) / 2].blue);

                swap(rgb[bih.height - i - 1][j].red, rgb[bih.height - i - 1 - (y2 - y1) / 2][j].red);
                swap(rgb[bih.height - i - 1][j].green, rgb[bih.height - i - 1 - (y2 - y1) / 2][j].green);
                swap(rgb[bih.height - i - 1][j].blue, rgb[bih.height - i - 1 - (y2 - y1) / 2][j].blue);

                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].red, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2 -
x1) / 2].red);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].green, rgb[bih.height - i - 1 - (y2 - y1) / 2][j +
(x2 - x1) / 2].green);
                swap(rgb[bih.height - i - 1 - (y2 - y1) / 2][j].blue, rgb[bih.height - i - 1 - (y2 - y1) / 2][j + (x2
- x1) / 2].blue);
```

```
        }
    }

    return 0;
}
```

info.cpp

```
#include "info.h"
#include "ui_info.h"

Info::Info(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Info)
{
    ui->setupUi(this);

}

Info::~Info()
{
    delete ui;
}

void Info::setInfo() {
    ui->_name->setText(info.name);
    ui->_path->setText(info.path);
    ui->_version->setText(QString::number(info.version));
    ui->_size->setText(QString::number(info.size) + " байт");
    ui->_width->setText(QString::number(info.width) + " пикселей");
    ui->_height->setText(QString::number(info.height) + " пикселей");
    ui->_readable->setText(info.isReadable ? "Есть" : "Нет");
    ui->_writable->setText(info.isWriteable ? "Есть" : "Нет");
}
```

main.cpp

```
#include "mainwindow.h"
#include <QApplication>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    img = new Image();
    picture = new MyGraphicView();
    reference = new Reference();
    info = new Info();
    rgb_window = new RGB_window();
    statusBar()->showMessage("rgb(0; 0; 0)");
    bool succ = connect(picture, SIGNAL(selection()), this, SLOT(selection()));
    Q_ASSERT(succ);
}

MainWindow::~MainWindow() {
    delete ui;
}

void MainWindow::on_open_triggered()
{
    QString str = QFileDialog::getOpenFileName(nullptr, "Выберите файл для открытия",
"/home/user", "*.bmp");

    if (str == nullptr) return;

    button_pressed = 0;
    picture->button_pressed = 0;
    switch (img->loadImage(str.toLocal8Bit().constData())) {
        case ALL_OK:
            picture->height = img->bih.height;
            picture->width = img->bih.width;
            info->info.name = QFileInfo(str).baseName();
            info->info.path = QFileInfo(str).path() + "/" + info->info.name + ".bmp";
            switch(img->bih.size){
                case CORE:
                    img->bih.height = 0;
                    img->bih.width = 0;
                    QMessageBox::critical(this, "Ошибка", "Версия CORE bmp изображения не
поддерживается.");
                    return;
                case V3:
                    info->info.version = 3;
                    break;
                case V4:
                    info->info.version = 4;
                    break;
                case V5:
                    info->info.version = 5;
                    break;
                default:
                    img->bih.height = 0;
```

```
                img->bih.width = 0;
                QMessageBox::critical(this, "Ошибка", "Некорректный заголовок у bmp файла");
                return;
            }
            info->info.size = img->bfh.size;
            info->info.width = img->bih.width;
            info->info.height = img->bih.height;
            info->info.isReadable = QFileInfo(str).isReadable();
            info->info.isWriteable = QFileInfo(str).isWritable();
            picture->update(img->getPixmap());
            ui->gridLayout->addWidget(picture);
            return;
        case LOAD_ERR:
            img->bih.height = 0;
            img->bih.width = 0;
            QMessageBox::critical(this, "Ошибка", "Возникла ошибка при открытии файла");
            return;
        case BIG_IMG:
            img->bih.height = 0;
            img->bih.width = 0;
            QMessageBox::critical(this, "Ошибка", "Размер изображения не должен превышать
10000x10000 пикселей");
            return;
    }
}

void MainWindow::on_save_triggered()
{
    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Невозможно сохранить пустое изображение");
        return;
    }
    button_pressed = 0;
    picture->button_pressed = 0;

    QString str = QFileDialog::getSaveFileName(nullptr, "Выберите папку для сохранения",
"/home/user", "*.bmp");

    if (str == nullptr) return;

    switch (img->saveImage((str + ".bmp").toLocal8Bit().constData())) {
        case ALL_OK:
            return;
        case SAVE_ERR:
            QMessageBox::critical(this, "Ошибка", "Возникла ошибка при сохранении файла");
            return;
    }
}

void MainWindow::selection() {
    switch (button_pressed) {

        case PIPETTE: {
```

```
        QPoint coord(picture->coord.x, picture->coord.y);
        if (coord.x() < 0 || coord.y() >= img->bih.height || coord.y() < 0 || coord.x() >= img-
>bih.width) {
            QMessageBox::critical(this, "Ошибка", "Точка находится вне изображения");
            return;
        }
        int y = img->bih.height - 1 - coord.y();
        int x = coord.x();
        statusBar()->showMessage("rgb(" + QString::number(img->rgb[y][x].red) + "; " +
QString::number(img->rgb[y][x].green) + "; " + QString::number(img->rgb[y][x].blue) + ")");
        return;
      }
      case SQUARE: {
        if (picture->coord.x < 0 || picture->coord.y < 0 || picture->coord.x >= img->bih.width  ||
picture->coord.y >= img->bih.height || picture->c_end.x < 0 || picture->c_end.y < 0 || picture-
>c_end.x >= img->bih.width  || picture->c_end.y >= img->bih.height){
            QMessageBox::critical(this, "Ошибка", "Выделенная область выходит за пределы
рисунка, попробуйте снова");
            button_pressed = 0;
            return;
        }
        int thick = ui->thick->value();
        if (picture->coord.x - thick + 1 < 0 || picture->coord.y - thick + 1 < 0 || picture->coord.x +
thick - 1 > img->bih.width || picture->coord.y + thick - 1 > img->bih.height || picture->c_end.x - thick
+ 1 < 0 || picture->c_end.y - thick + 1 < 0 || picture->c_end.x + thick - 1 > img->bih.width || picture-
>c_end.y + thick - 1 > img->bih.height) {
            QMessageBox::critical(this, "Ошибка", "Выделенная область выходит за пределы
рисунка, попробуйте снова");
            button_pressed = 0;
            return;
        }
        QPoint point1(picture->coord.x, picture->coord.y);
        QPoint point2(picture->c_end.x, picture->c_end.y);
        img->drawSquare(point1.x() > point2.x() ? point2.x() : point1.x(), point1.y() > point2.y() ?
point2.y() : point1.y(), abs(point2.x() - point1.x()), thick, color, check, fcolor);
        picture->update(img->getPixmap());
        check = false;
        button_pressed = 0;
        return;
      }
      case CHPARTS: {
        if (picture->coord.x < 0 || picture->coord.y < 0 || picture->coord.x >= img->bih.width  ||
picture->coord.y >= img->bih.height || picture->c_end.x < 0 || picture->c_end.y < 0 || picture-
>c_end.x >= img->bih.width  || picture->c_end.y >= img->bih.height){
            QMessageBox::critical(this, "Ошибка", "Выделенная область выходит за пределы
рисунка, попробуйте снова");
            button_pressed = 0;
            return;
        }
        QPoint point1(picture->coord.x, picture->coord.y);
        QPoint point2(picture->c_end.x, picture->c_end.y);
        int x1 = point1.x() > point2.x() ? point2.x() : point1.x();
        int y1 = point1.y() > point2.y() ? point2.y() : point1.y();
```

```
            int x2 = point1.x() < point2.x() ? point2.x() : point1.x();
            int y2 = point1.y() < point2.y() ? point2.y() : point1.y();
            img->chParts(x1, y1, x2, y2, ui->diagonally->isChecked());
            picture->update(img->getPixmap());
            button_pressed = 0;
            return;
        }
    }
}

void MainWindow::on_rgbcomp_clicked() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }

    button_pressed = 0;
    picture->button_pressed = 0;

    rgb_window->exec();

    if (rgb_window->_close) return;
    rgb_window->_close = 1;

    int comp = rgb_window->comp();
    int value = rgb_window->get_value();

    img->rgb_comp(comp, value);
    picture->update(img->getPixmap());
}

void MainWindow::on_pipette_clicked() {
    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }
    button_pressed = PIPETTE;
    picture->button_pressed = PIPETTE;
}

void MainWindow::on_draw_Square_clicked() {
    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }
    button_pressed = 0;
    picture->button_pressed = 0;
    color = QColorDialog::getColor(Qt::white, this, "Выберите цвет контура");
    if (!color.isValid()) return;
```

```
    check = ui->isFill->isChecked();
    if (check) {
        fcolor = QColorDialog::getColor(Qt::white, this, "Выберите цвет заливки");
        if (!fcolor.isValid()) return;
    }
    picture->color = color;
    picture->thick = ui->thick->value();
    button_pressed = SQUARE;
    picture->button_pressed = SQUARE;
}

void MainWindow::on_ch_of_pix_clicked() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }

    button_pressed = 0;
    picture->button_pressed = 0;

    color = QColorDialog::getColor(Qt::white, this, "Выберите цвет пикселя");
    if (!color.isValid()) return;

    img->edit_often_color(color);
    picture->update(img->getPixmap());
}

void MainWindow::on_chParts_clicked() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Перед использованием функций откройте
изображение");
        return;
    }

    button_pressed = CHPARTS;
    picture->button_pressed = CHPARTS;
}

void MainWindow::on_reference_triggered() {
    button_pressed = 0;
    picture->button_pressed = 0;
    reference->show();
}

void MainWindow::on_info_triggered() {

    if (img->bih.width == 0 || img->bih.height == 0) {
        QMessageBox::critical(this, "Ошибка", "Чтобы просмотреть информацию об изображении
- загрузите его");
        return;
```

```
    }

    button_pressed = 0;
    picture->button_pressed = 0;
    info->setInfo();
    info->exec();
}
```

mygraphicview.cpp

```
#include "mygraphicview.h"
#include "mainwindow.h"

MyGraphicView::MyGraphicView(QWidget *parent) : QGraphicsView(parent) {
    this->setAlignment(Qt::AlignLeft | Qt::AlignTop);
    group = new QGraphicsItemGroup();
}

void MyGraphicView::update(QPixmap pixmap) {
    scene = new QGraphicsScene();
    scene->addPixmap(pixmap);
    this->setScene(scene);
}

void MyGraphicView::mousePressEvent(QMouseEvent* event) {
    QPoint pos = mapToScene(event->pos()).toPoint();
    coord.x = pos.x();
    coord.y = pos.y();
    if (button_pressed != PIPETTE)
        mouse_press = true;
}

void MyGraphicView::mouseReleaseEvent(QMouseEvent* event) {
    if (mouse_press && button_pressed != SQUARE) {
        QPoint pos = mapToScene(event->pos()).toPoint();
        c_end.x = pos.x();
        c_end.y = pos.y();
        if (c_end.x < 0)
            c_end.x = 0;
        if (c_end.y < 0)
            c_end.y = 0;
    }
    mouse_press = false;
    this->deleteItemsFromGroup(group);

    if (button_pressed != PIPETTE) button_pressed = 0;

    emit selection();
}

void MyGraphicView::mouseMoveEvent(QMouseEvent *event) {
    if (mouse_press && button_pressed == SQUARE) {
        QPoint pos = mapToScene(event->pos()).toPoint();
```

```
        int x = pos.x();
        int y = pos.y();
        this->deleteItemsFromGroup(group);
        group = new QGraphicsItemGroup();
        QPen penColor(color);
        if (x < 0)
            x = 0;
        y = ((y >= coord.y) ? coord.y + abs(x - coord.x) : coord.y - abs(x - coord.x));
        if (y < 0) {
            y = 0;
        }
        if (y == 0) {
            x = (x >= coord.x) ? coord.x + coord.y : coord.x - coord.y;
        }
        group->addToGroup(scene->addLine(coord.x, coord.y, x, coord.y, penColor));
        group->addToGroup(scene->addLine(x, coord.y, x, y, penColor));
        group->addToGroup(scene->addLine(x, y, coord.x, y, penColor));
        group->addToGroup(scene->addLine(coord.x, y, coord.x, coord.y, penColor));
        scene->addItem(group);
        c_end.x = x;
        c_end.y = y;
    } else if (mouse_press && button_pressed == CHPARTS) {
        QPoint pos = mapToScene(event->pos()).toPoint();
        int x = pos.x();
        int y = pos.y();
        this->deleteItemsFromGroup(group);
        group = new QGraphicsItemGroup();
        QPen penBlack(Qt::black);
        if (x<0)
            x=0;
        if (y<0)
            y=0;
        group->addToGroup(scene->addLine(coord.x, coord.y, x, coord.y, penBlack));
        group->addToGroup(scene->addLine(x, coord.y, x, y, penBlack));
        group->addToGroup(scene->addLine(x, y, coord.x, y, penBlack));
        group->addToGroup(scene->addLine(coord.x, y, coord.x, coord.y, penBlack));
        scene->addItem(group);
    }
}

MyGraphicView::~MyGraphicView() {
    delete group;
    delete scene;
}

void MyGraphicView::deleteItemsFromGroup(QGraphicsItemGroup *group) {
    foreach(QGraphicsItem *item, scene->items())
      if(item->group() == group)
        delete item;
}
```

rgb_window.cpp

```
#include "rgb_window.h"
#include "ui_rgb_window.h"

RGB_window::RGB_window(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::RGB_window)
{
    ui->setupUi(this);
}

RGB_window::~RGB_window() {
    delete ui;
}

void RGB_window::on_ok_clicked()
{
    this->close();
    _close = 0;
}

int RGB_window::comp() {
    if (ui->comp_r->isChecked()) {
        return 1; /* red */
    }
    if (ui->comp_g->isChecked()) {
        return 2; /* green */
    }

    return 3; /* blue */
}

int RGB_window::get_value() {
    if (ui->min->isChecked())
        return ui->min->text().toInt(); /* return zero */
    return ui->max->text().toInt(); /* return 255 */
}
```

reference.cpp

```
#include "reference.h"
#include "ui_reference.h"

Reference::Reference(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::reference)
{
    ui->setupUi(this);
}

Reference::~Reference()
{
    delete ui;
}
```

info.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Info</class>
 <widget class="QDialog" name="Info">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>481</width>
    <height>371</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Информация об изображении</string>
  </property>
  <property name="styleSheet">
   <string notr="true">QDialog {
    background-color: rgb(255,255,255);
}</string>
  </property>
  <widget class="QTextBrowser" name="textBrowser">
   <property name="geometry">
    <rect>
     <x>10</x>
     <y>10</y>
     <width>461</width>
     <height>31</height>
    </rect>
   </property>
   <property name="verticalScrollBarPolicy">
    <enum>Qt::ScrollBarAlwaysOff</enum>
   </property>
   <property name="horizontalScrollBarPolicy">
    <enum>Qt::ScrollBarAlwaysOff</enum>
   </property>
   <property name="html">
    <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Ubuntu'; font-size:11pt; font-
weight:400; font-style:normal;&quot;&gt;
&lt;p align=&quot;center&quot; style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-size:14pt;
font-weight:600; color:#601189;&quot;&gt;Информация об
изображении&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
   </property>
   <property name="acceptRichText">
    <bool>true</bool>
```

```
   </property>
   <property name="openLinks">
    <bool>false</bool>
   </property>
  </widget>
  <widget class="QPushButton" name="pushButton">
   <property name="geometry">
    <rect>
     <x>188</x>
     <y>340</y>
     <width>101</width>
     <height>25</height>
    </rect>
   </property>
   <property name="styleSheet">
    <string notr="true">QPushButton {
    background-color: rgb(96, 17, 137);
    color: rgb(255, 255, 255)
}</string>
   </property>
   <property name="text">
    <string>Ок</string>
   </property>
  </widget>
  <widget class="QWidget" name="layoutWidget">
   <property name="geometry">
    <rect>
     <x>10</x>
     <y>40</y>
     <width>461</width>
     <height>291</height>
    </rect>
   </property>
   <layout class="QHBoxLayout" name="horizontalLayout">
    <item>
     <layout class="QVBoxLayout" name="verticalLayout">
      <item>
       <widget class="QLabel" name="name">
        <property name="text">
         <string>Имя:</string>
        </property>
        <property name="alignment">
         <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QLabel" name="path">
        <property name="text">
         <string>Полный путь:</string>
        </property>
        <property name="alignment">
         <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
```

```
  </property>
 </widget>
</item>
<item>
 <widget class="QLabel" name="type">
  <property name="text">
   <string>Тип изображения:</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
 </widget>
</item>
<item>
 <widget class="QLabel" name="version">
  <property name="text">
   <string>Версия:</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
 </widget>
</item>
<item>
 <widget class="QLabel" name="size">
  <property name="text">
   <string>Размер:</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
 </widget>
</item>
<item>
 <widget class="QLabel" name="width">
  <property name="text">
   <string>Ширина:</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
 </widget>
</item>
<item>
 <widget class="QLabel" name="height">
  <property name="text">
   <string>Высота:</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
 </widget>
</item>
```

```
  <item>
   <widget class="QLabel" name="writable">
    <property name="text">
     <string>Право на чтение:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="readable">
    <property name="text">
     <string>Право на запись:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
  </item>
 </layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_2">
  <item>
   <widget class="QLabel" name="_name">
    <property name="text">
     <string/>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="_path">
    <property name="text">
     <string/>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="_type">
    <property name="text">
     <string>BMP (bmp)</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="_version">
    <property name="text">
     <string/>
    </property>
   </widget>
  </item>
  <item>
```

```
      <widget class="QLabel" name="_size">
       <property name="text">
        <string/>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QLabel" name="_width">
       <property name="text">
        <string/>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QLabel" name="_height">
       <property name="text">
        <string/>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QLabel" name="_readable">
       <property name="text">
        <string/>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QLabel" name="_writable">
       <property name="text">
        <string/>
       </property>
      </widget>
     </item>
    </layout>
   </item>
  </layout>
 </widget>
</widget>
<resources/>
<connections>
 <connection>
  <sender>pushButton</sender>
  <signal>clicked()</signal>
  <receiver>Info</receiver>
  <slot>close()</slot>
  <hints>
   <hint type="sourcelabel">
    <x>258</x>
    <y>351</y>
   </hint>
   <hint type="destinationlabel">
    <x>311</x>
```

```
     <y>352</y>
    </hint>
   </hints>
  </connection>
 </connections>
</ui>
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1193</width>
    <height>788</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Graphics editor</string>
  </property>
  <property name="styleSheet">
   <string notr="true">QMainWindow {
  background-color: rgb(215, 215, 215);
}</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QWidget" name="gridLayoutWidget">
    <property name="geometry">
     <rect>
      <x>0</x>
      <y>110</y>
      <width>1191</width>
      <height>621</height>
     </rect>
    </property>
    <layout class="QGridLayout" name="gridLayout"/>
   </widget>
   <widget class="QLabel" name="back_ground2">
    <property name="geometry">
     <rect>
      <x>0</x>
      <y>100</y>
      <width>1201</width>
      <height>741</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">QLabel {
  background-color: rgb(255, 255, 255);
```

```
  border: 1px solid gray;
}</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QWidget" name="layoutWidget">
    <property name="geometry">
     <rect>
      <x>10</x>
      <y>0</y>
      <width>871</width>
      <height>101</height>
     </rect>
    </property>
    <layout class="QGridLayout" name="gridLayout_2">
     <item row="1" column="1">
      <layout class="QHBoxLayout" name="horizontalLayout">
       <item>
        <widget class="QLabel" name="l_thick">
         <property name="text">
          <string>      Толщина:</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QSpinBox" name="thick">
         <property name="correctionMode">
          <enum>QAbstractSpinBox::CorrectToPreviousValue</enum>
         </property>
         <property name="minimum">
          <number>1</number>
         </property>
         <property name="maximum">
          <number>25</number>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QCheckBox" name="isFill">
         <property name="styleSheet">
          <string notr="true"/>
         </property>
         <property name="text">
          <string>Заливка</string>
         </property>
         <property name="tristate">
          <bool>false</bool>
         </property>
        </widget>
       </item>
      </layout>
```

```
</item>
<item row="1" column="0">
 <widget class="QPushButton" name="pipette">
  <property name="styleSheet">
   <string notr="true"/>
  </property>
  <property name="text">
   <string>Пипетка</string>
  </property>
 </widget>
</item>
<item row="0" column="2">
 <widget class="QPushButton" name="chParts">
  <property name="text">
   <string>Поменять местами части изображения</string>
  </property>
 </widget>
</item>
<item row="0" column="0">
 <widget class="QPushButton" name="rgbcomp">
  <property name="styleSheet">
   <string notr="true"/>
  </property>
  <property name="text">
   <string>rgb-компонент</string>
  </property>
 </widget>
</item>
<item row="0" column="1">
 <widget class="QPushButton" name="draw_Square">
  <property name="styleSheet">
   <string notr="true"/>
  </property>
  <property name="text">
   <string>Рисование квадрата</string>
  </property>
 </widget>
</item>
<item row="1" column="2">
 <layout class="QHBoxLayout" name="horizontalLayout_2">
  <item>
   <widget class="QRadioButton" name="diagonally">
    <property name="text">
     <string>По диагонали</string>
    </property>
    <property name="checked">
     <bool>true</bool>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QRadioButton" name="clockwise">
    <property name="text">
```

```
          <string>По часовой</string>
         </property>
        </widget>
       </item>
      </layout>
     </item>
    </layout>
   </widget>
   <widget class="QLabel" name="back_ground3">
    <property name="geometry">
     <rect>
      <x>-20</x>
      <y>101</y>
      <width>1221</width>
      <height>9</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">QLabel {
  background-color: rgb(0, 0, 0);
  border: 1px solid gray;
}</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QPushButton" name="ch_of_pix">
    <property name="geometry">
     <rect>
      <x>890</x>
      <y>10</y>
      <width>291</width>
      <height>81</height>
     </rect>
    </property>
    <property name="text">
     <string>Заменить часто встречаемый пиксель</string>
    </property>
   </widget>
   <zorder>back_ground2</zorder>
   <zorder>layoutWidget</zorder>
   <zorder>gridLayoutWidget</zorder>
   <zorder>back_ground3</zorder>
   <zorder>ch_of_pix</zorder>
  </widget>
  <widget class="QStatusBar" name="statusBar">
   <property name="styleSheet">
    <string notr="true">QStatusBar {
  background-color: rgb(46, 52, 54);
  border: 1px solid gray;
  color: rgb(245, 253, 255)
}</string>
```

```
   </property>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <property name="styleSheet">
    <string notr="true">QToolBar {
  background-color: rgb(225, 225, 225);
  border: 1px solid black;
}</string>
   </property>
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
   <addaction name="open"/>
   <addaction name="separator"/>
   <addaction name="save"/>
   <addaction name="separator"/>
   <addaction name="info"/>
   <addaction name="separator"/>
   <addaction name="reference"/>
  </widget>
  <action name="open">
   <property name="text">
    <string>Открыть файл</string>
   </property>
   <property name="toolTip">
    <string>Открыть файл</string>
   </property>
  </action>
  <action name="save">
   <property name="text">
    <string>Сохранить файл</string>
   </property>
   <property name="toolTip">
    <string>Сохранить файл</string>
   </property>
  </action>
  <action name="reference">
   <property name="text">
    <string>Справка</string>
   </property>
   <property name="toolTip">
    <string>Справка</string>
   </property>
  </action>
  <action name="info">
   <property name="text">
    <string>Информация об изображении</string>
   </property>
   <property name="toolTip">
    <string>Информация об изображении</string>
```

```
   </property>
  </action>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```

rgb_window.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>RGB_window</class>
 <widget class="QDialog" name="RGB_window">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>172</width>
    <height>134</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Настройка компонента</string>
  </property>
  <widget class="QWidget" name="layoutWidget">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>171</width>
     <height>134</height>
    </rect>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout_3">
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout">
      <item>
       <widget class="QWidget" name="rgb" native="true">
        <layout class="QVBoxLayout" name="verticalLayout">
         <item>
          <widget class="QRadioButton" name="comp_r">
           <property name="text">
            <string>R</string>
           </property>
           <property name="checked">
            <bool>true</bool>
           </property>
          </widget>
         </item>
         <item>
          <widget class="QRadioButton" name="comp_g">
           <property name="text">
```

```
        <string>G</string>
       </property>
       <property name="checked">
        <bool>false</bool>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QRadioButton" name="comp_b">
       <property name="text">
        <string>B</string>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
  </item>
  <item>
   <widget class="QWidget" name="comp" native="true">
    <layout class="QVBoxLayout" name="verticalLayout_2">
     <item>
      <widget class="QRadioButton" name="min">
       <property name="text">
        <string>0</string>
       </property>
       <property name="checkable">
        <bool>true</bool>
       </property>
       <property name="checked">
        <bool>true</bool>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QRadioButton" name="max">
       <property name="text">
        <string>255</string>
       </property>
       <property name="checked">
        <bool>false</bool>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
  </item>
 </layout>
</item>
<item>
 <widget class="QPushButton" name="ok">
  <property name="text">
   <string>OK</string>
  </property>
```

```
    </widget>
   </item>
  </layout>
 </widget>
</widget>
<resources/>
<connections/>
</ui>
```

reference.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>reference</class>
 <widget class="QDialog" name="reference">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1010</width>
    <height>561</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Справка</string>
  </property>
  <widget class="QTextBrowser" name="textBrowser">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>1011</width>
     <height>561</height>
    </rect>
   </property>
   <property name="html">
    <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /&gt;&lt;style
type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Ubuntu'; font-size:11pt; font-
weight:400; font-style:normal;&quot;&gt;
&lt;p align=&quot;center&quot; style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-size:22pt;
font-weight:600; color:#710080;&quot;&gt;Справка &lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px;
margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;&quot;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-
indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-size:14pt;&quot;&gt; Для работы с
изображением кликнете &lt;/span&gt;&lt;span style=&quot; font-size:14pt; font-
weight:600;&quot;&gt;&amp;quot;Открыть файл&amp;quot; &lt;/span&gt;&lt;span style=&quot;
```

font-size:14pt;">и веберите путь к файлу формата .bmp. Обратите внимание, что программа обрабатывает изображения только формата BMP.</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">Для обработки изображения существуют следующие инструменты:</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"> -Фильтр rgb-компонент: кликнете на кнопку "rgb-компонент", выберите в появившемся окне какой из трёх компонентов R, G или B вы хотите установить в 0 или 255.</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"> -Рисование квадрата: выберите значение толщины конутра и флаг на заливку квадрата, кликнете на кнопку "Нарисовать квадрат", выберите цвет контура и цвет заливки, в случае установленного флажка на заливку, и выделите область изображения, на которой будет отрисован квадрат.
</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt; font-weight:600;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"> -Смена местами 4 куска области: кликнете на кнопку "Поменять местами части изображения", предварительно выбрав каким образом будут сменены части, есть возможность смены, как по диагонали, так и по часовой стрелке, и выделите прямоугольную область.</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"> -Смена цвета наиболее встречаемого пикселя: кликнете на кнопку "Заменить часто встречаемый пиксель". Вам будет предложено выбрать цвет, на который необходимо заменить часто встречаемый пиксель.</p>

<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">После работы с изображением вы можете сохранить его, кликнув на кнопку "Сохранить файл". После этого в появившемся окне выберите путь для сохранения файла.</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">Также вы можете посмотреть подробную информацию об изображении во вкладке "Информация об изображении".</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p align="right" style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">Автор программы: Бородкин Юрий Владимирович</p></body></html></string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>