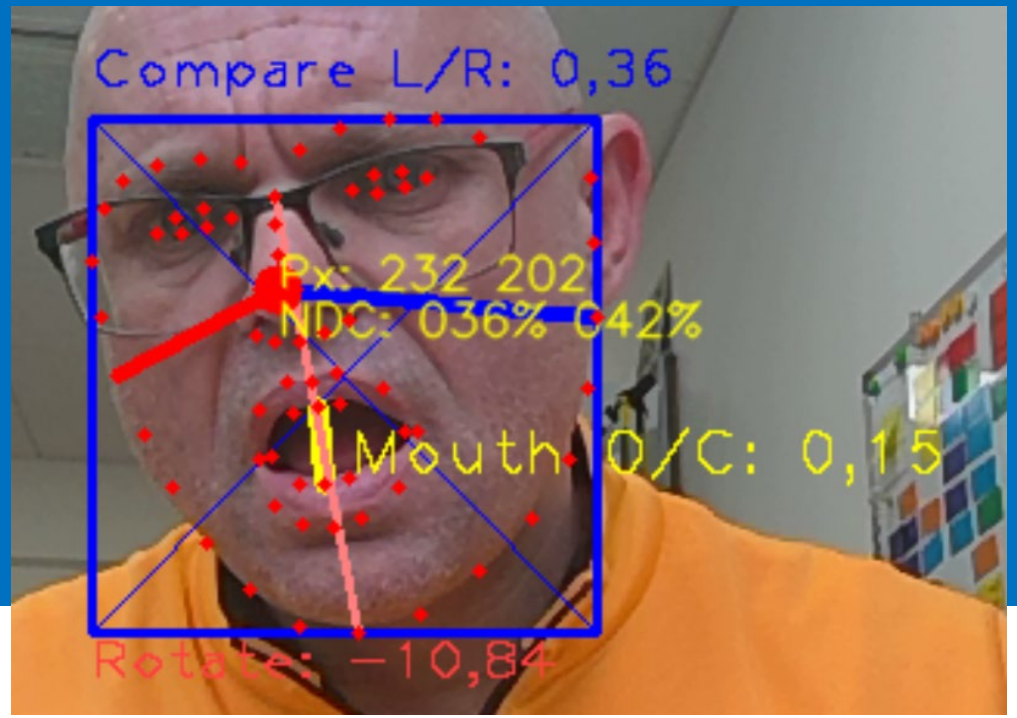
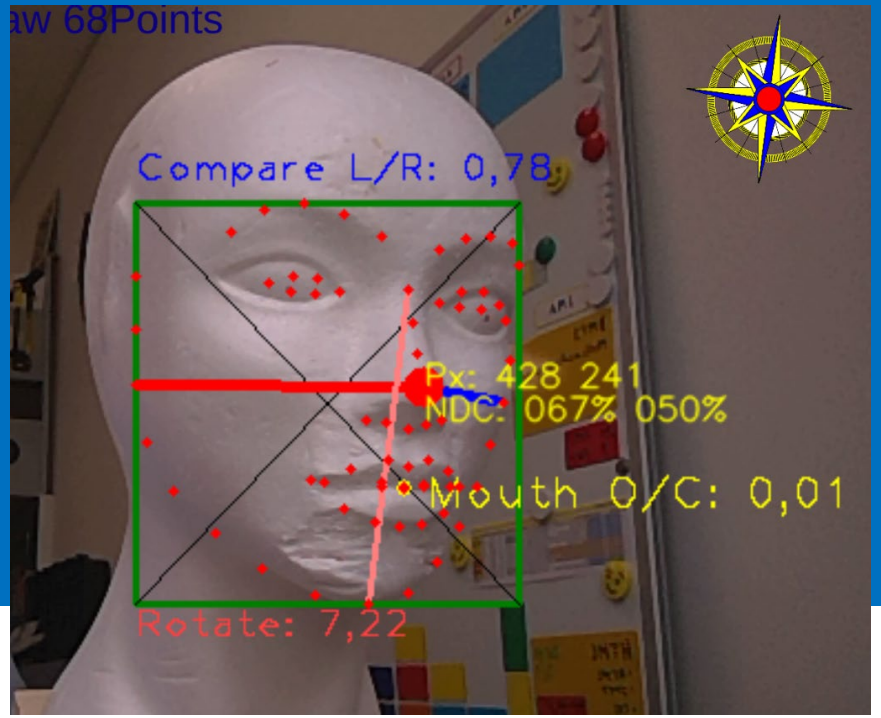
OpenCV
Computer Vision

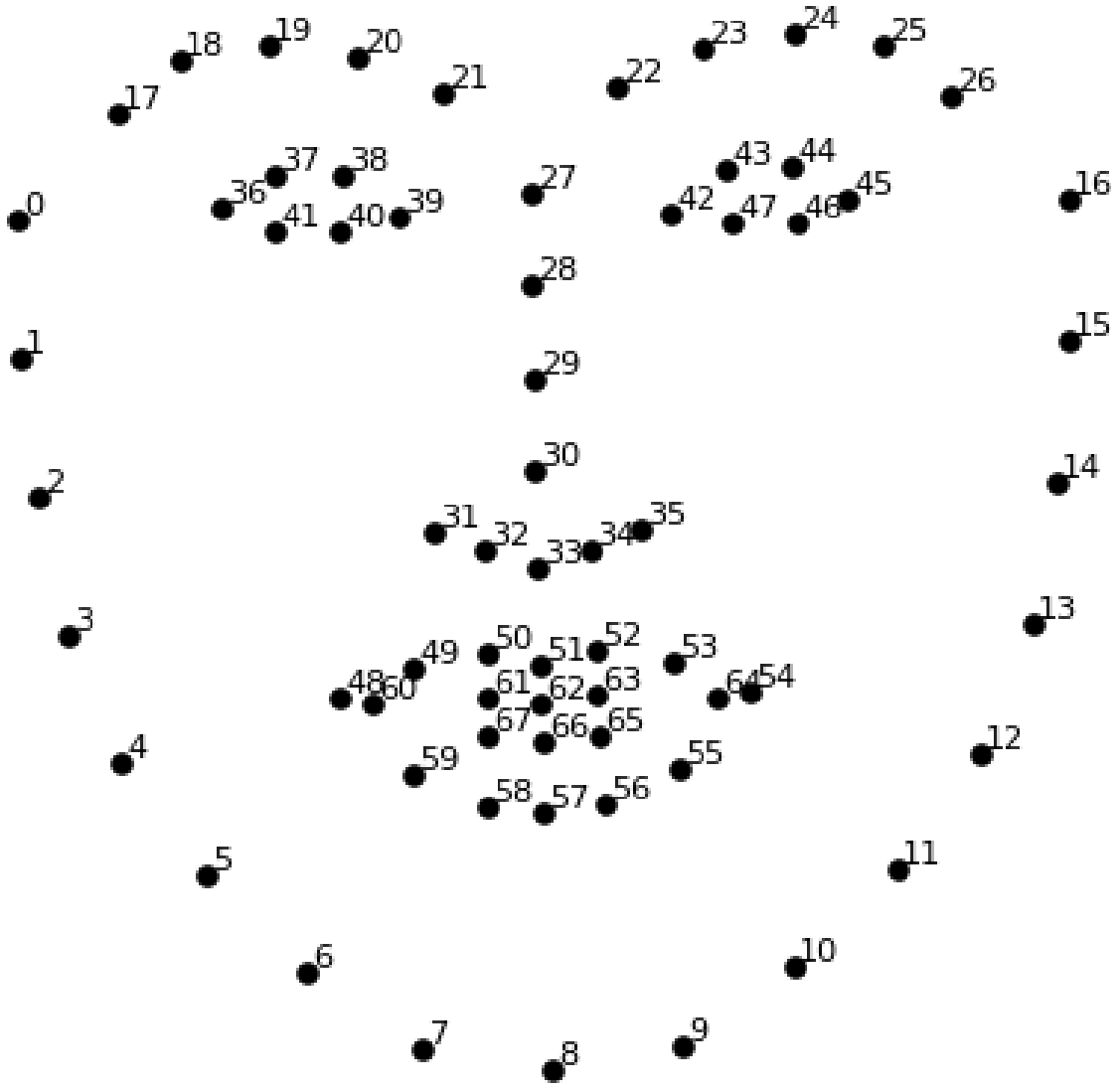
IMA DMT4

DI Dr. Alexander Nischelwitzer

Dlib Face

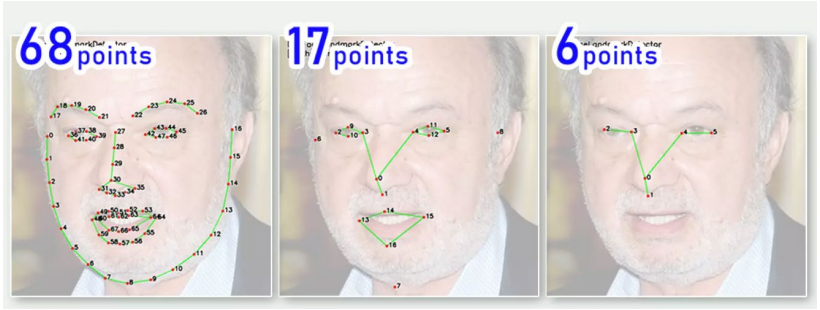

Compare L/R: 0,36
Px: 232 202
NDC: 036% 042%
Mouth O/C: 0,15
Rotate: −10,84

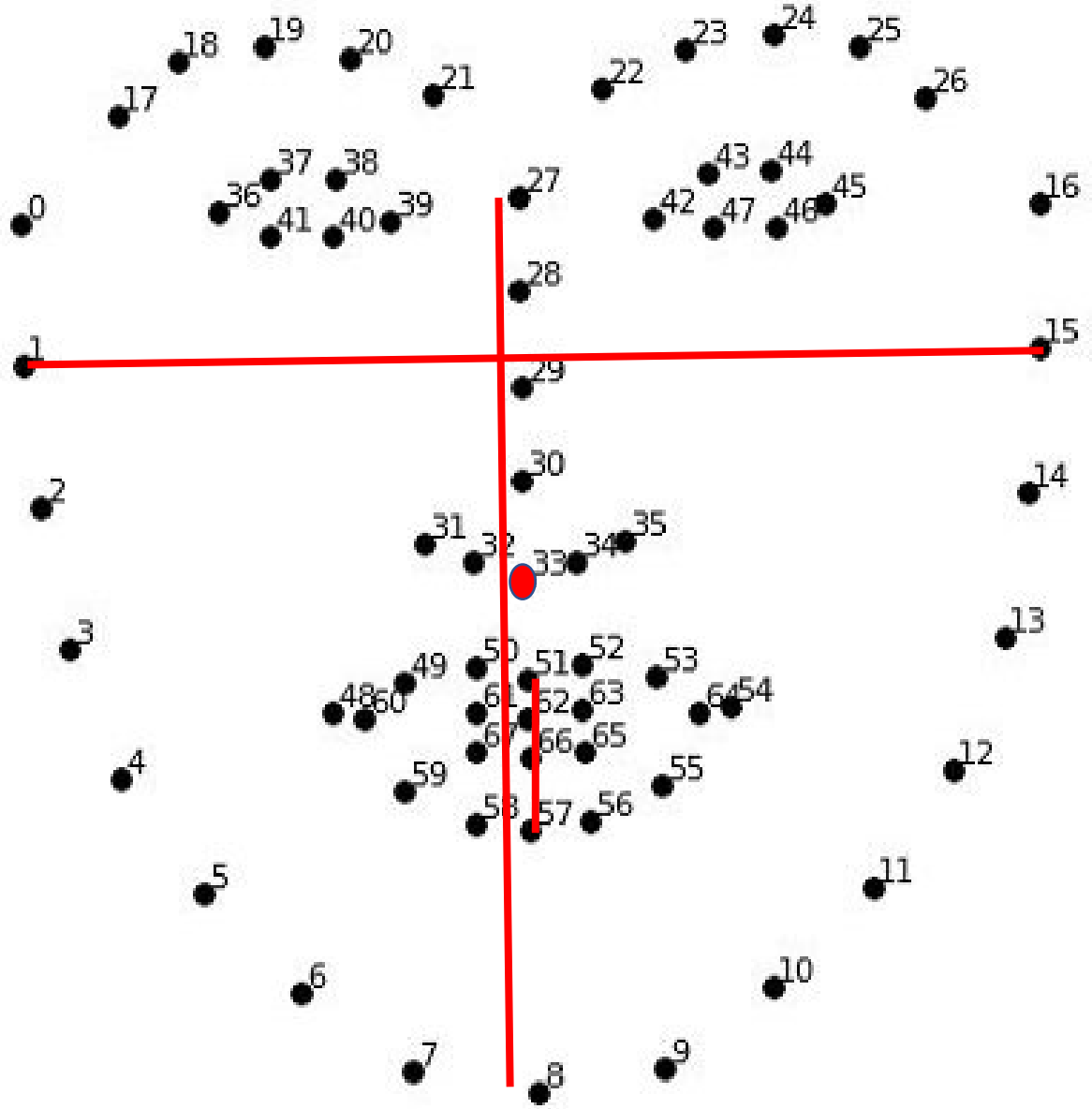

aw 68Points
Compare L/R: 0,78
Px: 428 241
NDC: 067% 050%
Mouth O/C: 0,01
Rotate: 7,22


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

Dlib Face Landmarks

- #68 Points





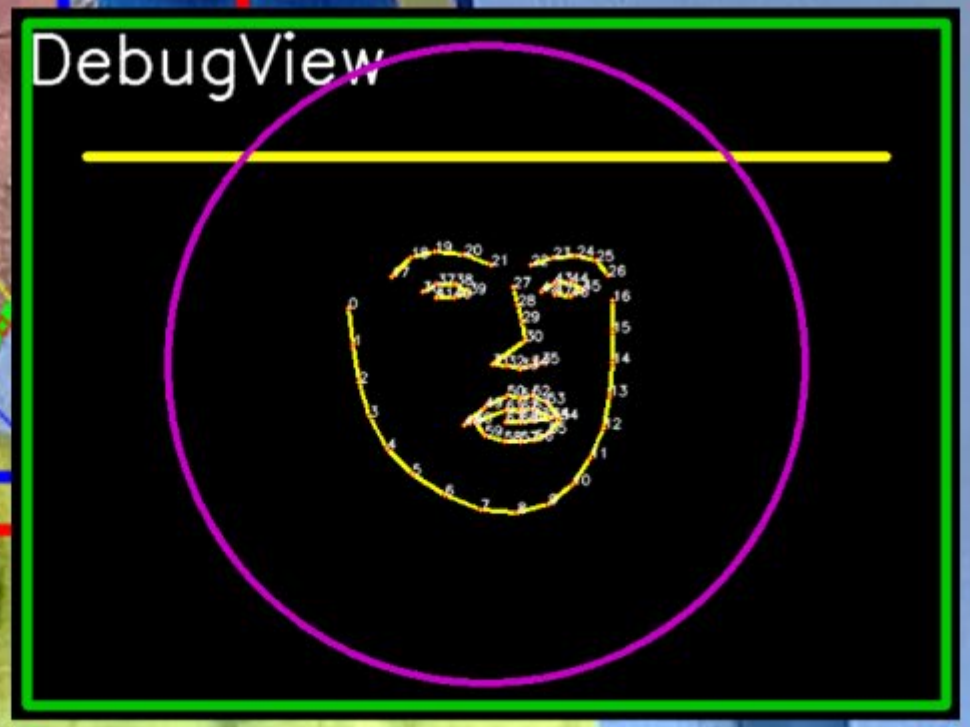
https://assetstore.unity.com/packages/tools/integration/dlib-facelandmark-detector-64314

OpenCV FaceLandmark Draw 68Points

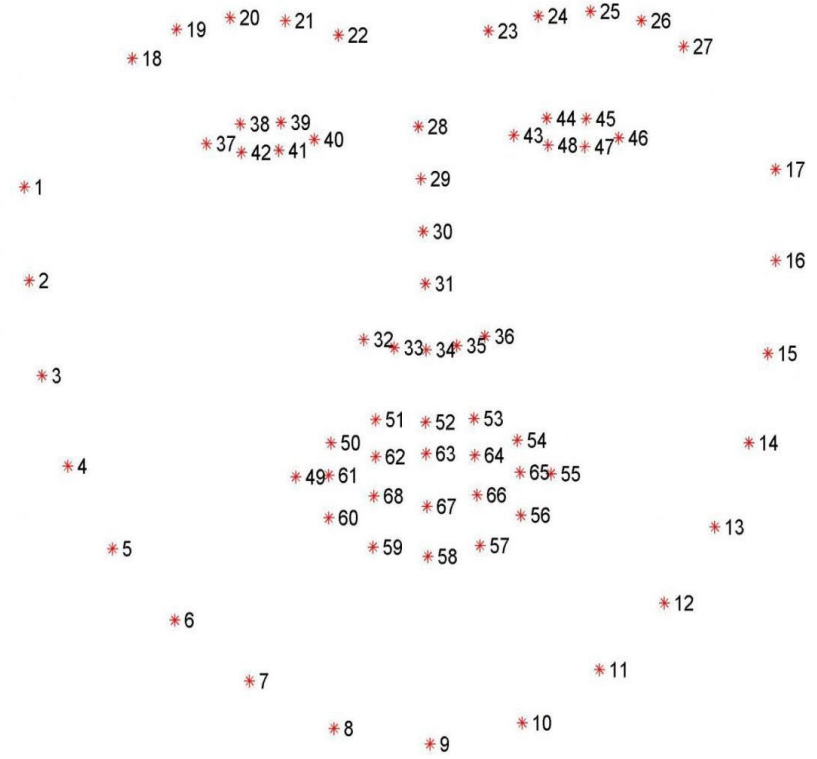
Faces: 1
BB Center: (341, 245)
BB CenterNDC: (0.53, 0.51)
Nose [30]: (345, 224)
NoseNDC [30]: (0.54, 0.47)

Usage:
SHIFT-1: InGame Debug Console
TAB: Toggle InfoScreen (put your own info here)
0,a,b .. Switch Cam 0..Cam0 a..Cam1 b..Cam2

StatusInfoTxt



Landmarks

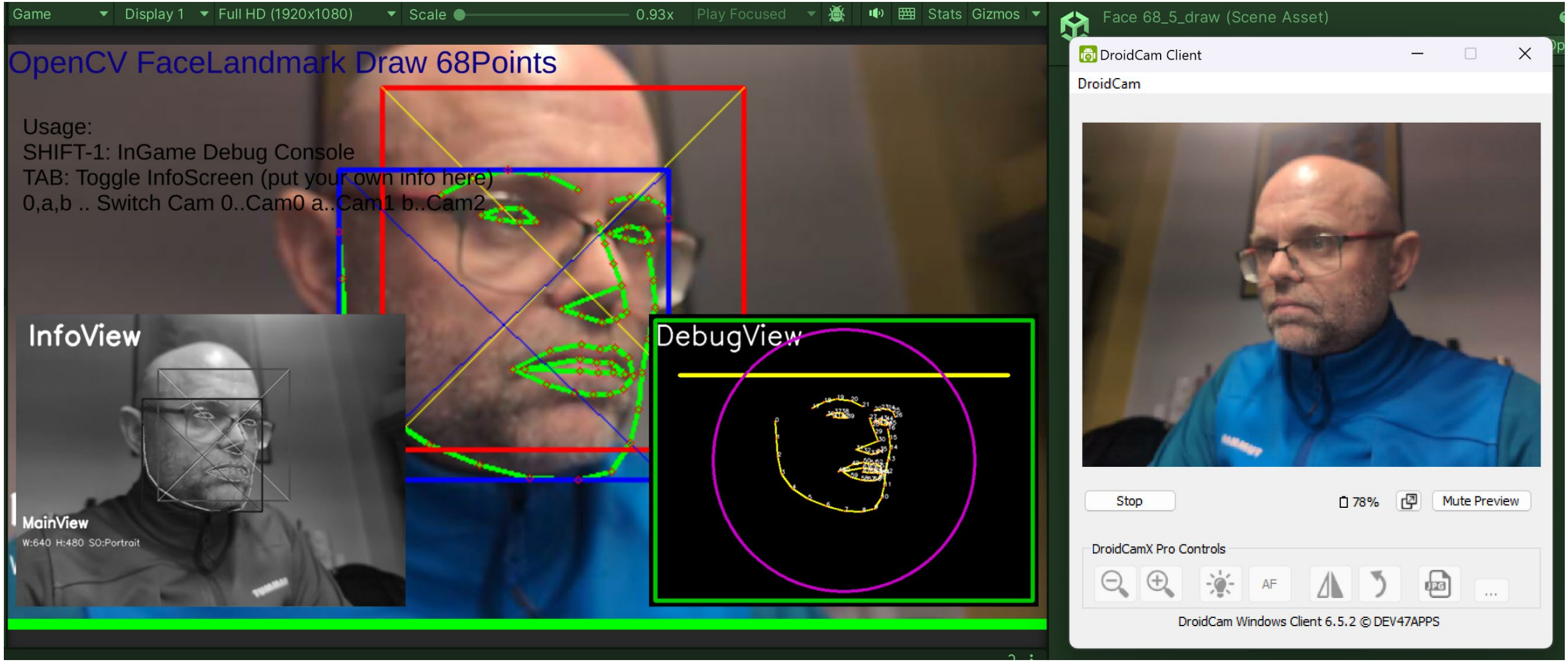


Datenstruktur der "Landmarks":

List<Point> landmarks = new List<Point>()
Point P = new Point(float x, float y)

```
// Baseline ########################################################

for (int i = 0; i < points.Count; i++)
{
    Imgproc.circle(imgMat, points[i], 2, green, -1);
    Imgproc.putText(imgMat, i.ToString(), points[i], Imgproc.FONT_HERSHEY_SIMPLEX, 0.5, white, 1, Imgproc.LINE_AA, false);
}
```

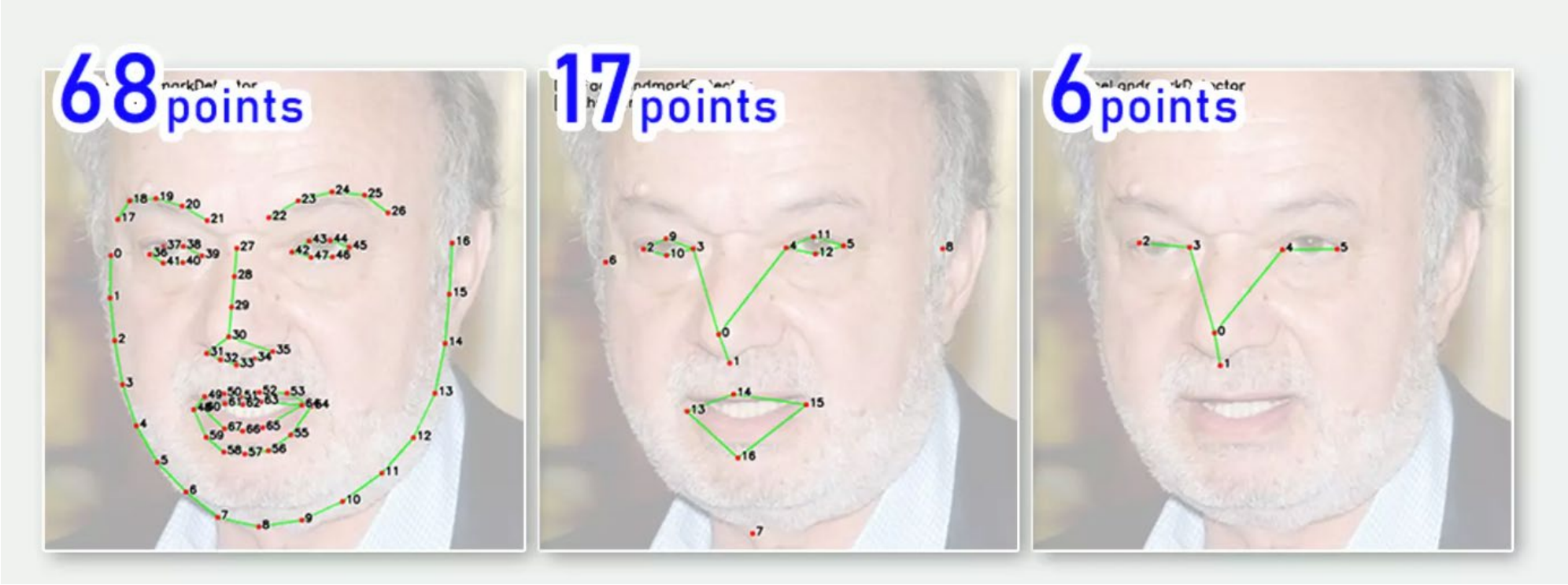
DroidCam Client



https://www.dev47apps.com/droidcam/windows/

OpenCV **Dlib FaceLandmark Detector**

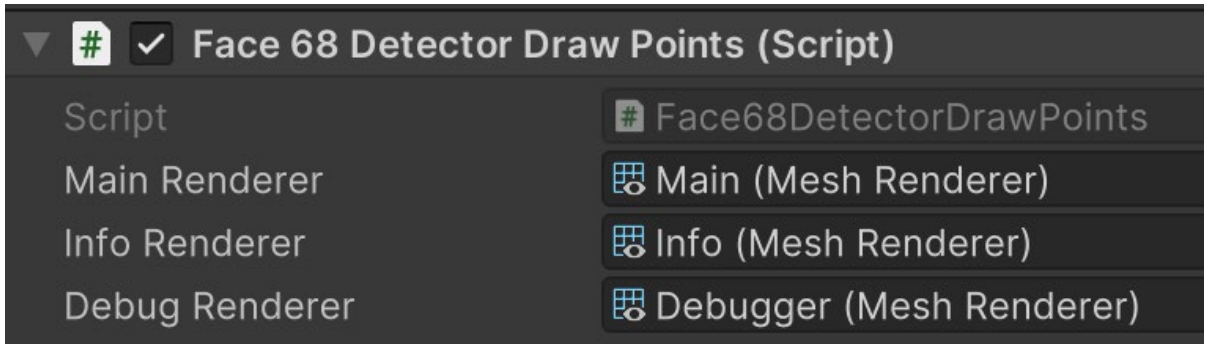
- We are using 68 face land marks

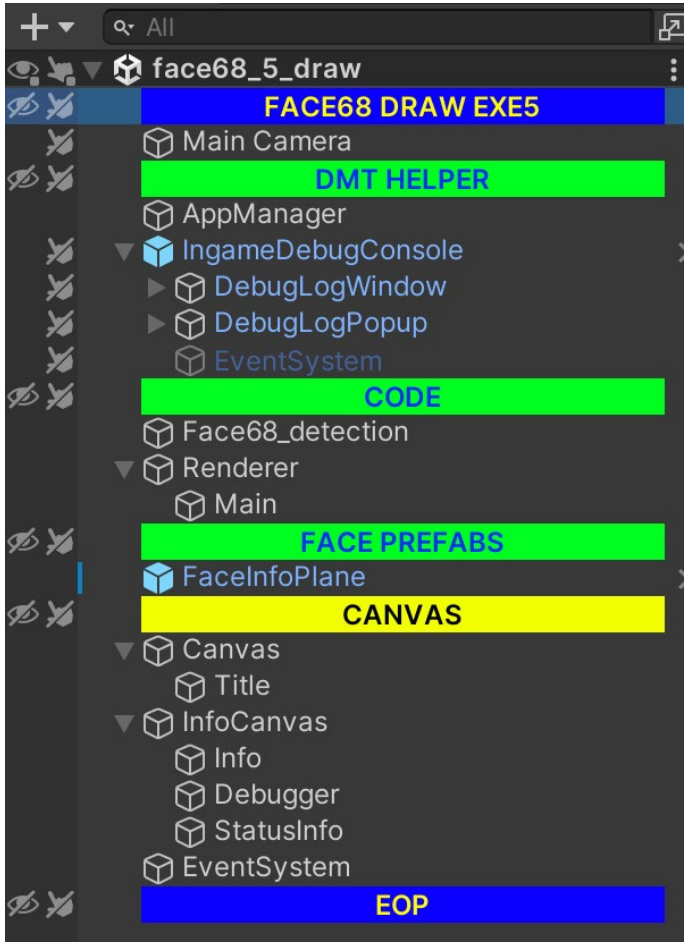


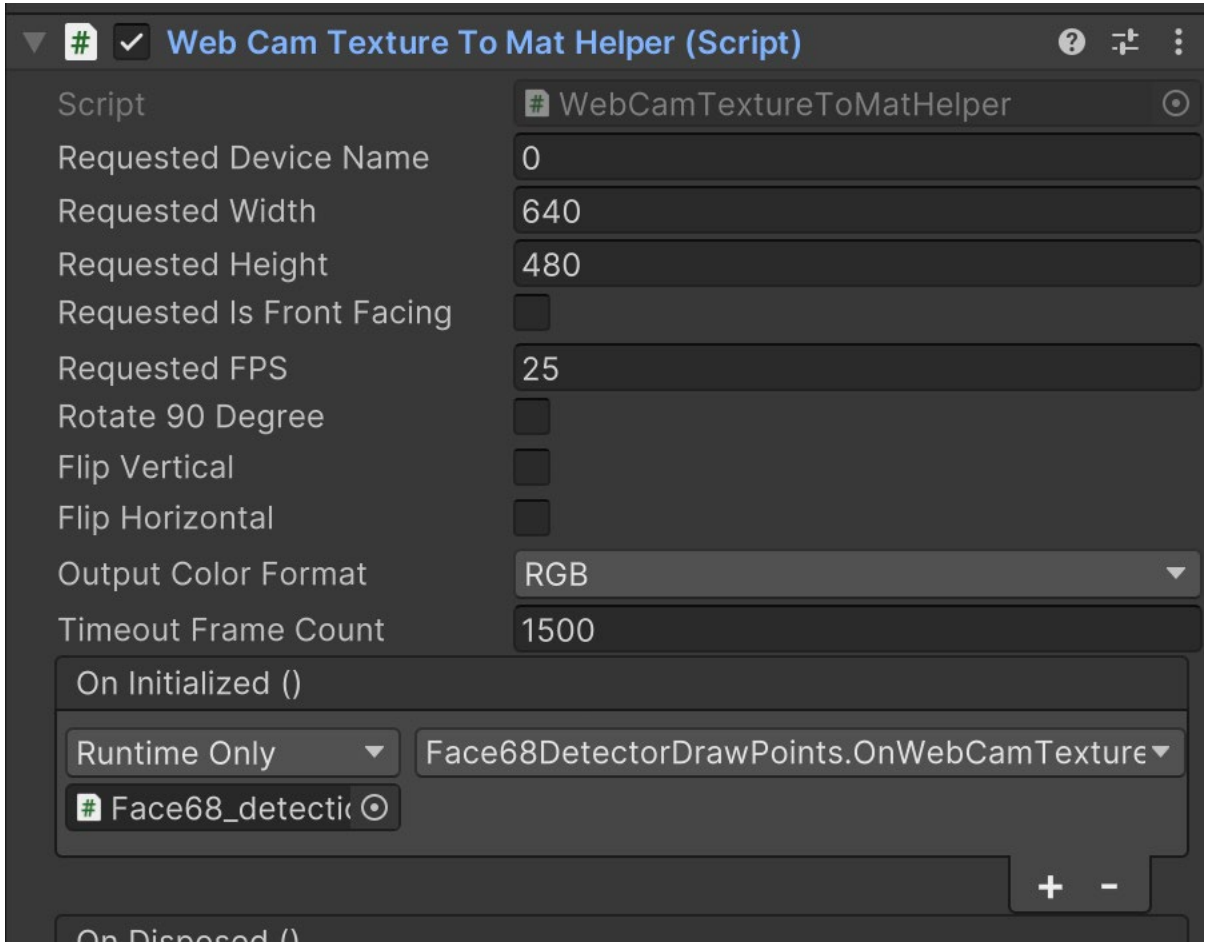
Tex 2 Mat

Face68Detector

- WebCam
- Tex2Mat



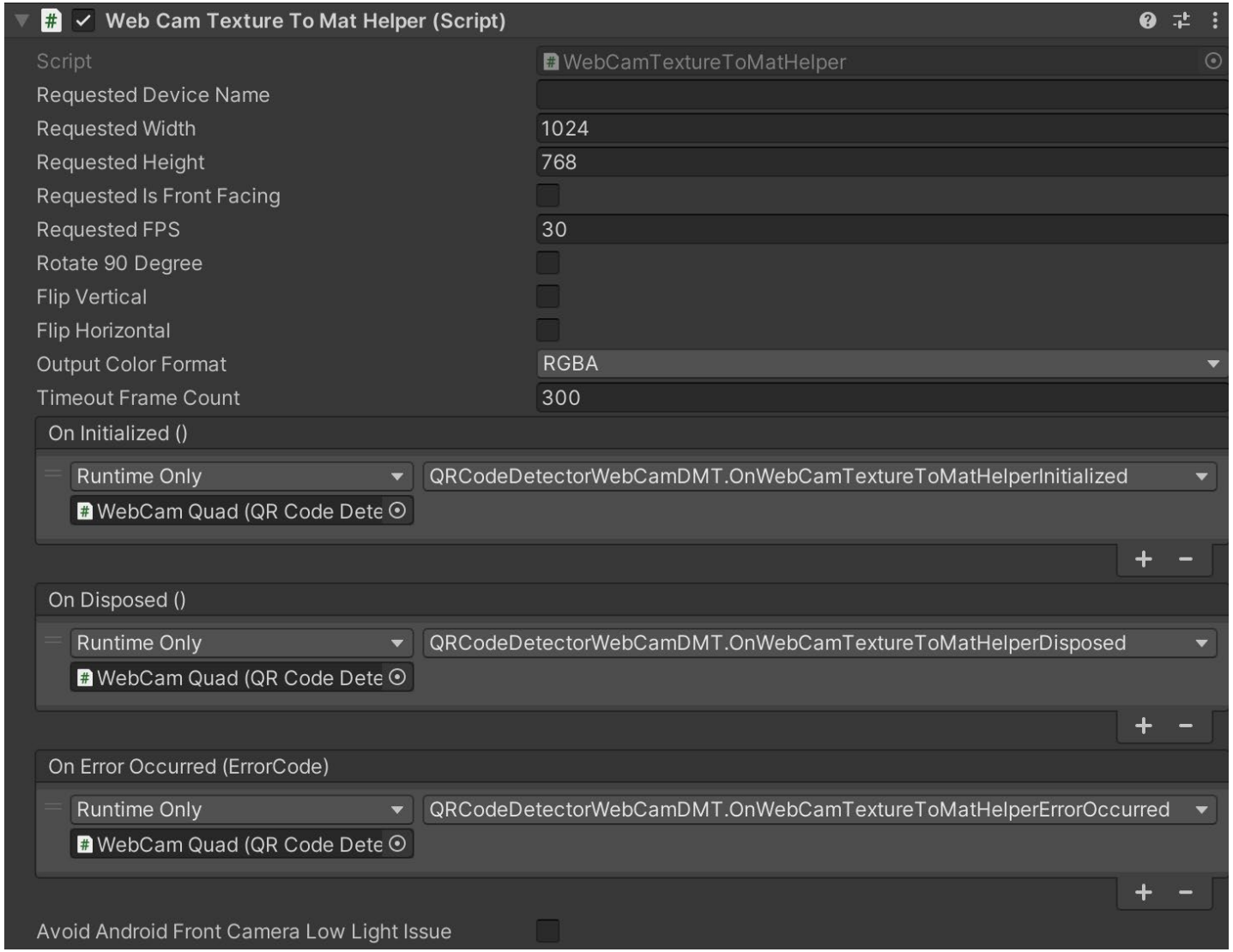




WebCamera

- Auflösung
- Color RGBA RBG

- Matrix **Mat**
- Texture Texture2D

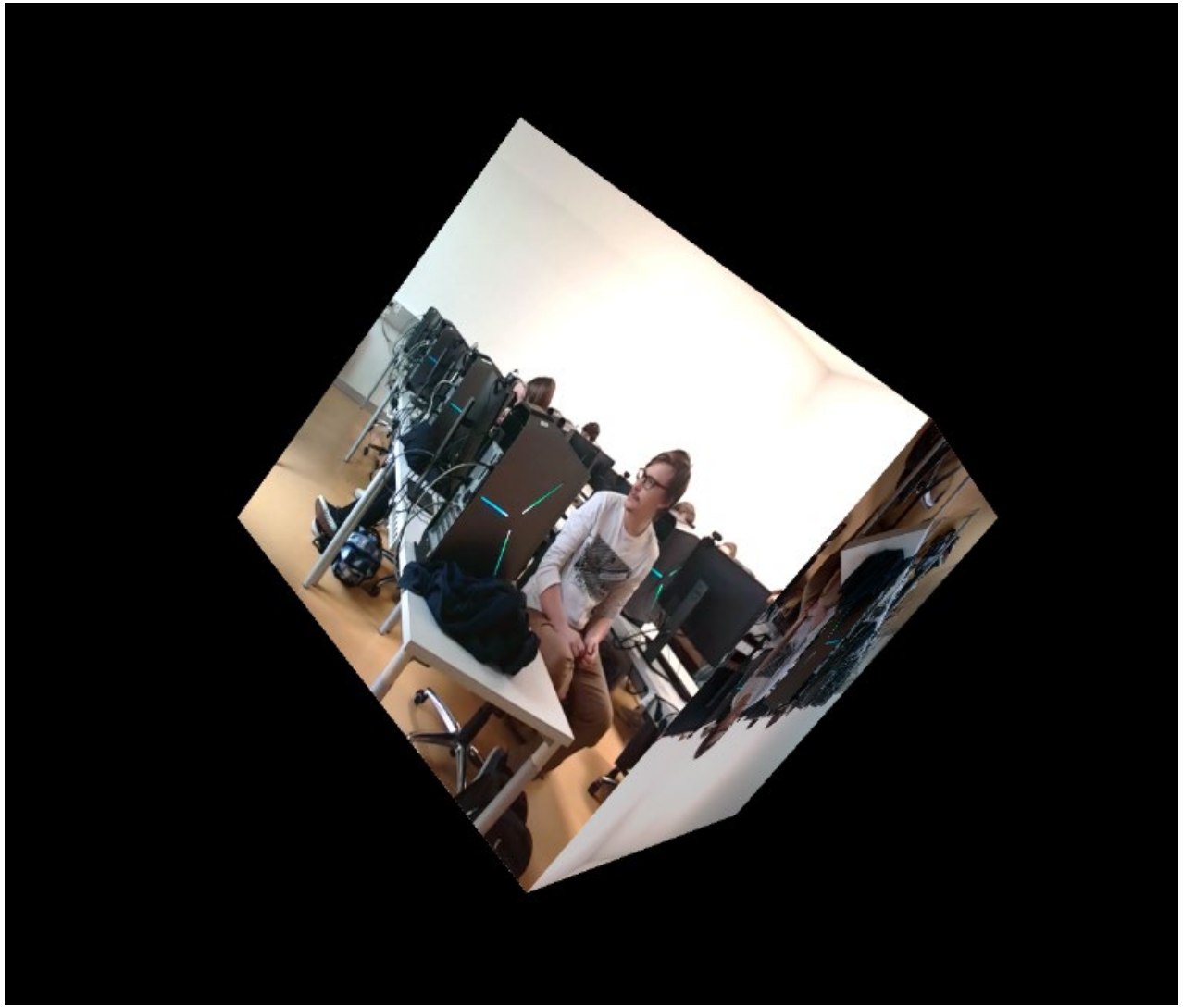



```
private Mat webCamInfo;
webCamInfo = webCamTextureToMatHelper.GetMat();
```

Assets\OpenCVForUnity\org\opencv\unity\helper**WebCamTextureToMatHelper**.cs

Übung

- Cube mit Webcam
- Rotation



Übung 2

- Quad im Hintergrund
- Gleiche Textur

```
private Texture2D texture;
```

Lösung: Übung 2

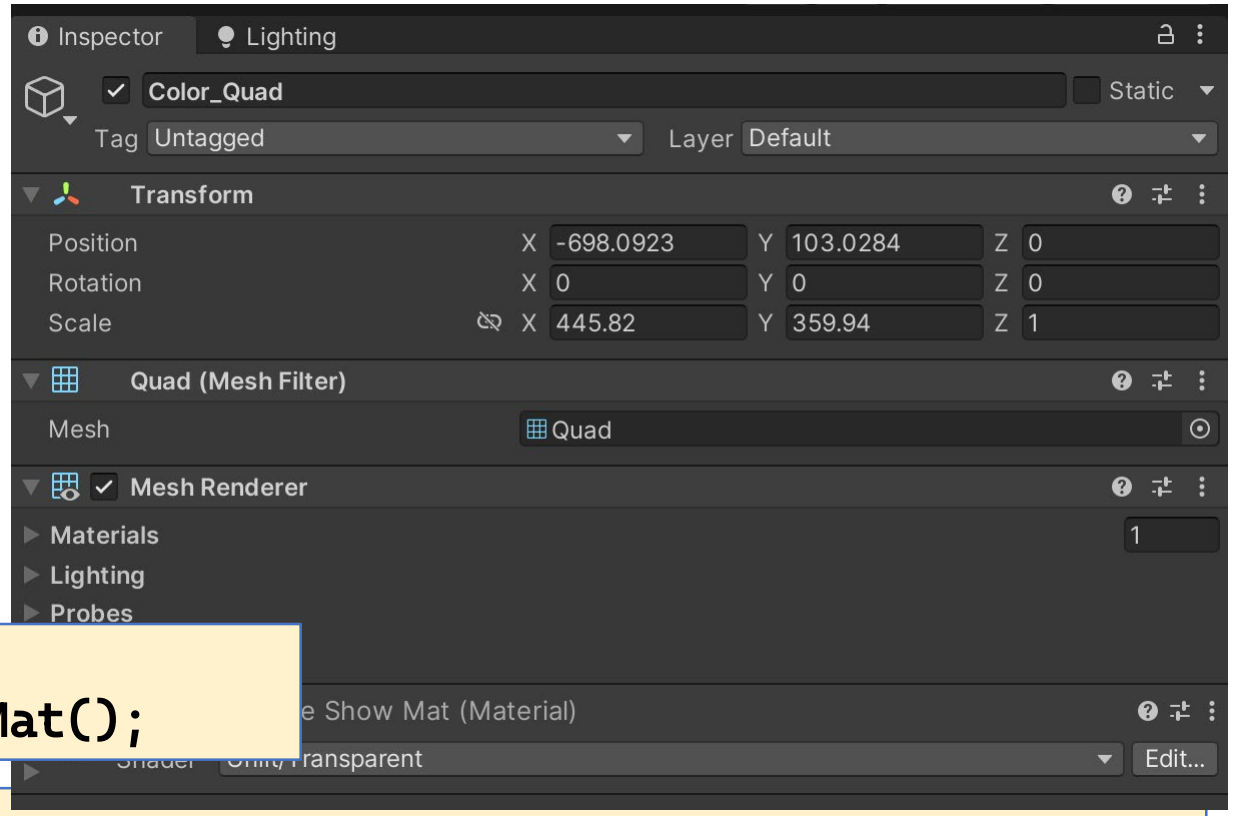
- Render vom GameObject
- Textur im Init zuweisen


```
public Renderer myBackgroundRender;
```

```
CamInit
myBackgroundRender.material.mainTexture = texture;
```

WebCamera

- Quad or Cube
  mit MeshRenderer
- Mat → Texture2D → Renderer




```
private Mat webCamInfo;
webCamInfo = webCamTextureToMatHelper.GetMat();
```

```
private Texture2D textureColor;
public  Renderer  colorShow;

textureColor = new Texture2D(webCamInfo.cols(), webCamInfo.rows(), TextureFormat.RGB24, false);
colorShow.material.mainTexture = textureColor;
```

```
Mat colorMat = webCamTextureToMatHelper.GetMat();
Utils.matToTexture2D(colorMat, textureColor);
```

OpenCV API

- http://enoxsoftware.github.io/OpenCVForUnity/3.0.0/
- http://docs.opencv.org/4.7.0/index.html
- OpenCV Java Clone
- Core
  https://docs.opencv.org/4.7.0/de/d7a/tutorial_table_of_content_core.html
- Image Processing
  https://docs.opencv.org/4.7.0/d7/da8/tutorial_table_of_content_imgproc.html

http://enoxsoftware.github.io/OpenCVForUnity/3.0.0/doc/html/namespace_open_c_v_for_unity_1_1_unity_utils_1_1_helper.html

matToTexture2D

◆ matToTexture2D() [1/2]

```
static void OpenCVForUnity.UnityUtils.Utils.matToTexture2D ( Mat         mat,
                                                             Texture2D   texture2D,
                                                             bool        flip = true,
                                                             int         flipCode = 0,
                                                             bool        flipAfter = false,
                                                             bool        updateMipmaps = false,
                                                             bool        makeNoLongerReadable = false
                                                           )
```

- Documentation
- Converts OpenCV Mat to Unity Texture2D

```
Utils.matToTexture2D(colorMat, textureColor);
```

```
using OpenCVForUnity.CoreModule;
using OpenCVForUnity.ImgprocModule;
using OpenCVForUnity.UnityUtils;
using OpenCVForUnity.UnityUtils.Helper;
```

- Mat Types:
  - CV_8UC1 (GRAY) 1 Byte
  - CV_8UC3 (RGB) 3 Byte
  - CV_8UC4 (RGBA) 4 Byte

http://enoxsoftware.github.io/OpenCVForUnity/3.0.0/doc/html/class_open_c_v_for_unity_1_1_unity_utils_1_1_utils.html

Detector

faceLandmarkDetector.Detect();

- Face68DetectorDrawPoints

```
[RequireComponent(typeof(WebCamTextureToMatHelper))]
public class Face68DetectorDrawPoints : MonoBehaviour
```

- Zeilen ab 210

```
211             //#####################################################################
212             //DETECTION: detect face rects
213             List<UnityEngine.Rect> detectResult = faceLandmarkDetector.Detect();
214
215             //#####################################################################
```

```
List<UnityEngine.Rect> detectResult = faceLandmarkDetector.Detect();
```

Drawing → Matrix

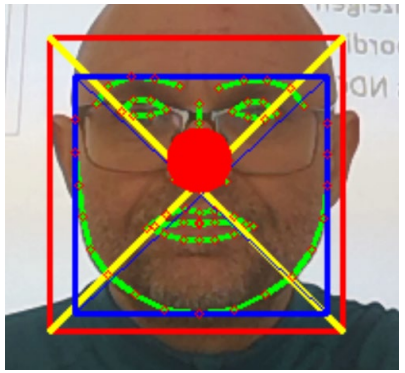
- Direkt ins Bild (Matrix) → Anzeige über Texture

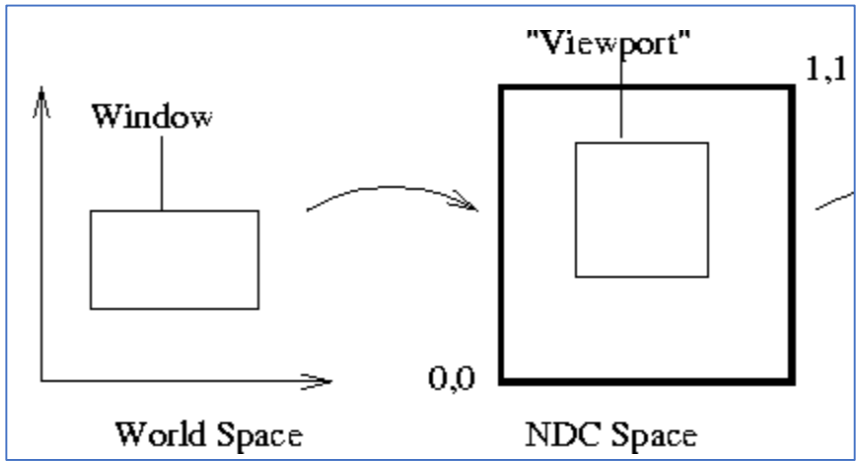
```
Imgproc.circle(rgbMat, new Point(imgWidth/2, imgHeight/2), 15, new
Scalar(200, 0, 200), 4);

Imgproc.line(matFaceNumber, new Point(50, 100), new Point(imgWidth - 50,
100), new Scalar(255, 255, 0), 5);

Imgproc.rectangle(matFaceNumber, new Point(10, 10), new Point(imgWidth - 10,
imgHeight - 10), new Scalar(0, 200, 0), 5);

Imgproc.putText(matFaceNumber, "DebugView", new Point(10, 50),
Imgproc.FONT_HERSHEY_SIMPLEX, 1.5, new Scalar(255, 255, 255), 2,
Imgproc.LINE_AA, false);
```

```
OpenCVForUnity.UnityUtils.Utils.matToTexture2D(matFaceNumber, debugTexture);
```

Point

Übung: Big red Nose



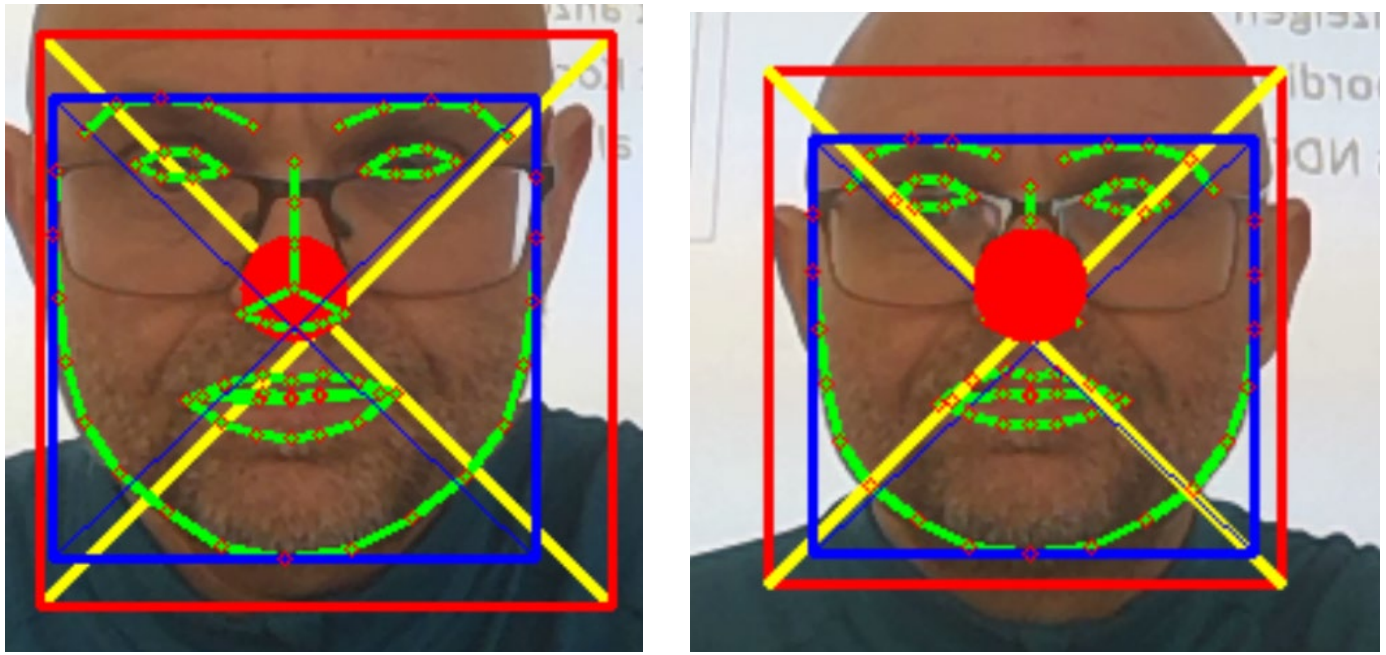


Face68DetectorDrawPoints

1. Nasenpunkt anzeigen
2. Nasenpunkt Koordinaten einzeichnen
3. Nasenpunkt als NDC einzeichnen

```
Point nosePoint = new Point(points[30].x, points[30].y);
Imgproc.circle(rgbMat, nosePoint, 12, new Scalar(255, 0, 0), 8);
```

RED NOSE: „Reihenfolge“ Mat



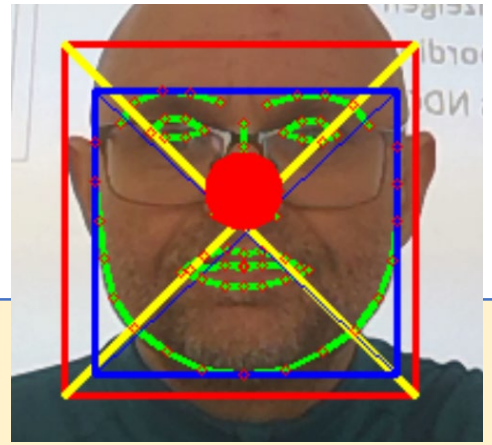
```
Point nosePoint = new Point(points[30].x, points[30].y);
Imgproc.circle(rgbMat, nosePoint, 15, new Scalar(255, 0, 0), 5);
```

void Imgproc.circle(Mat img, Point center, int radius, Scalar color, int thickness) (+ 3 Überladungen)

Übung: Big red Nose

Face68DetectorDrawPoints

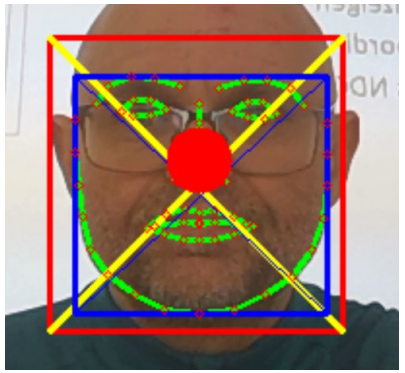
1. Nasenpunkt anzeigen
   relative Veränderung



```
Point faceBBMin = new Point(999, 999);
Point faceBBMax = new Point(0, 0);

for (run = 0; run < points.Count; run++) // check all Points
{
    if (points[run].x > faceBBMax.x) faceBBMax.x = points[run].x;
    if (points[run].y > faceBBMax.y) faceBBMax.y = points[run].y;
    if (points[run].x < faceBBMin.x) faceBBMin.x = points[run].x;
    if (points[run].y < faceBBMin.y) faceBBMin.y = points[run].y;
}
```

```
(faceBBMax.x - faceBBMin.x)
```

RELATIV: Big red Nose



RED NOSE DAY

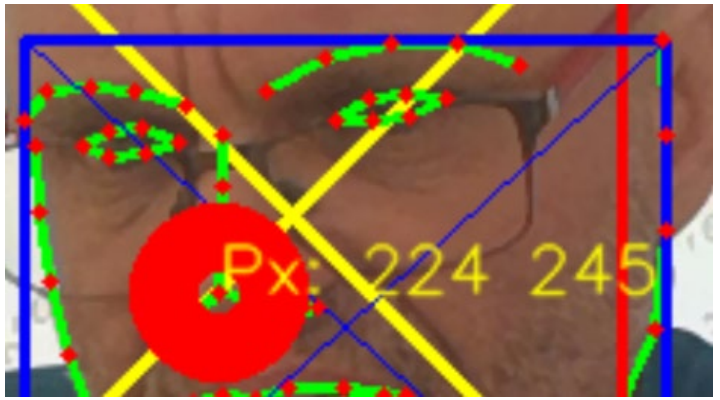
- Relative Anzeige der „Nasengröße“
- 10% der Breite der „blauen BoundingBox“

```
Point nosePoint = new Point(points[30].x, points[30].y);

int thickLine = Convert.ToInt32((faceBBMax.x - faceBBMin.x) * 0.1);


Imgproc.circle(rgbMat, nosePoint, 15, new Scalar(255, 0, 0), thickLine);
```

Übung: Big red Nose

`Face68DetectorDrawPoints`

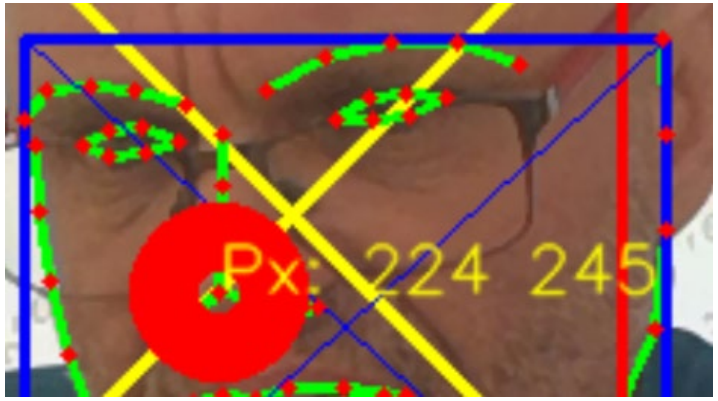
1. Nasenpunkt anzeigen
2. Nasenpunkt Koordinaten einzeichnen




```
void Imgproc.putText(Mat img, string text, Point org, int fontFace, double fontScale, Scalar color, int thickness, int lineType, bool bottomLeftOrigin) (+ 3 Überladungen)
```

```
void Imgproc.circle(Mat img, Point center, int radius, Scalar color, int thickness) (+ 3 Überladungen)
```

TXT: Big red Nose



Koordination als Text

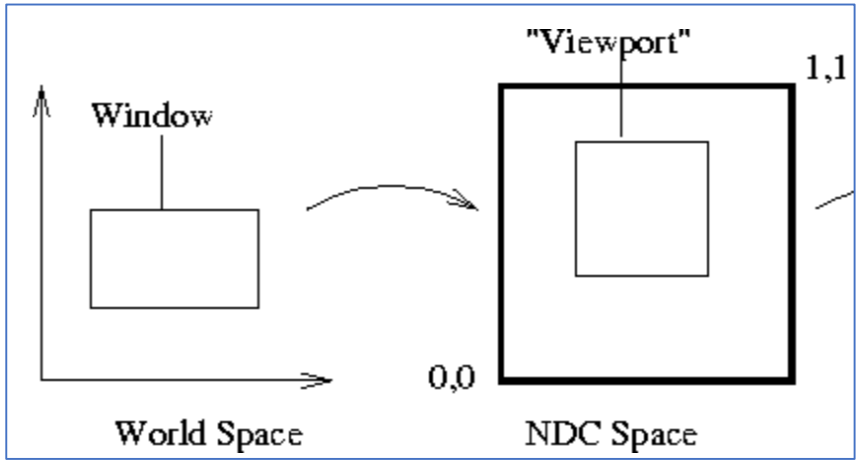
```
Point nosePoint = new Point(points[30].x, points[30].y);
Imgproc.circle(rgbMat, nosePoint, 15, new Scalar(255, 0, 0), 10);

Imgproc.putText(rgbMat, "Px: "+ points[30].x+" "+points[30].y, nosePoint,
    Imgproc.FONT_HERSHEY_SIMPLEX, 0.6, new Scalar(255, 255, 0), 1,
    Imgproc.LINE_AA, false);
```

void Imgproc.putText(Mat img, string text, Point org, int fontFace, double fontScale, Scalar color, int thickness, int lineType, bool bottomLeftOrigin) (+ 3 Überladungen)

Übung: Big red Nose

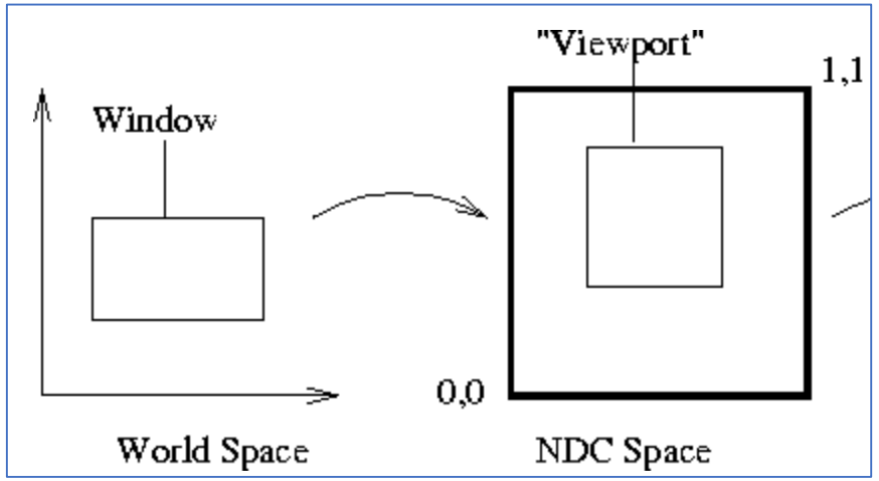
`Face68DetectorDrawPoints`

1. Nasenpunkt anzeigen
2. Nasenpunkt Koordinaten einzeichnen
3. Nasenpunkt als **NDC** einzeichnen




```
void Imgproc.circle(Mat img, Point center, int radius, Scalar color, int thickness) (+ 3 Überladungen)
```

NDC: Big red Nose




```
float noseX = points[30].x / rgbMat.width();
float noseY = points[30].y / rgbMat.height();

String noseNDC = "NDC: " + (noseX * 100).ToString("000") + "% "
                         + (noseY * 100).ToString("000") + "% ";

Imgproc.putText(rgbMat, noseNDC, new Point(nosePoint.x, nosePoint.y + 30),
                Imgproc.FONT_HERSHEY_SIMPLEX, 0.6,
                new Scalar(255, 255, 0), 1, Imgproc.LINE_AA, false);
```

Rect & Rectangle

- OpenCV

```
Imgproc.rectangle(rgbMat, faceBBMin, faceBBMax, new Scalar(0, 0, 255), 2);
```

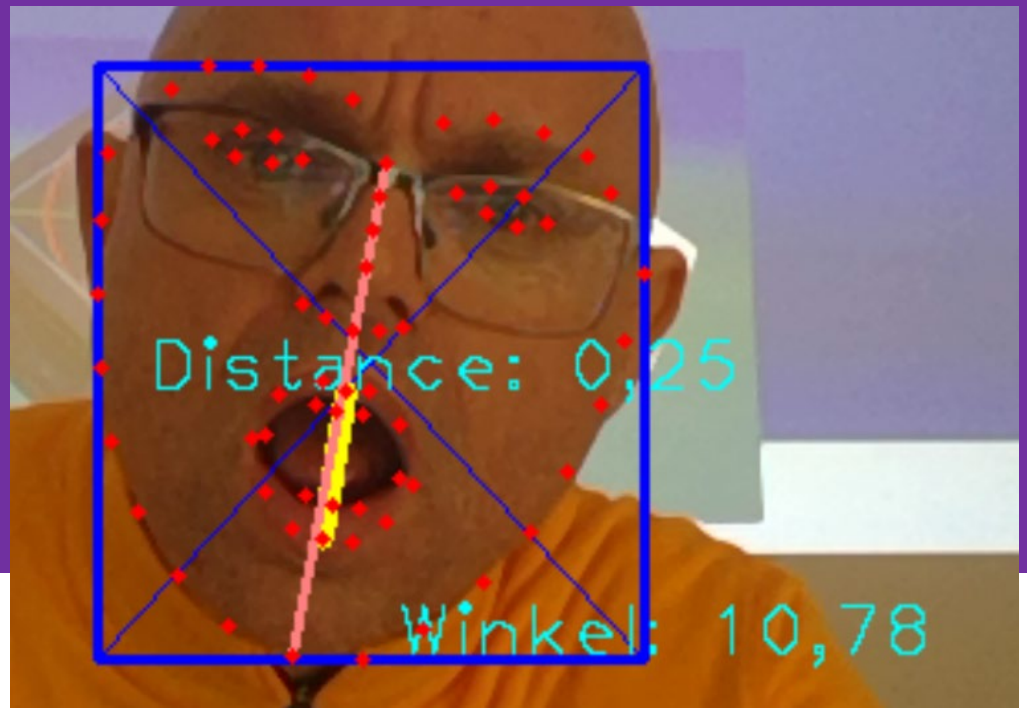
void Imgproc.rectangle(Mat img, Point pt1, Point pt2, Scalar color, int thickness) (+ 7 Überladungen)

- Datentype (Unity, C#)

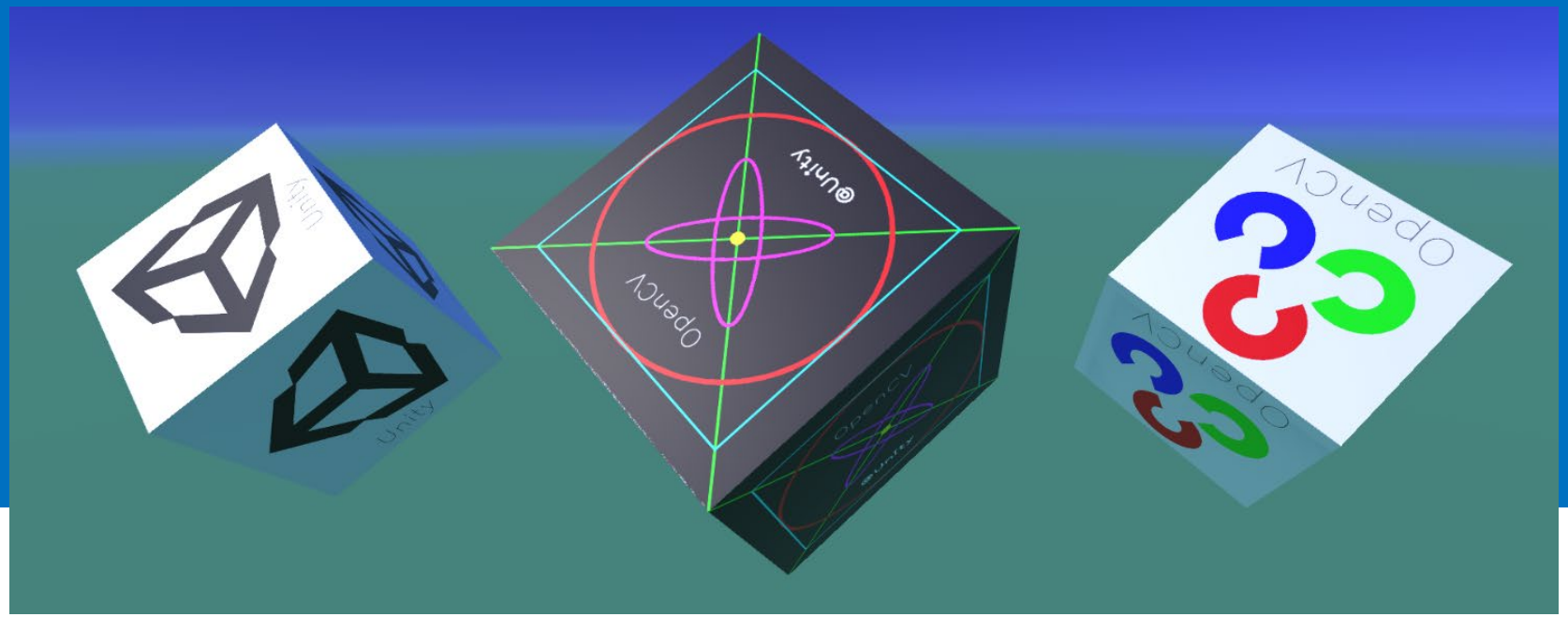
```
new UnityEngine.Rect((float)faceBBMin.x, (float)faceBBMin.y, (
```

UnityEngine.Rect.Rect(float x, float y, float width, float height) (+ 3 Überladungen)
Creates a new rectangle.

Lab-2

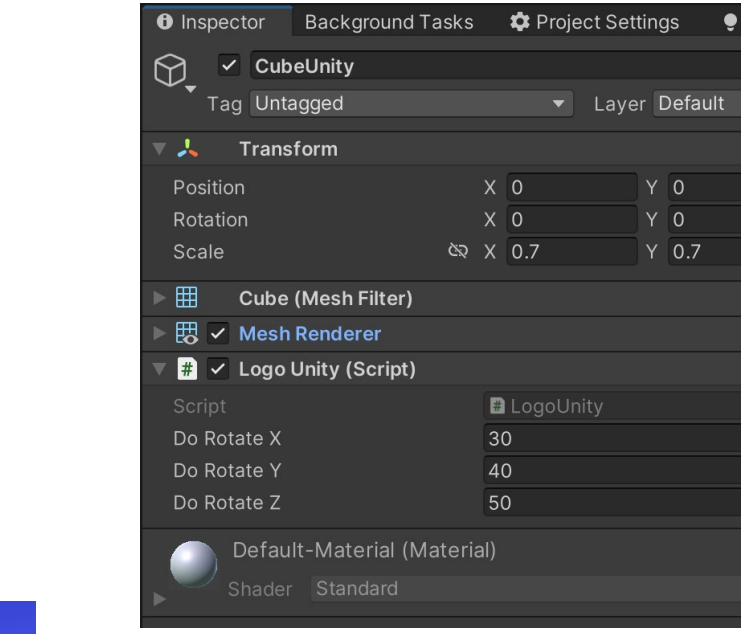


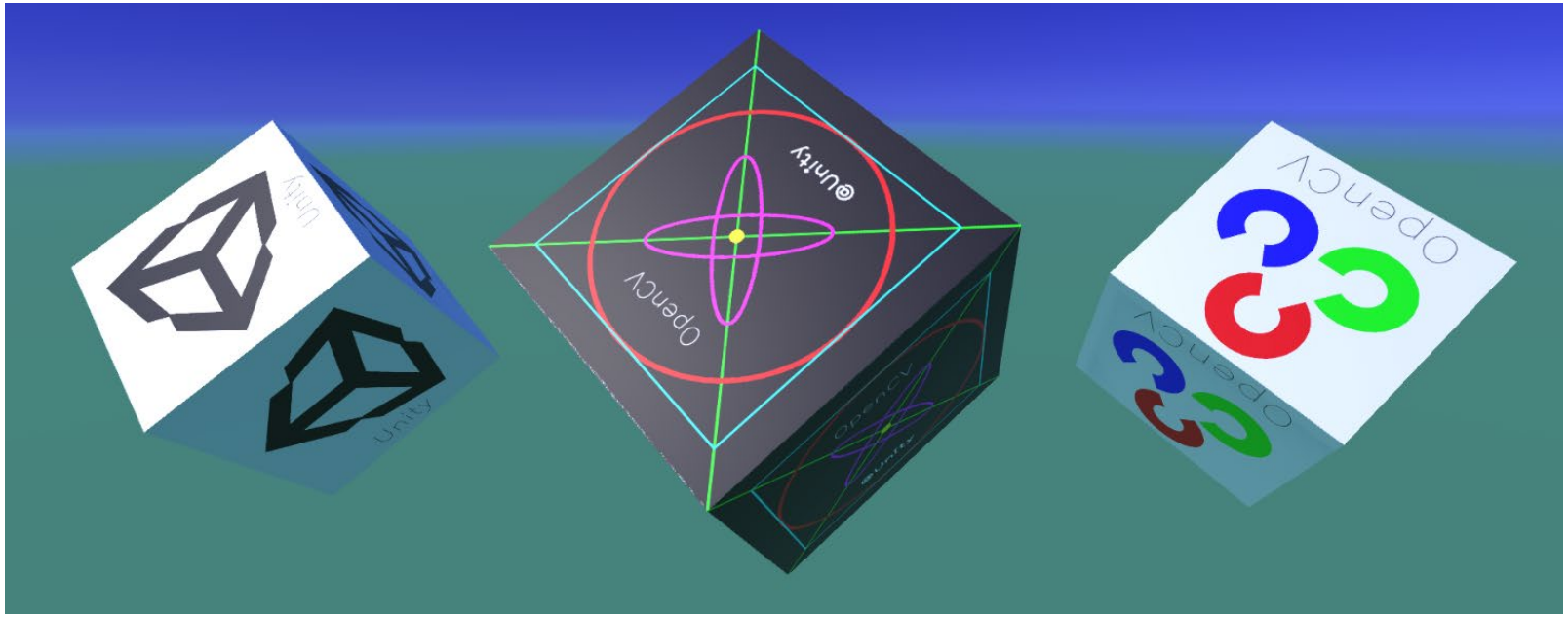
more Drawing



More Drawing in OpenCV

- Hr. Zeidler
  - Imgproc.ellipse
  - List<Point> MatOfPoint List<MatOfPoint>



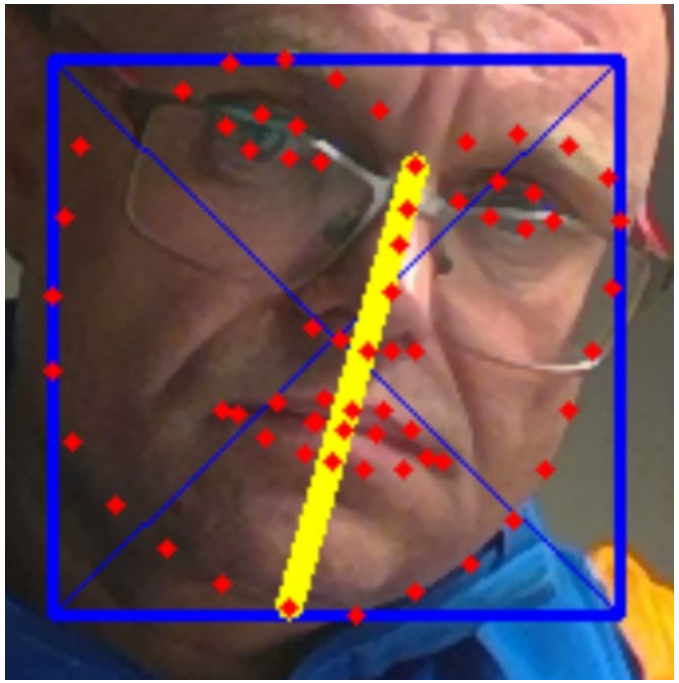


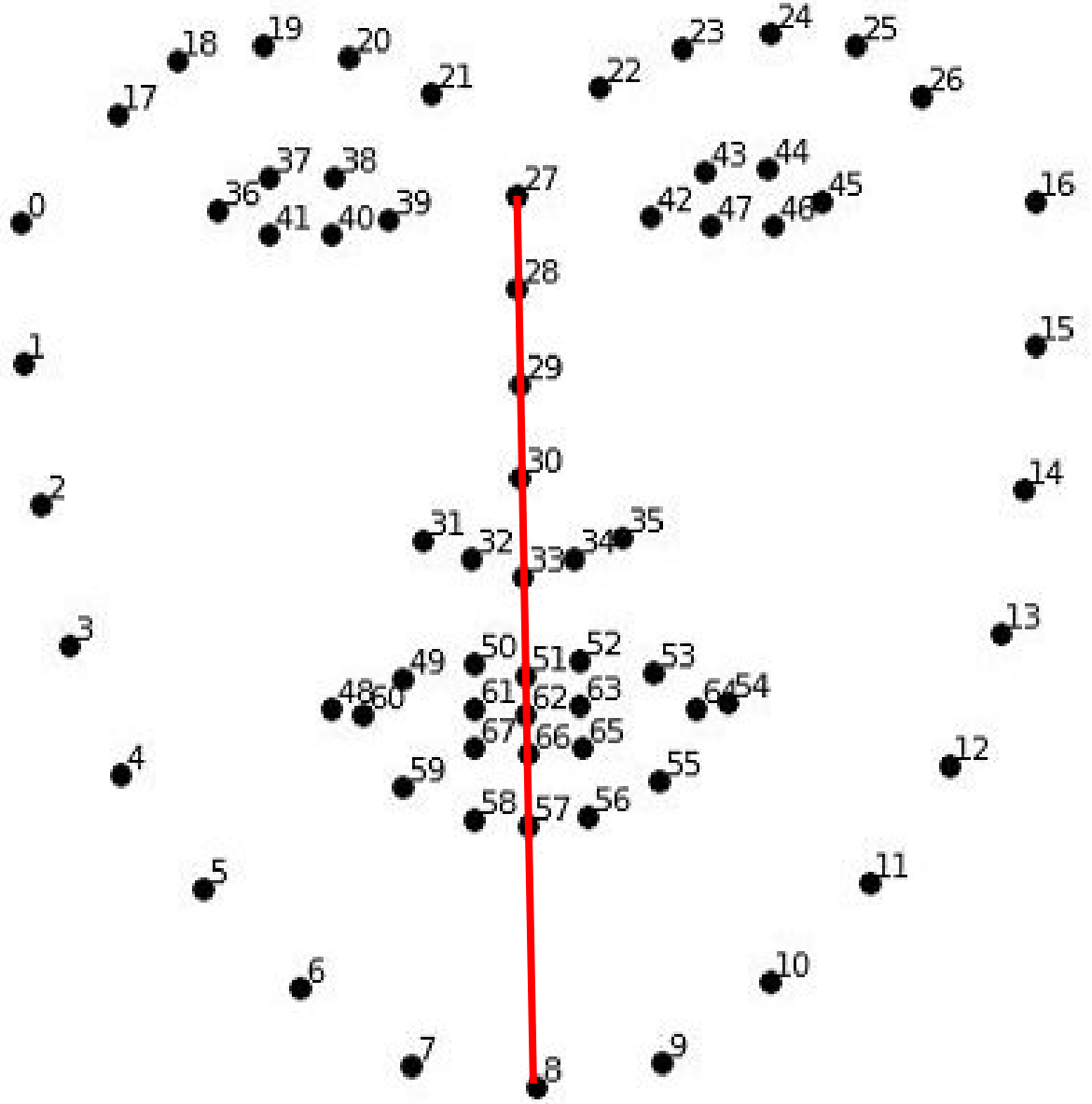
\\abacus\imaXX\skripten\4.Semester\DMT4_FaceCtrl\3_scripts\logo_drawing

Neigung

Winkel

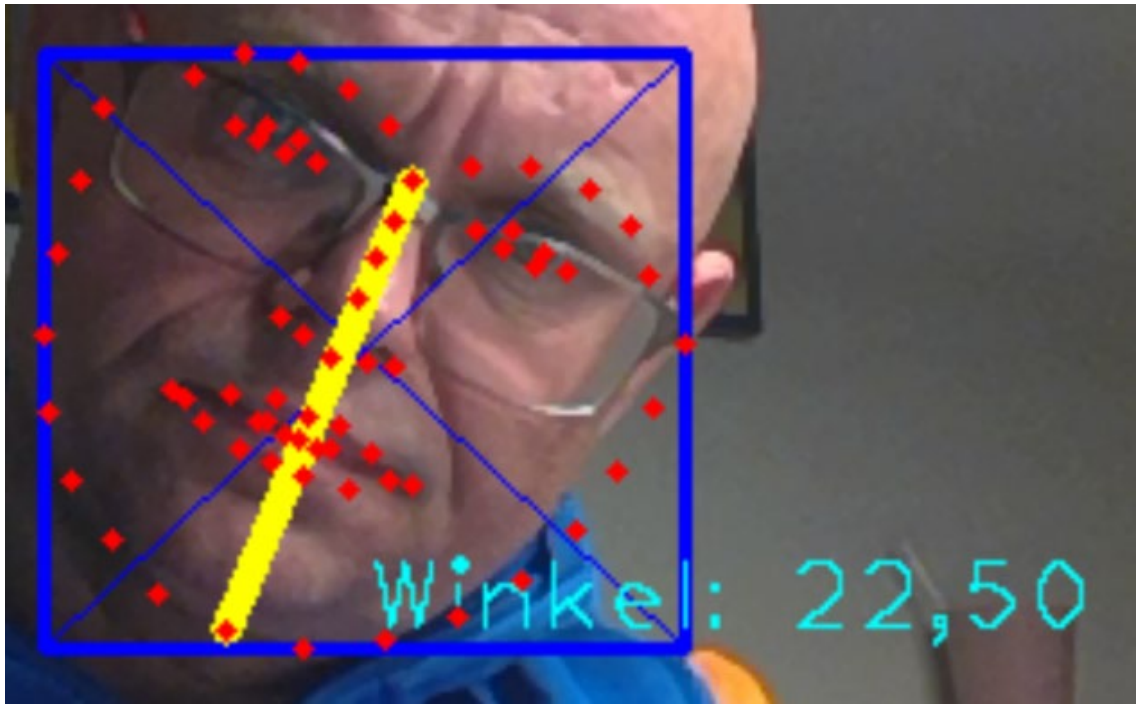
- Punkte 8 und 27





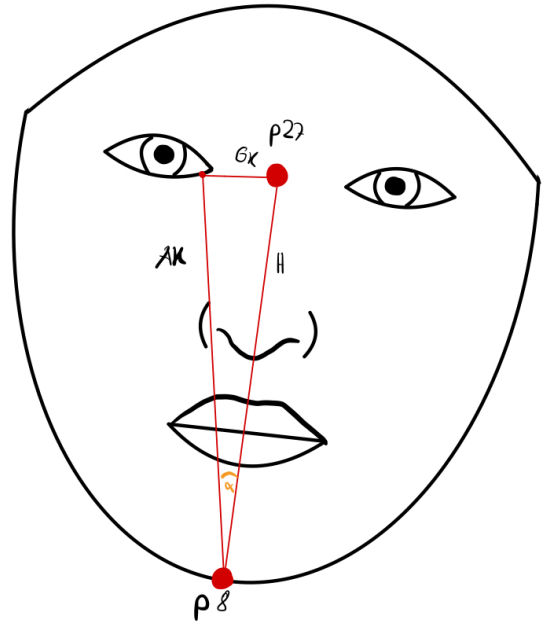
Ziel

- Berechnung der Kopfneigung in Grad
- Einzeichnen der Berechnungslinie
- Textausgabe des Winkels
- Speicherung im „Static Store“



ARCTAN2

Tipps: Kopfneigung



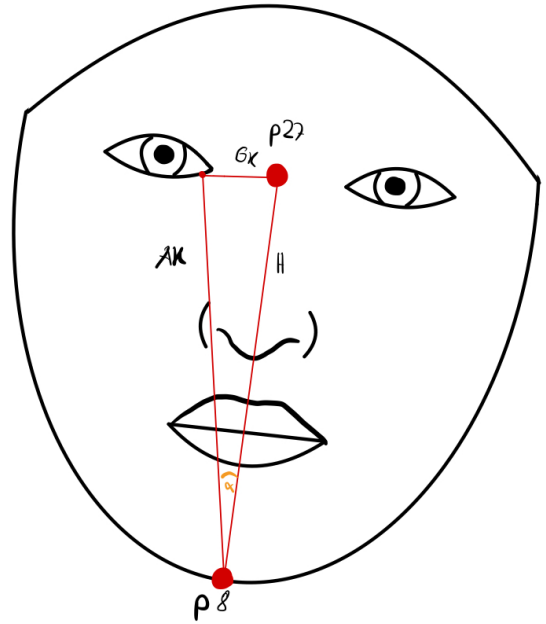
$$\tan\alpha = \frac{GK}{AK}$$

$$\alpha = \arctan\left(\frac{GK}{AK}\right)$$

$$\alpha = \arctan\left(\frac{x_2 - x_1}{y_2 - y_1}\right)$$

```
Point centerPoint = new Point(points[8].x, points[8].y);
Point winkelPoint = new Point(points[27].x, points[27].y);

// ToDo: Linie Zeichnen

double faceAngle = Math.Atan2(winkelPoint.y - centerPoint.y,
    winkelPoint.x - centerPoint.x) * 180.0f / Math.PI + 90.0f;

String angleTxt = "Winkel: " + faceAngle.ToString("0.00");

// ToDo: Text ausgeben
```

Lösung: Kopfneigung



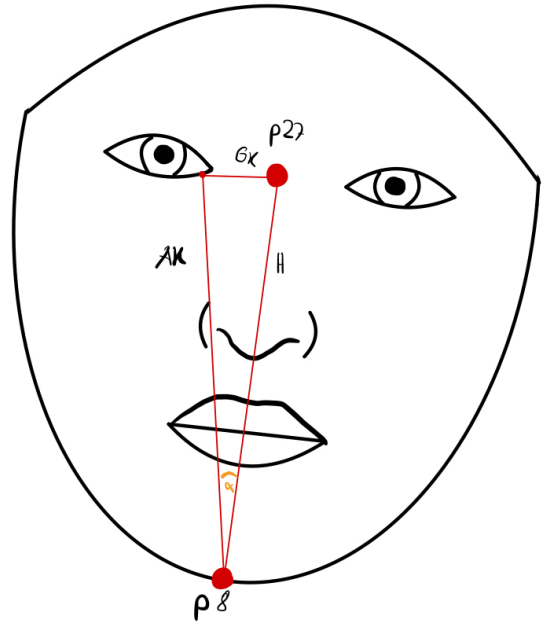
$$\tan\alpha = \frac{GK}{AK}$$

$$\alpha = \arctan\left(\frac{GK}{AK}\right)$$

$$\alpha = \arctan\left(\frac{x_2 - x_1}{y_2 - y_1}\right)$$

```
Point centerPoint = new Point(points[8].x, points[8].y);
Point winkelPoint = new Point(points[27].x, points[27].y);

Imgproc.line(rgbMat, centerPoint, winkelPoint, new Scalar(255, 255, 0), 5);

double faceAngle = Math.Atan2(winkelPoint.y - centerPoint.y,
    winkelPoint.x - centerPoint.x) * 180.0f / Math.PI + 90.0f;

String angleTxt = "Winkel: " + faceAngle.ToString("0.00");

// Code für Textausgabe…
```

Lösung: Kopfneigung



$$\tan\alpha = \frac{GK}{AK}$$

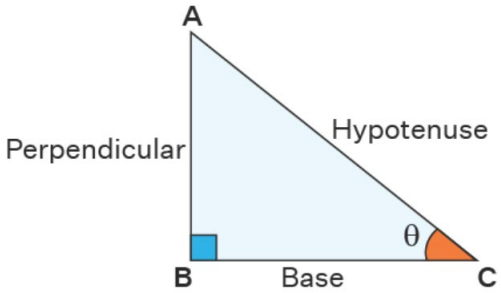
$$\alpha = \arctan\left(\frac{GK}{AK}\right)$$

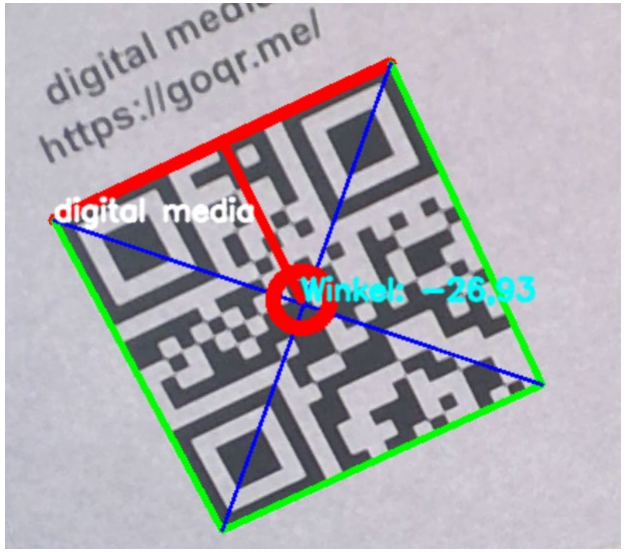
$$\alpha = \arctan\left(\frac{x_2 - x_1}{y_2 - y_1}\right)$$

```
Point centerPoint = new Point(points[8].x, points[8].y);
Point winkelPoint = new Point(points[27].x, points[27].y);

Imgproc.line(rgbMat, centerPoint, winkelPoint, new Scalar(255, 255, 0), 5);

double faceAngle = Math.Atan2(winkelPoint.y - centerPoint.y,
    winkelPoint.x - centerPoint.x) * 180.0f / Math.PI + 90.0f;

String angleTxt = "Winkel: " + faceAngle.ToString("0.00");
Imgproc.putText(rgbMat, angleTxt, centerPoint+new Point(30,0),
    Imgproc.FONT_HERSHEY_PLAIN, 1.5, new Scalar(0, 255, 255), 1, Imgproc.LINE_4, false);
```

Übung: Winkel berechnen

- Winkel berechnen mit ARCTAN2





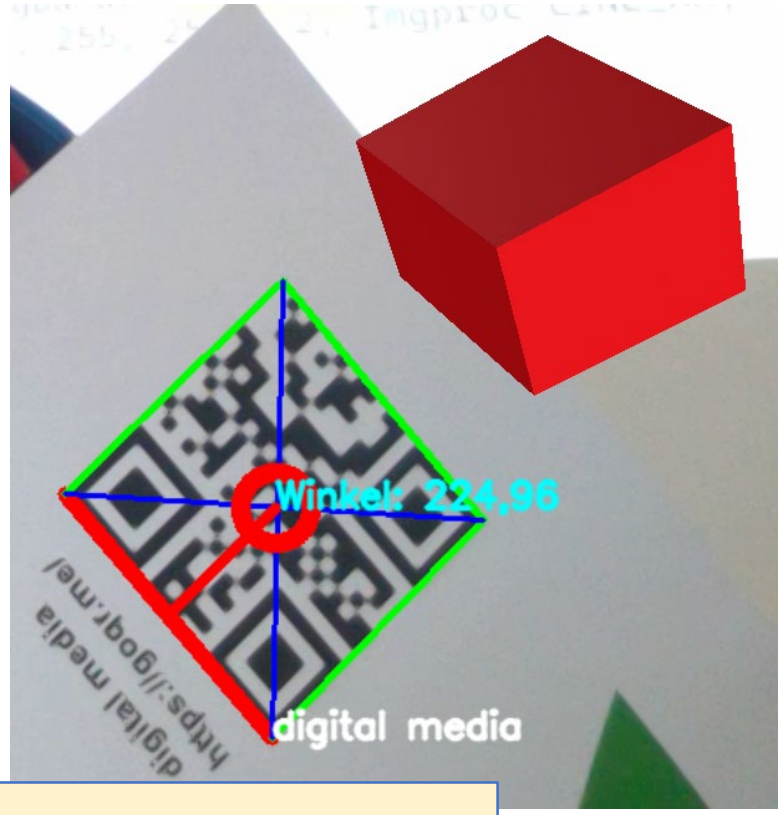
$$\tan\theta = \frac{\text{Perpendicular}}{\text{Base}}$$

$$\Rightarrow \theta = \tan^{-1}\left(\frac{\text{Perpendicular}}{\text{Base}}\right)$$

```
double qrAngle = Math.Atan2(winkelP.y - centerP.y,
                            winkelP.x - centerP.x)*180.0f/Math.PI + 90.0f;

String angleTxt = "Winkel: " + qrAngle.ToString("0.00");

Imgproc.putText(rgbaMat, angleTxt, centerP, Imgproc.FONT_HERSHEY_SIMPLEX, 0.7,
new Scalar(0, 255, 255, 255), 2, Imgproc.LINE_AA, false);
```

QR: Winkelberechnung

- Rad: 2 π
- Grad: 360°
- Neugrad: 400°




```
double qrAngle = Math.Atan2(winkelP.y - centerP.y,
                            winkelP.x - centerP.x)*180.0f/Math.PI + 90.0f;

 if (qrAngle < 0.0f) qrAngle += 360;
String angleTxt = "Winkel: " + qrAngle.ToString("0.00");

Imgproc.putText(rgbaMat, angleTxt, centerP, Imgproc.FONT_HERSHEY_SIMPLEX, 0.7,
new Scalar(0, 255, 255, 255), 2, Imgproc.LINE_AA, false);
```

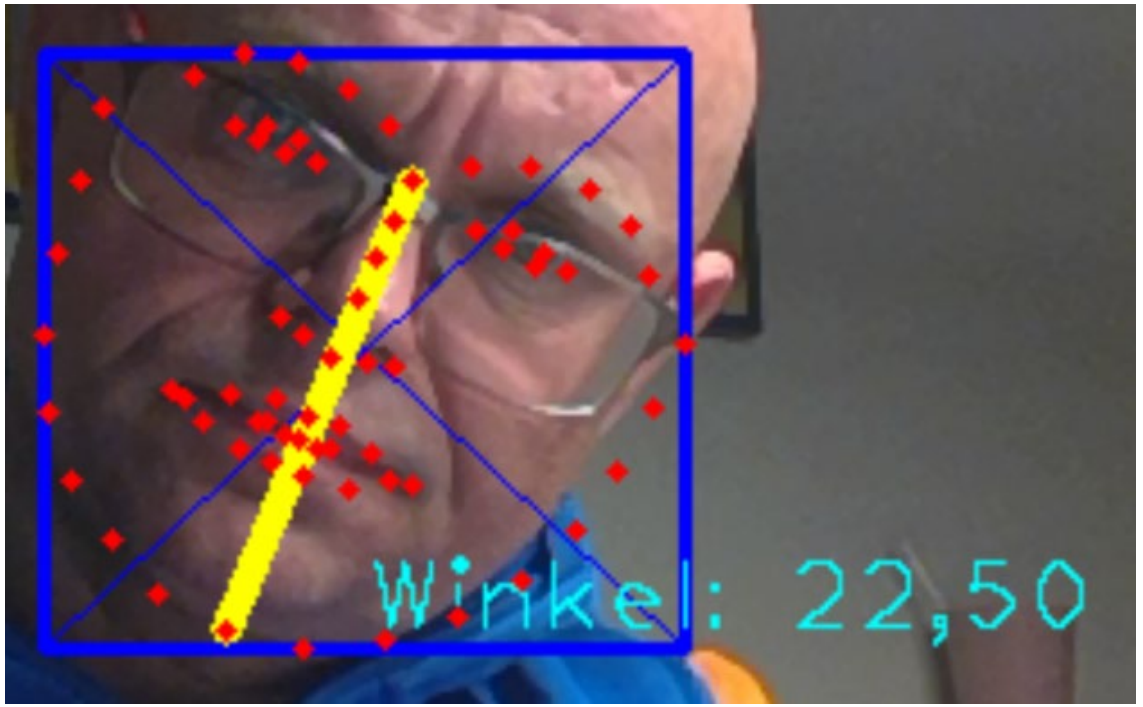
Fonts

- `Imgproc.FONT_HERSHEY_SIMPLEX`

Static Store

Ziel

- Berechnung der Kopfneigung in Grad
- Einzeichnen der Berechnungslinie
- Textausgabe (Matrix) des Winkels
- Speicherung im „Static Store“
- Textausgabe (TMPro) im InfoCanvas



Static Store

- int
- bool
- List<Vector2>
- Rect

```
8   namespace DMT
9   {
10      public static class StaticStore
11      {
12
13          private static int _myFaceCnt = 0;
14          private static bool _camInit = false;
15          private static List<Vector2> _faceList = null;
16          private static List<UnityEngine.Rect> _faceRectList = null;
17          private static UnityEngine.Rect _boundingBox = Rect.zero;
18
19          public const int imgWidth = 640;
20          public const int imgHeight = 480;
21
22          public static int myFaceCnt // face counter (Anzahl)...
40
41          public static List<Vector2> myFaceList // face Points...
50
51          public static List<UnityEngine.Rect> myFaceRects // faceRect - BoundingBoxes...
60
61          public static UnityEngine.Rect myBoundingBox // main BoundingBoxes...
70
71          public static bool camInit...
76
77      }
78  }
```

Static Store

- auto-implemented property

```
public static double FaceAngle { get; set; } // face angle (Winkel)
```

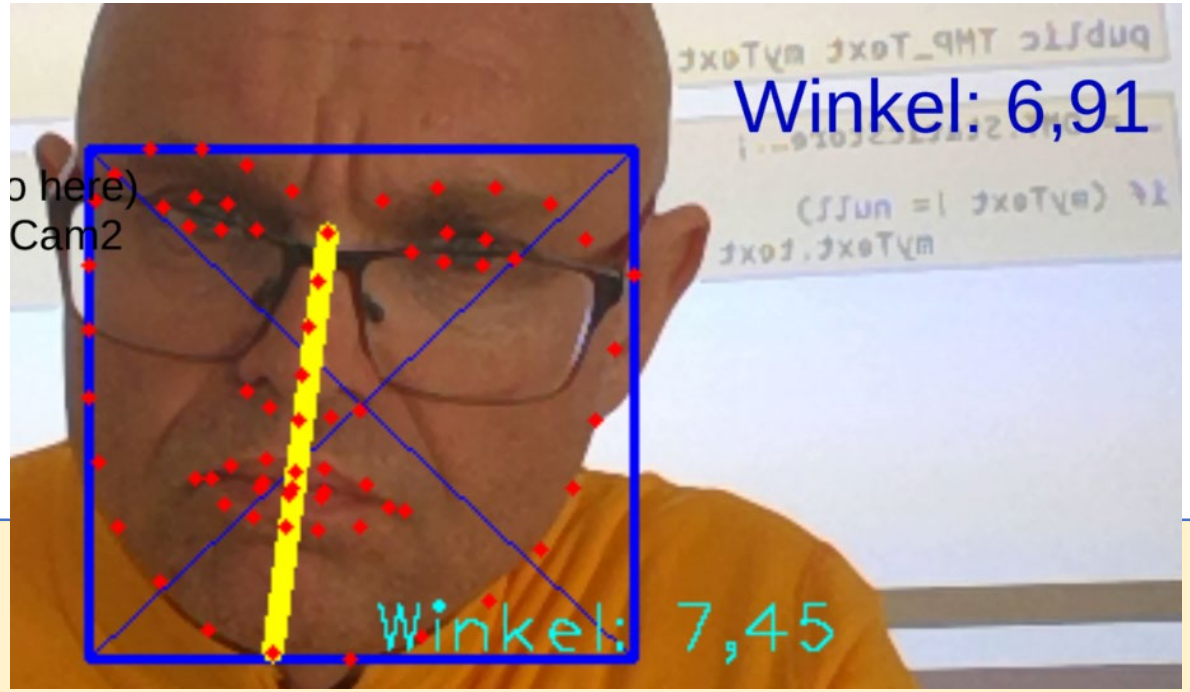
```
private static double _faceAngle = 0;

public static double FaceAngle
{
    get { return _faceAngle; }
    set { _faceAngle = value; }
}
```

- Zusweisung

```
DMT.StaticStore.FaceAngle = faceAngle;
```

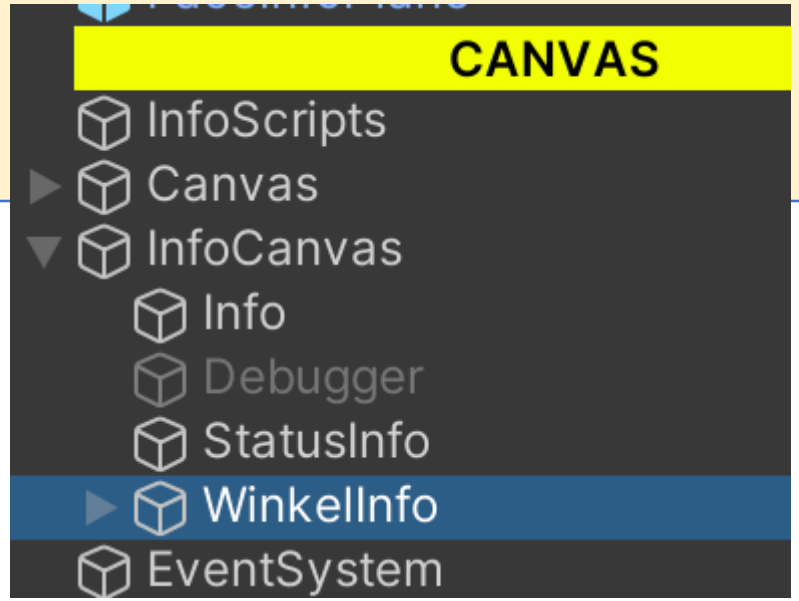
Text Mesh Pro

- WinkelTextShow Script



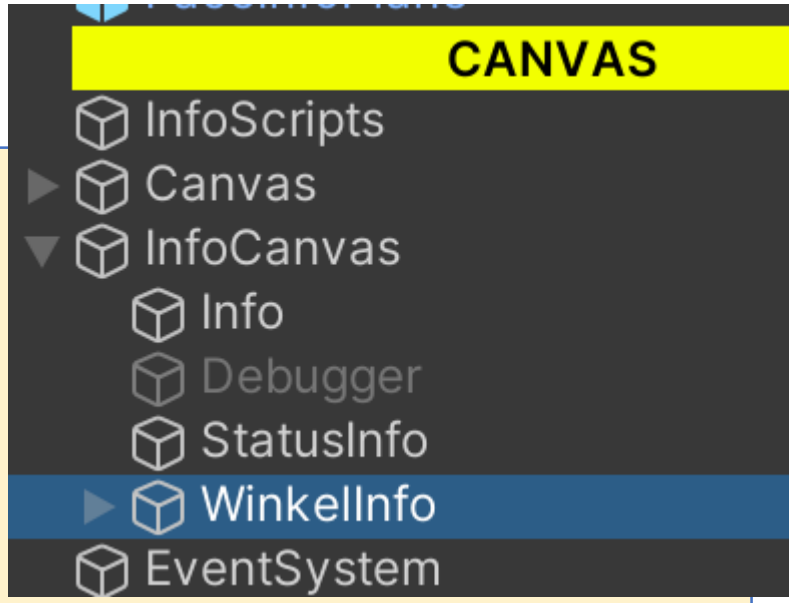

```
using TMPro;

public TMP_Text myText;
```

```
_____ = DMT.StaticStore. _____;

if (myText != null)
            myText.text = "Winkel: " + _____ + "\n";
```



Lösung: Winkel & Text Mesh Pro

- WinkelTextShow Script




```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class ShowWinkelInfo : MonoBehaviour
{
    public TMP_Text myText;

    void Update()
    {
       double myFaceAngle = DMT.StaticStore.FaceAngle;

        if (myText != null)
            myText.text = "Winkel: " + myFaceAngle.ToString("0.00") + "°";
    }
}
```

Übung: Mundwinkel

`Face68DetectorDrawPoints`

1. Mundwinkel anzeigen (big red line)
2. Mundwinkel Wert einzeichnen

`StaticStore`

1. Mundwinkel im StaticStore
2. Mundwinkel im InfoCanvas (TMPro) „NoseInfo“

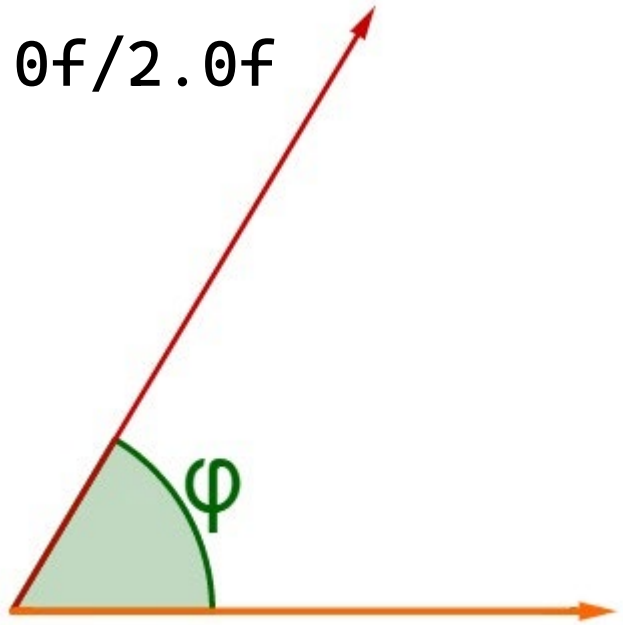
ARCCOS

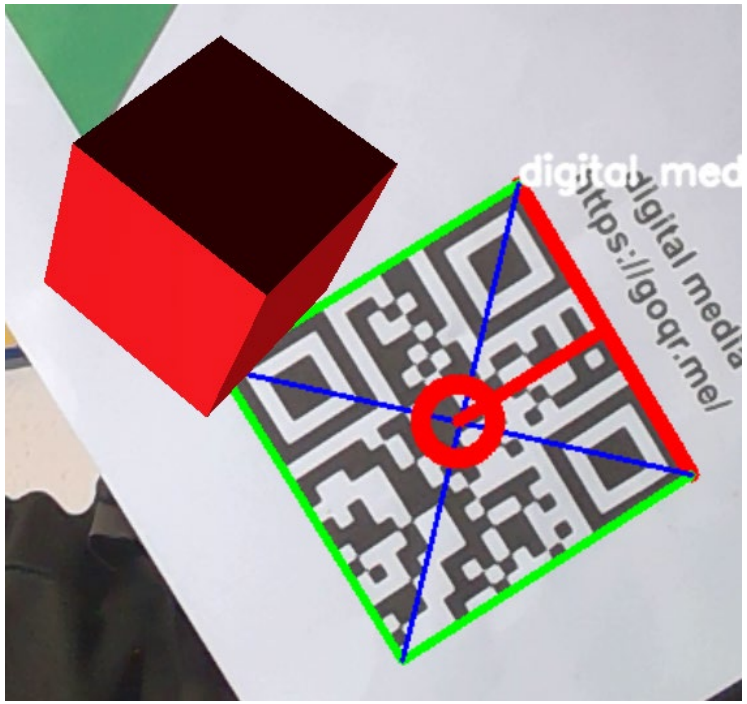
Übung: Winkel berechnen

$$\varphi = \arccos\left(\frac{\vec{u} \circ \vec{v}}{\left|\vec{u}\right| \cdot \left|\vec{v}\right|}\right)$$

- Winkel berechnen mit ARCCOS
- Winkel ausgeben
  Imgproc.**putText**
- `Rad/Grad: Math.PI*360.0f/2.0f`

```
using System;

Math.Acos()
Math.Sqrt()
Math.Pow()
Math.Atan2()

Math.PI Mathf.PI
```

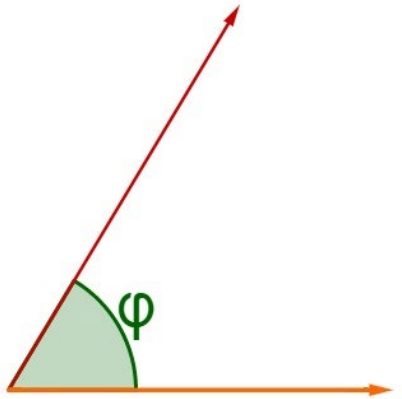


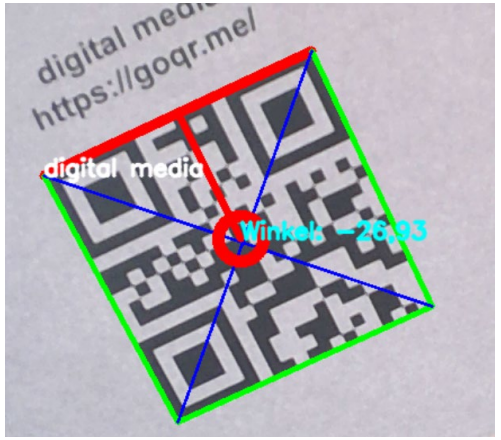


Übung: Winkel berechnen

$$\varphi = \arccos\left(\frac{\vec{u} \circ \vec{v}}{\left|\vec{u}\right| \cdot \left|\vec{v}\right|}\right)$$

- Winkel berechnen mit ARCCOS





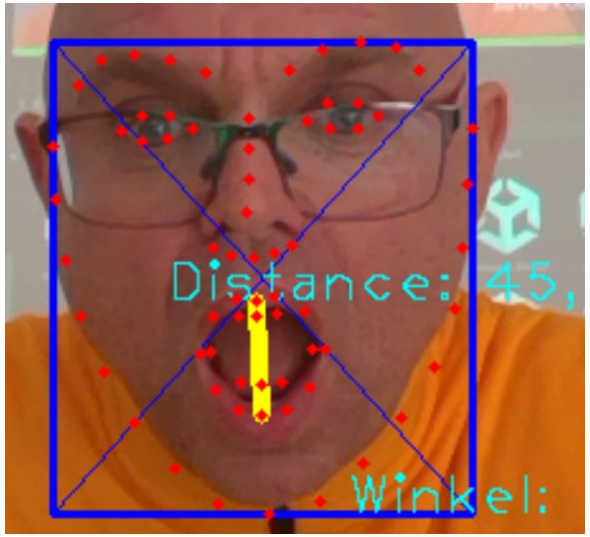

```
Point winkelCalc = new Point(winkelP.x - centerP.x, winkelP.y - centerP.y);

qrWinkel = Math.Acos(winkelCalc.x /
           Math.Sqrt(Math.Pow(winkelCalc.x, 2) + Math.Pow(winkelCalc.y, 2)));

DMT.StaticStore.myRotData = qrWinkel;
qrCenterP = centerP;
```

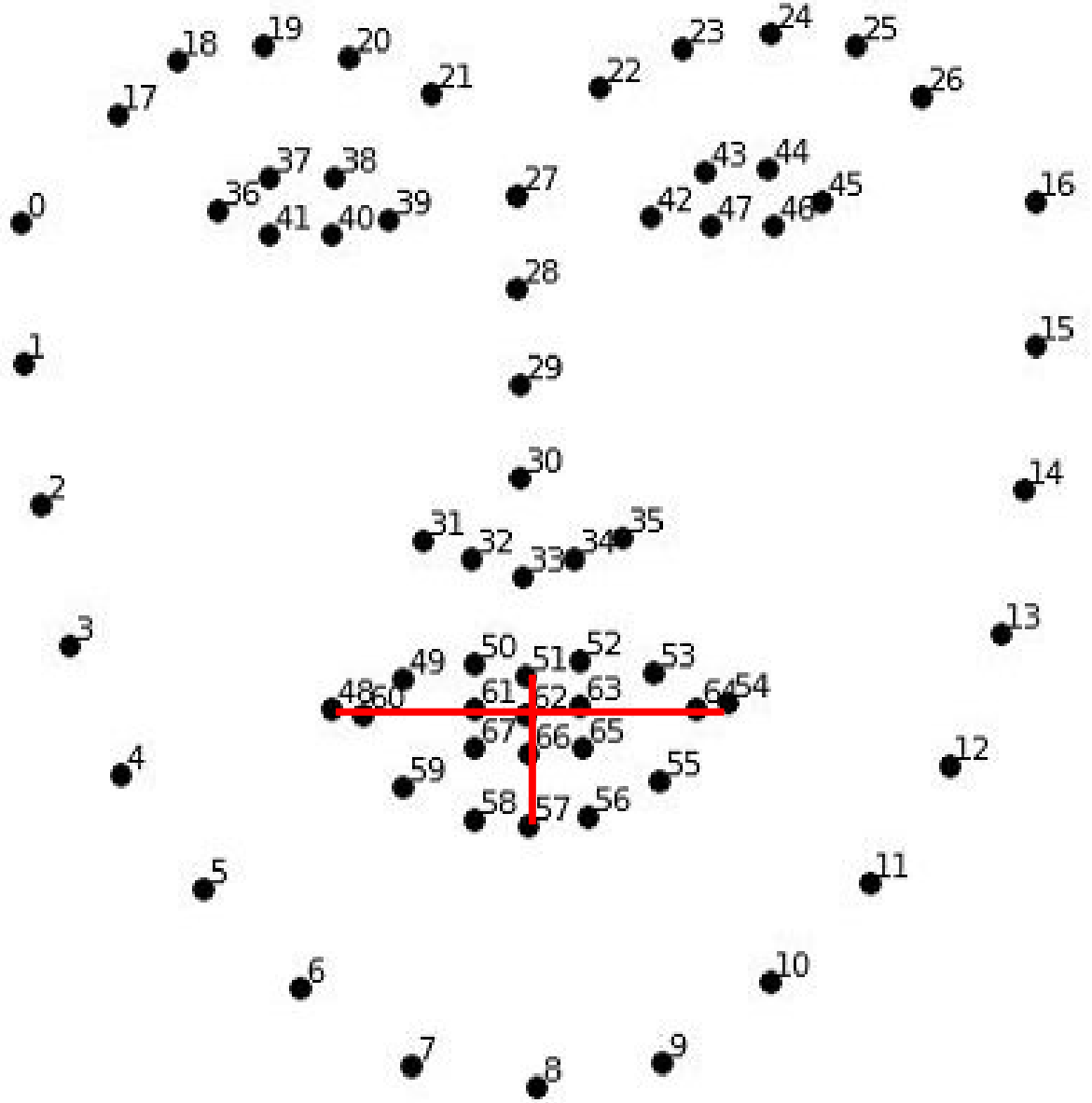
```
public double GetQRWinkel()
{
    return qrWinkel;
}
```

Distance

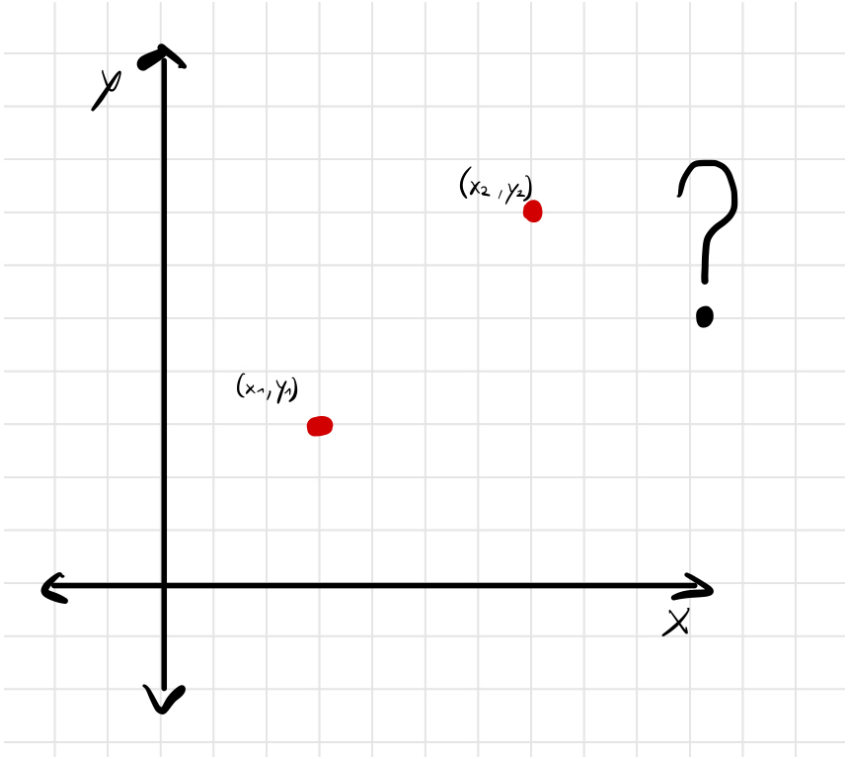
Distanzen „Mund“

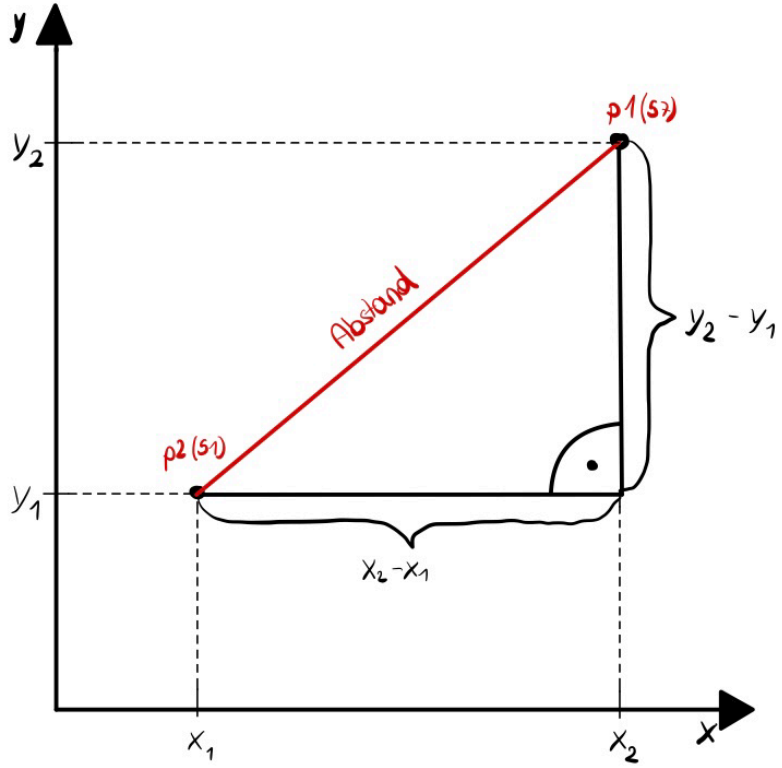
- Mundbreite
  - Punkte 48 und 54
- Mundöffnung
  - Punkte 51 und 57
    oder 62 und 66 (ohne Lippen)
- Verhältnis





Euklidische Distanz zweier Punkte im R2





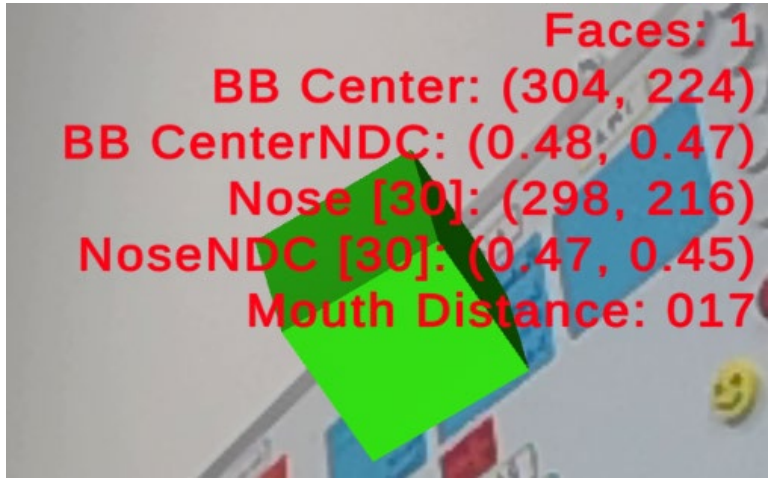
$$\text{Abstand} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$\text{Abstand} = \sqrt{(p_{1,x} - p_{2,x})^2 + (p_{1,y} - p_{2,y})^2}$$

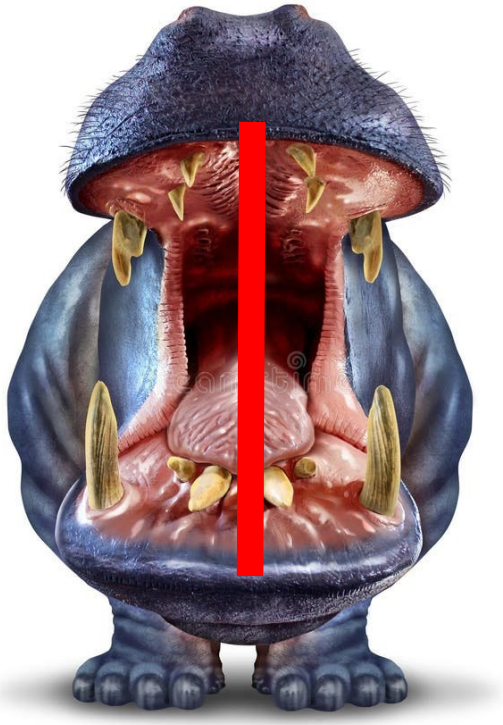
Mund - Punktdistanz

```
private const int mouthIndex1 = 51;
private const int mouthIndex2 = 57;

float mouthDistanz = Vector2.Distance(myFList[mouthIndex1], myFList[mouthIndex2]);
```

- auch als NDC möglich
  kann auch größer 1 werden (!)



Übung: Mouth



Face68DetectorDrawPoints

1. Mundöffnung anzeigen (big red line)
2. Mundöffnung Distanz als Wert (px) einzeichnen
3. Mundöffnung: NDC Wert einzeichnen
   Relativ zur Kopfhöhe

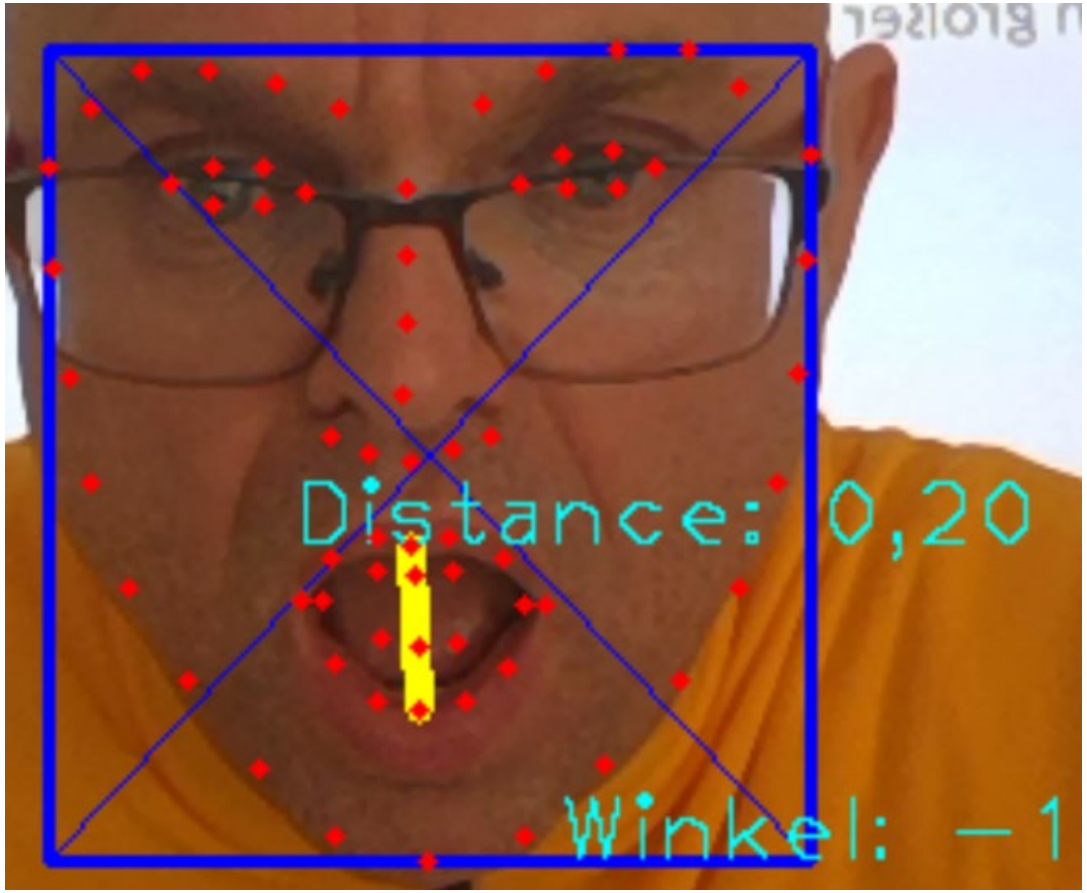
StaticStore

1. Mundöffnung im StaticStore
2. Mundöffnung im InfoCanvas (TMPro)

```
float mouthDistanz = Vector2.Distance(points[51], points[57]);
DMT.StaticStore.MouthOpen = mouthDistanz;
String mouthTxt = "Distance: " + mouthDistanz.ToString("0.00")
```

NDC Relativ

- Blue BoundingBox → Höhe schon im Code vorhanden faceBBMin und faceBBMax



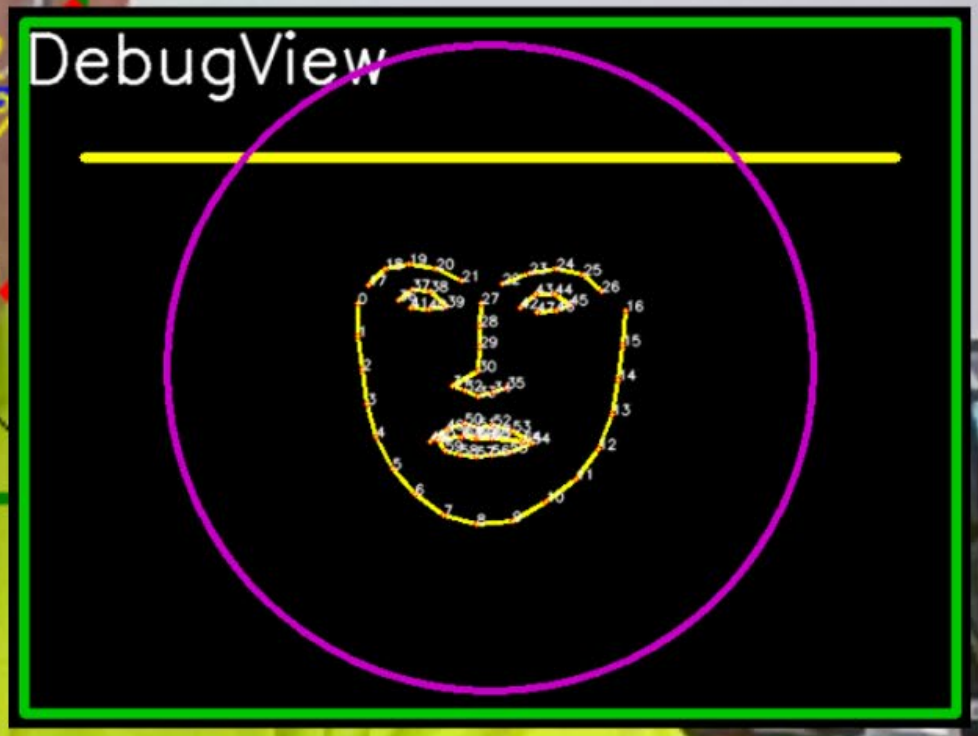

```
float mouthDistanz = Vector2.Distance(points[62], points[66]);

double blueHeight = faceBBMax.y - faceBBMin.y;
double mouthNDC = mouthDistanz / blueHeight;
```

Compass Prefab

OpenCV FaceLandmark Draw 68Points

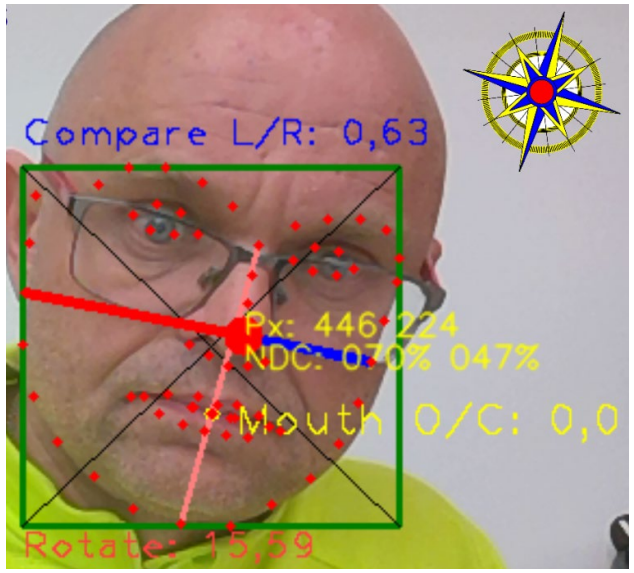
Usage:
SHIFT-1: InGame Debug Console
TAB: Toggle InfoScreen (put your own info here)
0,a,b .. Switch Cam 0..Cam0 a..Cam1 b..Cam2

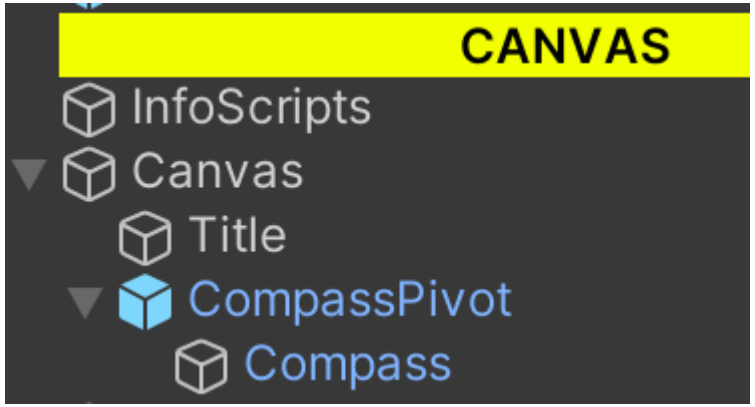
Winkel: 0,78°



Übung: Compass Prefab

- Winkel im StaticStore:
  „myFaceAngle“
- Prefab erstellen
  - Image „Compass“: compass_dmt.png → Sprite (2D and UI)
  - Script

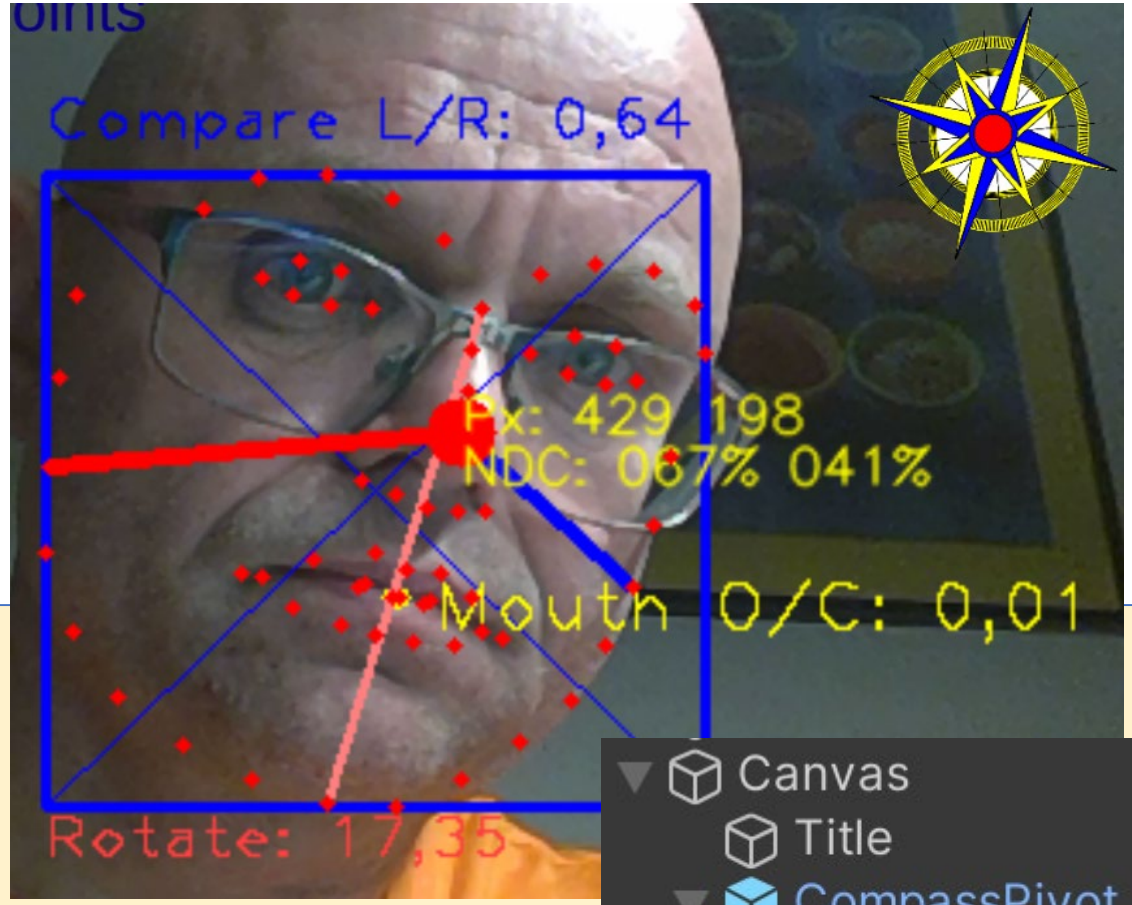





```
this.transform.rotation = Quaternion.Euler(0, 0, -1.0f * (float)myFaceAngle);
```

Compass Prefab

- Winkelanzeige als Prefab

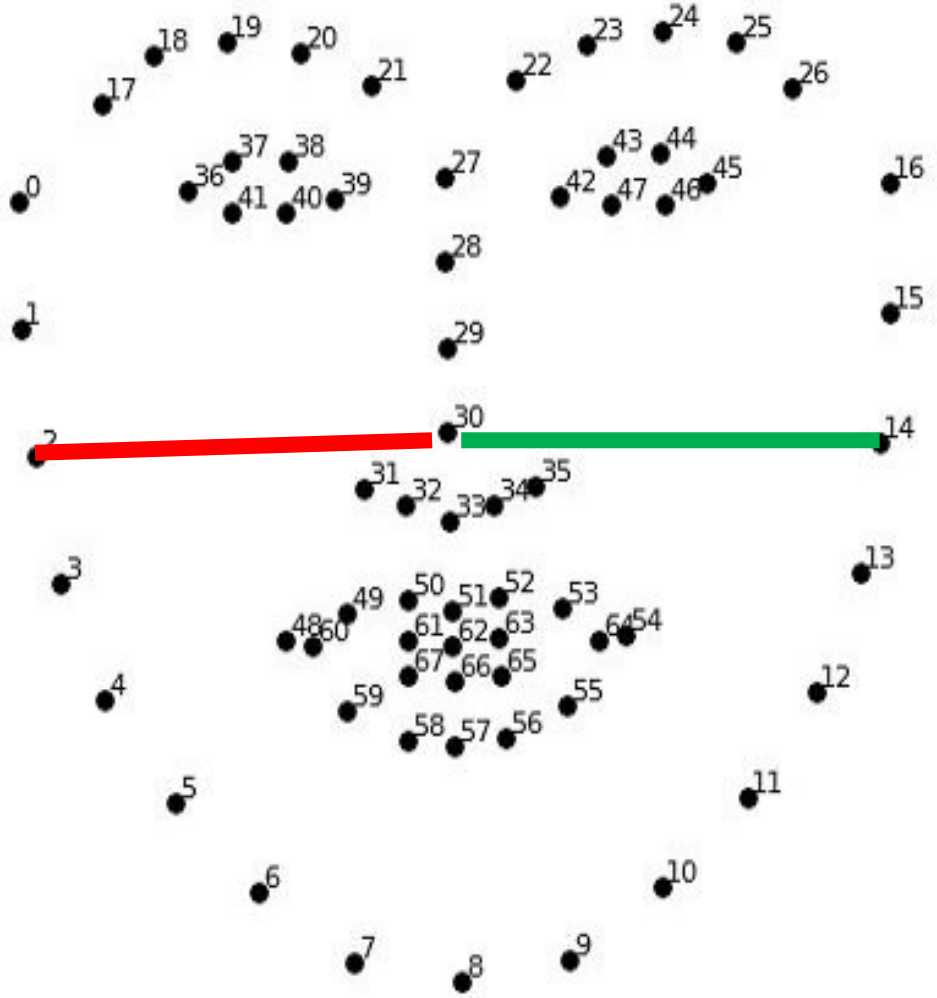



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WinkelRotate : MonoBehaviour
{
    void Update()
    {
        double myFaceAngle = DMT.StaticStore.FaceAngle;
        this.transform.rotation = Quaternion.Euler(0, 0, -1.0f * (float)myFaceAngle);
    }
}
```

Kopfbewegung

Gesichtsgesten left/right

- Ja
  rauf/runter
- Nein
  rechts/links
- Distanzverhältnis left/right
  2-30 und **30-14**

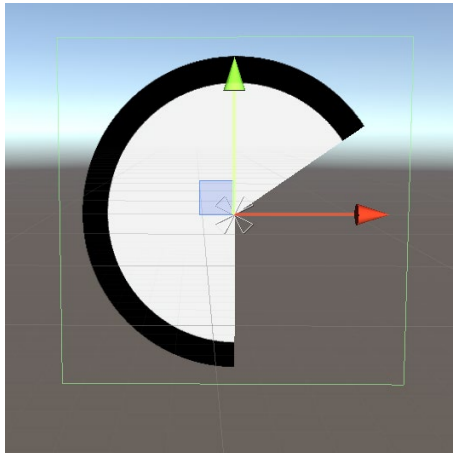


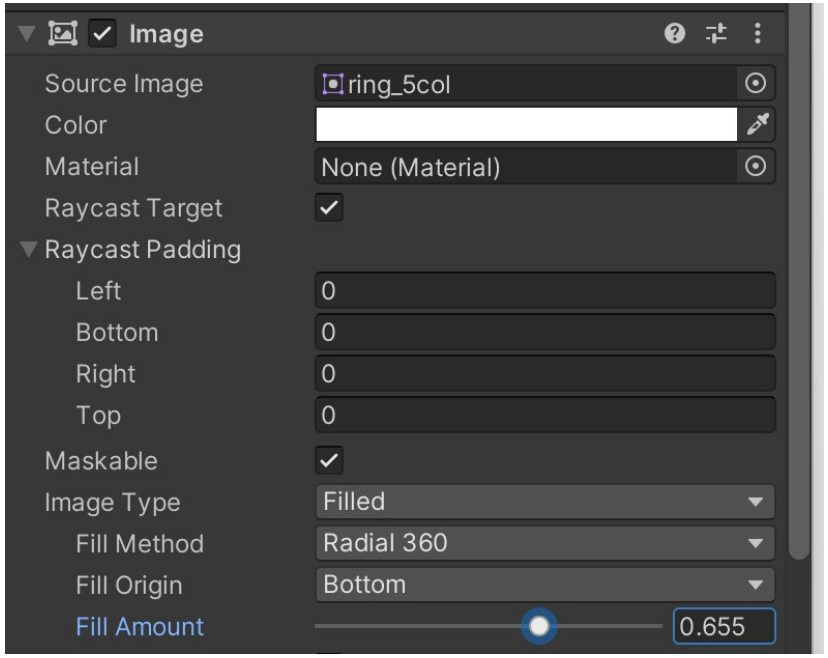

```
float leftDistanz = Vector2.Distance(points[2], points[30]);
float rightDistanz = Vector2.Distance(points[30], points[14]);

float compareDistanz = leftDistanz / (leftDistanz+rightDistanz);
```

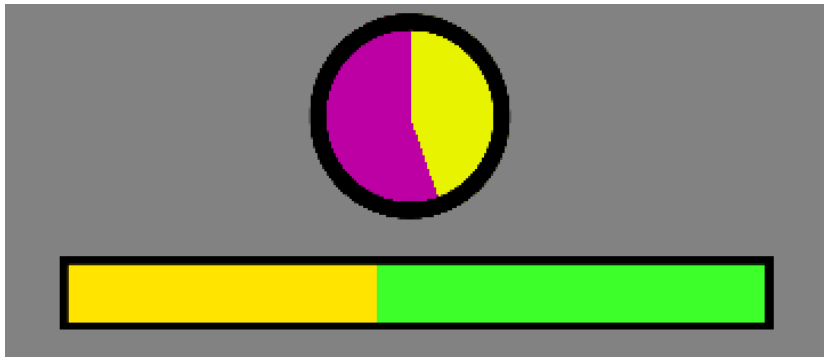
Circle Fill

- Radial progress bar
- Image/UI → Fill

```
radialIndicatorUI.enabled = true;
radialIndicatorUI.fillAmount = indicatorTimer;
```

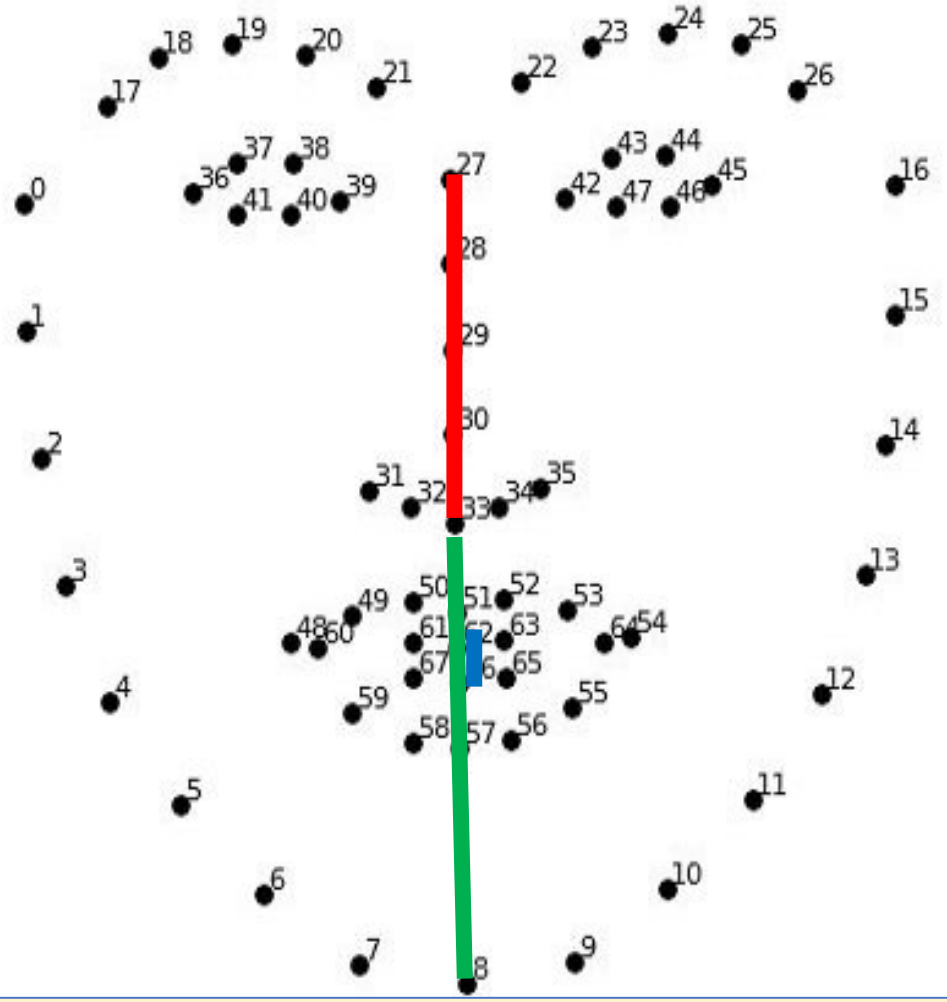






Gesichtsgesten up/down

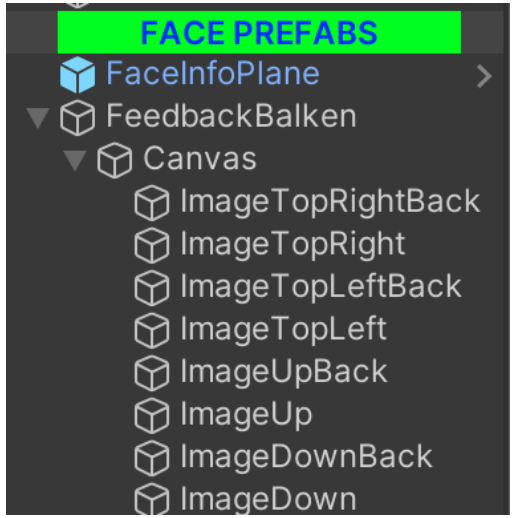
- Ja
  rauf/runter
- Nein
  rechts/links
- Distanzverhältnis up/down
  27-33 und **8-33 – 62-66**

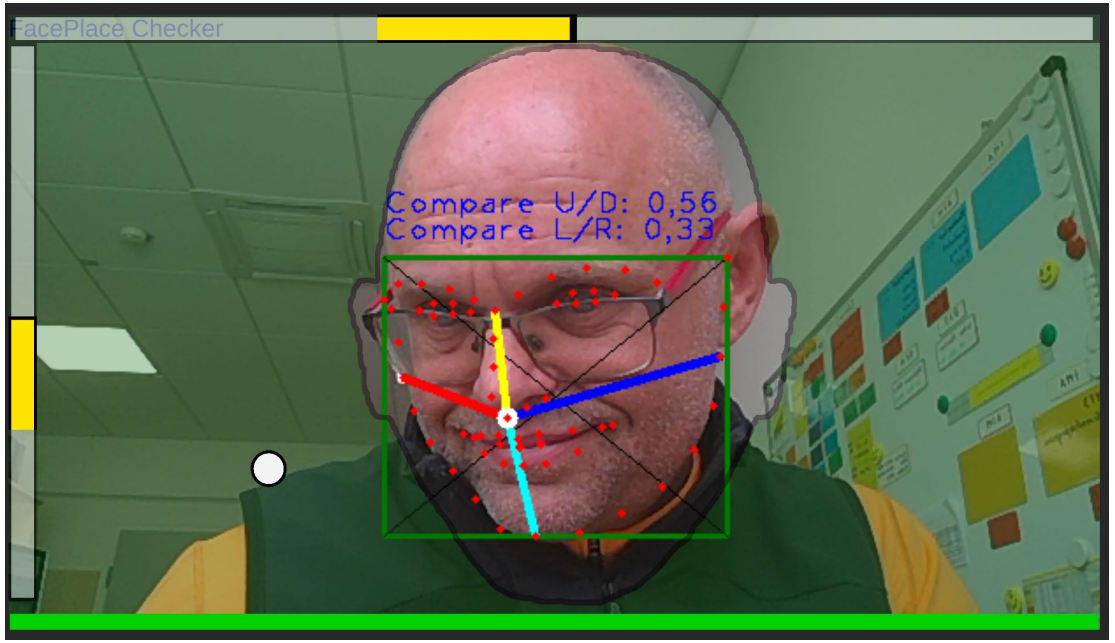


```
float raufDistanz = Vector2.Distance(points[pointOben], points[pointMitte])*1.40f;
float runterDistanz = Vector2.Distance(points[pointMitte], points[pointUnten]);

compareDistanzRaufRunter = raufDistanz / (raufDistanz + runterDistanz- mouthDistanz);
DMT.StaticStore.upDownFace = compareDistanzRaufRunter;
```

Info Anzeige

- Balken
- https://github.com/nischelwitzer/ClockButton

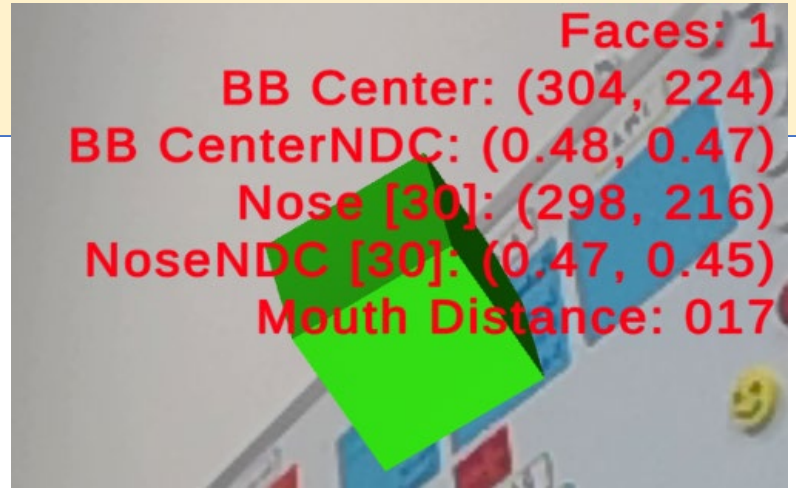




Move

Nose & NDC

```
private const int noseIndex = 30;

Vector2 sizeImg = new Vector2(DMT.StaticStore.imgWidth, DMT.StaticStore.imgHeight);

Vector2 nosePoint = myFList[noseIndex];
Vector2 nosePointNDC = nosePoint / sizeImg;
```



Nose & NDC → Move


```
private const int noseIndex = 30;

Vector2 sizeImg = new Vector2(DMT.StaticStore.imgWidth, DMT.StaticStore.imgHeight);

Vector2 nosePoint = myFList[noseIndex];
Vector2 nosePointNDC = nosePoint / sizeImg;
```

```
if (refPoint != Vector2.zero)
{
  float deltaX = refPoint.x - nosePoint.x;
  float deltaY = refPoint.y - nosePoint.y;

  if (moveAbs)
     this.transform.rotation = Quaternion.Euler(deltaY, deltaX, 0); // abs
  else
     this.transform.Rotate(deltaY / 100*moveVelocity, deltaX / 100*moveVelocity, 0);
}
```

Face OK?

Size & Shape

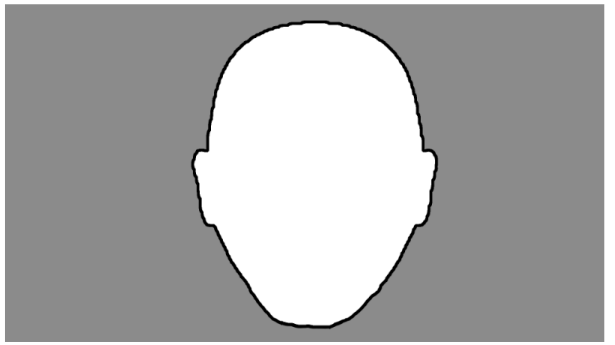
Maske & Code → git

- https://github.com/nischelwitzer?tab=repositories
- https://github.com/nischelwitzer/OCV-face68-FaceMask
  - **OCV-face68-StaticStore**
  - **OCV-face68-Nose-Mouth-BB**
  - **OCV-face68-2DOF-Rotation**

Check Nose Point

Check for correct face position.

```
Vector2 myNoseNDC = DMT.StaticStore.NoseNDC*100;

bool okNose = false;
if (((noseX - noseDeltaX) <= myNoseNDC.x) && (myNoseNDC.x <= (noseX
    ((noseY - noseDeltaY) <= myNoseNDC.y) && (myNoseNDC.y <= (noseY
```

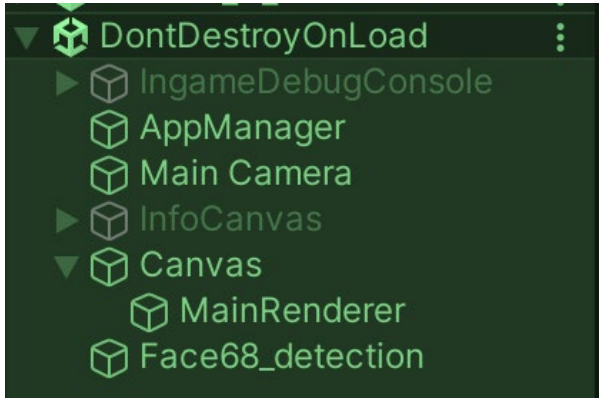


Feedback


FH | JOANNEUM
University of Applied Sciences
Face Game
Dlib Face

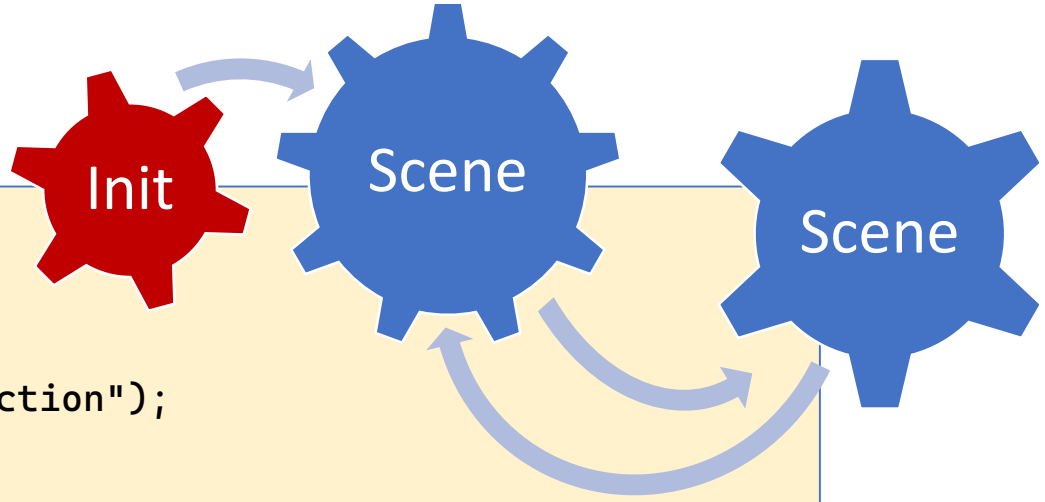
Feedback – Feedback – Feedback

- **CAM-Always** – CAM immer anzeigen (auch im Menü, Credits, …)
- Wartezeiten verkürzen
  - Sonst IMMER Feedback
  - Menüwechsel → z.B. 5 Sekunden „Ewigkeit“ ohne Feedback → **Usabiliy-Katastrophe**
    - Lösung: Einfärben und ein kurzer Ton
- Don´t Destroy on Load → wenn in allen Szenen verwendet
  - Application Manager
  - IngameDebugConsole
  - Canvas & InfoCanvas
  - WebCamCodes: Face68_detection



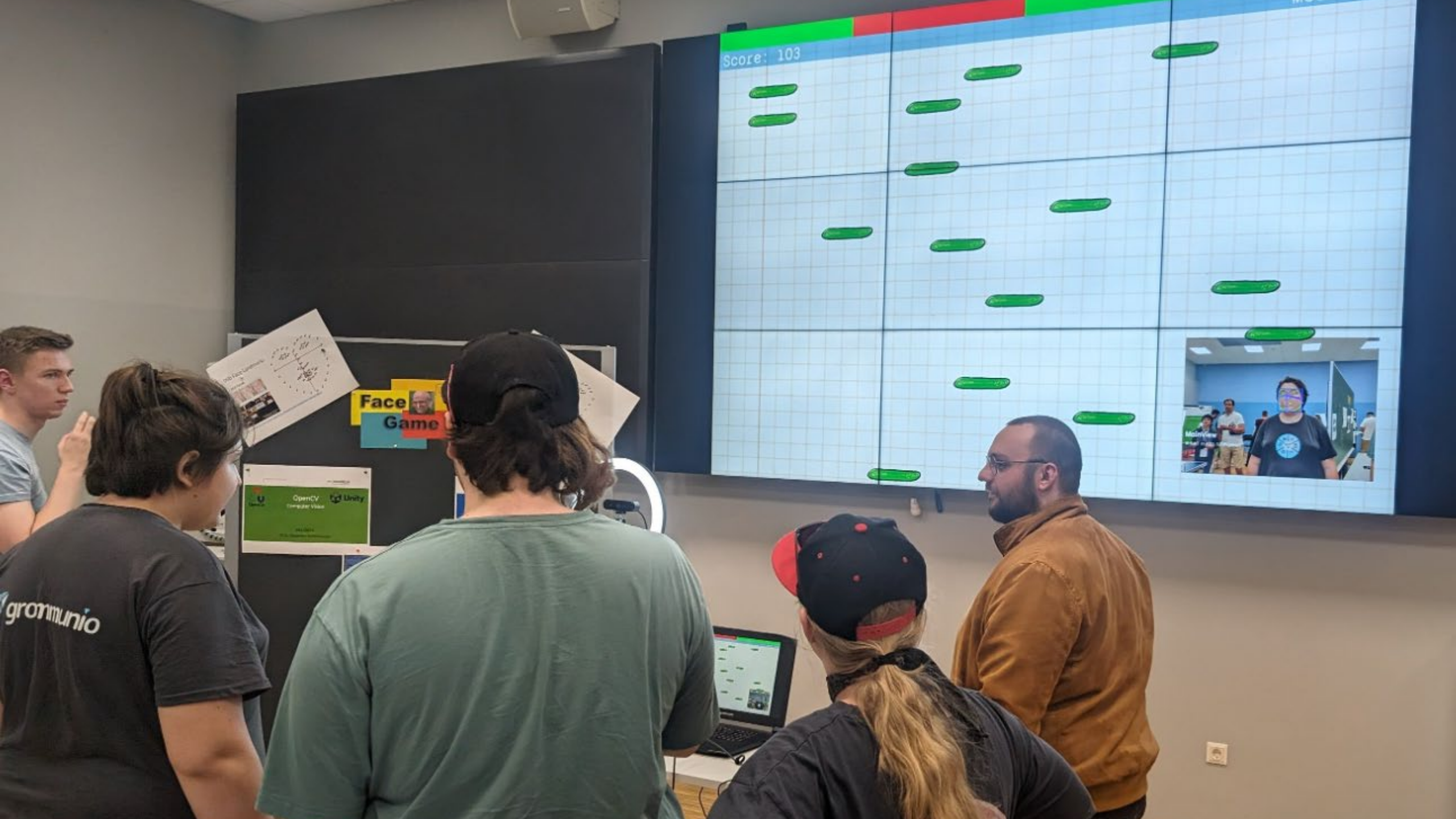
DontDestroyOnLoad

- Initialisierung (Init) nur einmal → DontDestroyOnLoad
- Weitergabe an Scenen
- Kamera „aufwecken“ (neue Scene)




```
public class AwakeCamera : MonoBehaviour
{
    void Start()
    {
        GameObject myCamGO = GameObject.Find("Face68_detection");

        string startCam = "";
        PlayerPrefs.GetString("startCam", startCam);
        myCamGO.GetComponent<WebCamTextureToMatHelper>().requestedDeviceName = startCam;
        myCamGO.GetComponent<WebCamTextureToMatHelper>().Initialize();
        Debug.Log("Awake/Refresh Camera: " + startCam);
    }
}
```

Ähnlich Code wie „Camera Switcher“ – in der neuen Scene!


Score: 103
Face
Game
OpenCV
Unity
grommunio

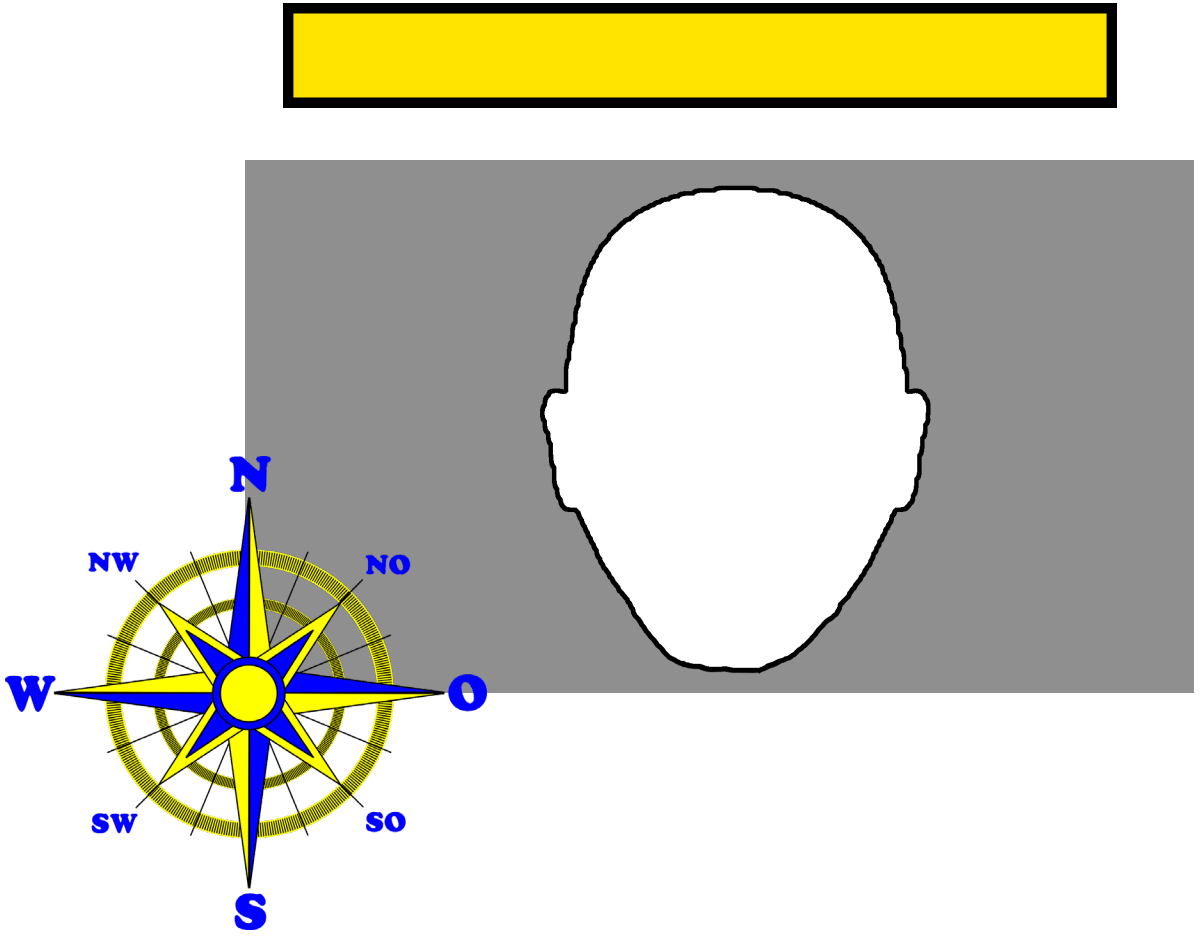
Infos

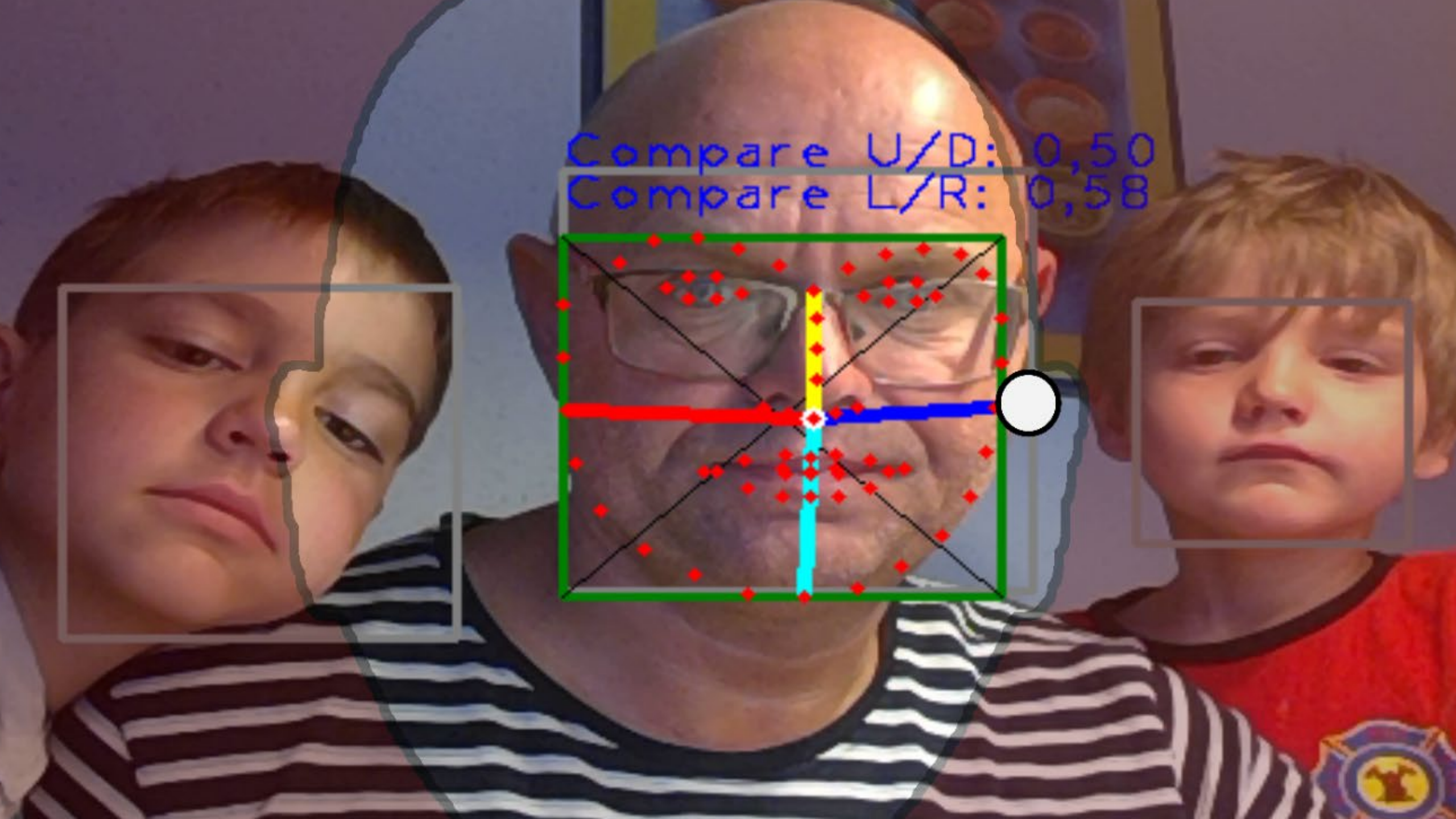
- Tutorial → nicht gleich dem Spiel
  - z.B. Pause machen > **Info** Mund öffnen > weiter nach der „Mundöffnung“
- Am Anfang „leicht“
  - 30 Sekunden „Easy“ Mode
  - Alle Spiele am Anfang **zu schwer**
- Fehlertoleranter
  - **Clear-Death** „Sterben“ muss eindeutig sein

Tools & Infos

FaceControl Usability

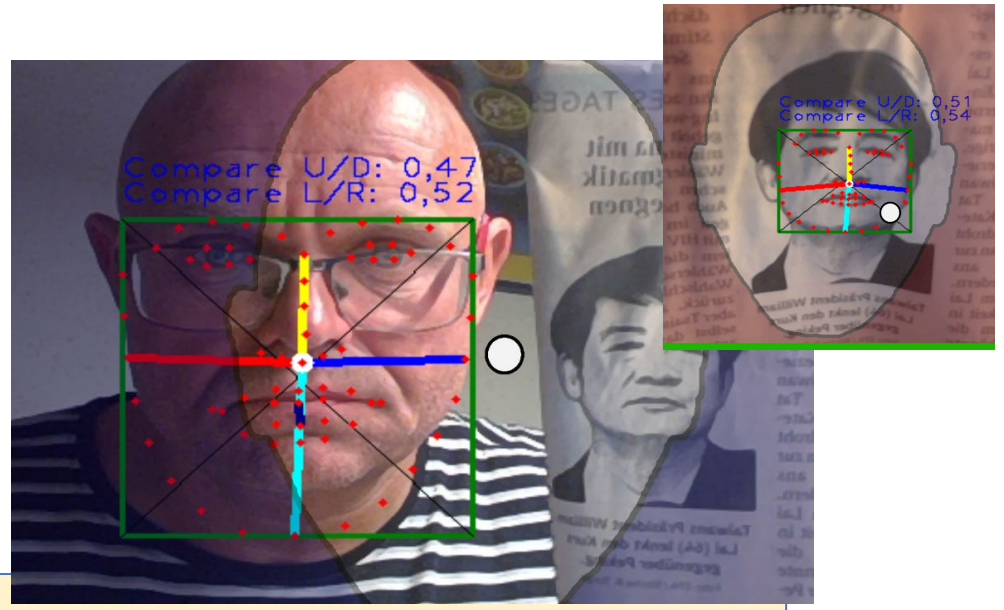
- Feedback
  - Positions-Balken
  - Neigungskompass
- Stabilität (Cam-Auswertung)
  - Mittelpunkt → NosePoint
  - Größe → FaceMask
- Fehlervermeidung
  - Mehr Faces → take largest Face




Compare U/D: 0,50
Compare L/R: 0,58

NearestPlayer

- Select only largest Face (BB)
- Line ~230 (Face68DetectorFacePlace.cs)

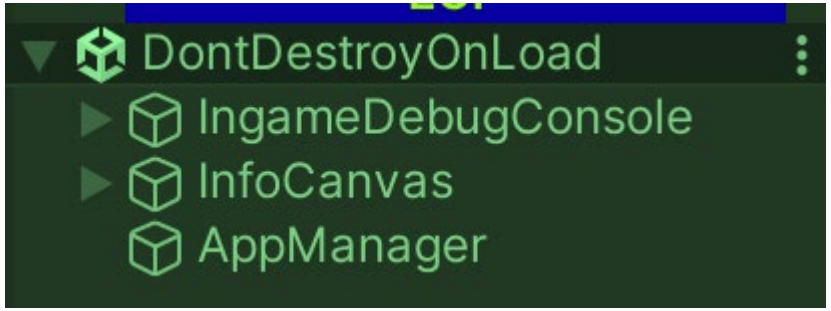



```
float rectMaxSize = 0;
UnityEngine.Rect useRect = new UnityEngine.Rect();

foreach (UnityEngine.Rect rect in detectResult)
{
    float rectSize = rect.width * rect.height;
    if (rectSize > rectMaxSize)
    {
        useRect = rect;
        rectMaxSize = rectSize;
    }
}
```

```
if (rectMaxSize > 0) // if largest face found
{
  ... userRect ...
```

```
·List<UnityEngine.Rect>·detectResult·=·faceLandmarkDetector.Detect();
```

InfoCanvas

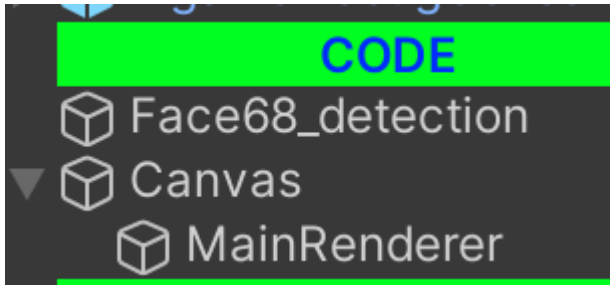


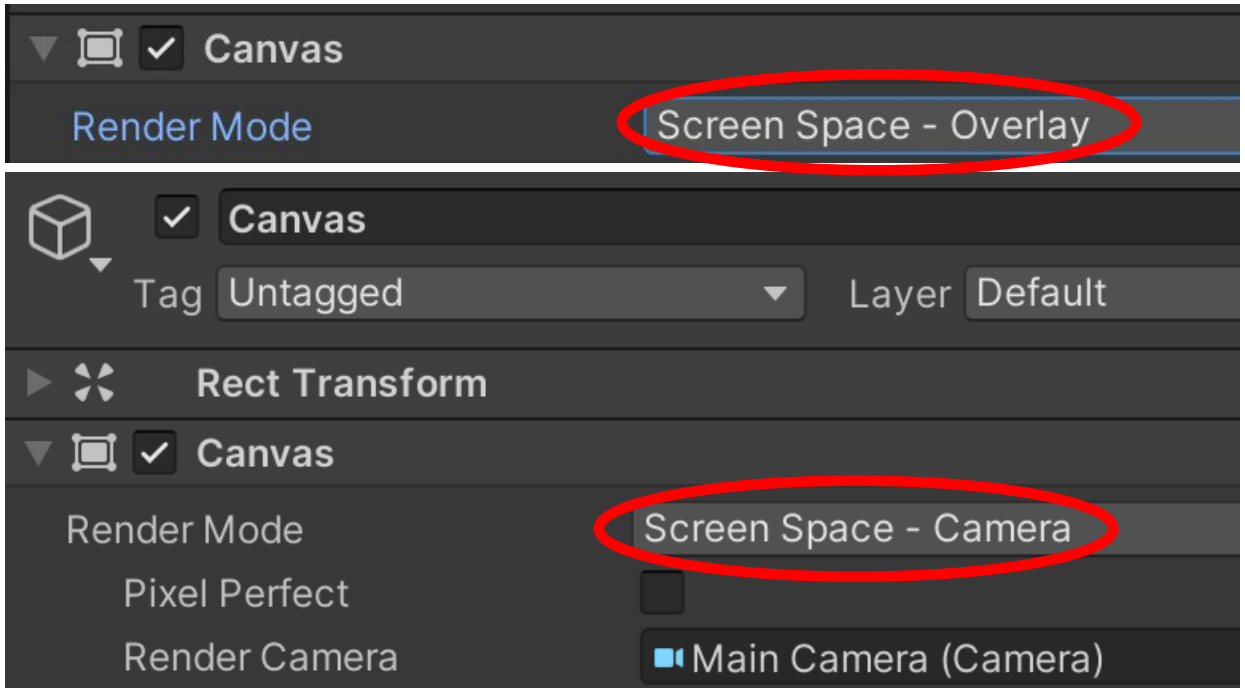
- Initialisierung von
  - ApplicationManager
  - InGame DebugConsole und
  - InfoCanvas
- DontDestroyOnLoad
- Mitnahme bei Scenenwechsel
- Nur **1x** exekutieren

```
using UnityEngine;

public class DoNotDestroyMe : MonoBehaviour
{
    private void Awake()
    {
        DontDestroyOnLoad(this.gameObject);
    }
}
```

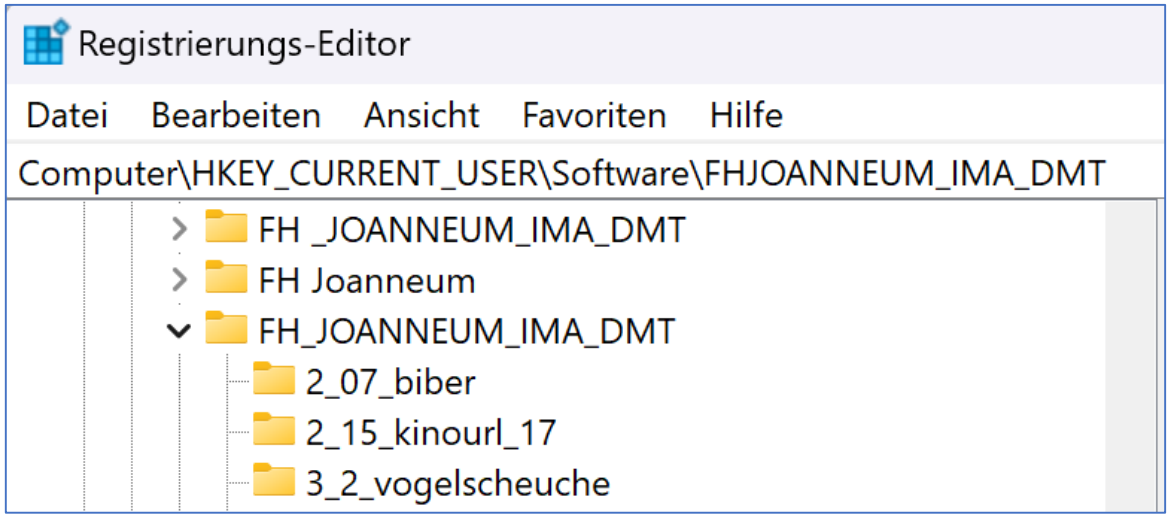
Camera & Renderer

- ScreenSpace - Overlay
- ScreenSpace - Camera
  - bei bewegter Kamera





Datenspeicherung

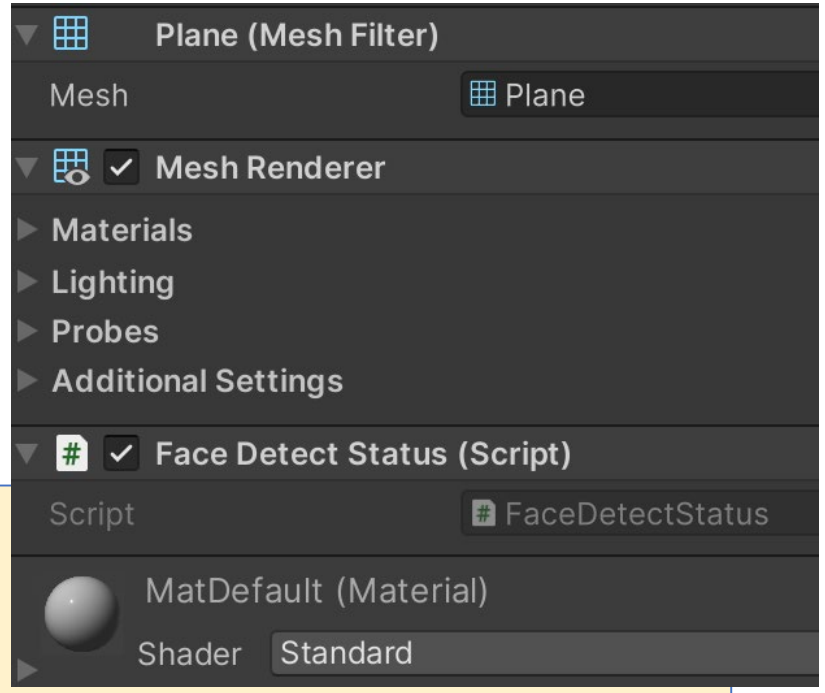


- JSON
- XML
- Simple: Registry

```
void IncStartLog()
{
    int startLog = PlayerPrefs.GetInt("startLog", 0);
    PlayerPrefs.SetInt("startLog", ++startLog);
}

void ResetPlayerPrefs()
{
    PlayerPrefs.DeleteAll();
    PlayerPrefs.SetInt("startLog", startLog);
}
```

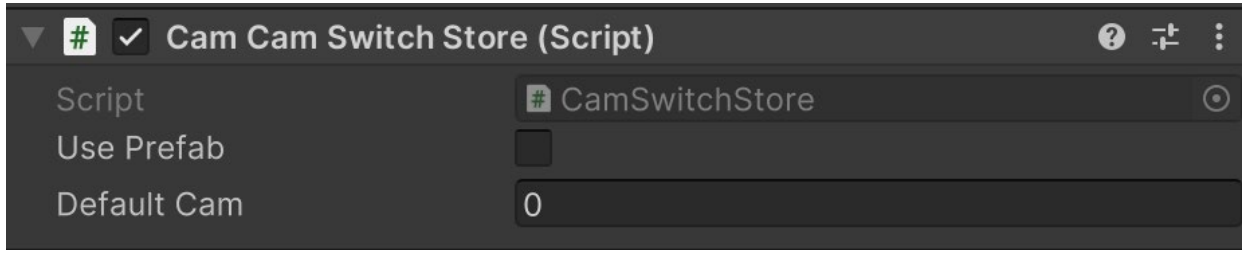
FaceDetectStatus

- Prefab




```
public class FaceDetectStatus : MonoBehaviour
{
    void Update()
    {
        int numFaces = DMT.StaticStore.myFaceCnt;

        if (numFaces > 0)
            this.GetComponent<Renderer>().material.color = new Color(0, 255, 0);
        else
            this.GetComponent<Renderer>().material.color = new Color(255, 0, 0);
    }
}
```

Cam Switcher



- Save selected Cam
- Switch: **0 | a | b** (0,1,2)

```
if (newCam != startCam) // new camera request
{
    PlayerPrefs.SetString("startCam", newCam);
    this.GetComponent<WebCamTextureToMatHelper>().requestedDeviceName = newCam;
    this.GetComponent<WebCamTextureToMatHelper>().Initialize();
    Debug.Log("Setting NEW Camera (playerpref startCam) to: " + newCam);
    startCam = newCam;
}
```

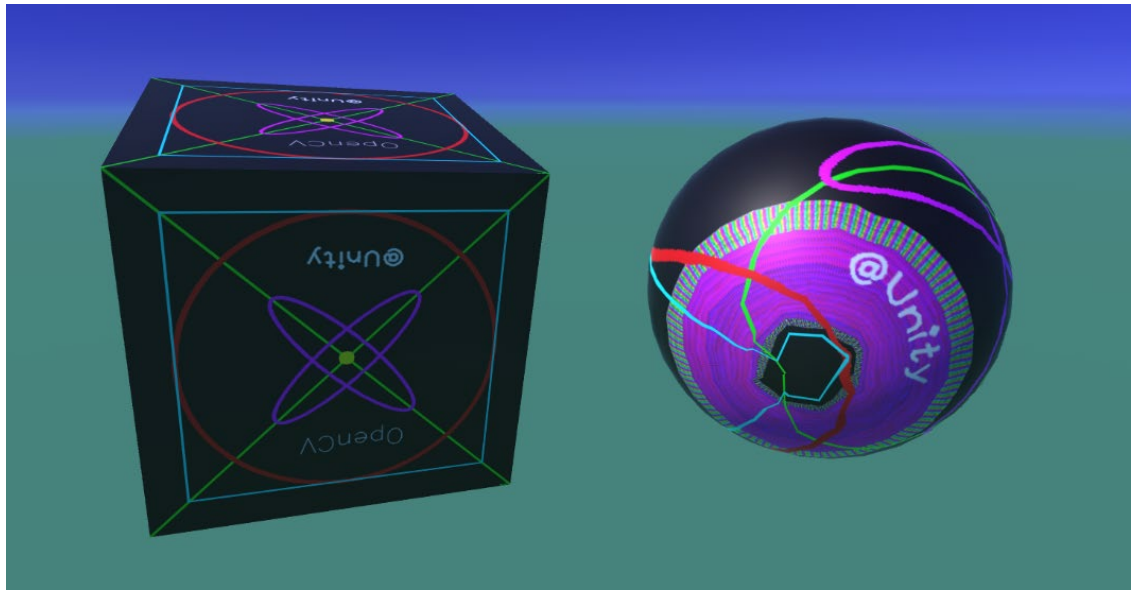
Abacus: → DMT4_FaceCtrl**3_scripts**\OpenCV**CamSwitchStore.cs**

Screenshot – Key „s“

```
public class FaceScreenshot : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown("s"))
            StartCoroutine(TakeScreenShot());
    }

    IEnumerator TakeScreenShot()
    {
        yield return new WaitForEndOfFrame();
        string filename = "face_screenshot_" +
            System.DateTime.Now.ToString("yyyyMMdd_HHmmss") + ".png";
        ScreenCapture.CaptureScreenshot(filename, 2);
        Debug.Log("Face Screenshot taken!");
    }
}
```

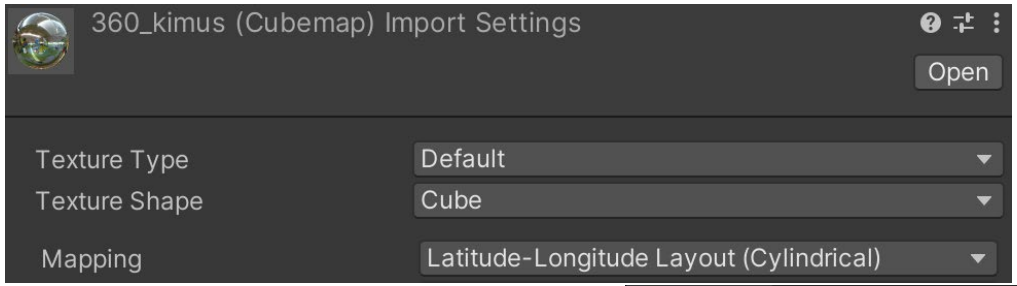
Only Drawing

- Renderer
- Texture
- Matrix

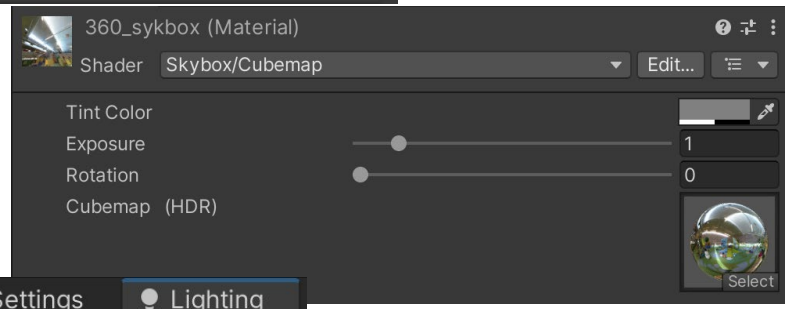


Equirectangular to Cubemaps

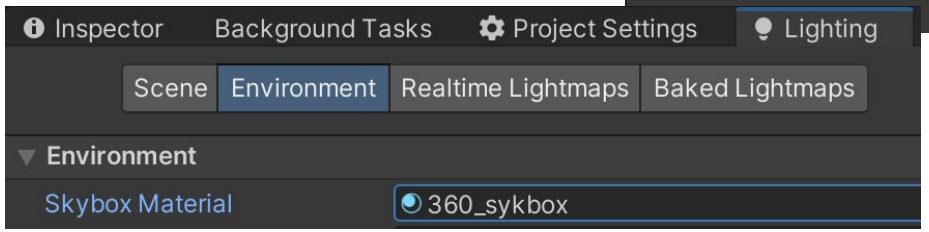
1. Texture 360 Grad Images

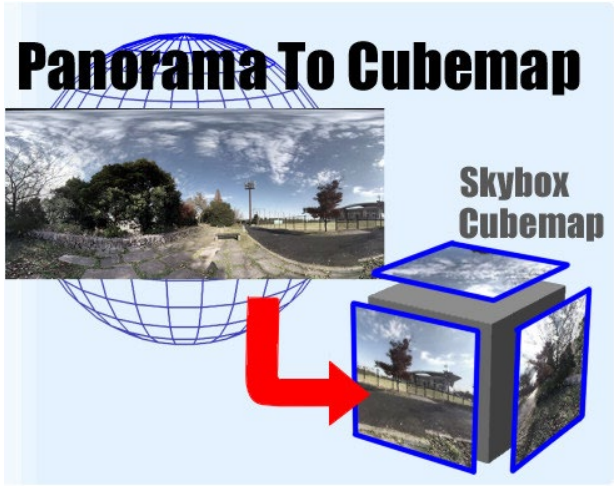


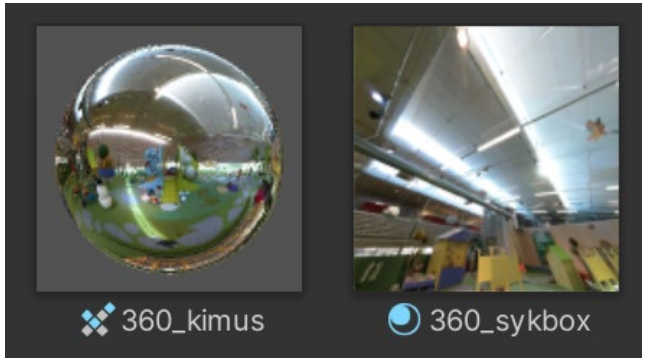
2. Material

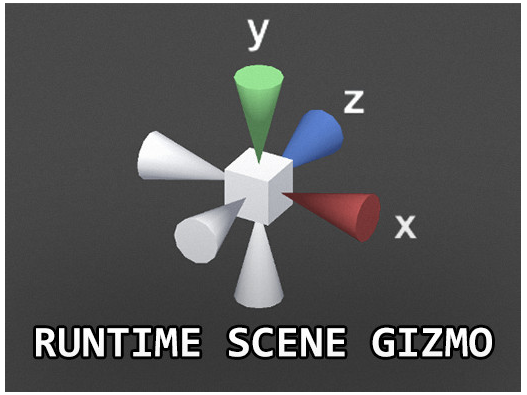


3. Lighting



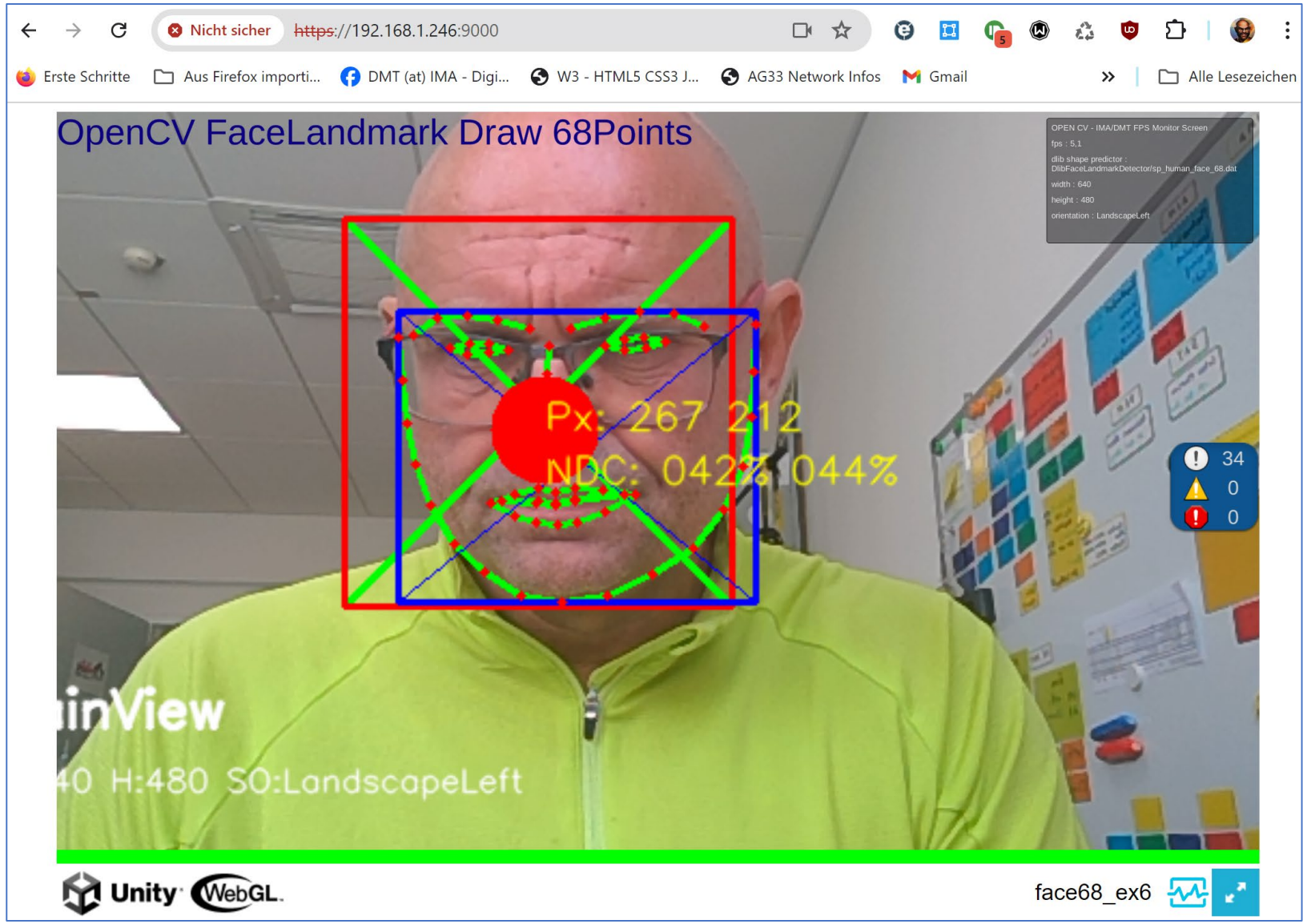






https://assetstore.unity.com/packages/tools/utilities/panorama-to-cubemap-13616

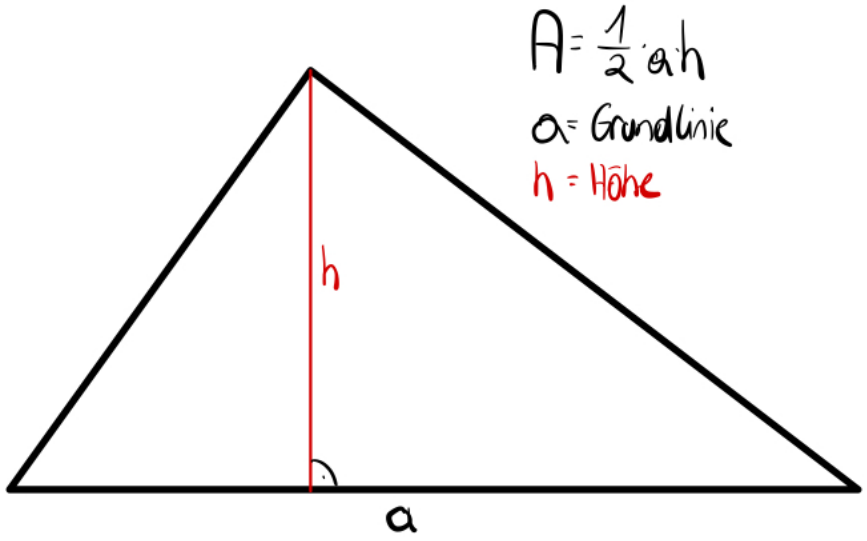
WebGL



- https://192.168.1.246:9000/

Fläche

Flächeninhalt eines Dreiecks



Umsetzung im Programmcode jedoch mittels Gaußscher Trapezformel (*shoelace formula*):

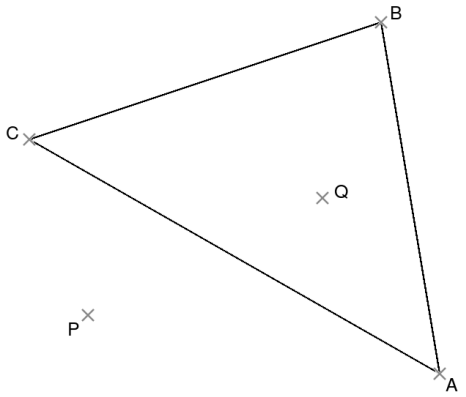
$$2A = \sum_{i=1}^{n} x_i \left(y_{i+1} - y_{i-1}\right)$$

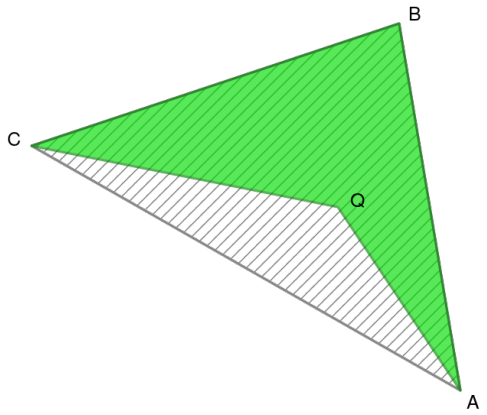
```
private double calcTriangleArea(Point p1, Point p2, Point p3)
{
    double area = Math.Abs((p1.x * (p2.y - p3.y) + p2.x * (p3.y - p1.y) + p3.x * (p1.y - p2.y)) / 2);
    return area;
}
```

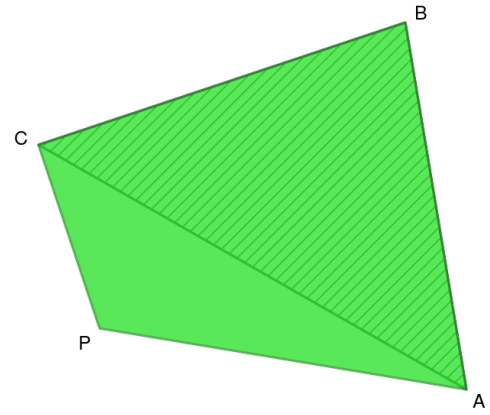
Liegt Punkt P in Dreieck ABC?

```
// Punkt in Dreieck ################################################

Point punkt = new Point(640, 360);
Imgproc.circle(imgMat, punkt, 5, red, -1);
Imgproc.line(imgMat, points[0], points[16], red);
Imgproc.line(imgMat, points[16], points[8], red);
Imgproc.line(imgMat, points[8], points[0], red);
Imgproc.putText(imgMat, ("Punkt im Dreieck: " + pointInTriangle(punkt, new List<Point> { points[0],points[16],points[8]}).ToString()), new Point(100, 220), Imgproc.FONT_HERSHEY_SIMPLEX, 0.5, black, 1, Imgproc.LINE_AA, false);
```

```
private bool pointInTriangle(Point P, List<Point> triangle)
{
    Point A = triangle[0];
    Point B = triangle[1];
    Point C = triangle[2];
    List<double> areas = new List<double> {
        calcTriangleArea(A,B,P) + calcTriangleArea(A,P,C),
        calcTriangleArea(A,B,P) + calcTriangleArea(B,P,C),
        calcTriangleArea(A,P,C) + calcTriangleArea(P,B,C)
    };

    float max1 = Mathf.Max((float)areas[0], (float)areas[1]);
    float max2 = Mathf.Max((float)areas[1], (float)areas[2]);
    float max = Mathf.Max(max1, max2);

    if (max - (float)calcTriangleArea(A, B, C) < 0)
    {
        return true;
    } else
    {
        return false;
    }
}
```







https://prlbr.de/2014/liegt-der-punkt-im-dreieck/

Flächen / FaceParts definieren

```
List<List<Point>> mouth = new List<List<Point>> {
    new List<Point> {points[48],points[49],points[59]},
    new List<Point> {points[49],points[50],points[59]},
    new List<Point> {points[59],points[50],points[58]},
    new List<Point> {points[50],points[58],points[51]},
    new List<Point> {points[58],points[51],points[57]},
    new List<Point> {points[51],points[52],points[57]},
    new List<Point> {points[52],points[56],points[57]},
    new List<Point> {points[52],points[56],points[55]},
    new List<Point> {points[55],points[52],points[53]},
    new List<Point> {points[53],points[55],points[54]}
};

drawTriangles(imgMat, mouth, white);
```

```
private void drawTriangles(Mat imgMat, List<List<Point>> region, Scalar color)
{
    foreach (List<Point> triangle in region)
    {
        Imgproc.line(imgMat, triangle[0], triangle[1], color);
        Imgproc.line(imgMat, triangle[1], triangle[2], color);
        Imgproc.line(imgMat, triangle[2], triangle[0], color);
    }
}
```

Beispiel: Punkt in Fläche

```
private bool pointInArea(Point P, List<List<Point>> triangles)
{
    foreach (List<Point> triangle in triangles)
    {
        if (pointInTriangle(P, triangle))
        {
            return true;
        }
    }
    return false;
}
```

```
// Punkt in Mund ##################################################

Imgproc.putText(imgMat, ("Punkt im Mund: " + pointInArea(punkt, mouth).ToString()), new Point(100, 240), Imgproc.FONT_HERSHEY_SIMPLEX, 0.5, black, 1, Imgproc.LINE_AA, false);
```

Beispiel: Endresultat

```
// Endresultat ######################################################

var faceParts = new Dictionary<string, List<List<Point>>>
{
    {"mouth",mouth}, {"nose",nose}, {"chin",chin}, {"eyeL",eyeL}, {"eyeR",eyeR}, {"cheekL",cheekL}, {"cheekR",cheekR}
};

foreach (KeyValuePair<string, List<List<Point>>> facePart in faceParts)
{
    //drawTriangles(imgMat, facePart.Value, white);
    if (pointInArea(punkt, facePart.Value))
    {
        Imgproc.putText(imgMat, ("Punkt in: " + facePart.Key), new Point(100, 260), Imgproc.FONT_HERSHEY_SIMPLEX, 0.5, black, 1, Imgproc.LINE_AA, false);
    }
}
```

EOF