Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего образования

«Московский политехнический университет»

Кафедра «СМАРТ-технологии»

**Лабораторная работа №5**

По дисциплине: «Технологии визуализации данных систем управления»

По теме: «Применение технологий дополненной реальности»

Группа: 211-327

Выполнил студент:

Роднов К.А. \_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватели: Ковалева В. В.,

Идиатуллов Т. Т.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2024

**Цель работы:** разработать приложение для визуализации данных технической системы с использованием технологий дополненной реальности.

**Задачи:**

* Разработать приложение для визуализации технической информации по роботизированной производственной линии.
* Коммуникацию по сбору данных с производственного оборудования обеспечить с использованием платформы ThingWorx, иной технологии сбора данных или посредством непосредственного подключения к оборудованию.
* Реализовать детекцию положения системы координат и визуализацию данных средствами технологий дополненной реальности.
* Выполнить разработку сцены дополненной реальности в соответствии индивидуальным заданием.

**Ход работы:**

1) Было разработано web-приложение, выполняющее роль сервера на локальном хосте для отправки закодированных данных на основное приложение. Приложение использует в качестве интерфейса html-страницу. Приём сообщений также ведётся с помощью класса HttpClient.

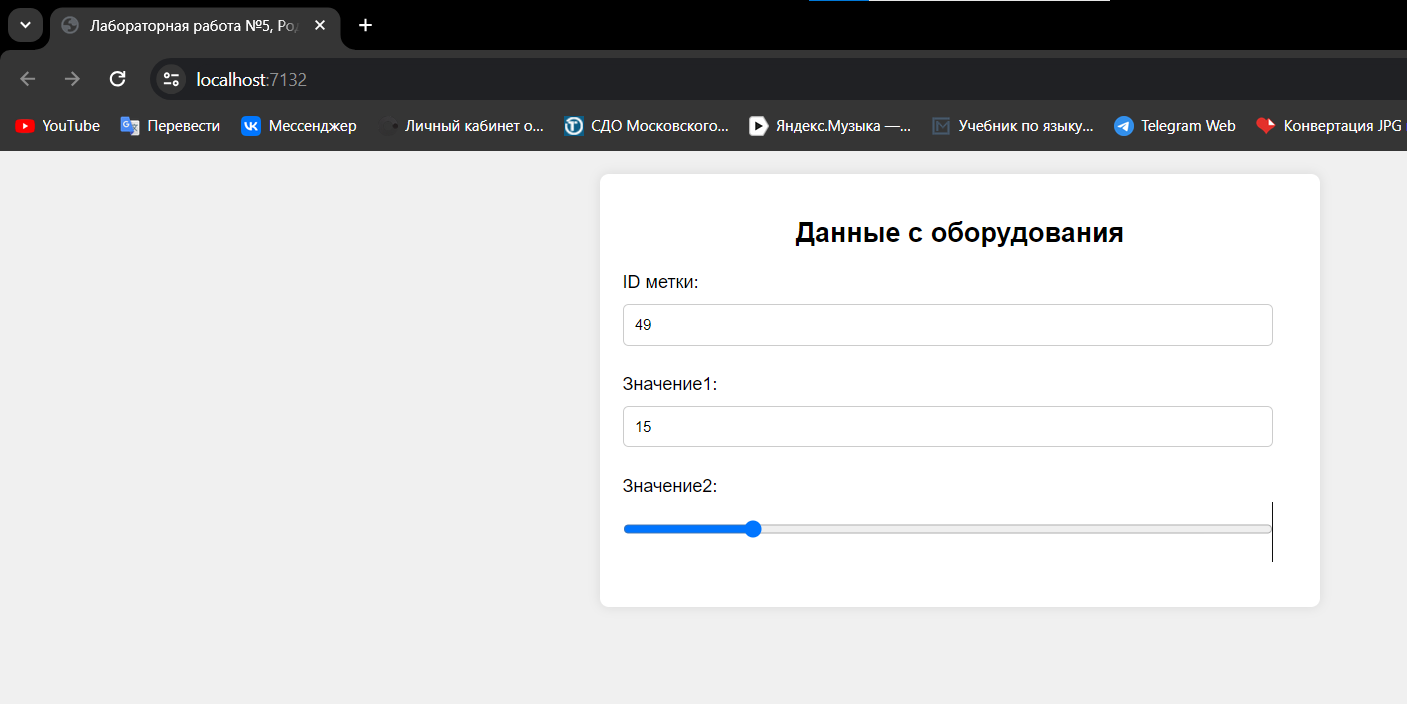


Рисунок 1 – Интерфейс html-страницы

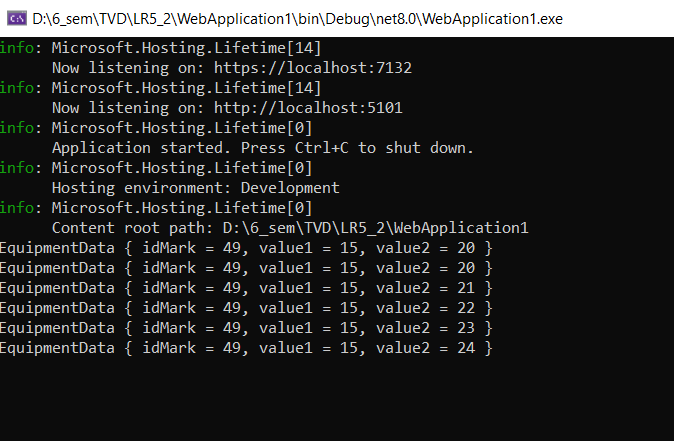


Рисунок 2 – Приём данных с сервера

2) Было создано основное приложение, реализующее приём данных по средством http-запросов к серверу.



Рисунок 3 – Форма основного приложения

3) Был реализован отладочный режим визуализации данных, при котором производится захват изображения с камеры с последующим распознаванием ARuCo меток на нём. Данные об ID и координатах метки заносятся в таблицу, после чего метка обводится контуром. После распознавания метки, высчитывается матрица поворота относительно текстуры камеры и дорисовывается объёмный куб с заданным ID метки.

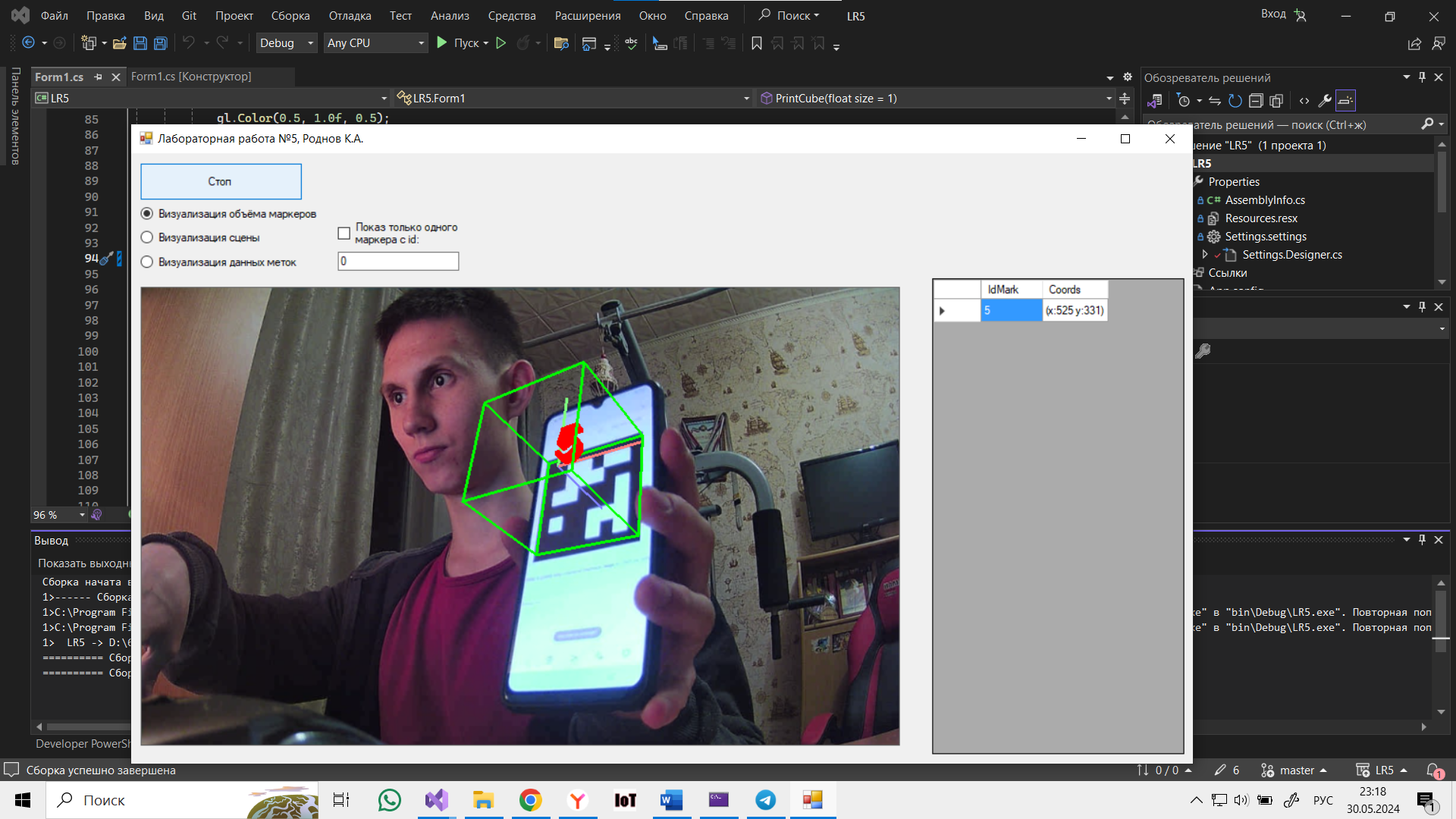


Рисунок 4 – Визуализации распознанной метки

4) Была разработана функция отрисовки сцены в окне OpenGL

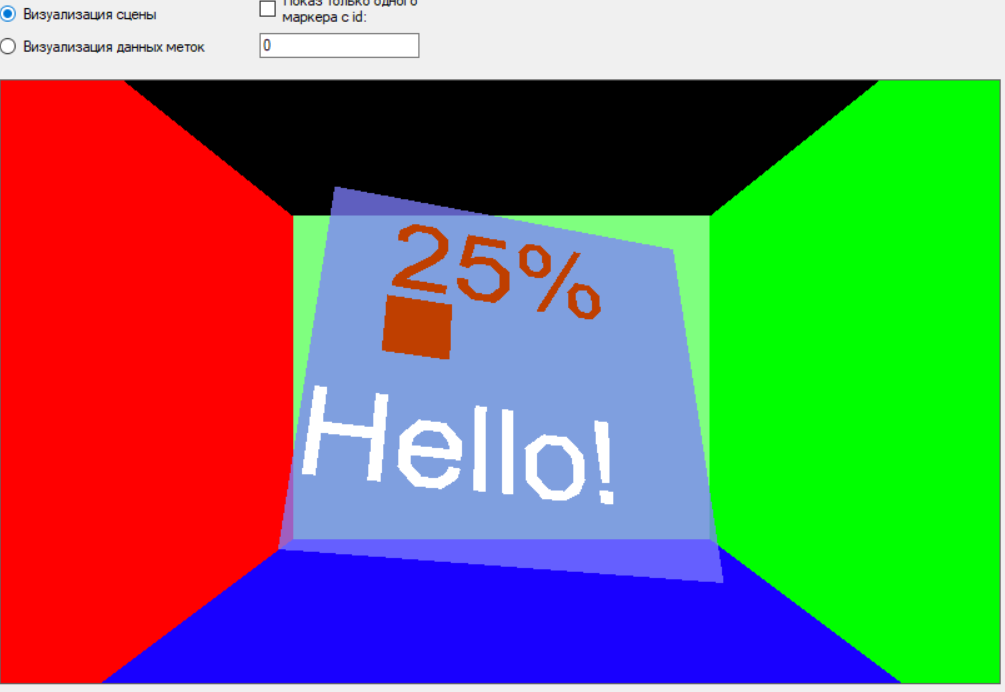


Рисунок 5 – Визуализация статичной сцены

5) Был реализован стандартный режим отображения для данных маркеров, приходящих с сервера. Данные отображаются в виде текстуры с изменяемыми в реальном времени параметрами.

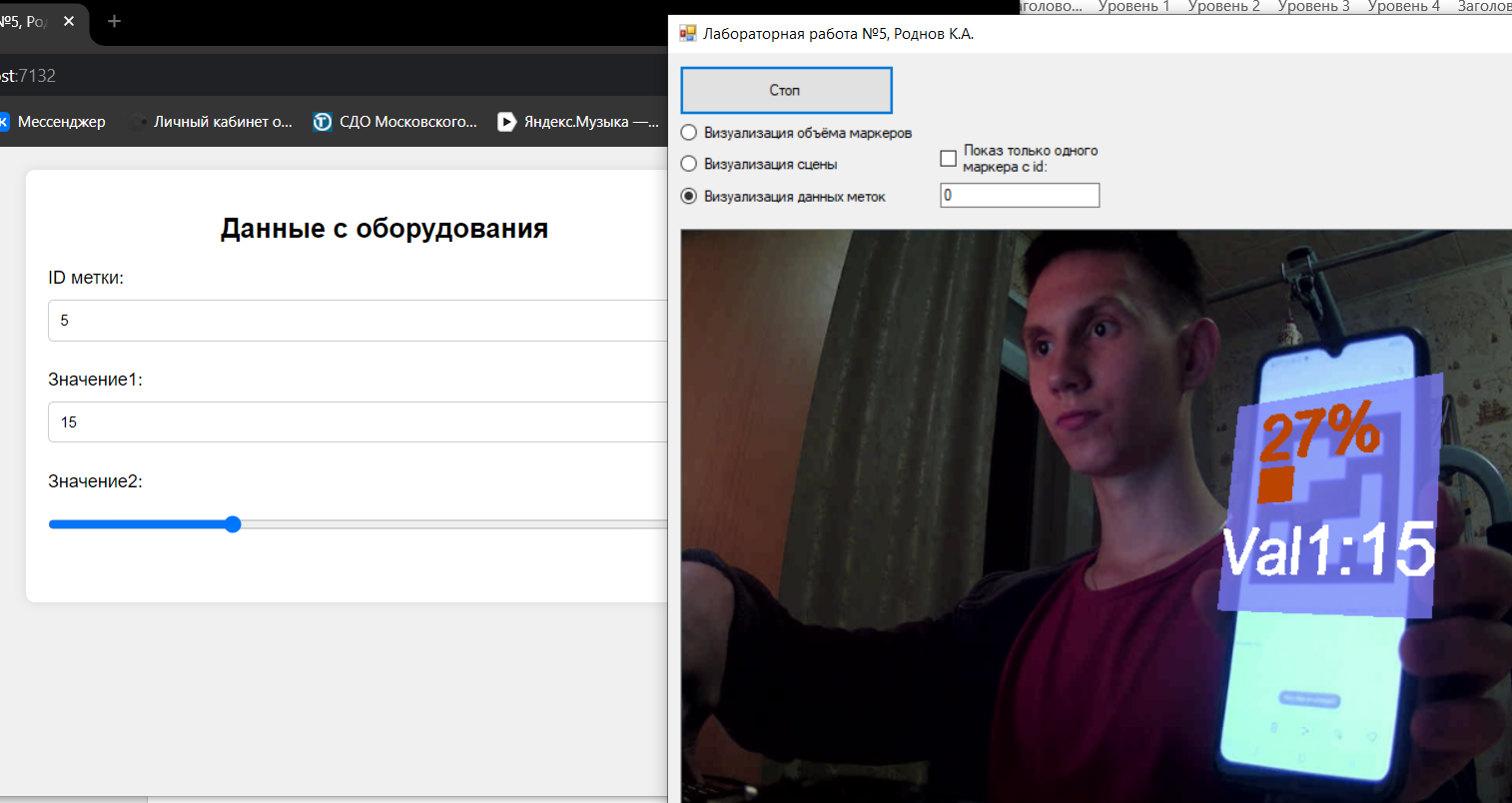


Рисунок 6 - Стандартный режим дополненной реальности

6) Реализована функция показа только одного маркера с заданным ID для лучшего управления приходящими данными, в случае если меток больше двух.

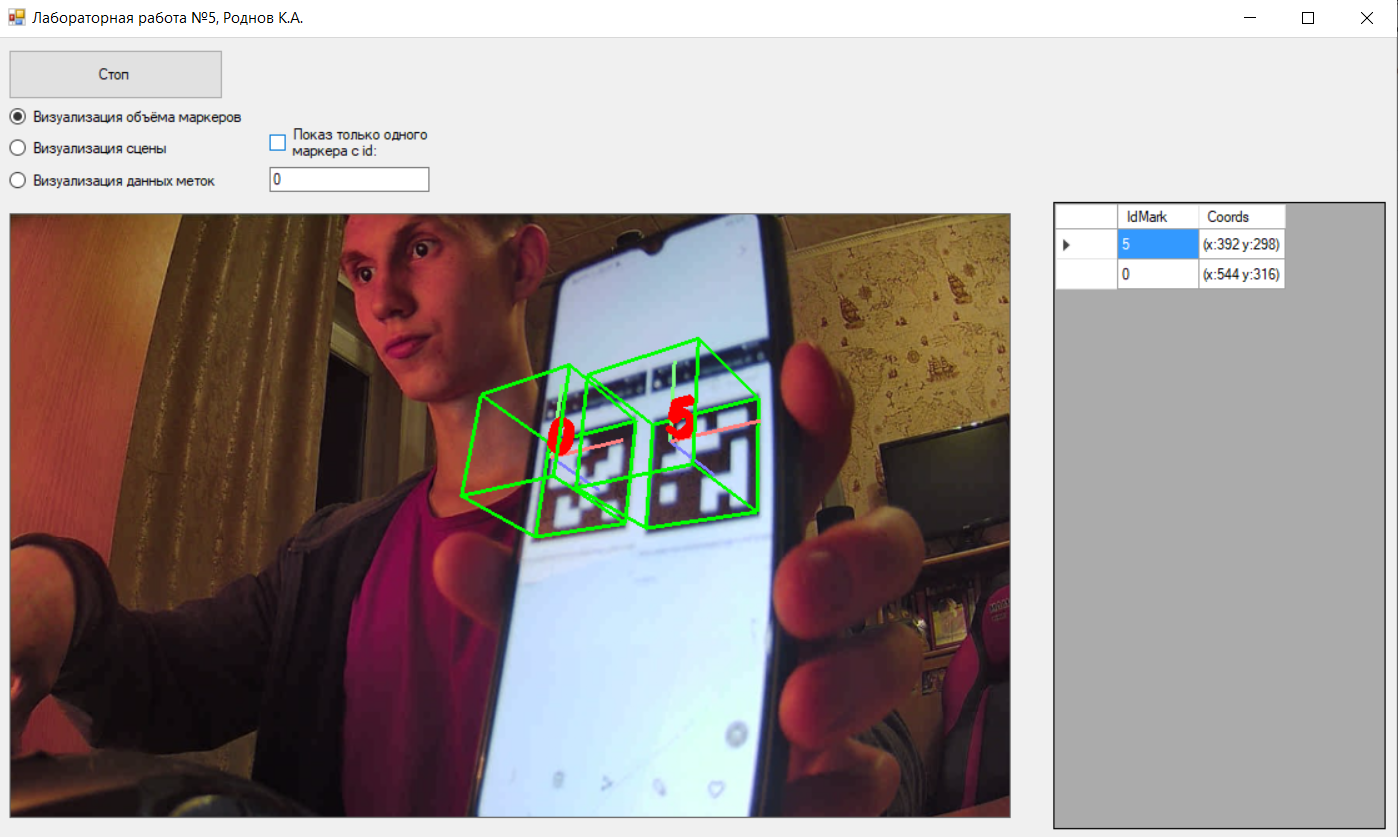


Рисунок 7 – Показ двух меток

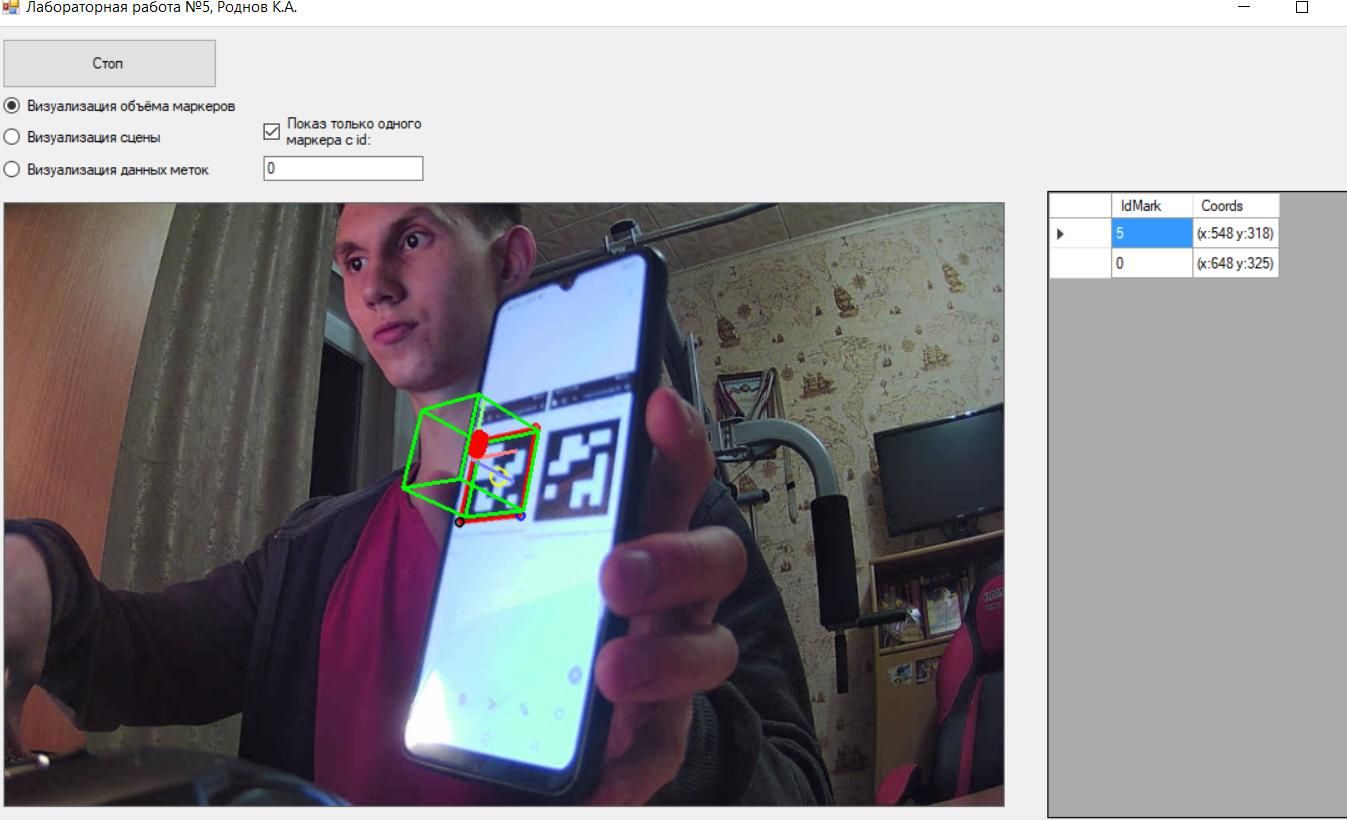


Рисунок 8 – Распознавание только одного маркера с заданным ID

**Вывод:** в ходе лабораторной работы было разработано приложение для визуализации данных технической системы с использованием технологий дополненной реальности.

**Приложение А**

Листинг А1 – Код Web-сервера

**namespace** WebApplication1

**{**

**public** class Program

**{**

**public** static void Main**(**string**[]** args**)**

**{**

var builder **=** WebApplication**.**CreateBuilder**();**

var app **=** builder**.**Build**();**

var message **=** ""**;**

app**.**Run**(async** **(**context**)** **=>**

**{**

var response **=** context**.**Response**;**

var request **=** context**.**Request**;**

**if** **(**request**.**Path **==** "/data"**)**

**{**

**if** **(**request**.**HasJsonContentType**())**

**{**

var person **=** **await** request**.**ReadFromJsonAsync**<**EquipmentData**>();**

**if** **(**person **!=** **null)**

message **=** person**.**ToString**();**

**}**

**await** response**.**WriteAsJsonAsync**(**message**);**

Console**.**WriteLine**(**message**);**

**}**

**else**

**{**

response**.**ContentType **=** "text/html; charset=utf-8"**;**

**await** response**.**SendFileAsync**(**"html/index.html"**);**

**}**

**});**

app**.**Run**();**

**}**

**public** record EquipmentData**(**int idMark**,** int value1**,** int value2**);**

**}**

**}**

Листинг А2 – Код html-страницы

**<!**DOCTYPE html**>**

**<**html lang**=**"ru"**>**

**<**head**>**

**<**meta charset**=**"UTF-8"**>**

**<**meta name**=**"viewport" content**=**"width=device-width, initial-scale=1.0"**>**

**<**title**>**Лабораторная работа №5**,** Роднов К**.**А**.</**title**>**

**<**style**>**

body **{**

font**-**family**:** Arial**,** sans**-**serif**;**

margin**:** 0**;**

padding**:** 0**;**

background**-**color**:** #f0f0f0**;**

**}**

#container {

max**-**width**:** 600px**;**

margin**:** 20px auto**;**

padding**:** 20px**;**

background**-**color**:** #fff**;**

border**-**radius**:** 8px**;**

box**-**shadow**:** 0 0 10px rgba**(**0**,** 0**,** 0**,** 0.1**);**

**}**

h2 **{**

text**-**align**:** center**;**

**}**

**.**form**-group** **{**

margin**-**bottom**:** 20px**;**

**}**

label **{**

display**:** block**;**

margin**-**bottom**:** 5px**;**

**}**

input**[**type**=**"number"**],**

input**[**type**=**"range"**]** **{**

width**:** calc**(**100**%** **-** 22px**);**

padding**:** 10px**;**

margin**:** 5px 0**;**

box**-**sizing**:** border**-**box**;**

border**:** 1px solid #ccc**;**

border**-**radius**:** 5px**;**

**}**

input**[**type**=**"range"**]** **{**

margin**-**bottom**:** 10px**;**

**}**

button **{**

width**:** 100**%;**

padding**:** 10px**;**

margin**:** 5px 0**;**

box**-**sizing**:** border**-**box**;**

border**:** none**;**

border**-**radius**:** 5px**;**

background**-**color**:** #007bff**;**

color**:** #fff**;**

cursor**:** pointer**;**

transition**:** background**-**color 0.3s ease**;**

**}**

button**:**hover **{**

background**-**color**:** #0056b3**;**

**}**

#message {

margin**-**top**:** 20px**;**

padding**:** 10px**;**

background**-**color**:** #f7f7f7**;**

border**-**radius**:** 5px**;**

**}**

**</**style**>**

**</**head**>**

**<**body**>**

**<**div id**=**"container"**>**

**<**h2**>**Данные с оборудования**</**h2**>**

**<**div class**=**"form-group"**>**

**<**label **for=**"idMark"**>**ID метки**:</**label**>**

**<**input name**=**"idMark" id**=**"idMark" type**=**"number"**,** **value=**"49"**>**

**</**div**>**

**<**div class**=**"form-group"**>**

**<**label **for=**"value1"**>**Значение1**:</**label**>**

**<**input name**=**"value1" id**=**"value1" type**=**"number"**,** **value=**"15"**>**

**</**div**>**

**<**div class**=**"form-group"**>**

**<**label **for=**"value2"**>**Значение2**:</**label**>**

**<**input type**=**"range" id**=**"value2" min**=**"0" max**=**"100" **value=**"18"**>**

**</**div**>**

**</**div**>**

**<**script**>**

const idMarkInput **=** document**.**getElementById**(**"idMark"**);**

const value1Input **=** document**.**getElementById**(**"value1"**);**

const value2Input **=** document**.**getElementById**(**"value2"**);**

const sendBtn **=** document**.**getElementById**(**"sendBtn"**);**

value1Input**.**addEventListener**(**"input"**,** send**);**

value2Input**.**addEventListener**(**"input"**,** send**);**

**async** function send**()** **{**

const response **=** **await** fetch**(**"/data"**,** **{**

method**:** "POST"**,**

headers**:** **{** "Accept"**:** "application/json"**,** "Content-Type"**:** "application/json" **},**

body**:** JSON**.**stringify**({**

idMark**:** idMarkInput**.value,**

value1**:** value1Input**.value,**

value2**:** value2Input**.value**

**})**

**});**

const message **=** **await** response**.**json**();**

**}**

**</**script**>**

**</**body**>**

**</**html**>**

Листинг А3 – Код приложения-клиента

**using** OpenCvSharp**;**

**using** OpenCvSharp**.**Aruco**;**

**using** OpenCvSharp**.**Extensions**;**

**using** SharpGL**;**

**using** SharpGL**.**Enumerations**;**

**using** SharpGL**.**SceneGraph**.**Assets**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Net**.**Http**;**

**using** System**.**Text**.**RegularExpressions**;**

**using** System**.**Threading**;**

**using** System**.**Windows**.**Forms**;**

**using** Point **=** OpenCvSharp**.**Point**;**

**namespace** LR5

**{**

**public** **partial** class Form1 **:** Form

**{**

OpenGL gl**;**

Thread cameraThread**,** dataFromApiThread**;**

bool runVideo**;**

VideoCapture capture**;**

DetectorParameters detectorParam **=** **new** DetectorParameters**();**

Mat matInput**;**

Point2f**[][]** corners**;**

int**[]** ids **=** **new** int**[]** **{** **};**

Mat rvec**,** tvec**;**

bool**[]** displayMode **=** **new** bool**[]** **{** **true,** **false,** **false** **};**

bool showOneMarker**;**

int idMark **=** 0**;**

List**<**EquipmentData**>** data **=** **new** List**<**EquipmentData**>();**

**readonly** Mat cameraMatrix **=** **new** Mat**(**3**,** 3**,** MatType**.**CV\_64F**,** **new** double**[]** **{** 968.08624052**,** 0**,** 644.63057786**,** 0**,** 955.40821957**,** 364.58219136**,** 0**,** 0**,** 1 **});**

**readonly** Mat distCoeffs **=** **new** Mat**(**5**,** 1**,** MatType**.**CV\_64F**,** **new** double**[]** **{** **-**4.73423959e-02**,** **-**1.25875642e+00**,** 1.12354237e-04**,** 4.72701099e-03**,** 6.96144663e+00 **});**

**readonly** Point3f**[]** objectPoints **=** **new** Point3f**[]** **{** **new** Point3f**(**1.0f**,** 1.0f**,** 0**),** **new** Point3f**(-**1.0f**,** 1.0f**,** 0**),** **new** Point3f**(-**1.0f**,** **-**1.0f**,** 0**),** **new** Point3f**(**1.0f**,** **-**1.0f**,** 0**)** **};**

**readonly** Scalar**[]** colorsScalar **=** **new** Scalar**[]** **{** Scalar**.**Red**,** Scalar**.**Green**,** Scalar**.**Black**,** Scalar**.**Blue **};**

**readonly** Dictionary dictionary**;**

**readonly** Texture cameraTexture **=** **new** Texture**();**

**private** void button1\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**runVideo**)**

**{**

runVideo **=** **false;**

timer1**.**Stop**();**

DisposeVideo**();**

button1**.**Text **=** "Начать захват изображения"**;**

**}**

**else**

**{**

timer1**.**Start**();**

runVideo **=** **true;**

matInput **=** **new** Mat**();**

capture **=** **new** VideoCapture**(**0**)**

**{**

FrameHeight **=** 720**,**

FrameWidth **=** 1280**,**

AutoFocus **=** **true**

**};**

cameraThread **=** **new** Thread**(new** ThreadStart**(**CaptureCameraCallback**));**

cameraThread**.**Start**();**

dataFromApiThread **=** **new** Thread**(new** ThreadStart**(**ReceivingData**));**

dataFromApiThread**.**Start**();**

button1**.**Text **=** "Стоп"**;**

**}**

**}**

**private** void DisposeVideo**()**

**{**

**if** **(**cameraThread **!=** **null** **&&** cameraThread**.**IsAlive**)** cameraThread**.**Abort**();**

**if** **(**dataFromApiThread **!=** **null** **&&** dataFromApiThread**.**IsAlive**)** dataFromApiThread**.**Abort**();**

matInput**?.**Dispose**();**

capture**?.**Dispose**();**

**}**

**public** void PrintCube**(**float size **=** 1.0f**)**

**{**

gl**.**LineWidth**(**3**);**

gl**.**Begin**(**OpenGL**.**GL\_LINES**);**

gl**.**Color**(**1.0f**,** 0.5**,** 0.5**);**

gl**.**Vertex**(**0.0f**,** 0**,** 0**);**

gl**.**Vertex**(**size **\*** 1.5**,** 0**,** 0**);**

gl**.**Color**(**0.5**,** 1.0f**,** 0.5**);**

gl**.**Vertex**(**0.0f**,** 0**,** 0**);**

gl**.**Vertex**(**0**,** size **\*** 1.5**,** 0**);**

gl**.**Color**(**0.5**,** 0.5**,** 1.0f**);**

gl**.**Vertex**(**0**,** 0**,** 0.0f**);**

gl**.**Vertex**(**0**,** 0**,** size **\*** 1.5**);**

gl**.**End**();**

gl**.**Color**(**0.0**,** 1**,** 0.0f**);**

gl**.**PolygonMode**(**OpenGL**.**GL\_FRONT\_AND\_BACK**,** OpenGL**.**GL\_LINE**);**

gl**.**Begin**(**BeginMode**.**Quads**);**

gl**.**Vertex**(-**size**,** size**,** **-**size**);** // top

gl**.**Vertex**(**size**,** size**,** **-**size**);**

gl**.**Vertex**(**size**,** size**,** size**);**

gl**.**Vertex**(-**size**,** size**,** size**);**

gl**.**Vertex**(-**size**,** **-**size**,** **-**size**);** // bott0m

gl**.**Vertex**(**size**,** **-**size**,** **-**size**);**

gl**.**Vertex**(**size**,** **-**size**,** size**);**

gl**.**Vertex**(-**size**,** **-**size**,** size**);**

gl**.**Vertex**(-**size**,** **-**size**,** **-**size**);** // left

gl**.**Vertex**(-**size**,** size**,** **-**size**);**

gl**.**Vertex**(-**size**,** size**,** size**);**

gl**.**Vertex**(-**size**,** **-**size**,** size**);**

gl**.**Vertex**(**size**,** **-**size**,** **-**size**);** // right

gl**.**Vertex**(**size**,** size**,** **-**size**);**

gl**.**Vertex**(**size**,** size**,** size**);**

gl**.**Vertex**(**size**,** **-**size**,** size**);**

gl**.**End**();**

**}**

**public** Form1**()**

**{**

InitializeComponent**();**

dictionary **=** CvAruco**.**GetPredefinedDictionary**(**PredefinedDictionaryName**.**Dict6X6\_50**);**

**}**

**private** void openGLControl1\_OpenGLInitialized**(object** sender**,** EventArgs e**)**

**{**

gl **=** openGLControl1**.**OpenGL**;**

gl**.**Clear**(**OpenGL**.**GL\_COLOR\_BUFFER\_BIT **|** OpenGL**.**GL\_DEPTH\_BUFFER\_BIT**);**

gl**.**Enable**(**OpenGL**.**GL\_TEXTURE\_2D**);**

gl**.**Enable**(**OpenGL**.**GL\_DEPTH\_TEST**);**

gl**.**Enable**(**OpenGL**.**GL\_BLEND**);**

gl**.**BlendFunc**(**OpenGL**.**GL\_SRC\_ALPHA**,** OpenGL**.**GL\_ONE\_MINUS\_SRC\_ALPHA**);**

**}**

**private** void openGLControl1\_OpenGLDraw**(object** sender**,** RenderEventArgs args**)**

**{**

gl**.**Clear**(**OpenGL**.**GL\_COLOR\_BUFFER\_BIT **|** OpenGL**.**GL\_DEPTH\_BUFFER\_BIT**);**

**if** **(**runVideo **&&** **!**matInput**.**Empty**())**

**{**

**if** **(!**displayMode**[**1**])** ShowCameraGL**();**

gl**.**MatrixMode**(**OpenGL**.**GL\_PROJECTION**);**

gl**.**LoadIdentity**();**

gl**.**Perspective**(**45**,** openGLControl1**.**Width **/** **(**double**)**openGLControl1**.**Height**,** 0.1f**,** 100.0f**);**

gl**.**MatrixMode**(**OpenGL**.**GL\_MODELVIEW**);**

gl**.**LoadIdentity**();**

**if** **(**ids**.**Length **>** 0**)**

**{**

**for** **(**int i **=** 0**;** i **<** ids**.**Length**;** i**++)**

**{**

**if** **(**showOneMarker **&&** ids**[**i**]** **!=** idMark**)** **continue;**

rvec **=** **new** Mat**();**

tvec **=** **new** Mat**();**

Cv2**.**SolvePnP**(**InputArray**.**Create**(**objectPoints**),** InputArray**.**Create**(**corners**[**i**]),** cameraMatrix**,** distCoeffs**,** rvec**,** tvec**);**

**if** **(!**rvec**.**Empty**()** **&&** **!**tvec**.**Empty**())**

**{**

var matrix **=** TransitionToMark**(**rvec**,** tvec**);**

gl**.**LoadMatrix**(**matrix**);**

// Отладочный режим

**if** **(**displayMode**[**0**])**

**{**

gl**.**Color**(**1.0f**,** 0.0f**,** 0.0f**);**

gl**.**PolygonMode**(**OpenGL**.**GL\_FRONT\_AND\_BACK**,** OpenGL**.**GL\_FILL**);**

gl**.**Translate**(**0.0f**,** 0.0f**,** **-**1.0f**);**

gl**.**DrawText3D**(**"a"**,** 0.2f**,** 0.2f**,** ids**[**i**].**ToString**());**

gl**.**LoadMatrix**(**matrix**);**

gl**.**Translate**(**0.0f**,** 0.0f**,** **-**1.0f**);**

PrintCube**();**

**}**

// Стандартый режим

**else**

**{**

var existingData **=** data**.**FirstOrDefault**(**d **=>** d**.**ID **==** ids**[**i**]);**

**if** **(**existingData **!=** **null)**

**{**

PrintEquipmentData\_1**(**existingData**.**Value1**,** existingData**.**Value2**,** matrix**);**

**}**

**}**

**}**

**}**

**}**

**}**

// Визуализация сцены

**if** **(**displayMode**[**1**])**

**{**

gl**.**Clear**(**OpenGL**.**GL\_COLOR\_BUFFER\_BIT **|** OpenGL**.**GL\_DEPTH\_BUFFER\_BIT**);**

gl**.**LoadIdentity**();**

gl**.**Translate**(**0**,** 0.0f**,** **-**5.0f**);**

gl**.**Color**(**0.1f**,** 0.0f**,** 1.0f**);**

gl**.**PolygonMode**(**OpenGL**.**GL\_FRONT\_AND\_BACK**,** OpenGL**.**GL\_FILL**);**

gl**.**Begin**(**BeginMode**.**Quads**);**

gl**.**Vertex**(-**2.0f**,** **-**1.5f**,** **-**2.0f**);** // b0ttom

gl**.**Vertex**(**2.0f**,** **-**1.5f**,** **-**2.0f**);**

gl**.**Vertex**(**2.0f**,** **-**1.5f**,** 3.0f**);**

gl**.**Vertex**(-**2.0f**,** **-**1.5f**,** 3.0f**);**

gl**.**Color**(**1.0f**,** 0.0f**,** 0.0f**);**

gl**.**Vertex**(-**2.5f**,** **-**2**,** **-**4f**);** // left

gl**.**Vertex**(-**2.5f**,** 2**,** **-**4f**);**

gl**.**Vertex**(-**2.5f**,** 2**,** 2.5f**);**

gl**.**Vertex**(-**2.5f**,** **-**2**,** 2.5f**);**

gl**.**Color**(**0.0f**,** 1.0f**,** 0.0f**);**

gl**.**Vertex**(**2.5f**,** **-**2**,** **-**4f**);** // right

gl**.**Vertex**(**2.5f**,** 2**,** **-**4f**);**

gl**.**Vertex**(**2.5f**,** 2**,** 2.5f**);**

gl**.**Vertex**(**2.5f**,** **-**2**,** 2.5f**);**

gl**.**Color**(**0.5f**,** 1.0f**,** 0.5f**);**

gl**.**Vertex**(-**2.0f**,** 1.6f**,** **-**2.0f**);** // back

gl**.**Vertex**(**2.0f**,** 1.6f**,** **-**2.0f**);**

gl**.**Vertex**(**2.0f**,** **-**2f**,** **-**2.0f**);**

gl**.**Vertex**(-**2.0f**,** **-**2f**,** **-**2.0f**);**

gl**.**End**();**

gl**.**LoadIdentity**();**

gl**.**Translate**(**0.1**,** 0.1**,** **-**5.5f**);**

gl**.**Rotate**(-**30**,** 15**,** 0**);**

/\* Texture markTexture = new Texture();

markTexture.Create(gl, "mark.bmp");

markTexture.Bind(gl);

gl.PolygonMode(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_FILL);

gl.Color(1.0f, 1.0f, 1.0f, 1.0f);

gl.Begin(OpenGL.GL\_QUADS);

gl.TexCoord(0.0f, 0.0f); gl.Vertex(1.0f, 1.0f, 0);

gl.TexCoord(1.0f, 0.0f); gl.Vertex(-1.0f, 1.0f, 0);

gl.TexCoord(1.0f, 1.0f); gl.Vertex(-1.0f, -1.0f, 0);

gl.TexCoord(0.0f, 1.0f); gl.Vertex(1.0f, -1.0f, 0);

gl.End();\*/

PrintEquipmentData**(**31**,** 25**);**

**}**

gl**.**Flush**();**

**}**

**private** void PrintEquipmentData\_1**(**int val1**,** int val2**,** double**[]** matrix**)**

**{**

gl**.**Begin**(**OpenGL**.**GL\_QUADS**);**

gl**.**Color**(**0.5f**,** 0.5f**,** 1f**,**0.75f**);**

gl**.**Vertex**(-**1.5f**,** **-**1.5f**);**

gl**.**Vertex**(-**1.5f**,** 1.5f**);**

gl**.**Vertex**(**1.5f**,** 1.5f**);**

gl**.**Vertex**(**1.5f**,** **-**1.5f**);**

gl**.**End**();**

Color barColor **=** GetProgressBarColor**(**val2**);**

gl**.**Color**(**barColor**.**R **/** 255.0**,** barColor**.**G **/** 255.0**,** barColor**.**B **/** 255.0**);**

gl**.**Translate**(-**1**,** 0.6**,** **-**0.01f**);**

gl**.**DrawText3D**(**"a"**,** 0.1f**,** 0.1f**,** $"{val2}%"**);**

gl**.**LoadMatrix**(**matrix**);**

gl**.**Translate**(**0**,** 0**,** **-**0.01f**);**

gl**.**Begin**(**OpenGL**.**GL\_QUADS**);**

gl**.**Vertex**(-**1.0f**,** 0f**);**

gl**.**Vertex**(-**1.0f**,** 0.5f**);**

gl**.**Vertex**((**val2 **/** 100.0f**)** **\*** 2 **-** 1**,** 0.5f**);**

gl**.**Vertex**((**val2 **/** 100.0f**)** **\*** 2 **-** 1**,** 0f**);**

gl**.**End**();**

gl**.**Translate**(-**1.5f**,** **-**1**,** **-**0.01f**);**

gl**.**Color**(**1f**,** 1f**,** 1f**);**

gl**.**DrawText3D**(**"a"**,** 0.05f**,** 0.05f**,** $"Val1:{val1}"**);**

**}**

**private** void PrintEquipmentData**(**int val1**,** int val2**)**

**{**

gl**.**Begin**(**OpenGL**.**GL\_QUADS**);**

gl**.**Color**(**0.5f**,** 0.5f**,** 1f**,** 0.75f**);**

gl**.**Vertex**(-**1.5f**,** **-**1.5f**);**

gl**.**Vertex**(-**1.5f**,** 1.5f**);**

gl**.**Vertex**(**1.5f**,** 1.5f**);**

gl**.**Vertex**(**1.5f**,** **-**1.5f**);**

gl**.**End**();**

Color barColor **=** GetProgressBarColor**(**val2**);**

gl**.**Color**(**barColor**.**R **/** 255.0**,** barColor**.**G **/** 255.0**,** barColor**.**B **/** 255.0**);**

gl**.**Translate**(-**1**,** 0.6**,** 0.01f**);**

gl**.**DrawText3D**(**"a"**,** 0.1f**,** 0.1f**,** $"{val2}%"**);**

gl**.**LoadIdentity**();**

gl**.**Translate**(**0.1**,** 0.1**,** **-**5.5f**);**

gl**.**Rotate**(-**30**,** 15**,** 0**);**

gl**.**Translate**(**0**,** 0**,** 0.01f**);**

gl**.**Begin**(**OpenGL**.**GL\_QUADS**);**

gl**.**Vertex**(-**1.0f**,** 0f**);**

gl**.**Vertex**(-**1.0f**,** 0.5f**);**

gl**.**Vertex**((**val2 **/** 100.0f**)** **\*** 2 **-** 1**,** 0.5f**);**

gl**.**Vertex**((**val2 **/** 100.0f**)** **\*** 2 **-** 1**,** 0f**);**

gl**.**End**();**

gl**.**Translate**(-**1.5f**,** **-**1**,** 0.01f**);**

gl**.**Color**(**1f**,** 1f**,** 1f**);**

gl**.**DrawText3D**(**"a"**,** 0.05f**,** 0.05f**,** $"Hello!"**);**

**}**

**private** Color GetProgressBarColor**(**int progress**)**

**{**

int red **=** **(**int**)(**255 **\*** **(**1 **-** **(**double**)**progress **/** 100**));**

int green **=** **(**int**)(**255 **\*** **((**double**)**progress **/** 100**));**

**return** Color**.**FromArgb**(**red**,** green**,** 0**);**

**}**

**private** double**[]** TransitionToMark**(**Mat localRvec**,** Mat localTvec**)**

**{**

gl**.**LoadIdentity**();**

Mat rotation **=** **new** Mat**();**

Mat viewMatrix **=** Mat**.**Zeros**(**rows**:** 4**,** 4**,** MatType**.**CV\_64F**);**

Cv2**.**Rodrigues**(**localRvec**,** rotation**);**

**for** **(**int row **=** 0**;** row **<** 3**;** **++**row**)**

**{**

**for** **(**int col **=** 0**;** col **<** 3**;** **++**col**)**

**{**

viewMatrix**.**At**<**double**>(**row**,** col**)** **=** rotation**.**At**<**double**>(**row**,** col**);**

**}**

viewMatrix**.**At**<**double**>(**row**,** 3**)** **=** localTvec**.**At**<**double**>(**row**,** 0**);**

**}**

viewMatrix**.**At**<**double**>(**3**,** 3**)** **=** 1.0f**;**

Mat cvToGl **=** Mat**.**Zeros**(**rows**:** 4**,** 4**,** MatType**.**CV\_64F**);**

cvToGl**.**At**<**double**>(**0**,** 0**)** **=** **-**1.0f**;** // Invert the x axis из-за отзеркаливания

cvToGl**.**At**<**double**>(**1**,** 1**)** **=** **-**1.0f**;** // Invert the y axis

cvToGl**.**At**<**double**>(**2**,** 2**)** **=** **-**1.0f**;** // invert the z axis

cvToGl**.**At**<**double**>(**3**,** 3**)** **=** 1.0f**;**

viewMatrix **=** cvToGl **\*** viewMatrix**;**

Mat glViewMatrix **=** Mat**.**Zeros**(**rows**:** 4**,** 4**,** MatType**.**CV\_64F**);**

Cv2**.**Transpose**(**viewMatrix**,** glViewMatrix**);**

double**[]** doubleArray **=** **new** double**[**glViewMatrix**.**Rows **\*** glViewMatrix**.**Cols**];**

**for** **(**int row **=** 0**;** row **<** glViewMatrix**.**Rows**;** row**++)**

**{**

**for** **(**int col **=** 0**;** col **<** glViewMatrix**.**Cols**;** col**++)**

**{**

doubleArray**[**row **\*** glViewMatrix**.**Cols **+** col**]** **=** glViewMatrix**.**At**<**double**>(**row**,** col**);**

**}**

**}**

**return** doubleArray**;**

**}**

**private** void ShowCameraGL**()**

**{**

**if** **(**matInput**.**Empty**())** **return;**

gl**.**MatrixMode**(**OpenGL**.**GL\_PROJECTION**);**

gl**.**LoadIdentity**();**

gl**.**Ortho**(-**1**,** 1**,** **-**1**,** 1**,** 0.1f**,** 100**);**

gl**.**MatrixMode**(**OpenGL**.**GL\_MODELVIEW**);**

gl**.**LoadIdentity**();**

gl**.**Translate**(**0.0f**,** 0.0f**,** **-**99.9f**);**

cameraTexture**.**Create**(**gl**,** matInput**.**ToBitmap**());**

cameraTexture**.**Bind**(**gl**);**

gl**.**PolygonMode**(**OpenGL**.**GL\_FRONT\_AND\_BACK**,** OpenGL**.**GL\_FILL**);**

gl**.**Color**(**1.0f**,** 1.0f**,** 1.0f**,** 1.0f**);**

gl**.**Begin**(**OpenGL**.**GL\_QUADS**);**

gl**.**TexCoord**(**0.0f**,** 0.0f**);** gl**.**Vertex**(**1.0f**,** 1.0f**,** 0**);**

gl**.**TexCoord**(**1.0f**,** 0.0f**);** gl**.**Vertex**(-**1.0f**,** 1.0f**,** 0**);**

gl**.**TexCoord**(**1.0f**,** 1.0f**);** gl**.**Vertex**(-**1.0f**,** **-**1.0f**,** 0**);**

gl**.**TexCoord**(**0.0f**,** 1.0f**);** gl**.**Vertex**(**1.0f**,** **-**1.0f**,** 0**);**

gl**.**End**();**

gl**.**BindTexture**(**OpenGL**.**GL\_TEXTURE\_2D**,** 0**);** // Отвязываем текстуру

**}**

**private** void CaptureCameraCallback**()**

**{**

**while** **(**runVideo**)**

**{**

**if** **(**displayMode**[**1**])** **continue;**

matInput **=** capture**.**RetrieveMat**();**

CvAruco**.**DetectMarkers**(**matInput**,** dictionary**,** **out** corners**,** **out** ids**,** detectorParam**,** **out** \_**);**

**if** **(**ids**.**Length **>** 0**)**

**{**

**if** **(**displayMode**[**0**])**

**{**

**for** **(**int j **=** 0**;** j **<** ids**.**Length**;** j**++)**

**{**

**if** **(**showOneMarker **&&** ids**[**j**]** **!=** idMark**)** **continue;**

Point center **=** **new** Point**((**corners**[**j**][**0**].**X **+** corners**[**j**][**1**].**X **+** corners**[**j**][**2**].**X **+** corners**[**j**][**3**].**X**)** **/** 4**,**

**(**corners**[**j**][**0**].**Y **+** corners**[**j**][**1**].**Y **+** corners**[**j**][**2**].**Y **+** corners**[**j**][**3**].**Y**)** **/** 4**);**

Cv2**.**Circle**(**matInput**,** center**,** 10**,** Scalar**.**Yellow**,** 2**);**

**for** **(**int i **=** 0**;** i **<** 4**;** i**++)**

**{**

Cv2**.**Line**(**matInput**,** corners**[**j**][**i**].**ToPoint**(),** corners**[**j**][(**i **+** 1**)** **%** 4**].**ToPoint**(),** Scalar**.**Red**,** 3**);**

Cv2**.**Circle**(**matInput**,** corners**[**j**][**i**].**ToPoint**(),** 5**,** colorsScalar**[**i**],** 2**);**

**}**

**}**

**}**

**}**

Invoke**(new** Action**(()** **=>**

**{**

GC**.**Collect**();**

GC**.**WaitForPendingFinalizers**();**

**}));**

**}**

**}**

**private** void radioButton1\_CheckedChanged**(object** sender**,** EventArgs e**)**

**{**

displayMode**[**0**]** **=** **true;**

displayMode**[**1**]** **=** **false;**

displayMode**[**2**]** **=** **false;**

**}**

**private** void radioButton2\_CheckedChanged**(object** sender**,** EventArgs e**)**

**{**

displayMode**[**0**]** **=** **false;**

displayMode**[**1**]** **=** **true;**

displayMode**[**2**]** **=** **false;**

**}**

**private** void radioButton3\_CheckedChanged**(object** sender**,** EventArgs e**)**

**{**

displayMode**[**0**]** **=** **false;**

displayMode**[**1**]** **=** **false;**

displayMode**[**2**]** **=** **true;**

**}**

**private** void checkBox1\_CheckedChanged**(object** sender**,** EventArgs e**)**

**{**

showOneMarker **=** checkBox1**.**Checked**;**

**}**

**private** void textBox1\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

int**.**TryParse**(**textBox1**.**Text**,** **out** idMark**);**

**}**

**private** void Form1\_Load**(object** sender**,** EventArgs e**)**

**{**

dataGridView1**.**Rows**.**Clear**();**

dataGridView1**.**Columns**.**Add**(**"IdMark"**,** "IdMark"**);**

dataGridView1**.**Columns**.**Add**(**"Coords"**,** "Coords"**);**

**}**

**private** void timer1\_Tick**(object** sender**,** EventArgs e**)**

**{**

**if** **(**ids**.**Length **>** 0**)**

**{**

dataGridView1**.**Rows**.**Clear**();**

**for** **(**int i **=** 0**;** i **<** ids**.**Length**;** i**++)**

**{**

Point center **=** **new** Point**((**corners**[**i**][**0**].**X **+** corners**[**i**][**1**].**X **+** corners**[**i**][**2**].**X **+** corners**[**i**][**3**].**X**)** **/** 4**,**

**(**corners**[**i**][**0**].**Y **+** corners**[**i**][**1**].**Y **+** corners**[**i**][**2**].**Y **+** corners**[**i**][**3**].**Y**)** **/** 4**);**

dataGridView1**.**Rows**.**Add**(**$"{ids[i]}"**,** $"{center}"**);**

**}**

**}**

**}**

**private** **async** void ReceivingData**()**

**{**

HttpClient client **=** **new** HttpClient**();**

**while** **(true)**

**{**

HttpResponseMessage response **=** **await** client**.**GetAsync**(**"http://localhost:5101/data"**);**

var responseBody **=** **await** response**.**Content**.**ReadAsStringAsync**();**

string pattern **=** @"idMark\s\*=\s\*(?<idMark>\d+),\s\*value1\s\*=\s\*(?<value1>\d+),\s\*value2\s\*=\s\*(?<value2>\d+)"**;**

Match match **=** Regex**.**Match**(**responseBody**,** pattern**);**

**if** **(**match**.**Success**)**

**{**

int idMark **=** int**.**Parse**(**match**.**Groups**[**"idMark"**].**Value**);**

int value1 **=** int**.**Parse**(**match**.**Groups**[**"value1"**].**Value**);**

int value2 **=** int**.**Parse**(**match**.**Groups**[**"value2"**].**Value**);**

var tempData **=** **new** EquipmentData**(**idMark**,** value1**,** value2**);**

var existingData **=** data**.**FirstOrDefault**(**d **=>** d**.**ID **==** idMark**);**

**if** **(**existingData **!=** **null)**

**{**

existingData**.**Value1 **=** value1**;**

existingData**.**Value2 **=** value2**;**

**}**

**else**

**{**

data**.**Add**(**tempData**);**

**}**

**}**

**}**

**}**

**private** void Form1\_FormClosing**(object** sender**,** FormClosingEventArgs e**)**

**{**

DisposeVideo**();**

**}**

**}**

**public** class EquipmentData

**{**

**public** int ID**;**

**public** int Value1**;**

**public** int Value2**;**

**public** EquipmentData**(**int ID**,** int Value1**,** int Value2**)**

**{**

**this.**ID **=** ID**;**

**this.**Value1 **=** Value1**;**

**this.**Value2 **=** Value2**;**

**}**

**}**

**}**