hochschule aschaffenburg
university of applied sciences

Beispiel: PanoViewer

Computergrafik: Realitätserfassung

Block I: Realitätserfassung

Image/Video Capture; Image/Video Creation

- **Lektion 1: Computergrafik-Projekt „PanoViewer“**
  - Einstieg: Unity Grundlagen
  - 360 Grad Panorama erfassen, verarbeiten und darstellen
    - Theorie: 360 Grad Bilder umrechnen
    - Code Review: Umrechnungsprogramm in C#
  - Texturen, UV Map

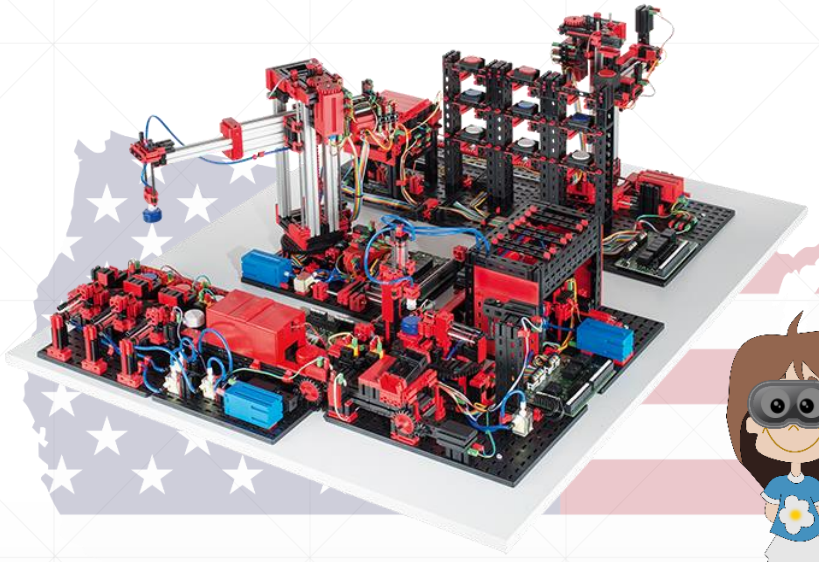
Literaturempfehlungen

- Anwendungsszenario
- Grundlagenschulung
- Hardware Tests

Projektspezifikation

Computergrafik

Anwendungsszenario



Augmented Reality

Collaboration in
Virtual Space

Virtual Reality

Voraussetzungen

Grundlagen-Schulung: Unity

- Benutzeroberfläche
- 3D Objekte erstellen
  - Eigenschaften (Transform, Mesh)
  - Erscheinung (Texturen, Materialien)
- Lichtquelle, Kamera
- Standard Assets, FPS
- Deployment: Stand-alone Programm erzeugen

Hardware Check

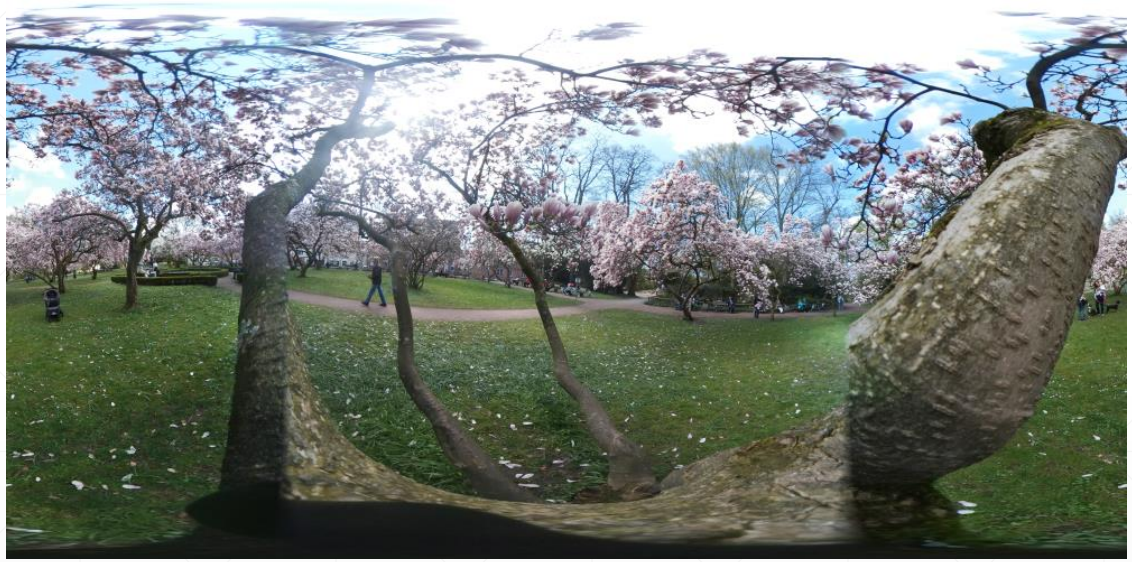
- Abbildungen, Projektionen

Theorie

Computergrafik

Equirektanguläre Projektion („Plattkarte)

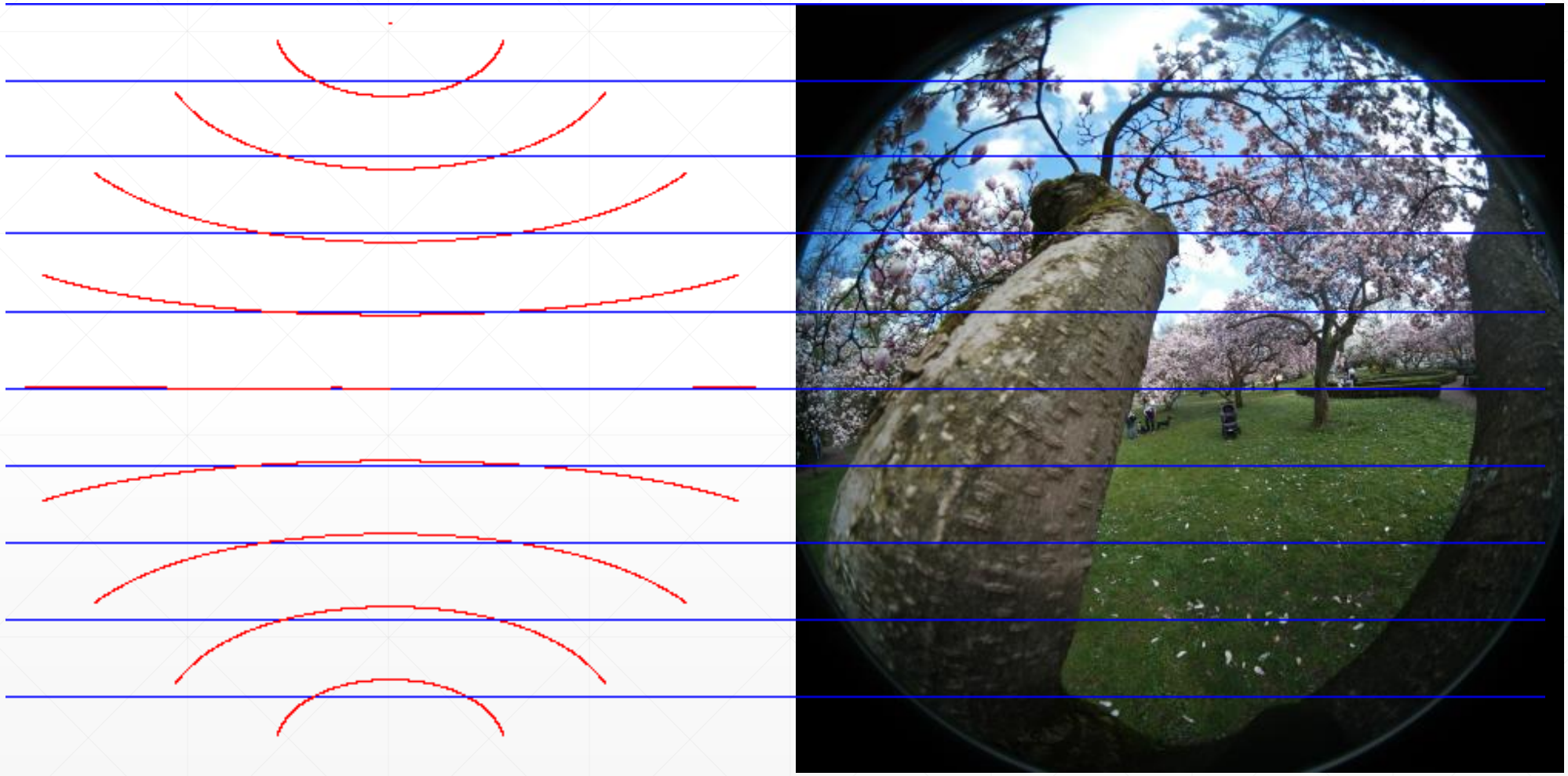
2:1



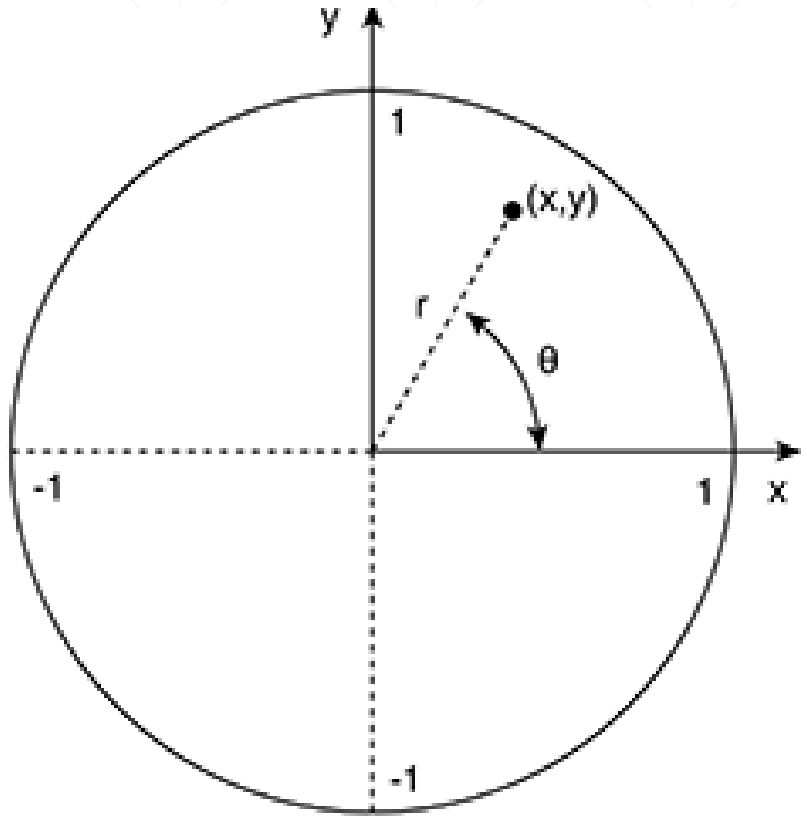
Raw:
Jedes Pixel auf das eine Position im Processed Image abbilden.
Injektive Abbildung, Interpolation.

Processed:
Für jedes Pixel die beste Zuordnung im Raw Image schätzen.
Surjektive Abbildung.

„Equi-“ = Abstandstreu, „-rectangle“ = Sphärisch nach flach



2D Fischauge nach 3D Raumvektor



Normalised fisheye coordinates

$r = 2 \operatorname{atan2}(\sqrt{(P_x^2+P_z^2)}, P_y) / \text{aperture}$

$\theta = \operatorname{atan2}(P_z, P_x)$

$x = r \cos(\theta)$

$y = r \sin(\theta)$

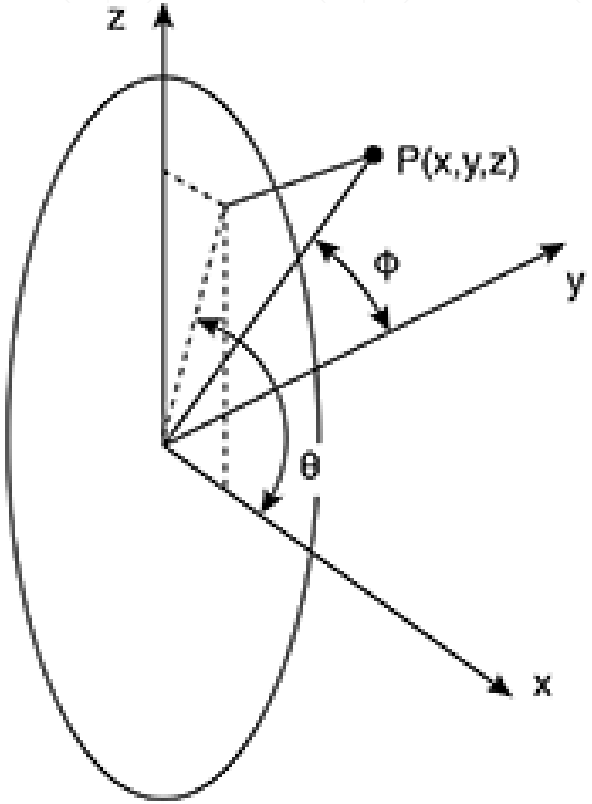
3D vector to 2D fisheye

2D fisheye to 3D vector

$\Phi = r \cdot \text{aperture} / 2$

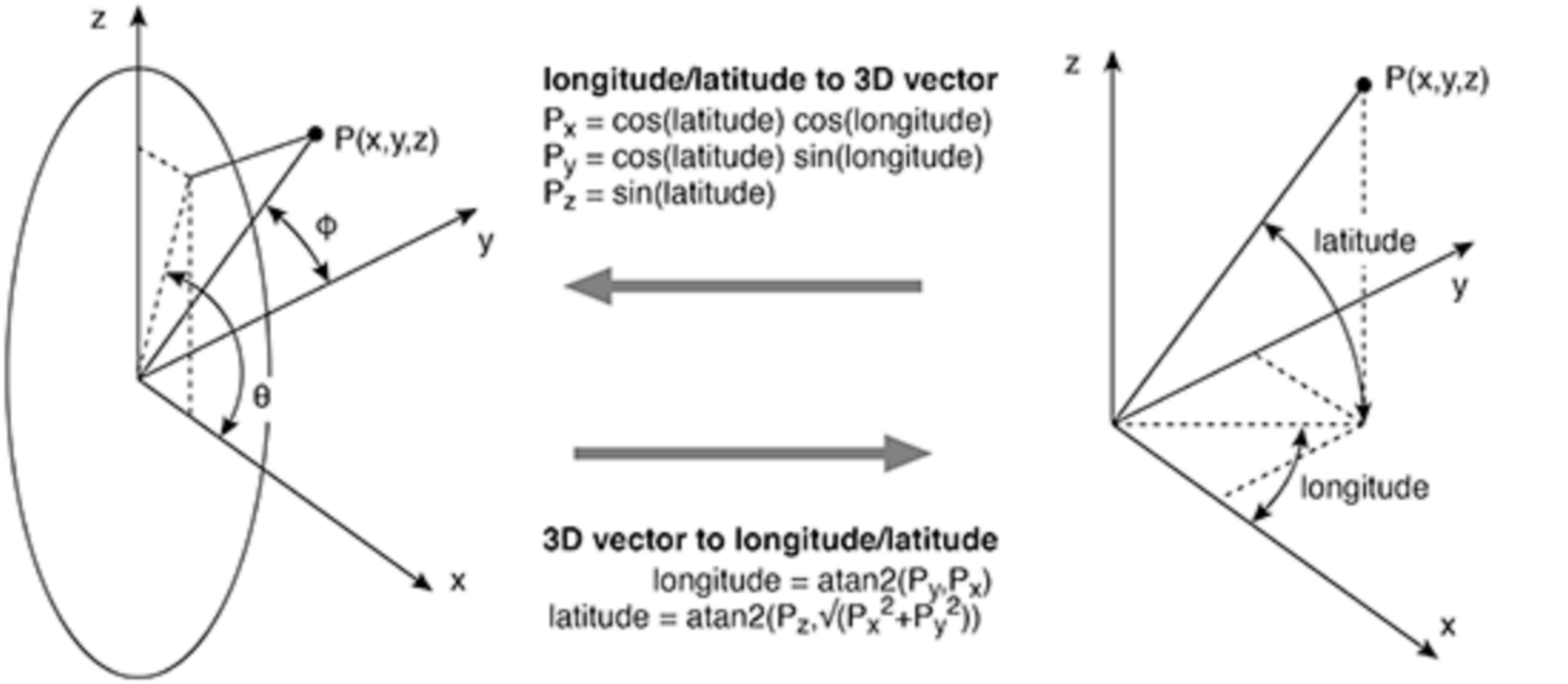
$\theta = \operatorname{atan2}(y,x)$

Px, Py, Pz ?



(Quelle: http://paulbourke.net/dome/dualfish2sphere/)

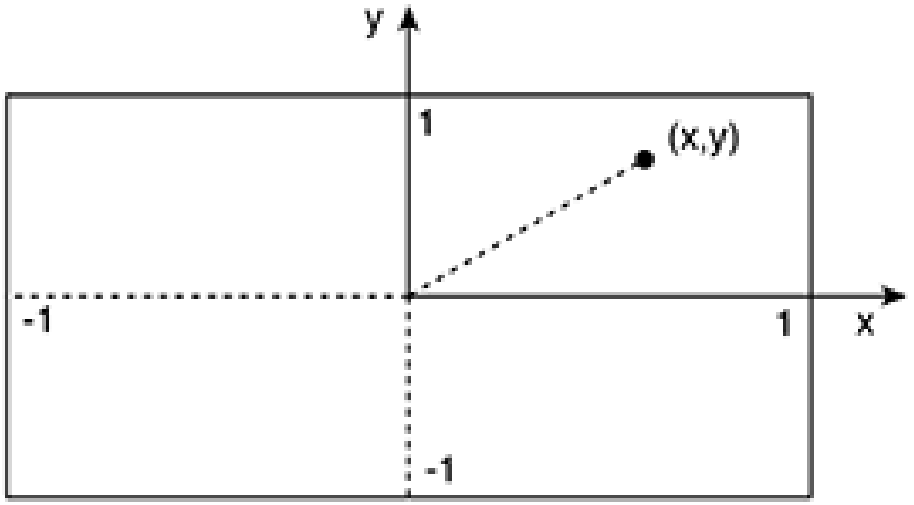
3D Raumvektor nach Breiten-/Längengrade



(Quelle: http://paulbourke.net/dome/dualfish2sphere/)

Breiten-/Längengrade nach „Landkarte“

Normalised Equirectangular coordinates

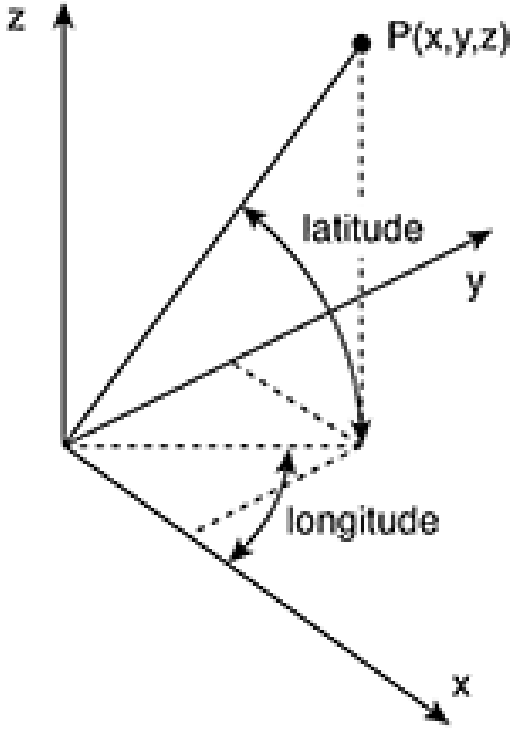


x = longitude / π
y = 2 latitude / π

3D vector to 2D equirectangular

2D equirectangular to 3D vector

longitude = x π
latitude = y π / 2
P_x = cos(latitude) cos(longitude)
P_y = cos(latitude) sin(longitude)
P_z = sin(latitude)



(Quelle: http://paulbourke.net/dome/dualfish2sphere/)

Umrechung 2x 190 Grad Bilder

Transformation

+ Stitching
+ Verzeichnungen
+ Farbabgleich

Algorithmus

```
private static double[] ReverseCalculation(double x0, double y0)
{
    // 2d equirectangular
    var longitude = x0 * Math.PI;
    var latitude = y0 * Math.PI / 2;

    // 3d coords on unit sphere
    var px = Math.Cos(latitude) * Math.Cos(longitude);
    var py = Math.Cos(latitude) * Math.Sin(longitude);
    var pz = Math.Sin(latitude);

    // 2d fisheye polar coords
    var r = 2 * Math.Atan2(Math.Sqrt(px * px + pz * pz), py) / _aperture;
    var theta = Math.Atan2(pz, px);

    // return 2d fisheye coords as integer
    return new[] {(r * Math.Cos(theta)), (r * Math.Sin(theta))};
}
```

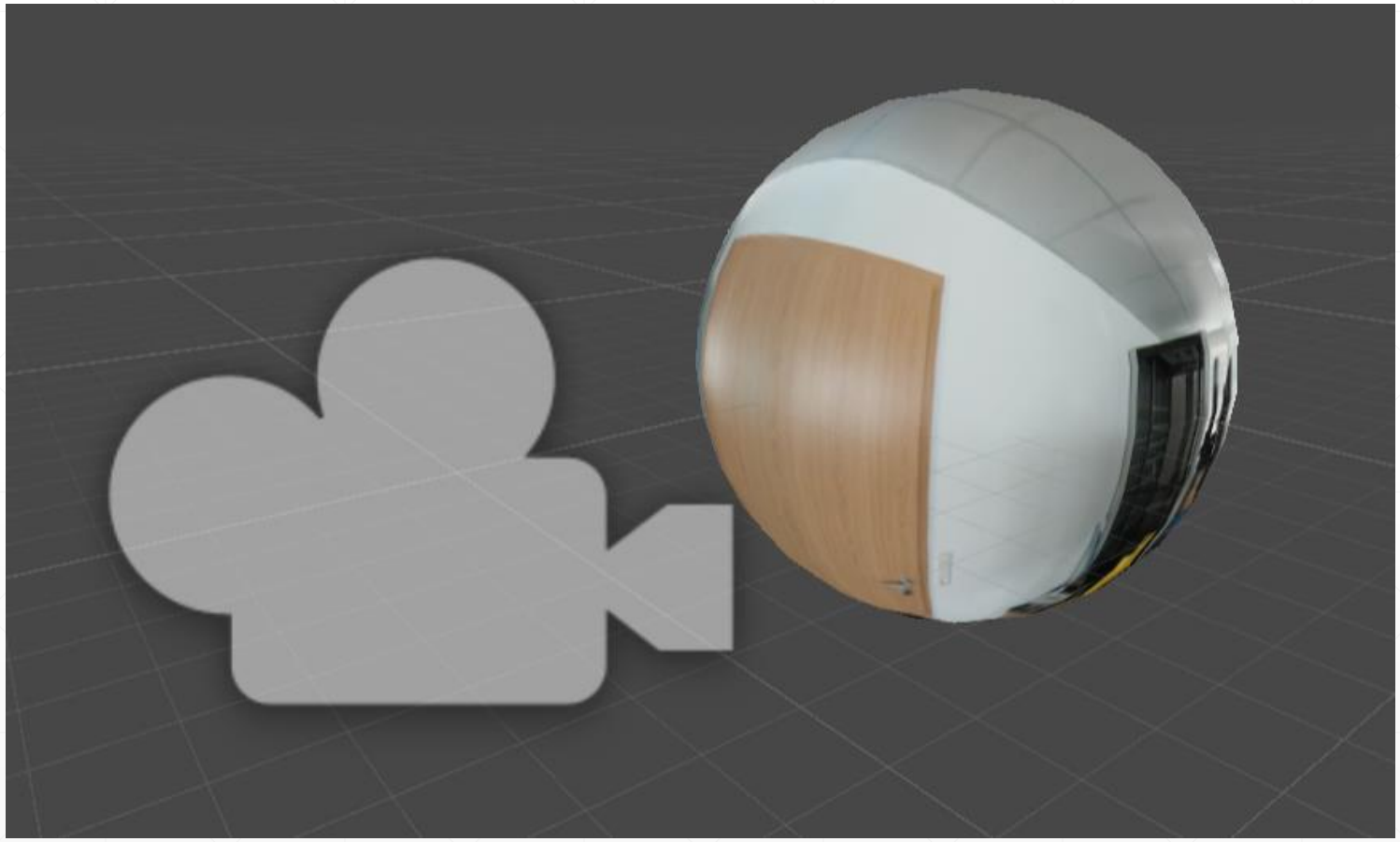
Do it!

- Methode 1: Projektion auf Kugel
- Einschub: Texturen und Beobachter
- Methode 2: Skybox
- Deployment auf VR (Android)

Implementierung

Computergrafik

PanoViewerSphere: Projektion auf Kugel

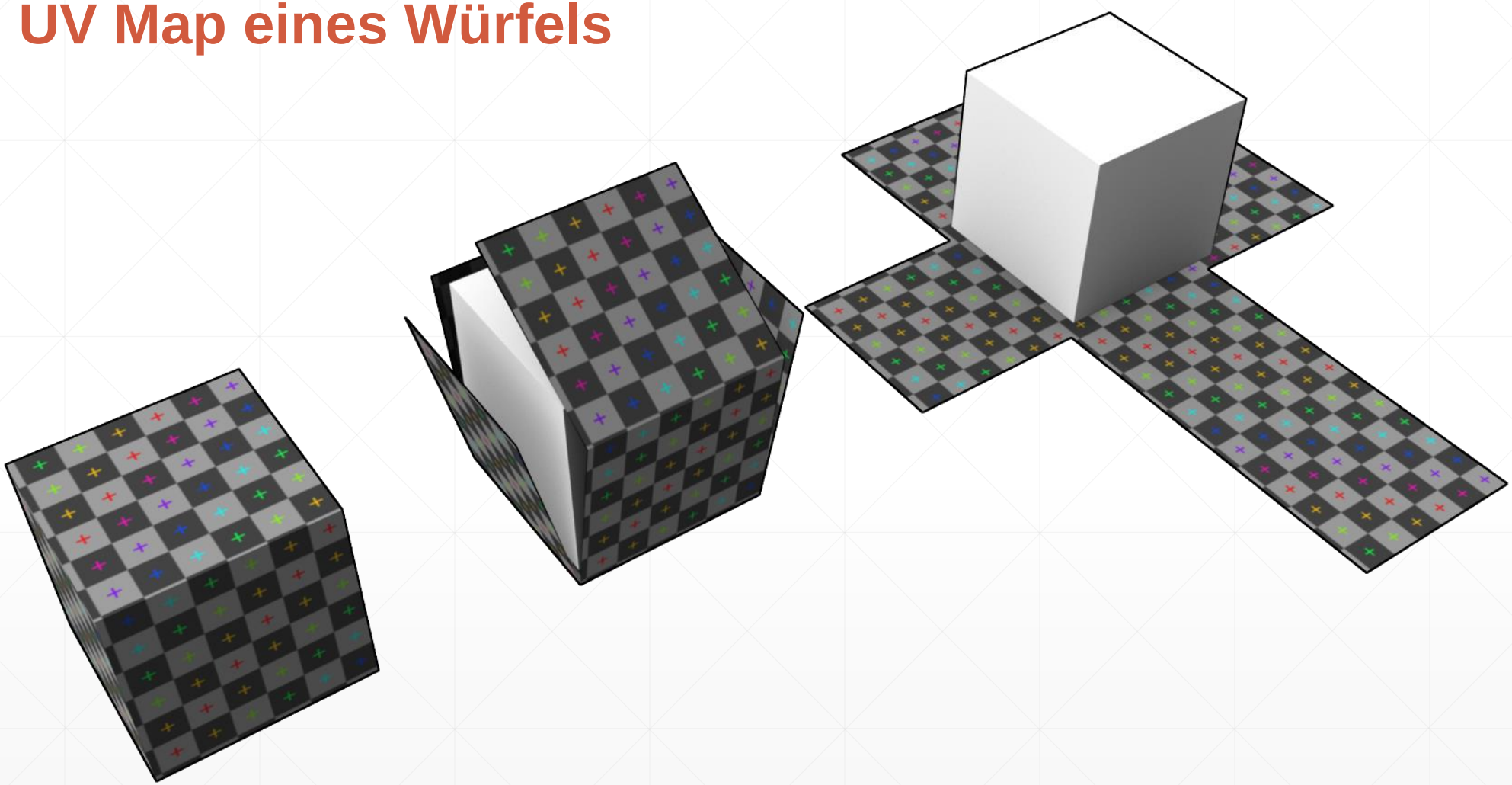


Einschub: Texturen und Beobachter

- Woher weiß die Kugel, wie sie das Bild anzeigen muss?
  - Texturen
  - UV Map
- Warum wird das Bild nur auf der Außenseite angezeigt?
- Was ist „die Außenseite“?

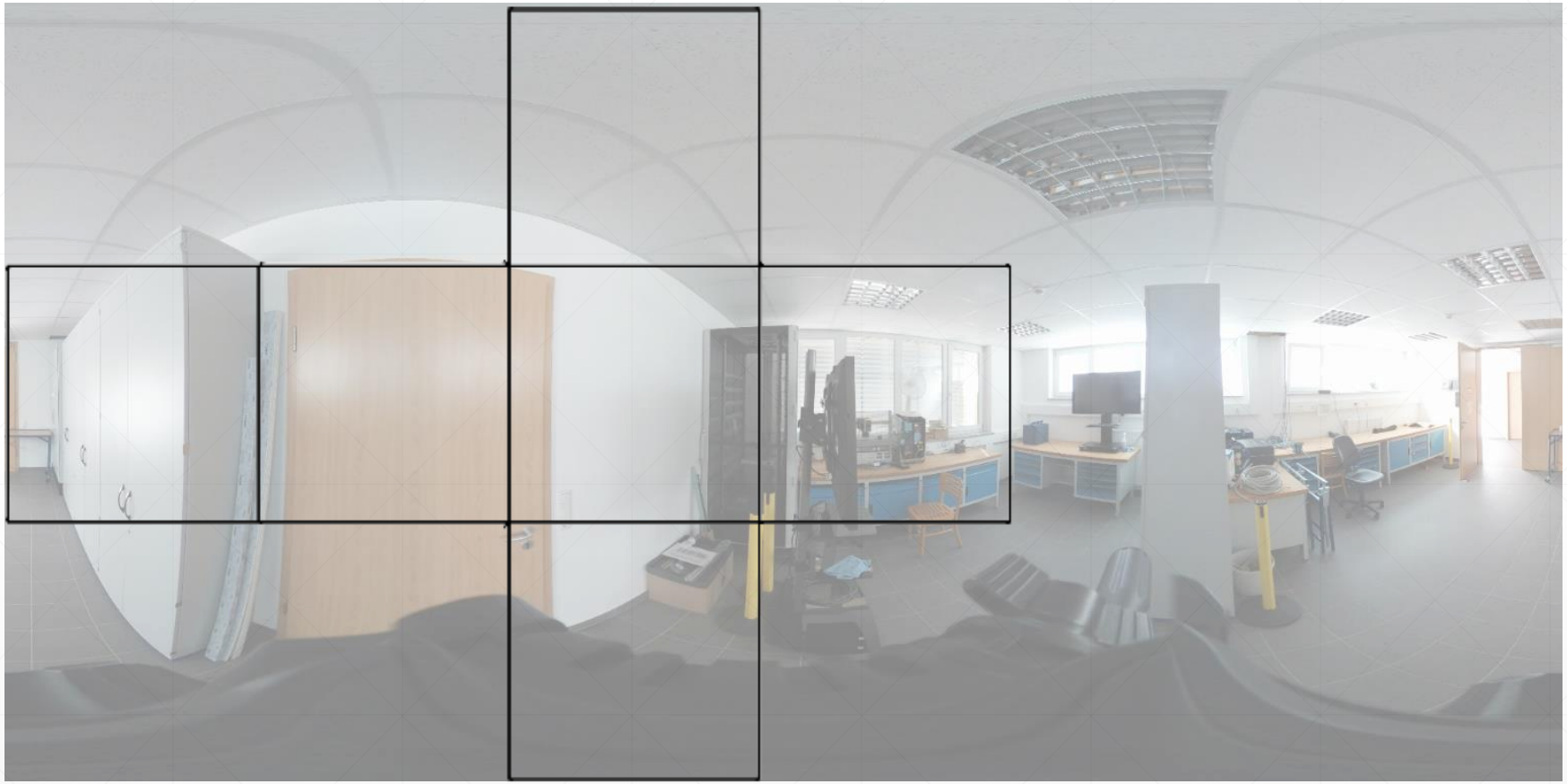
Bilder sind in Unity Texturen

UV Map eines Würfels

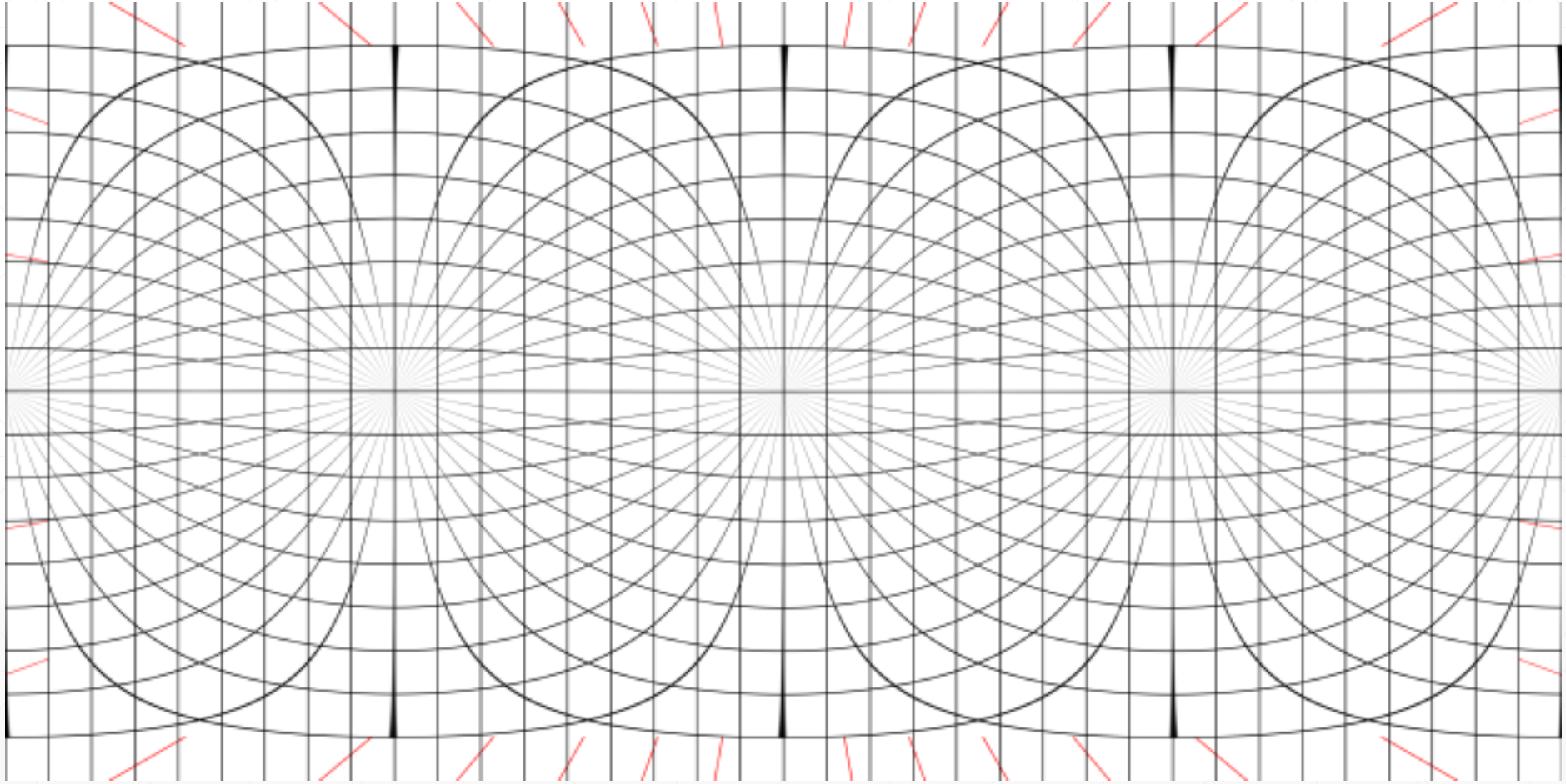


(Quelle: Wikipedia)

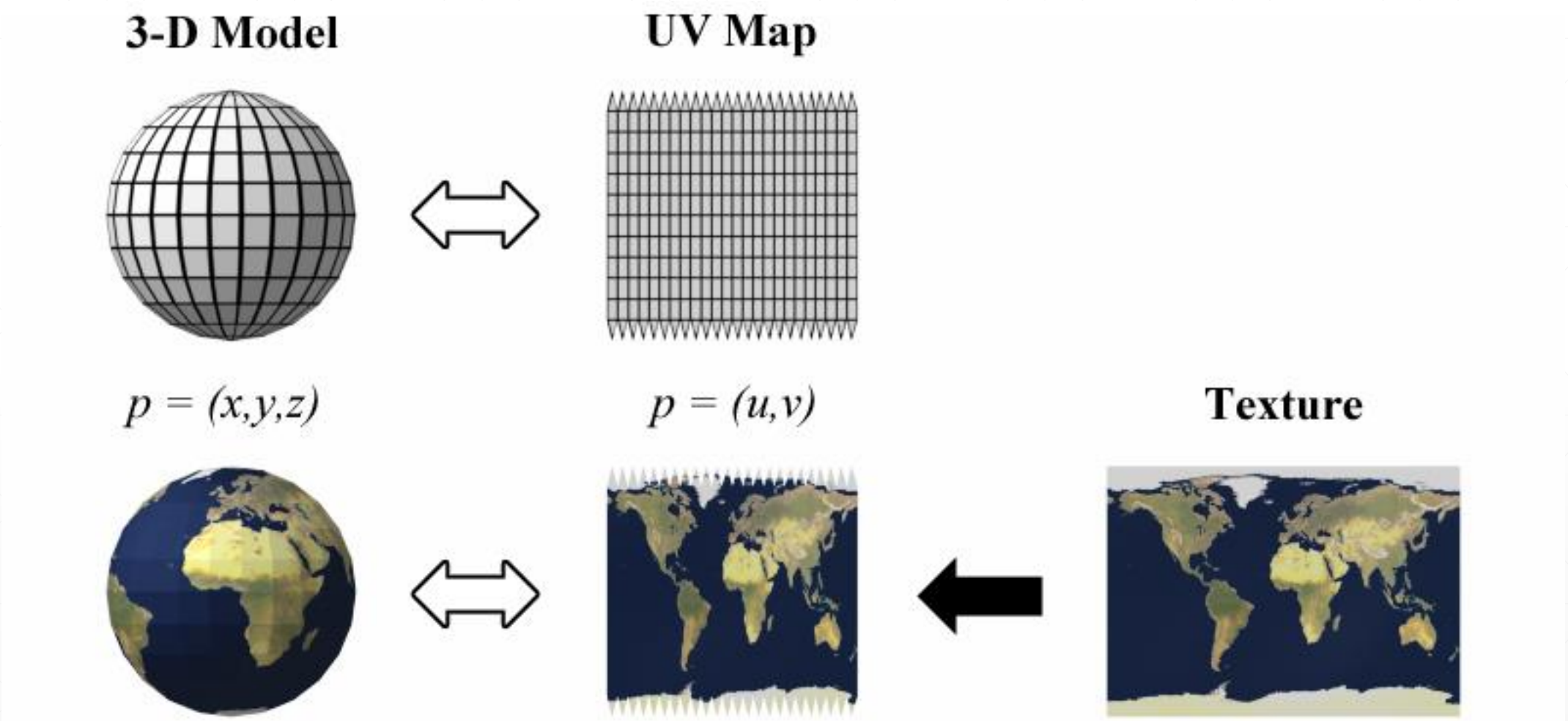
Unser Beispiel auf einem Würfel



UV Map einer Kugel



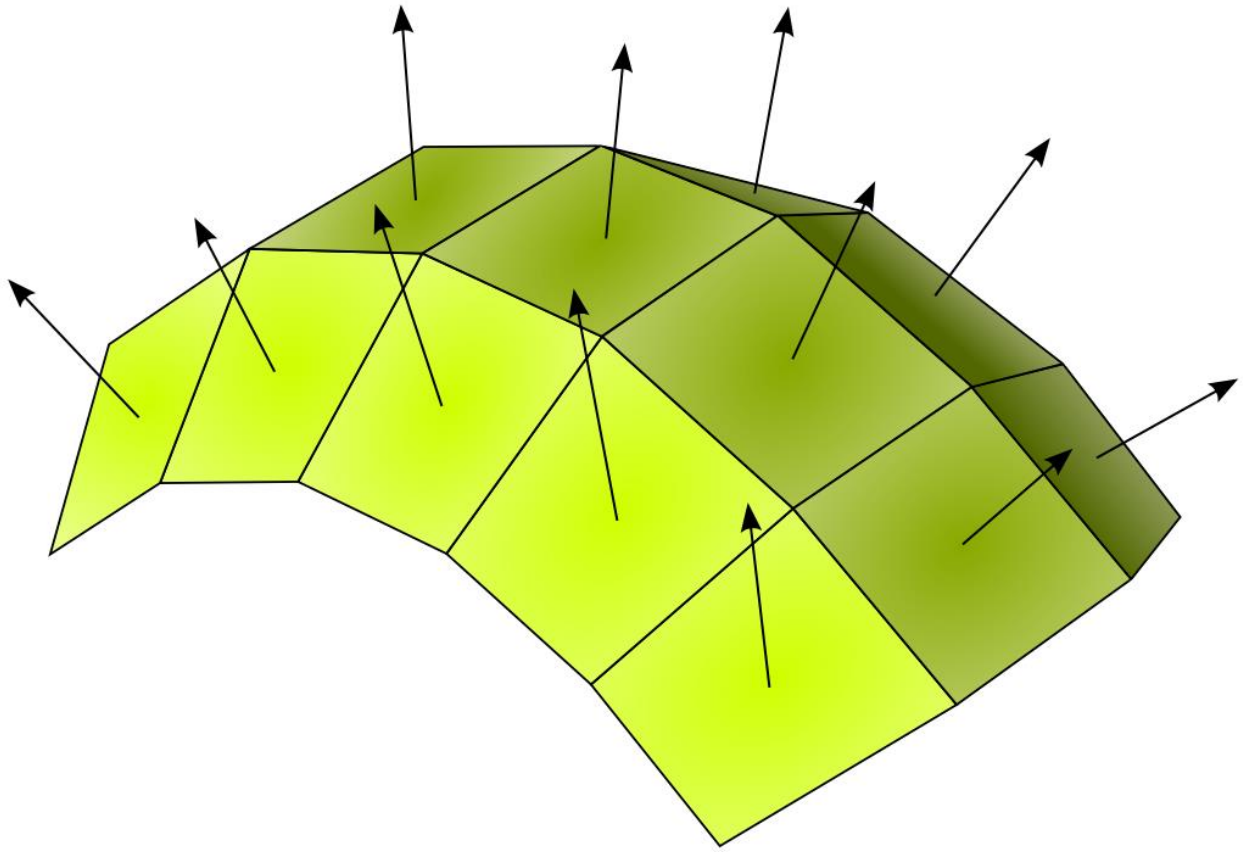
UV Map einer Polygon-Kugel (Unity)



Innenseite / Außenseite

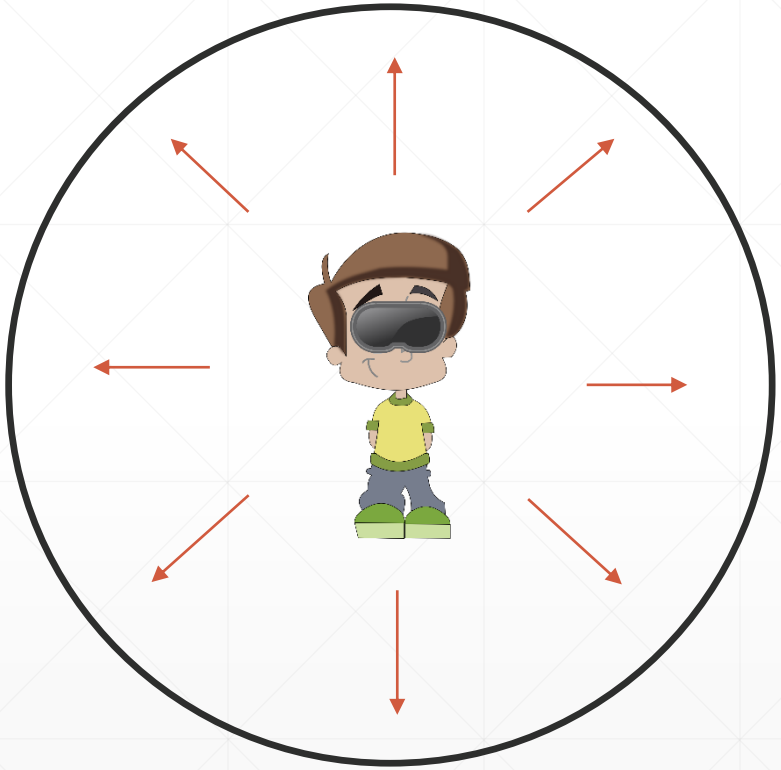
„Auf der anderen Seite sein“ =

Textur wird nicht verarbeitet.



(Quelle: Wikipedia)

PanoViewerSkybox: VR Szenario



Option 1:

Normalenvektoren umkehren

+ Umlaufsinn der Fragmente umkehren

PanoViewerSkybox: VR Szenario

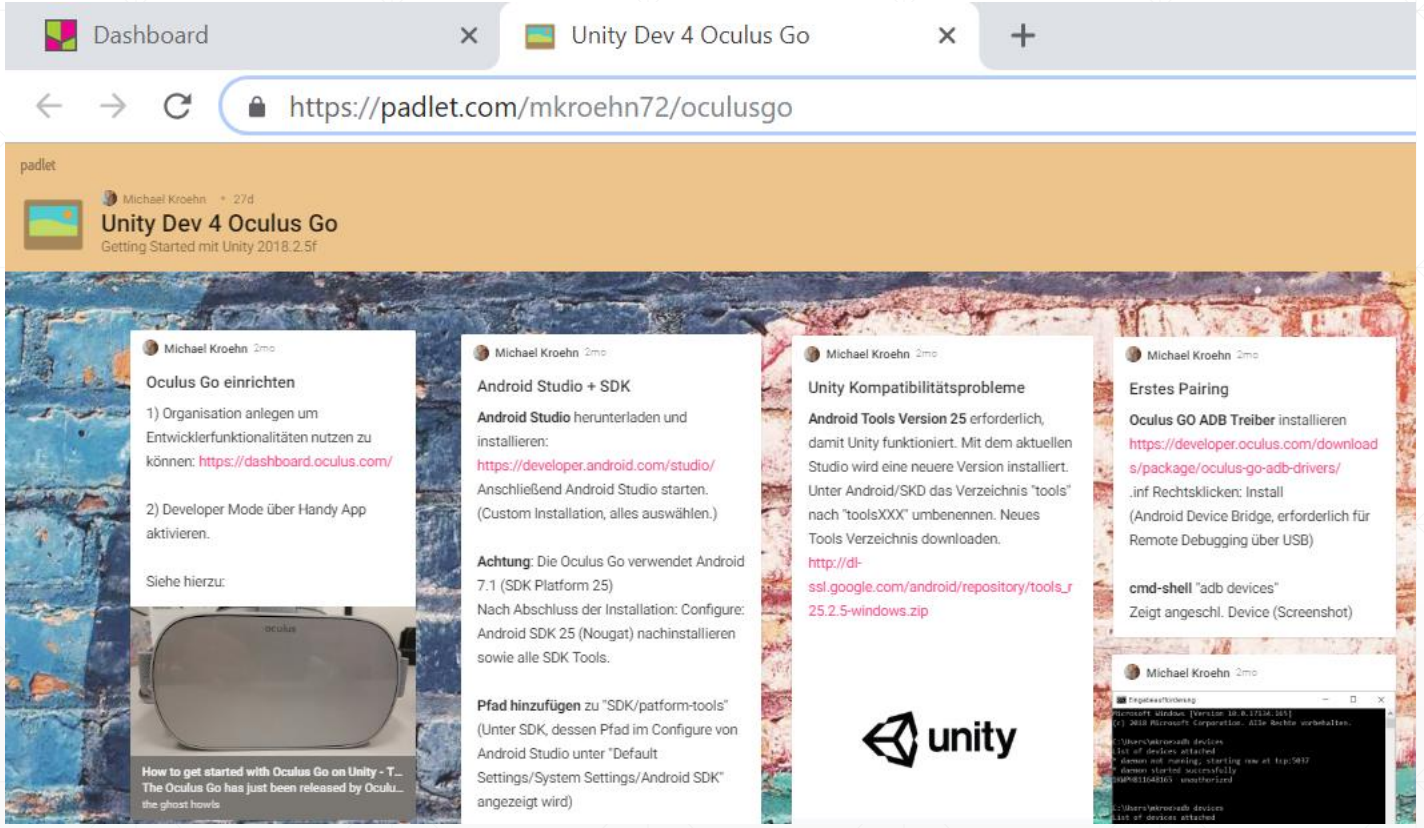
```
// Reverse triangles
var triangles = meshNormal.triangles;
for (int i = 0; i < triangles.Length; i = i + 3)
{
    var tmp = triangles[i];
    triangles[i] = triangles[i + 2];
    triangles[i + 2] = tmp;
}

meshFilterInverted.sharedMesh.triangles = triangles;

// Reverse Normals
var normals = meshNormal.normals;
for (int i = 0; i < normals.Length; i++)
{
    normals[i] = -normals[i];
}
```

Deployment auf Android, insbesondere VR

- **Herausforderungen**
  - Versionskonflikte
  - Entwicklermodus
  - Android Device Bridge



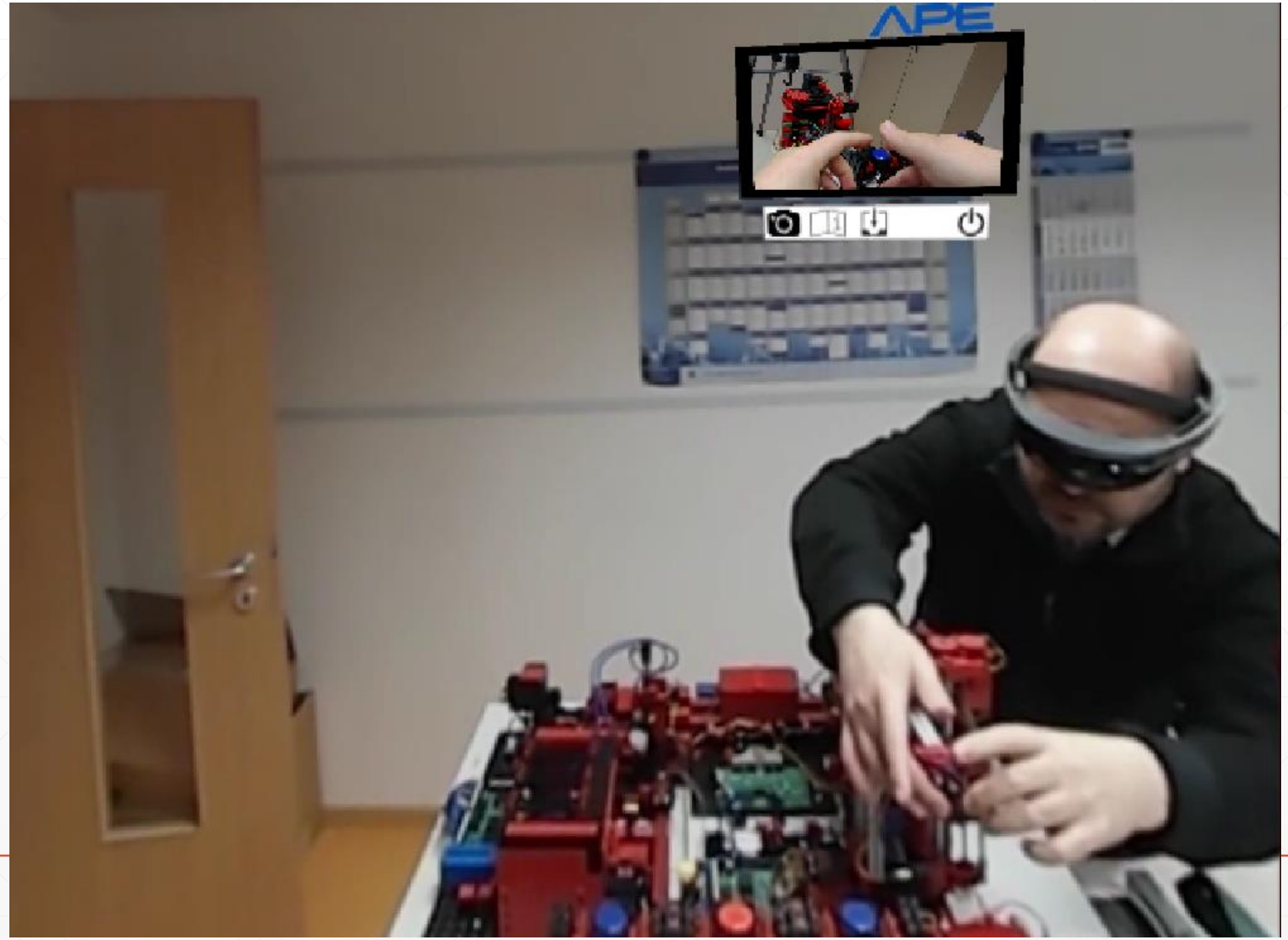
https://padlet.com/mkroehn72/oculusgo

- Unity Einstieg
- Transformationen mit C#
- Begriffe: Texturen, UV Maps
- Projekt: PanoViewer

Fazit

Computergrafik

Projektabschluss

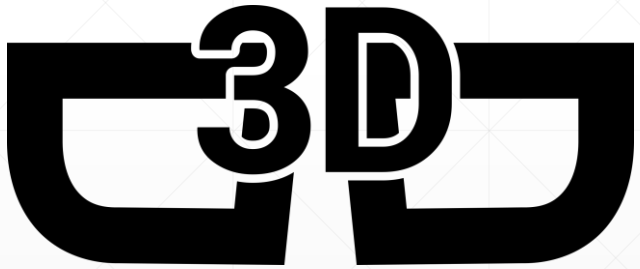


Herausforderungen

Reduktion
Kompression

Netzwerk

Umrechnung (>30fps)



Performance
(„mobile shader“)