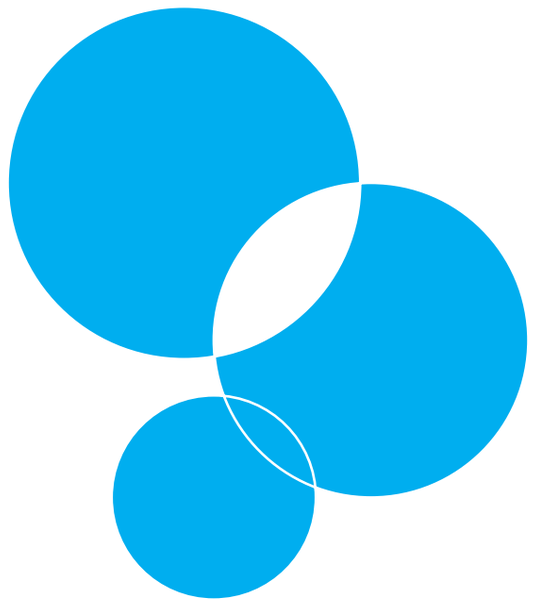


Interaktives Fluidsystem

Übersicht

- Einführung in das Thema
- Inspiration und Ansatz
- Umsetzung

Einführung

- Flüssigkeiten Interaktion
  bei angemessener Performance

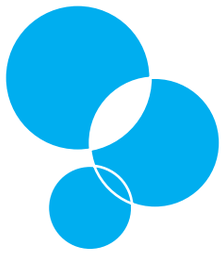
- Funktionalität > Realismus
  // Optik haben andere Übernommen

- Inspiration: WindWaker

Jump
Put Away
FREE
089

Inspiration

- Bäche haben keinen echten „Fluss“
- wirken nur durch eine animierte Textur bewegt
- Ziel:
  Performante Lösung von fließendem Wasser / Flüssigkeit mit (erhöhter) Reaktionsfähigkeit
  >> ausbaufähiges System

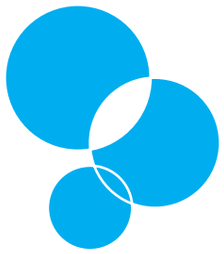


Ansatz

- Screen Space Fluid Rendering for Games
  // Simon Green
  // Nvidia

- Particle Based Fluid Simulation

Quelle



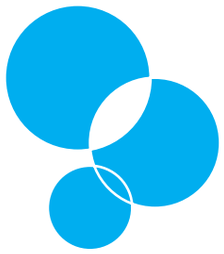
Ansatz

- Emitter für Partikel
- Partikel mit Lifetime und Interpartikel-Kollision
- Shader, awesome!

Showcase

Umsetzung

- PartikelSystem
- Shader, aww Yeah!
- Komposition
- Weiterführende Arbeit



PartikelSystem

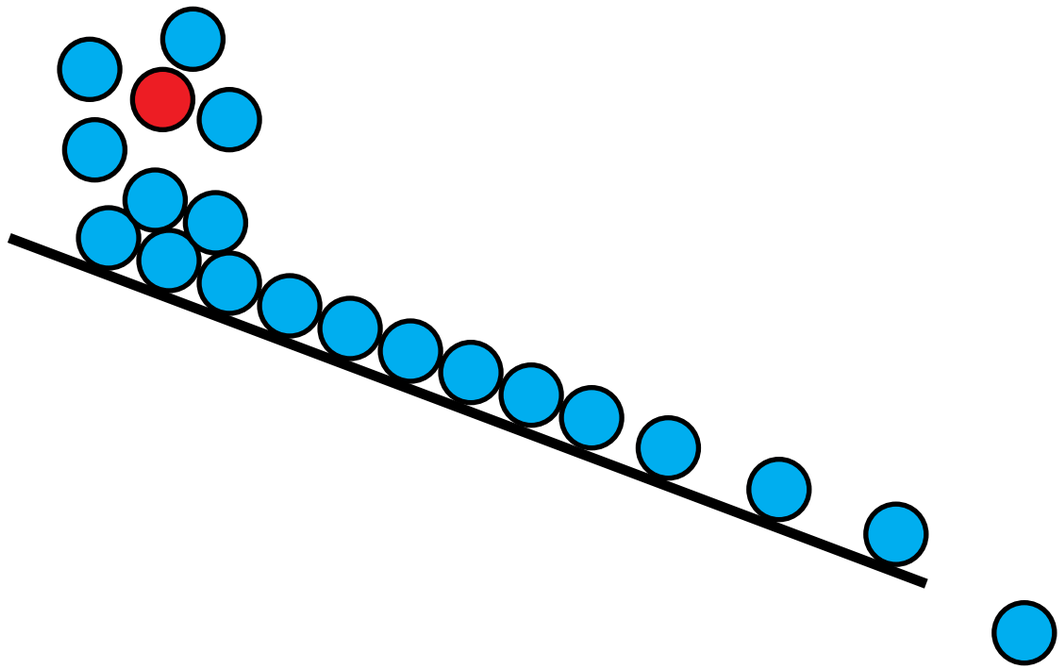
- Emitter

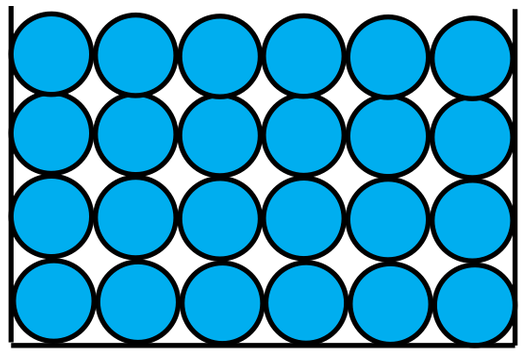
- Prefab mit Material und weiteren Eigenschaften wie Lifetime, Verhalten,

Performance?
Kollision: n^2

- Kugeln -> {Magie} -> Flüssigkeit? ¯_(ツ)_/¯

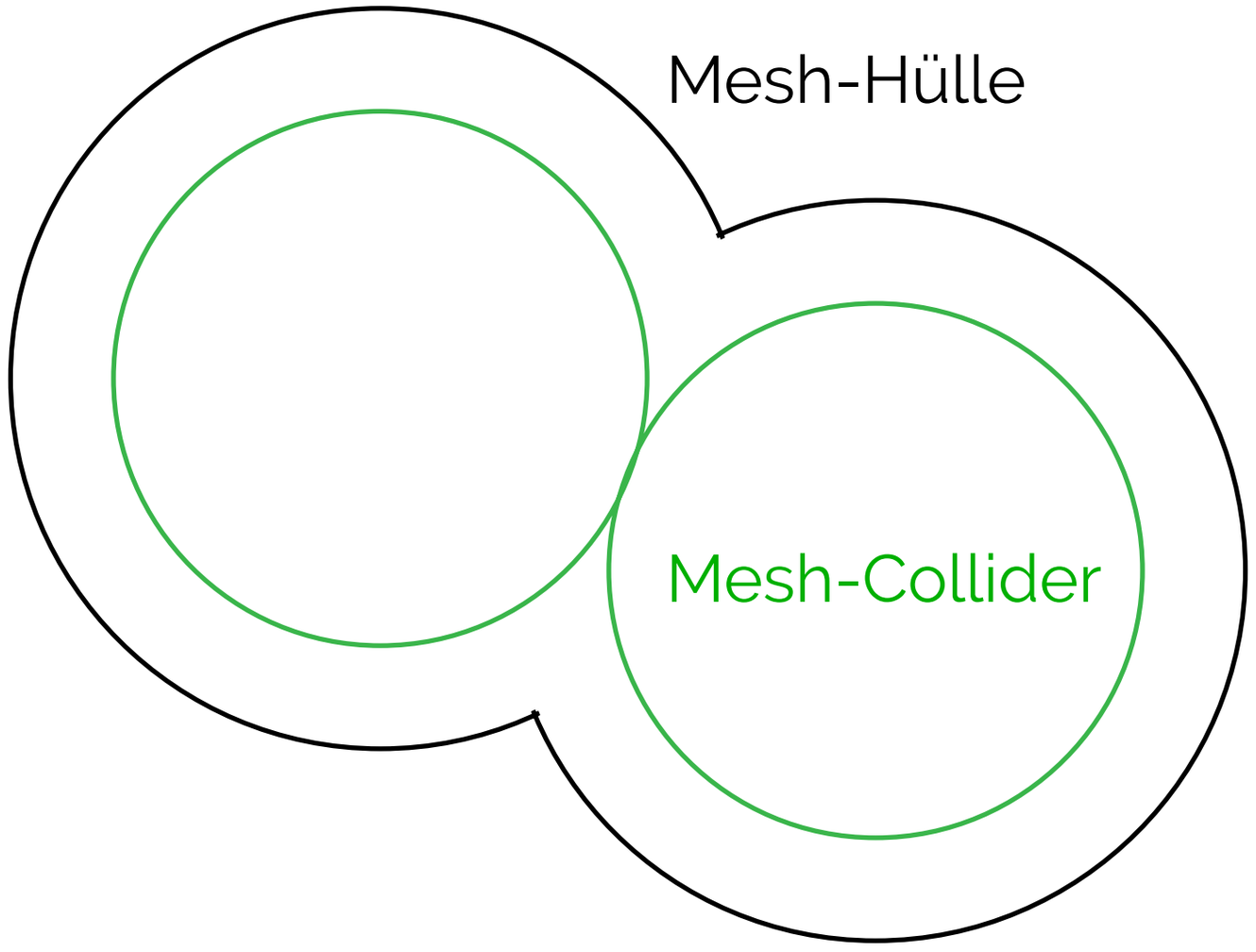
PartikelSystem



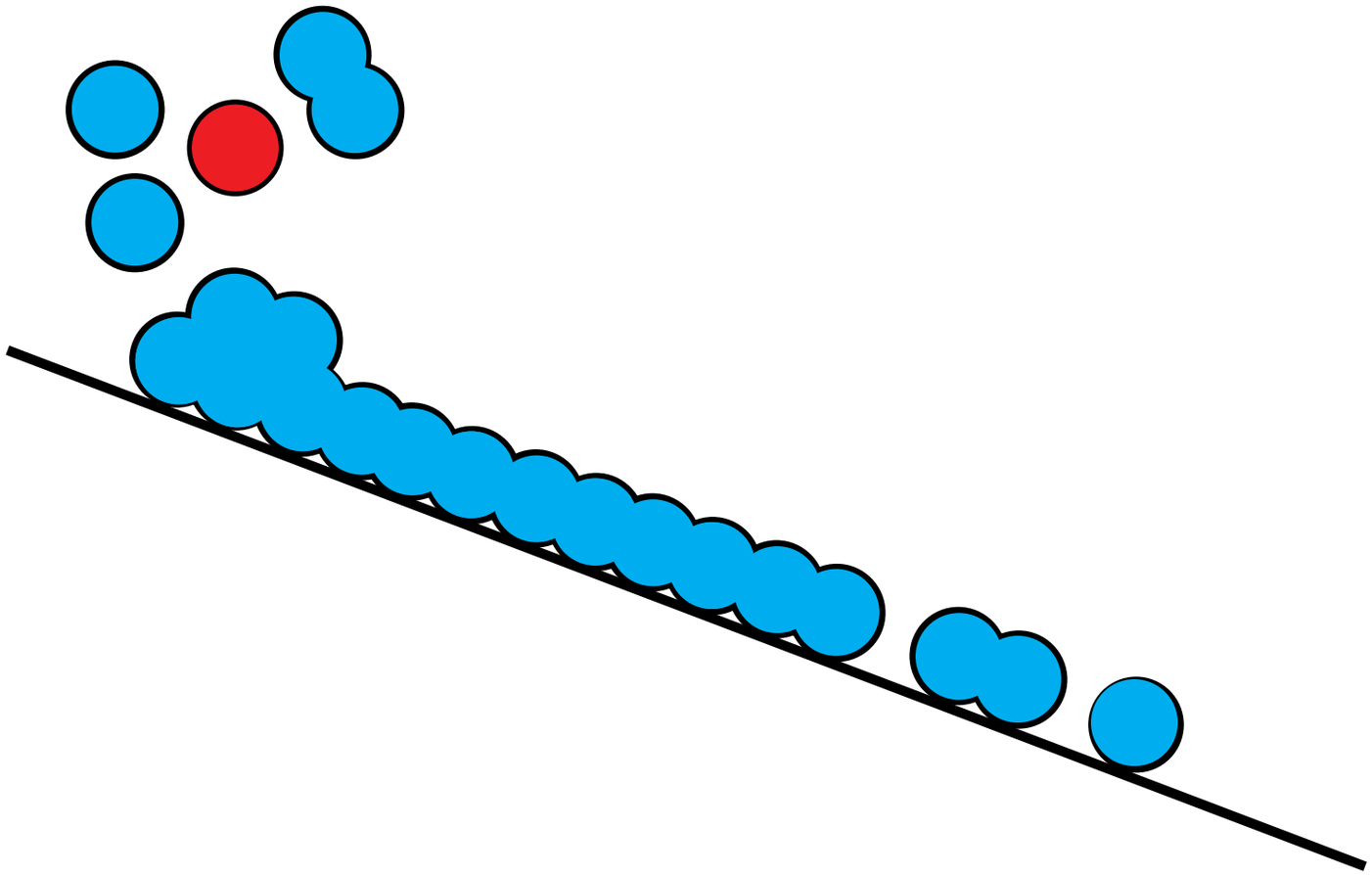


100 Spheres = 100 Volumeneinheiten

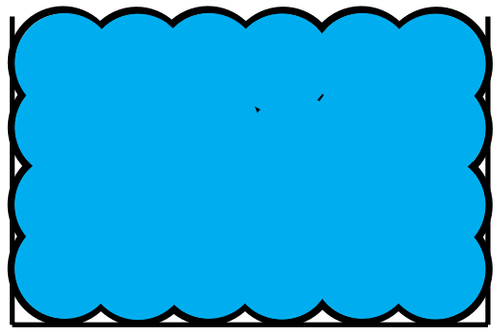
PartikelSystem



PartikelSystem

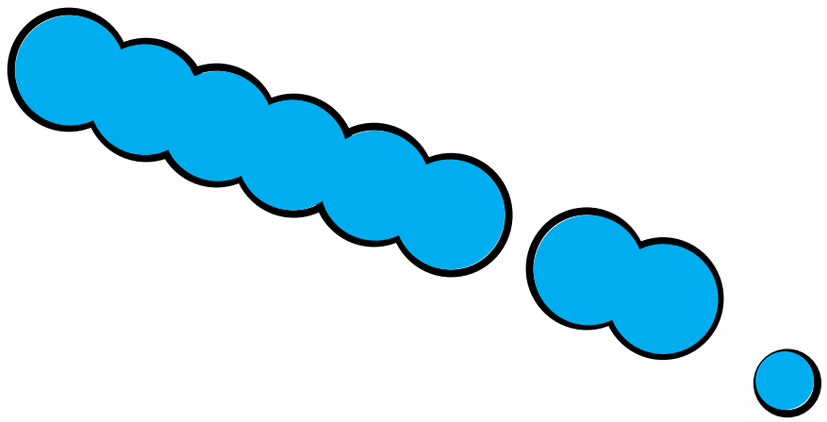


100 Spheres = 70 Volumeneinheiten



Performance?
Kollision: n^2

PartikelSystem

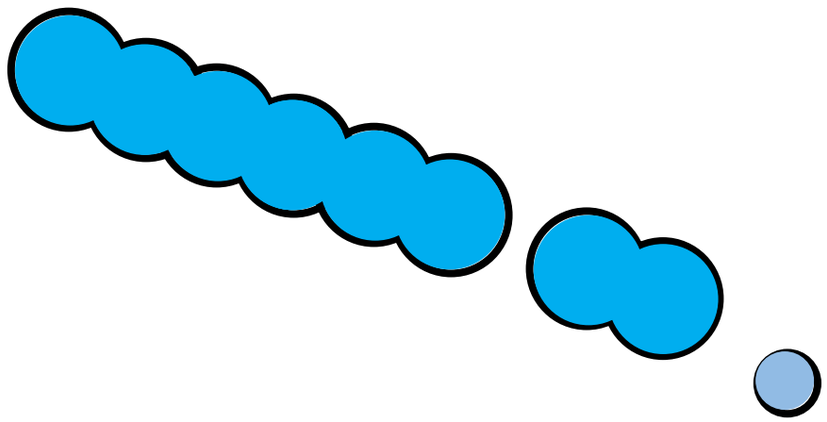


Prüfung der Gruppenzugehörigkeit durch Kollision mit anderen Partikeln

```
void OnCollisionEnter(Collision col)
{
    if (col.gameObject.tag == "Bubble") timeSinceLastCol = Time.time;
}
```

```
if (Time.time - timeSinceLastCol >= secondsUntilFoam && Time.time - startTime >= foamStartThreshold)
{
    isFoam = true;
}
if (Time.time - startTime >= lifeTime)
{
    isFoam = true;
}
```

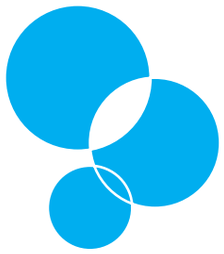
PartikelSystem



Prüfung der Gruppenzugehörigkeit durch Kollision mit anderen Partikeln

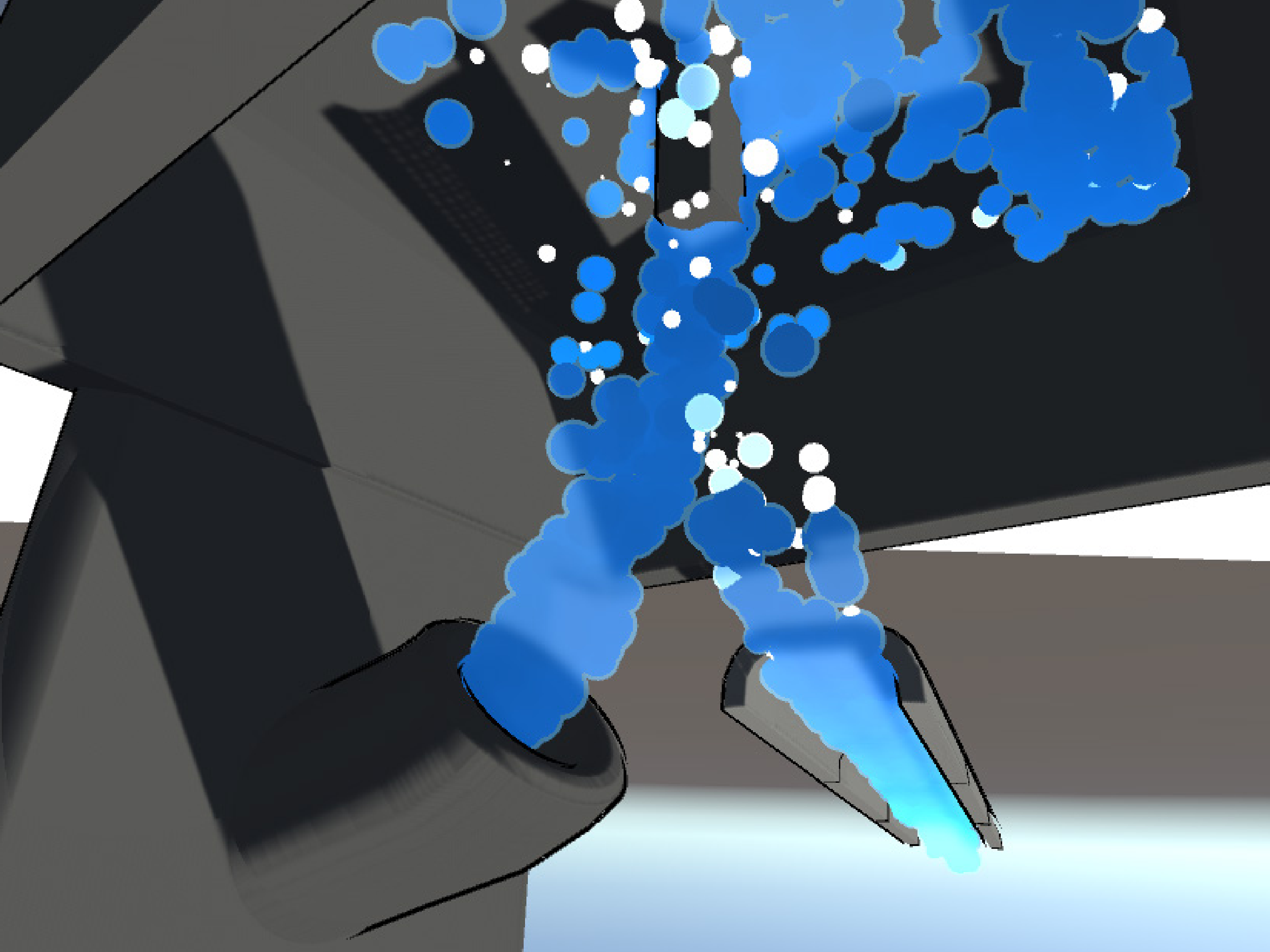
```
if (isFoam)
{

    this.gameObject.GetComponent<Renderer>().material.SetFloat("_WhiteFactor", 3 - this.gameObject.transform.localScale.x);

    if (this.gameObject.transform.localScale.x <= 0.3)
    {
        Destroy(this.gameObject);
    }

    this.gameObject.transform.localScale *= foamScale;

}
```

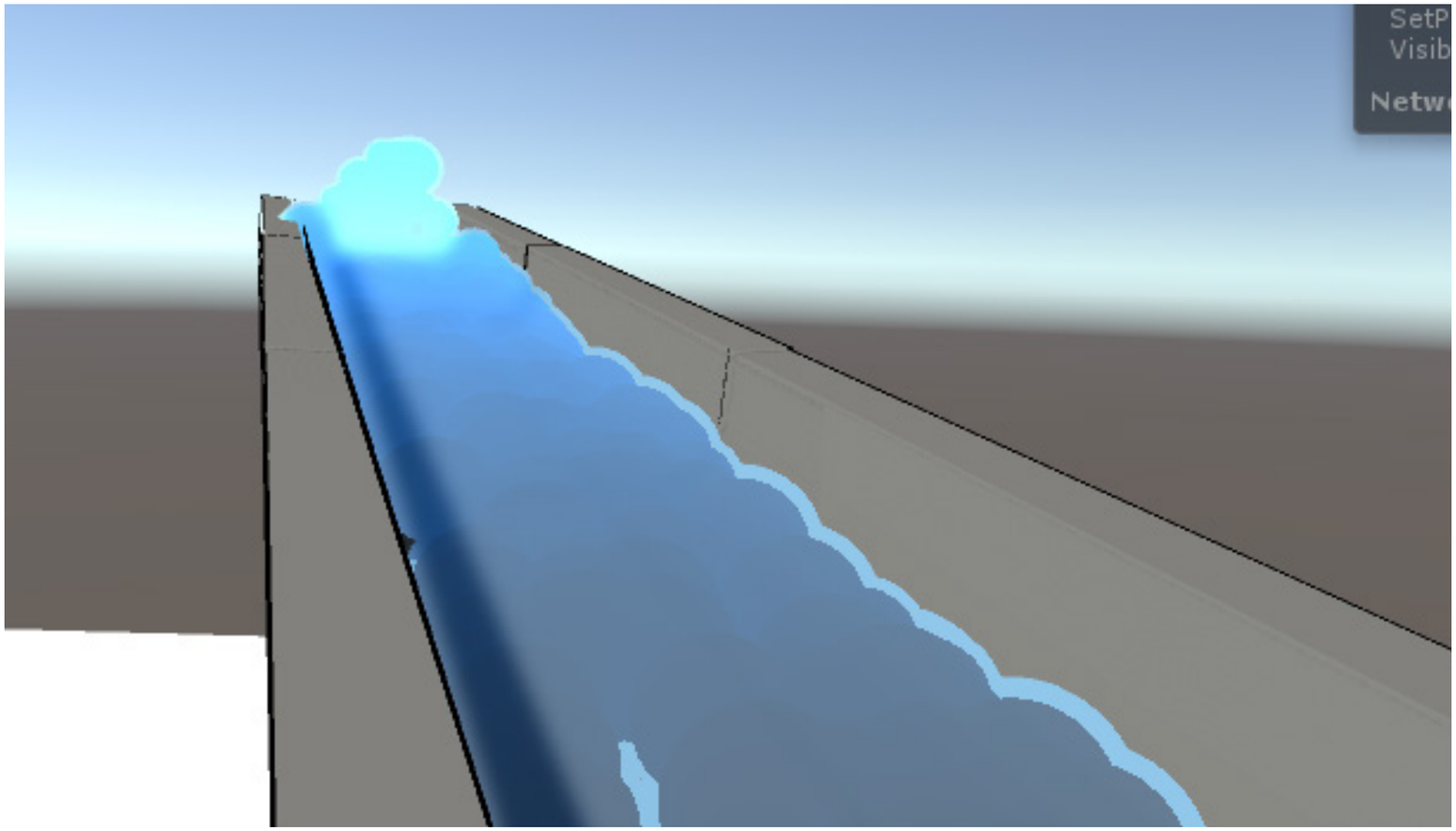
Shader, cool!



Shader, yeah!

- DepthTexture
- Outline (DepthTexture?)
  - Komposition
- Transparenz
- Schaum

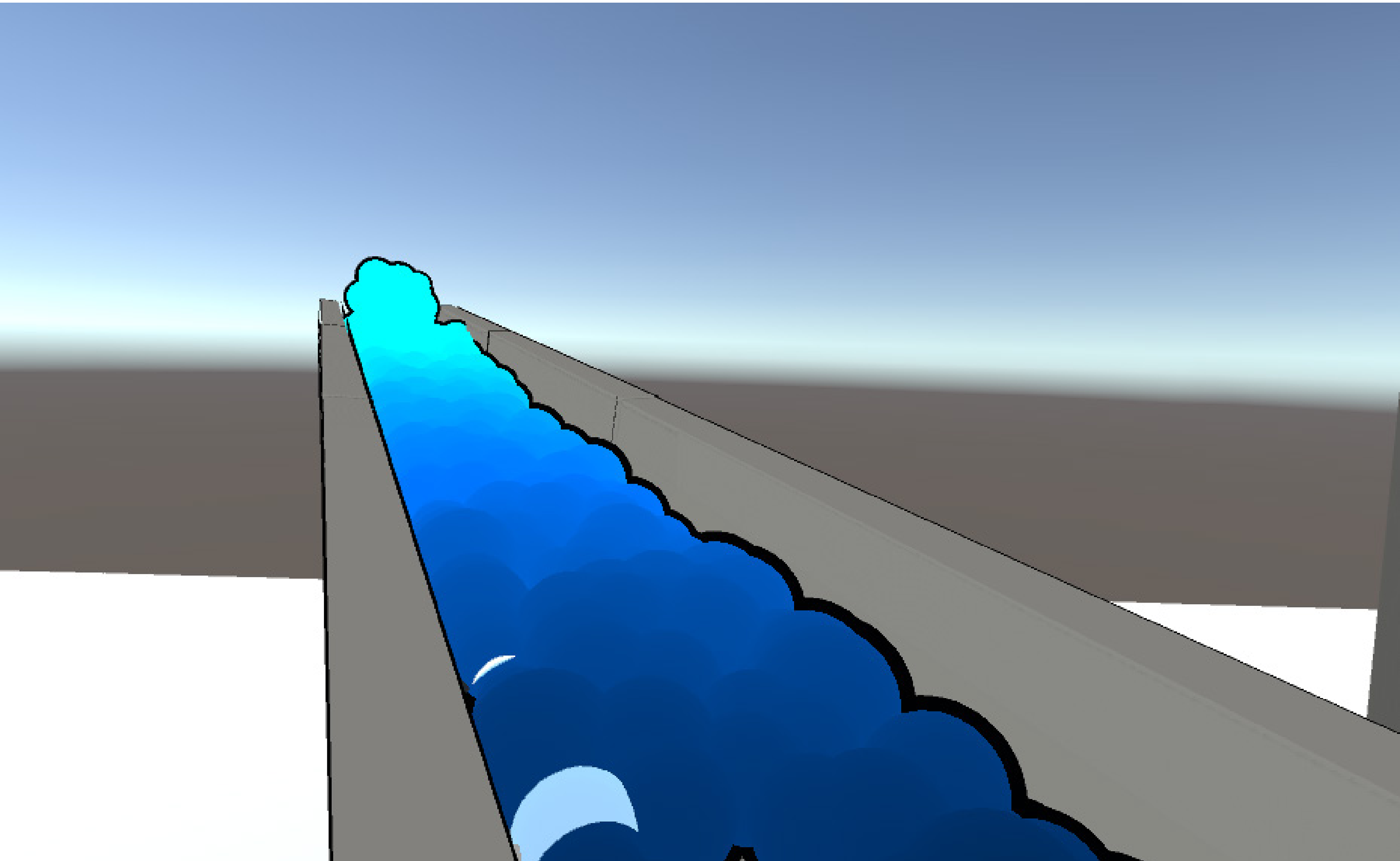


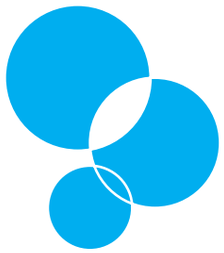


Shader

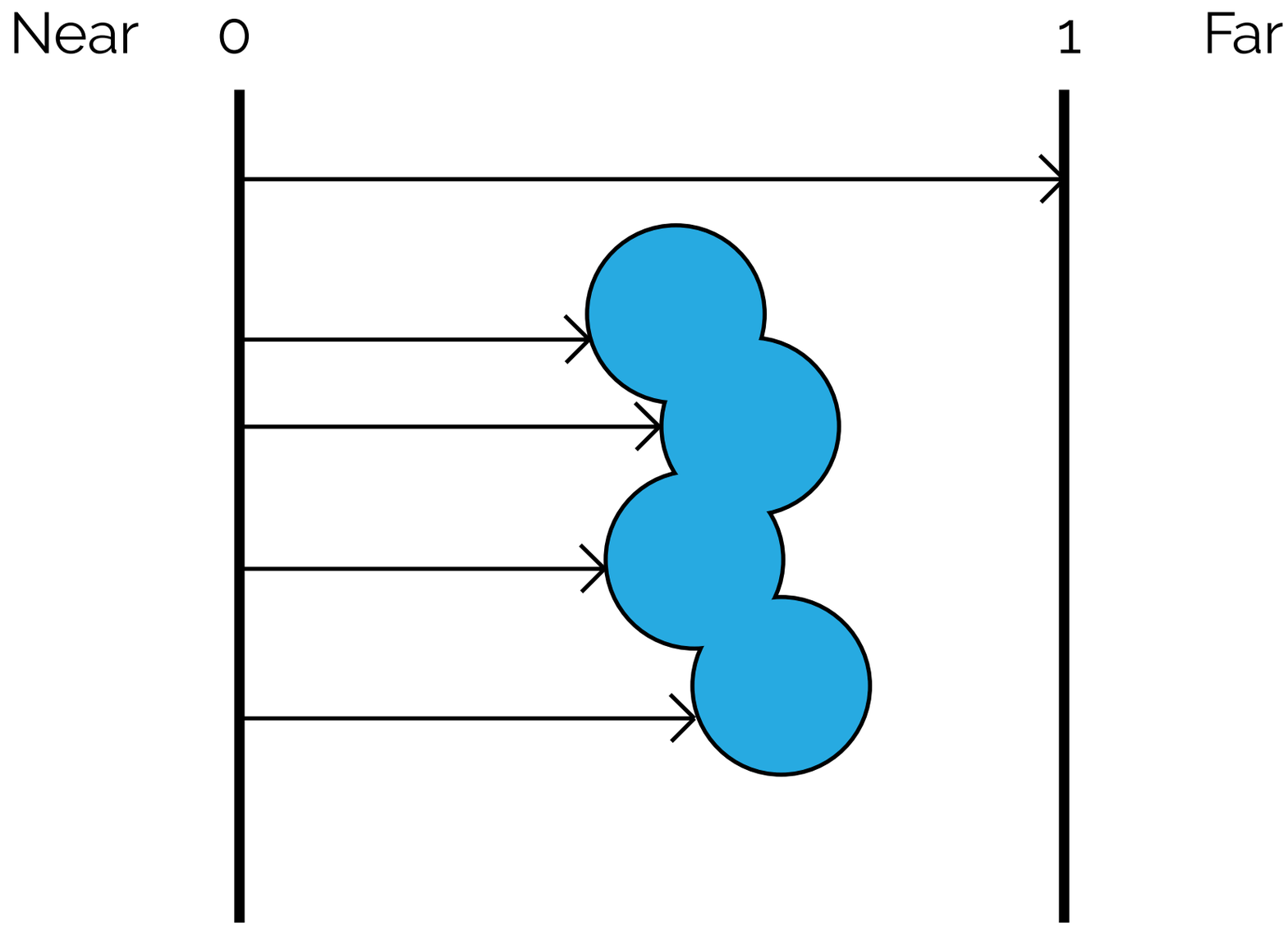
DepthTexture

DepthTexture





DepthTexture

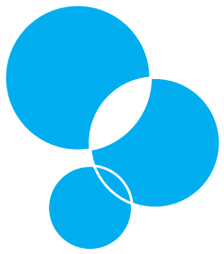


DepthTexture
mögl. Anwendung

- 3D Effekt
- ScreenSpace Effects
- Weichzeichnen der FluidOberfläche
- Siehe Simon Green
- Aber die ist doch Grau, oder?

Parameter

```
Properties{

    _MainTex("Transparenz Textur", 2D) = "" {}
    _Color("Color", Color) = (1,1,1,1)
    _OutlineColor("OutColor",Color) = (1,1,1,1)
    _ColorFactor("Intensität", float) = 5
    _WhiteFactor("Schaum Faktor",float) = 0
    _FoamIsWhite("Weiß 1 / Schwarz 0",int) = 0
    _Transparency("Transparenz", Range(0.0 , 1)) = 0.5
    _OutlineThickness("Outline Stärke", Range(0.0,1.0)) = 0.2

}
```



DepthTexture

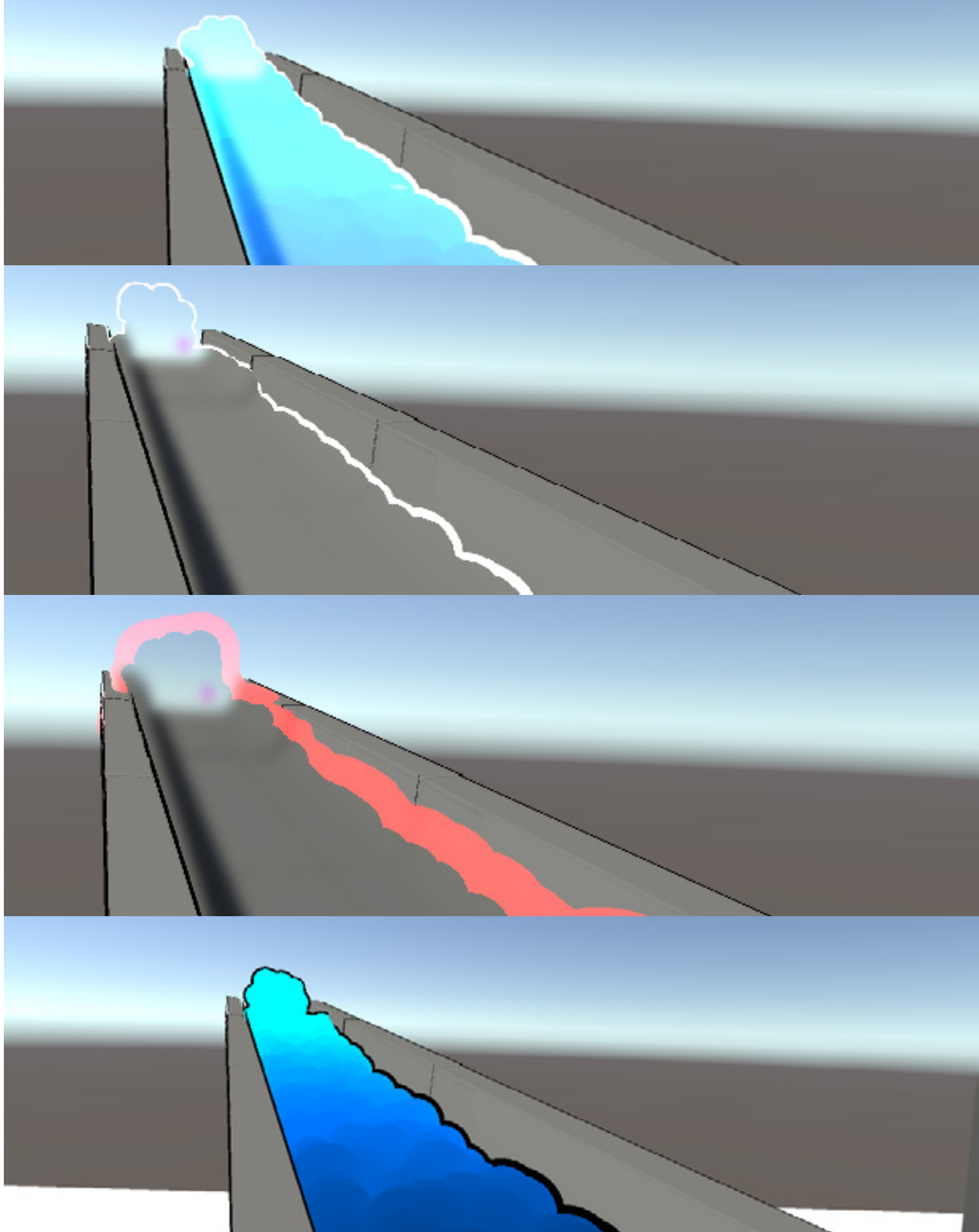
```
//Fragement Shader Abschnitt -> Einfärbung nach Tiefe

half4 frag(v2f i) : COLOR
{
    //Ausgabe Werte vorbereiten

    half4 depthOut = _Color;

    float depthValue;
    depthValue = Linear01Depth(tex2Dproj(_CameraDepthTexture, UNITY_PROJ_COORD(i.scrPos)).r);


  depthOut.r *= depthValue*_ColorFactor;
  depthOut.g *= depthValue*_ColorFactor;
  depthOut.b *= depthValue*_ColorFactor;
```

Outline

- Einbettung in Szenerie und Stil
- Verdeutlichen des Fluidkörpers
- *Fancy*

Outline

- Farbe wählbar
- Dicke wählbar
- DepthTexture abhängig (Vertex Shader)



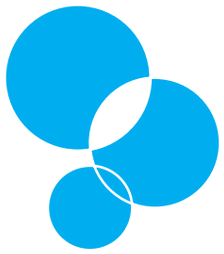
Outline

Magie

```
//Vertex Shader Abschnitt
v2f vert(appdata_full v)
{
    v2f o;
    v.vertex.xyz *= 1+_OutlineThickness; // outline
    o.pos = mul(UNITY_MATRIX_MVP, v.vertex);
    o.scrPos = ComputeScreenPos(o.pos);

    return o;
}
```

- im Fragment Shader:

```
if (depthValue >= 0.9) //OutlineFarbe bei größerer Tiefe
{
    depthOut.rgb = _OutlineColor;
}
```



Outline

depthValue?

- Warum kann auf Basis des Tiefen-wertes eine Outline gezeichnet werden, obwohl Geometrie direkt dahinter liegt?

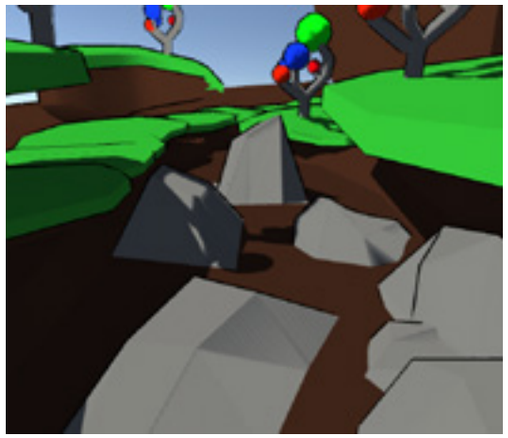
- Komposition

Komposition

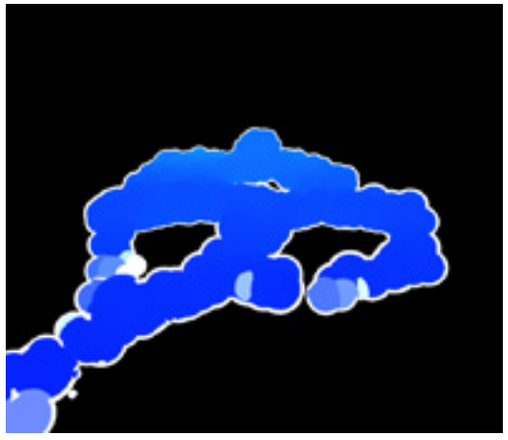
- Zusammenfügen diverse Kameras und Effekte in der Szene
- 3 Kameras
  Tiefe/Szene, Partikel, BlurSzene
- Warum Blur nicht im Shader?

Komposition

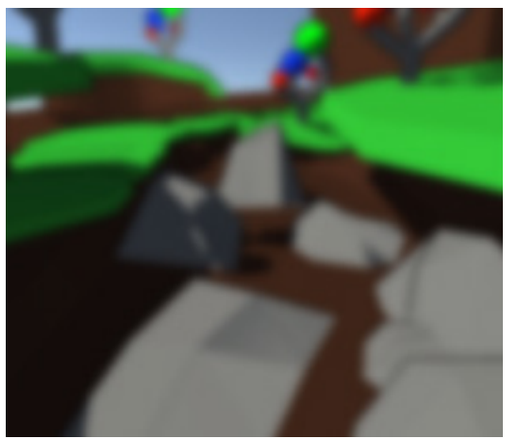
Szene



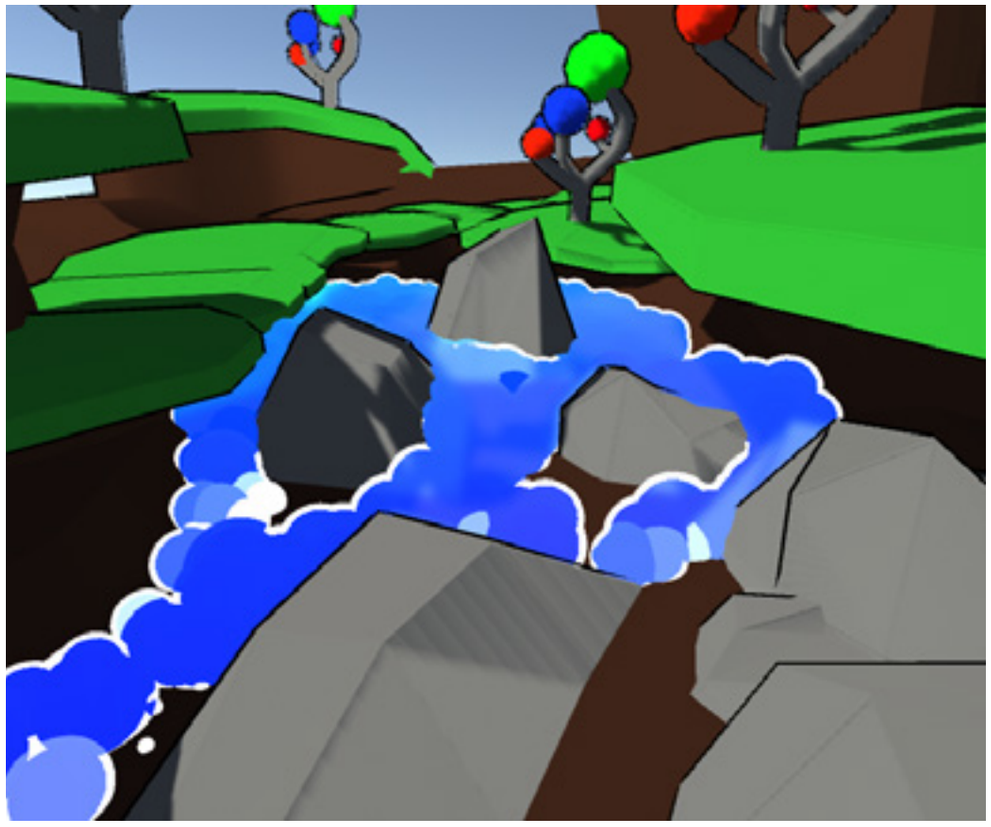
Partikel



Szene



Final

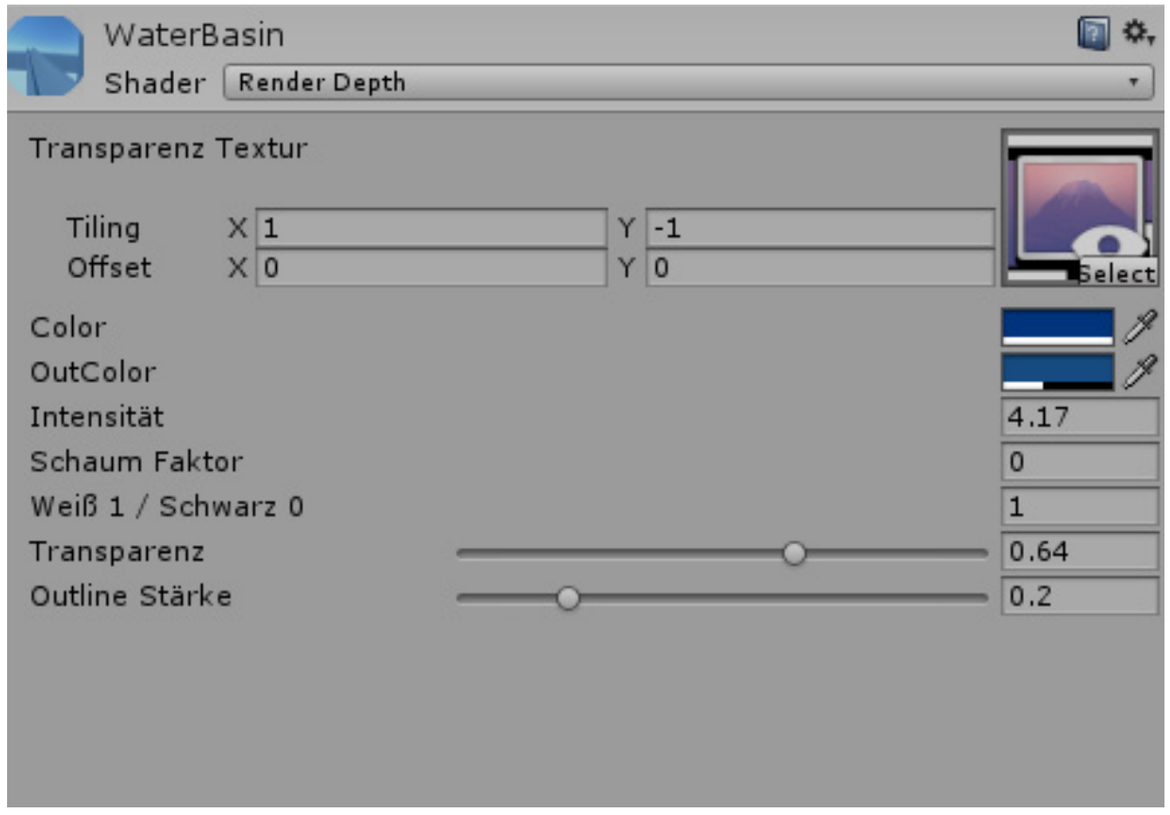


Outline

depthValue!

- Partikelkamera betrachtet nur die Partikel
- Tiefenwerte werden für die Partikel in Farbe gewandelt
- betrachtet die Partikelkamera bereiche um den Partikel herum: depthValue >= 0.9
- Es sei denn: Partikel hinter Partikel -> Outline nur am Rand des Fluids

Transparenz



- Betrachten des Hintergrundes mit der 3. Kamera -> RenderTexture
- ImageEffects -> Blur

- Kombinieren des Hintergrundes als Textur im Shader

Transparenz

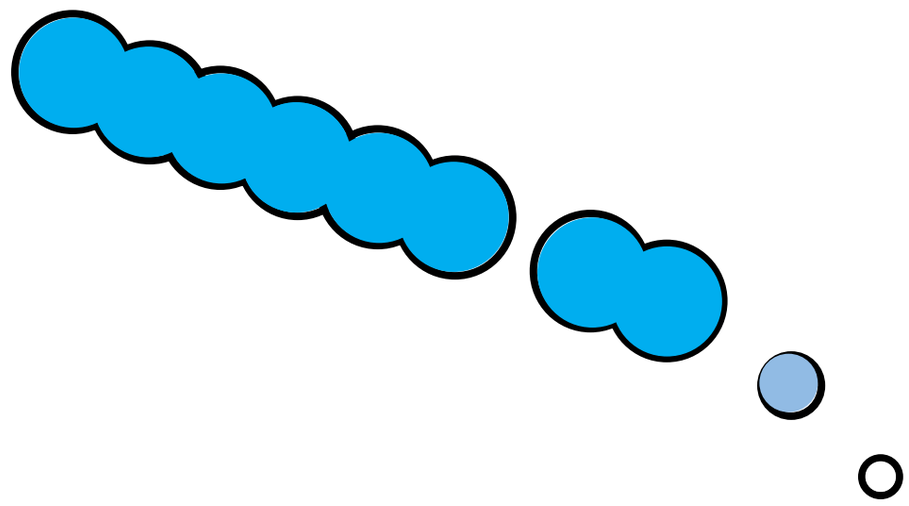
- Blur Image positionieren (Vertex Shader)

```
fixed4 col = tex2Dproj(_MainTex, UNITY_PROJ_COORD(i.scrPos));
```

- berechnete Farbwerte mit Blur Image und Transparenzwert kombinieren (Fragment Shader)

```
return depthOut + col * _Transparency;
```

Schaum

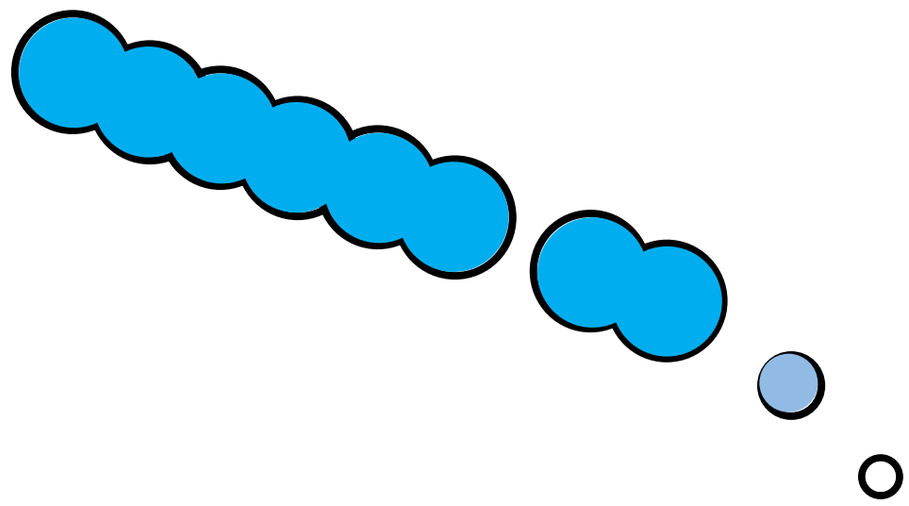


- Größe und Kontrolle durch Skript

```
if (isFoam)
{

    this.gameObject.GetComponent<Renderer>().material.SetFloat("_WhiteFactor", 3 - this.gameObject.transform.localScale.x);

    if (this.gameObject.transform.localScale.x <= 0.3)
    {
        Destroy(this.gameObject);
    }

    this.gameObject.transform.localScale *= foamScale;

}
```

- Addieren der Helligkeitsvariable zur allgemeinen Farbe

Schaum



- Addieren/Subtrahieren der Helligkeitsvariable

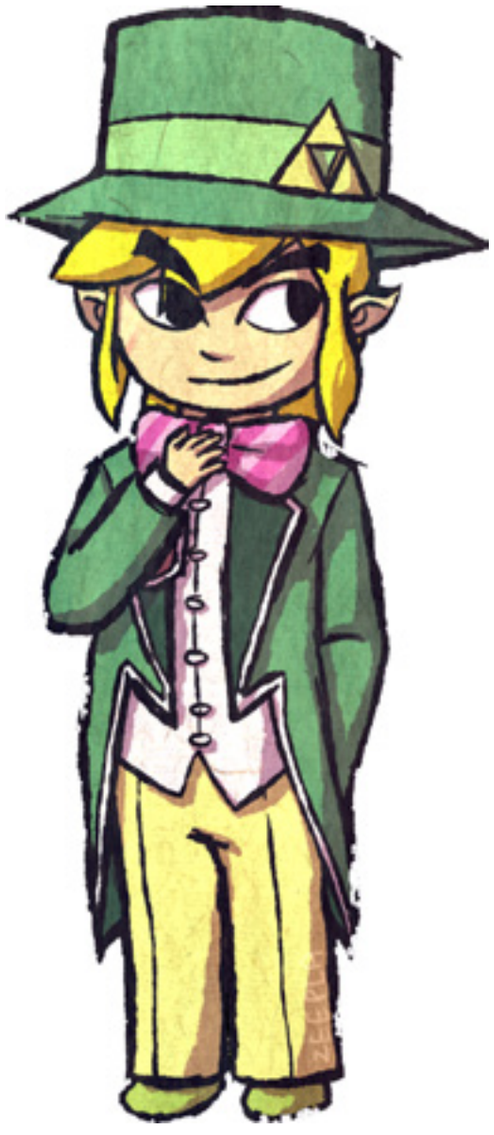
```
if (_FoamIsWhite == 1)
{
    depthOut += _WhiteFactor;
}
else
{
    depthOut -= _WhiteFactor;
}
```

Weiterführende Arbeit

- ScreenSpace Effects auf Basis der DepthTexture oder DepthNormalTexture

- Anpassen und Einfügen weiterer Parameter

Danke



Felix Bukatz
550105
MI-5