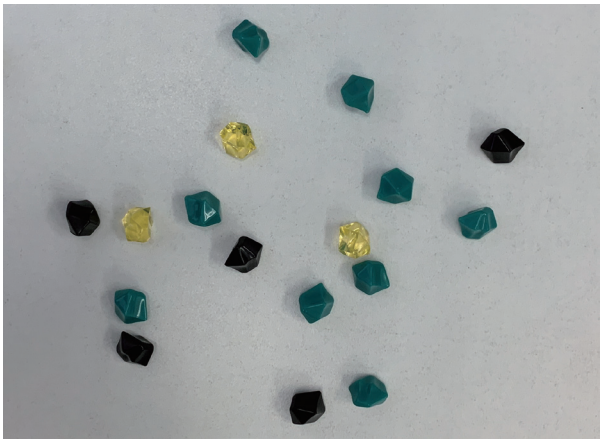
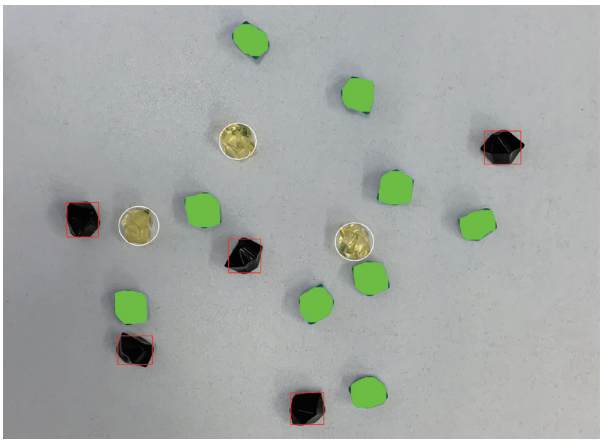
各宝石を検出



入力画像



出力画像

```
black = 5
green = 9
golden = 3
```

出力

```cpp
cv::Mat hsv(const cv::Mat& hsv_img, const cv::Scalar& min_hsv, const cv::Scalar& max_hsv) {
    cv::Mat bin_img = cv::Mat::zeros(hsv_img.size(), CV_8UC1);
    for (int y = 0; y < hsv_img.rows; y++) {
        for (int x = 0; x < hsv_img.cols; x++) {
            cv::Vec3b p = hsv_img.at<cv::Vec3b>(y, x);
            if (p[0] >= min_hsv[0] && p[0] <= max_hsv[0] &&
                p[1] >= min_hsv[1] && p[1] <= max_hsv[1] &&
                p[2] >= min_hsv[2] && p[2] <= max_hsv[2]) {
                bin_img.at<uchar>(y, x) = 255;
            }
        }
    }
    return bin_img;
}
void kensyutu(const cv::Mat& src_img, cv::Mat& dst_img, const cv::Scalar& min_hsv, const
    cv::Scalar& max_hsv, const cv::Scalar& draw_color, int draw_shape, std::string gem_name) {
    cv::Mat hsv_img, bin_img;
    cv::cvtColor(src_img, hsv_img, cv::COLOR_BGR2HSV);
    bin_img = hsv(hsv_img, min_hsv, max_hsv);

    //膨張収縮
    cv::dilate(bin_img, bin_img, cv::Mat(), cv::Point(-1, -1), COUNT);
    cv::erode(bin_img, bin_img, cv::Mat(), cv::Point(-1, -1), COUNT*2);
    cv::dilate(bin_img, bin_img, cv::Mat(), cv::Point(-1, -1), COUNT);

    std::vector<std::vector<cv::Point>> contours;
    cv::findContours(bin_img, contours, cv::RETR_EXTERNAL, cv::CHAIN_APPROX_SIMPLE);

    std::cout << gem_name << " = " << contours.size() << std::endl;

    for (const auto& contour : contours) {
        if (draw_shape == 0) { //外接長方形
            cv::Rect rect = cv::boundingRect(contour);
            cv::rectangle(dst_img, rect, draw_color, 2);
        } else if (draw_shape == 1) { //塗りつぶし
            cv::drawContours(dst_img, std::vector<std::vector<cv::Point>>{contour}, -1,
                draw_color, cv::FILLED);
        } else if (draw_shape == 2) { //外接円
            cv::Point2f center;
            float radius;
            cv::minEnclosingCircle(contour, center, radius);
            cv::circle(dst_img, center, static_cast<int>(radius), draw_color, 2);
        }
    }

    // 画像の表示
    cv::imshow(WINDOW_NAME_BINARY, bin_img);
}
```

```cpp
int main() {
    cv::Mat src_img = cv::imread(FILE_NAME, cv::IMREAD_COLOR);
    if (src_img.empty()) {
        fprintf(stderr, "Cannot read image file: %s.\n", FILE_NAME);
        return -1;
    }

    cv::Mat dst_img = src_img.clone();

    //黒曜石の検出
    kensyutu(src_img, dst_img, cv::Scalar(0, 0, 0), cv::Scalar(180, 255, 50), cv::Scalar(0,
        0, 255), 0, "black");

    //トルコ石の検出
    kensyutu(src_img, dst_img, cv::Scalar(60, 60, 50), cv::Scalar(100, 255, 255),
        cv::Scalar(0, 255, 0), 1, "green");

    //黄金の検出
    kensyutu(src_img, dst_img, cv::Scalar(20, 100, 100), cv::Scalar(30, 255, 255),
        cv::Scalar(255, 255, 255), 2, "golden");

    //表示
    cv::imshow(WINDOW_NAME_INPUT, src_img);
    cv::imshow(WINDOW_NAME_OUTPUT, dst_img);
    cv::waitKey(0);

    return 0;
}
```

プログラム (一部)

黒曜石・トルコ石・黄金それぞれの個数をコンソールに出力し、同時にそれぞれの画像位置に描画を行うプログラムを書いた。
・黒曜石（Black gem）は赤で外接長方形
・トルコ石（Green gem）は緑で塗りつぶし
・黄金（Golden gem）は白で外接円

Information

・制作時間・・・・3 時間

・制作時期・・・・2024 年 7 月

・制作人数・・・・1 人

・使用ソフト・・・Xcode

・使用言語・・・・C++