How it works:

1. Setup() and new function

```
void ofApp::setup() {
    cam.setup(320, 240);
    colorImg.allocate(320, 240);
    hueImg.allocate(320, 240);
    satImg.allocate(320, 240);
    valImg.allocate(320, 240);
    blueMask.allocate(320, 240);

    spawnInitialGreenCircles();
    spawnRedRectangles();
}
```

Import webcam, allocate color image, hue channel, saturation channel, value channel and mask for bule detection. Add two new function for create green circles and red rectangles from start of the game.

spawnInitialGreenCircles()

```
void ofApp::spawnInitialGreenCircles() {
    greenCircles.clear();
    for (int i = 0; i < 5; i++) {
        greenCircles.push_back(ofVec2f(ofRandom(320), ofRandom(240)));
    }
}
```

Here is the function create green circles when start the game. Loop 5 times to create 5 green circles in random position. Using push back for pushing the position value into draw function.

spawnRedRectangles()

```
void ofApp::spawnRedRectangles() {
    redRectangles.clear();
    for (int i = 0; i < 3; i++) {
        redRectangles.push_back(ofVec2f(ofRandom(320), ofRandom(240)));
    }
}
```

It works the same way as spawnInitialGreenCircles() function.

2. Update()

```
void ofApp::update() {
    cam.update();

    if (cam.isFrameNew() && !isGameOver) {
        colorImg.setFromPixels(cam.getPixels());    // webcam pixels to OpenCV image

        // convert to HSV
        ofxCvColorImage hsvImg;
        hsvImg.allocate(320, 240);
        hsvImg = colorImg;
        hsvImg.convertRgbToHsv();                   // RGB to HSV

        // extract HSV channels
        hsvImg.convertToGrayscalePlanarImages(hueImg, satImg, valImg);

        // create a mask for blue detection
        for (int i = 0; i < 320 * 240; i++) {
            unsigned char hue = hueImg.getPixels()[i];
            unsigned char sat = satImg.getPixels()[i];
            unsigned char val = valImg.getPixels()[i];

            // check pixel is in the blue colour value
            if (hue >= 100 && hue <= 140 && sat >= 100 && val >= 50) {
                blueMask.getPixels()[i] = 255;  // blue
            }
            else {
                blueMask.getPixels()[i] = 0;    // !blue
            }
        }

        blueMask.flagImageChanged();            // update mask

        // find blue blob
        contourFinder.findContours(blueMask, 20, 6400, 1, false);

        //blue circle detected
        if (contourFinder.nBlobs > 0) {
            ofVec2f blueCirclePos = contourFinder.blobs[0].centroid; // circle's position
            checkCollisions(blueCirclePos);                          // collisions checking
        }

        // create a new green circle every 3 seconds
        if (ofGetElapsedTimef() - lastSpawnTime > spawnInterval) {
            spawnNewGreenCircle();
            lastSpawnTime = ofGetElapsedTimef();
        }
    }
}
```

First update the webcam. Next make a if statement for detecting webcam new frame and game state (GameOver is true or false). In the statement, convert the webcam pixels to opencv image. Then convert to HSV and RGB to HSV. Also need to extract the HSV channels of the image. Next create a mask for colour detection. Here I want to detect blue color. Loop through all the image pixel, get the hue, sat and val of each image pixels. Then set a range that is in blue colour range. When detected set blue mask to 255 and update the mask. Next find blue blob using the mask. When a blue circle is detected get the position. I have a new function call checkCollisions() for check the collisions.

Also the game will create new green cicle ever 3 secounds, using ofGetElaspsedTimef() to counting 4 secounds, then use new function call spawnNewGreenCircle() to spawn new circle.

checkCollisions()

```
void ofApp::checkCollisions(ofVec2f blueCirclePos) {

    for (int i = greenCircles.size() - 1; i >= 0; i--) {
        if (blueCirclePos.distance(greenCircles[i]) < 15) {
            greenCircles.erase(greenCircles.begin() + i);
            score++;
        }
    }


    for (auto& rect : redRectangles) {
        if (blueCirclePos.x > rect.x && blueCirclePos.x < rect.x + 20 &&
            blueCirclePos.y > rect.y && blueCirclePos.y < rect.y + 20) {
            isGameOver = true;
            cam.close();
            break;
        }
    }
}
```

Here check the collision with green circles and red rectangles. When collision with green circles score +1, when collision with red rectangles set isGameOver to true, turn the webcam off.

spawnNewGreenCircle()

```
void ofApp::spawnNewGreenCircle() {
    greenCircles.push_back(ofVec2f(ofRandom(320), ofRandom(240)));
}
```

Create new green circle in random position

3. Draw()

```
void ofApp::draw() {
    cam.draw(0, 0);

    // Draw green circles
    ofSetColor(0, 255, 0);
    for (auto& circle : greenCircles) {
        ofDrawCircle(circle.x, circle.y, 10);
    }

    // Draw red rectangles
    ofSetColor(255, 0, 0);
    for (auto& rect : redRectangles) {
        ofDrawRectangle(rect.x, rect.y, 20, 20);
    }

    // Draw the score
    ofSetColor(255, 255, 255);
    ofDrawBitmapString("Score: " + ofToString(score), 10, 20);

    // gameover print "Finish" and final score
    if (isGameOver) {
        ofDrawBitmapString("Finish", 150, 120);
        ofDrawBitmapString("Final Score: " + ofToString(score), 140, 140);
    }
}
```

Here draw the webcam, green circle, red rectangles, the score board in to the screen. Make a if statement when the game is over, draw “finish” and “Final Score” into the screen.

4. Game restart

```
void ofApp::keyPressed(int key) {
    if (isGameOver && key == 'r') {
        isGameOver = false;
        score = 0;
        greenCircles.clear();
        redRectangles.clear();
        spawnInitialGreenCircles();
        spawnRedRectangles();
        cam.setup(320, 240);
    }
```

When game is over, pressing button ‘r’ will reset the game. Juse reset the isGameOver to false, set score to 0, clear the circles and rectangles in the screen, recreate new circles and rectangles, last is to setup webcam again.