Game Technology

Lecture 3 – 31.10.2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Dipl-Inf. Robert Konrad
Dr.-Ing. Florian Mehm

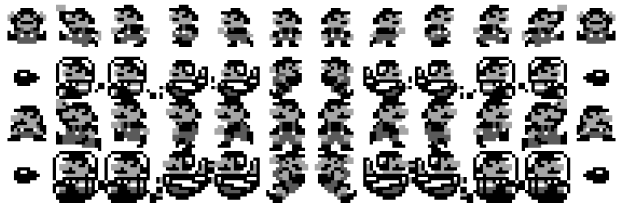
Prof. Dr.-Ing. Ralf Steinmetz
KOM - Multimedia Communications Lab

Preliminary timetable

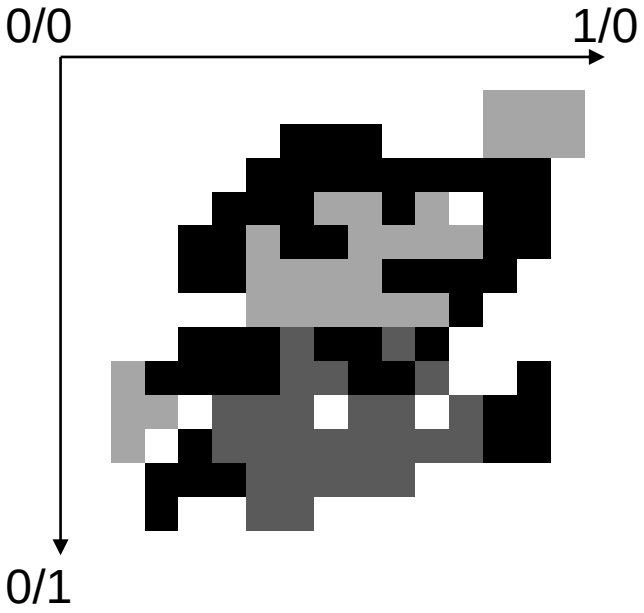
TECHNISCHE UNIVERSITÄT DARMSTADT

Lecture No.	Date	Topic
1	17.10.2014	Basic Input & Output
2	24.10.2014	Timing & Basic Game Mechanics
3	31.10.2014	Software Rendering 1
4	07.11.2014	Software Rendering 2
5	14.11.2014	Basic Hardware Rendering
6	21.11.2014	Animations
7	28.11.2014	Physically-based Rendering
8	05.12.2014	Physics 1
9	12.12.2014	Physics 2
10	19.12.2014	Scripting
11	16.01.2015	Compression & Streaming
12	23.01.2015	Multiplayer
13	30.01.2015	Audio
14	06.02.2015	Procedural Content Generation
15	13.02.2015	AI

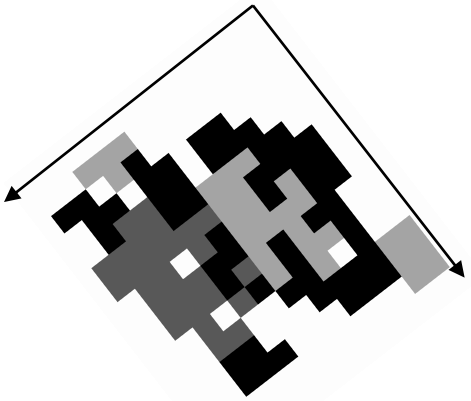
2D Rendering



Scaling



Rotations



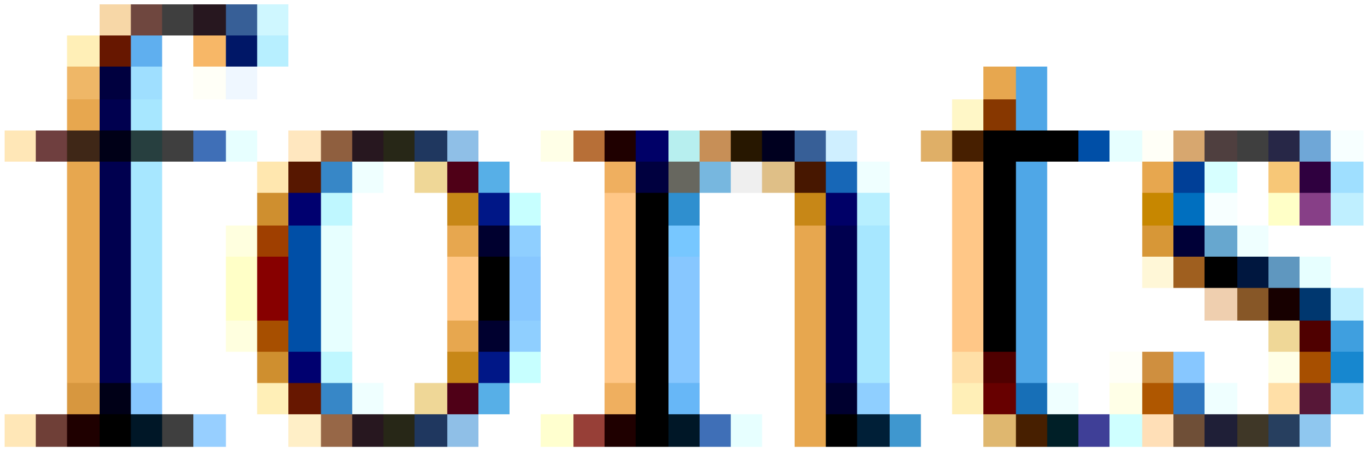
Fonts

TECHNISCHE UNIVERSITÄT DARMSTADT

- **TrueType format**
  - Line segments and Bézier curves
  - Kerning
    - VA
  - Pixel snapping
  - „TrueType systems include a virtual machine that executes programs inside the font”
  - Unicode
    - More than 110,000 characters

Subpixel Rendering

TECHNISCHE UNIVERSITÄT DARMSTADT



Bitmap Fonts

Font Libs

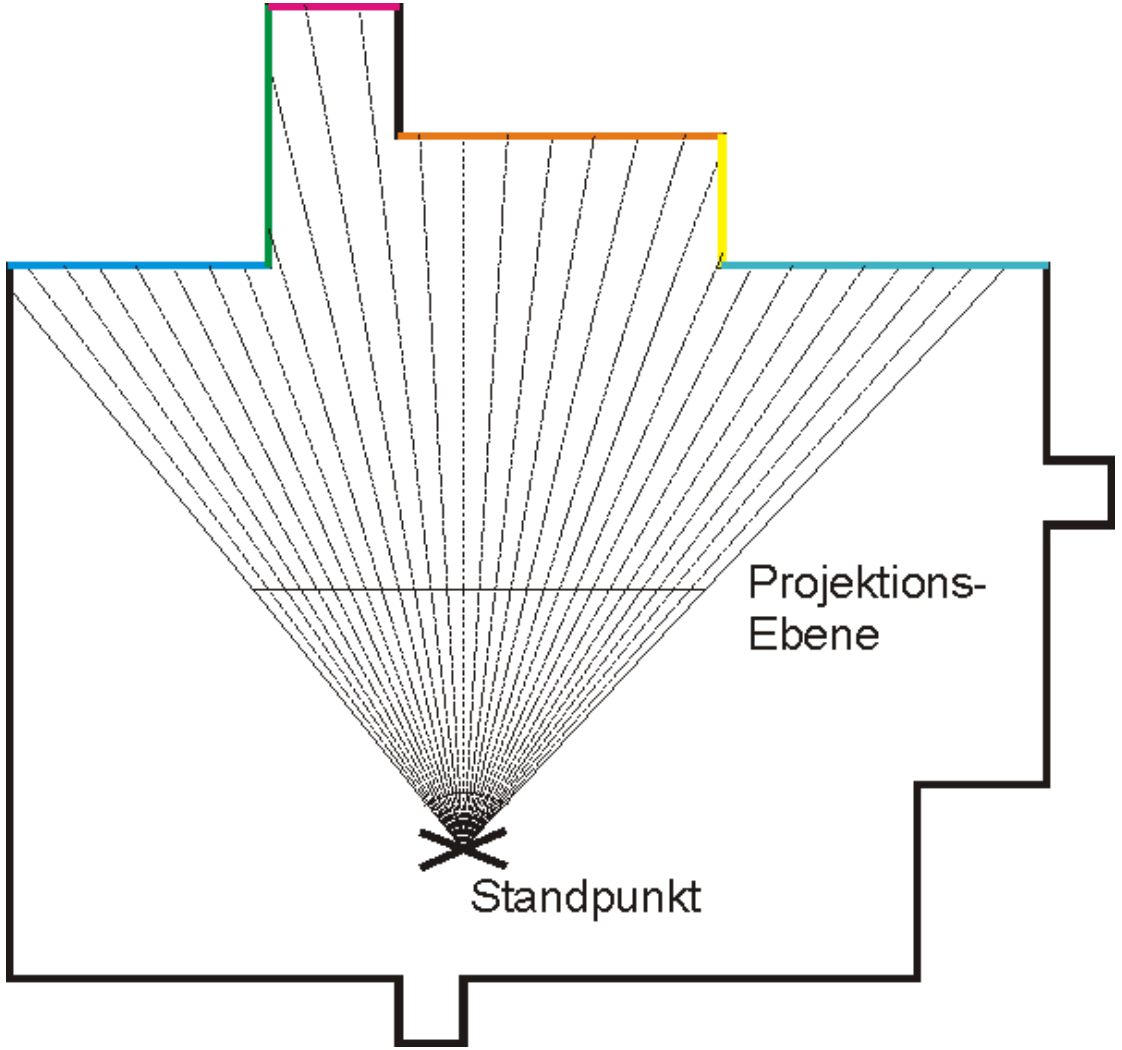
TECHNISCHE UNIVERSITÄT DARMSTADT

- **Freetype**
  - http://www.freetype.org
- **stb_truetype**
  - https://github.com/nothings/stb

„Raycasting“

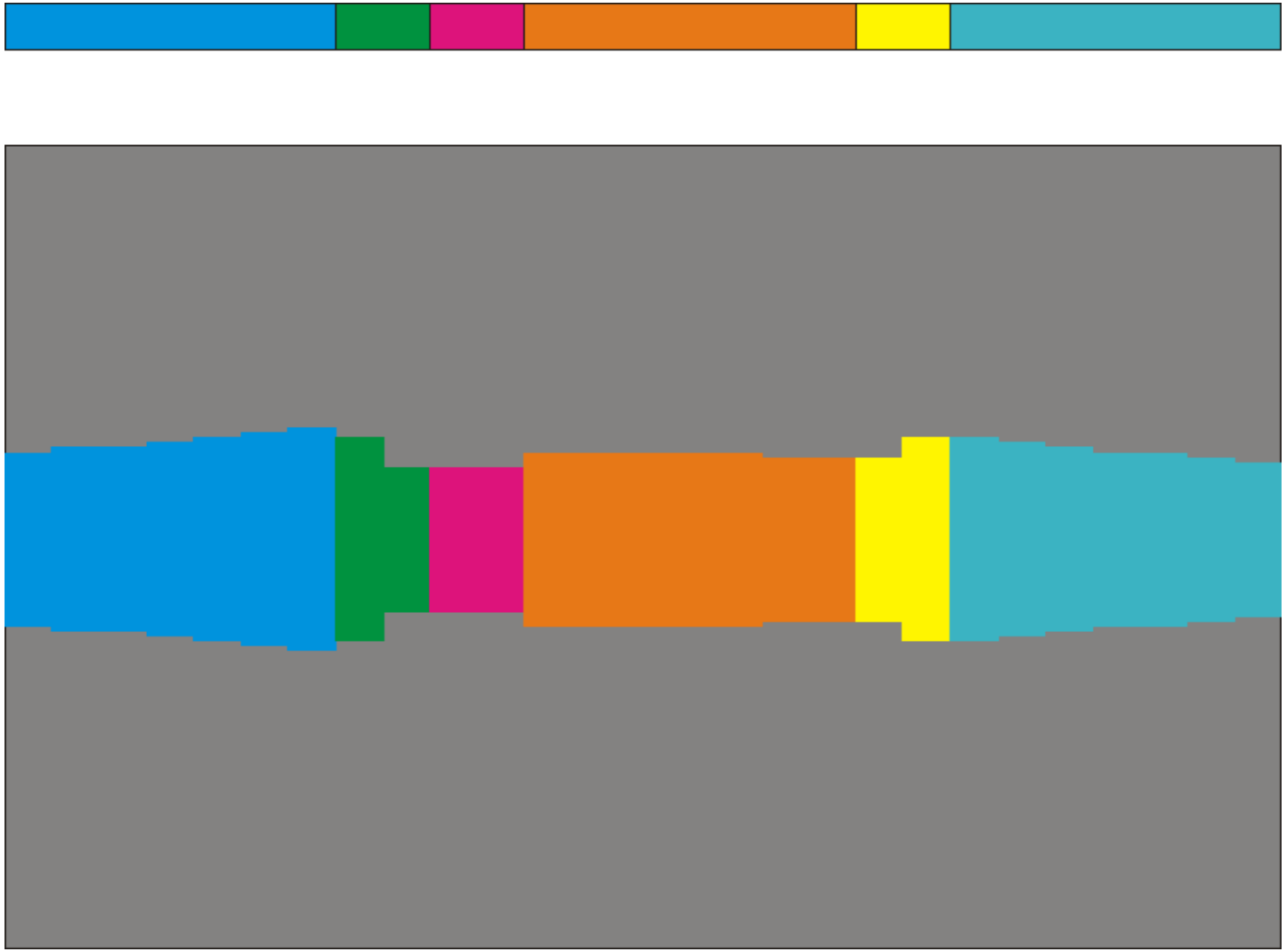

5
White Hallways
BOLTS
NUKES
HEALS
KEYS
SCROLLS
SCORE
5800
N
W
E
S

„Raycasting“



„Raycasting“

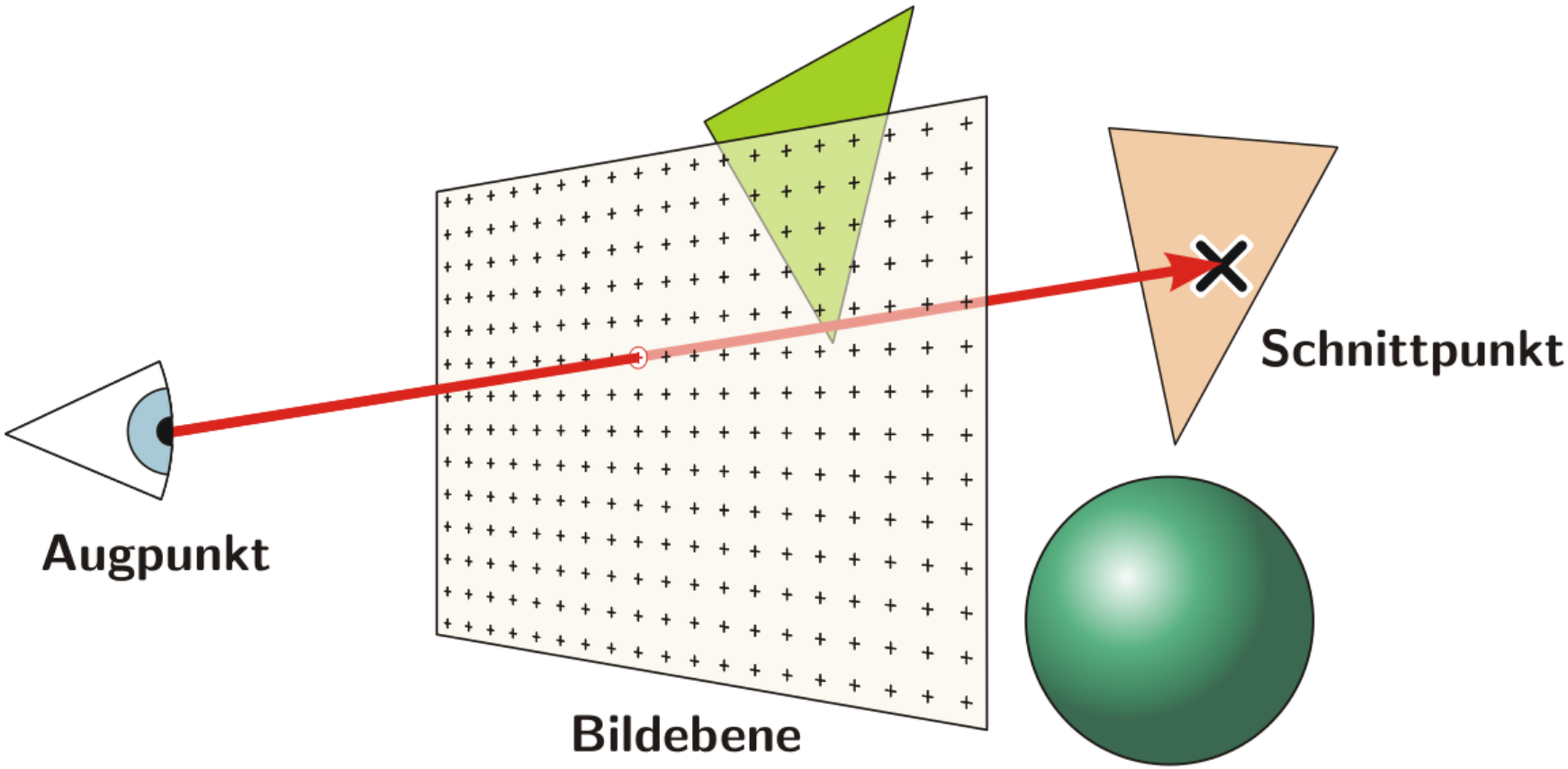
TECHNISCHE UNIVERSITÄT DARMSTADT



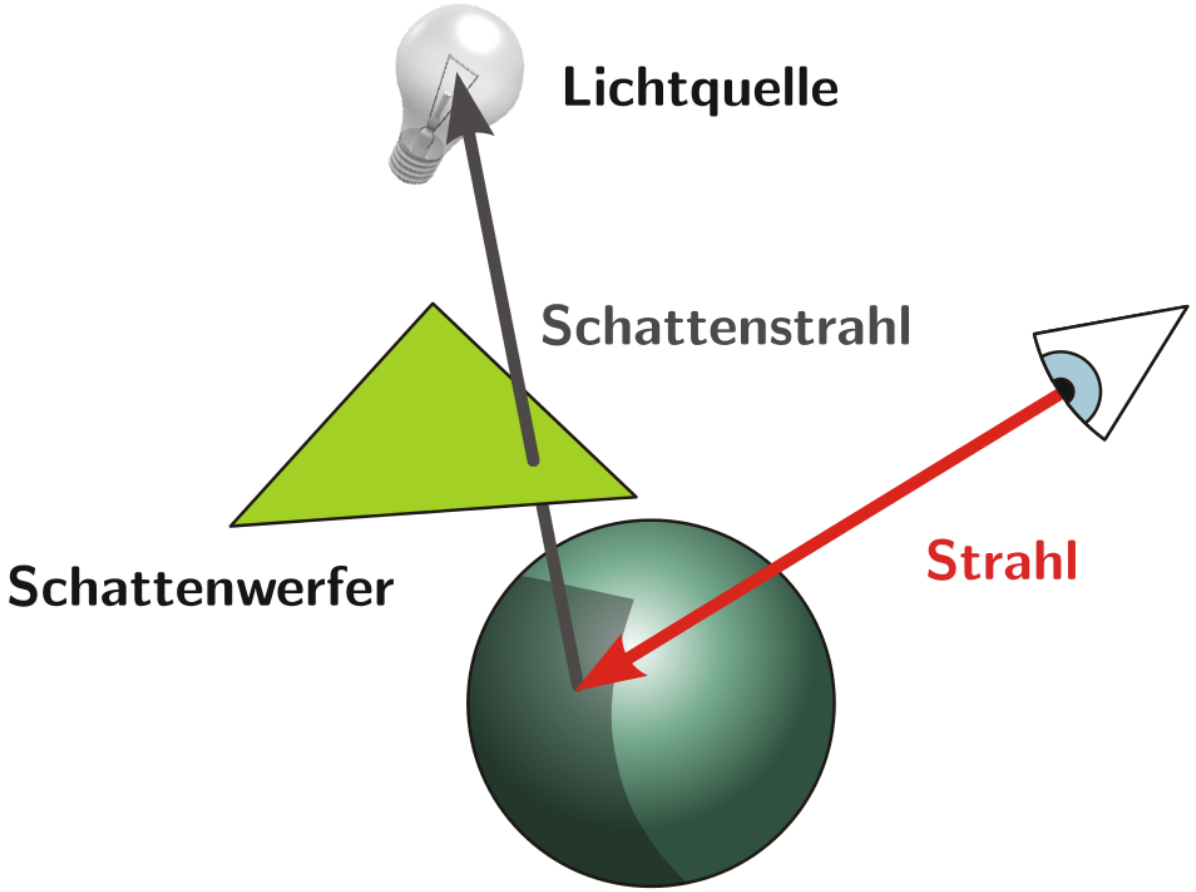
„Raycasting“


GAUNTLET
LEVEL 3
WARRIOR
SCORE HEALTH
4390 1628
VALKYRIE
SCORE HEALTH
0 0
INSERT COIN
WIZARD
SCORE HEALTH
0 0
INSERT COIN
ELF
SCORE HEALTH
0 0
INSERT COIN
1 COIN =
700 HEALTH
©1985
ATARI GAMES

Raytracing

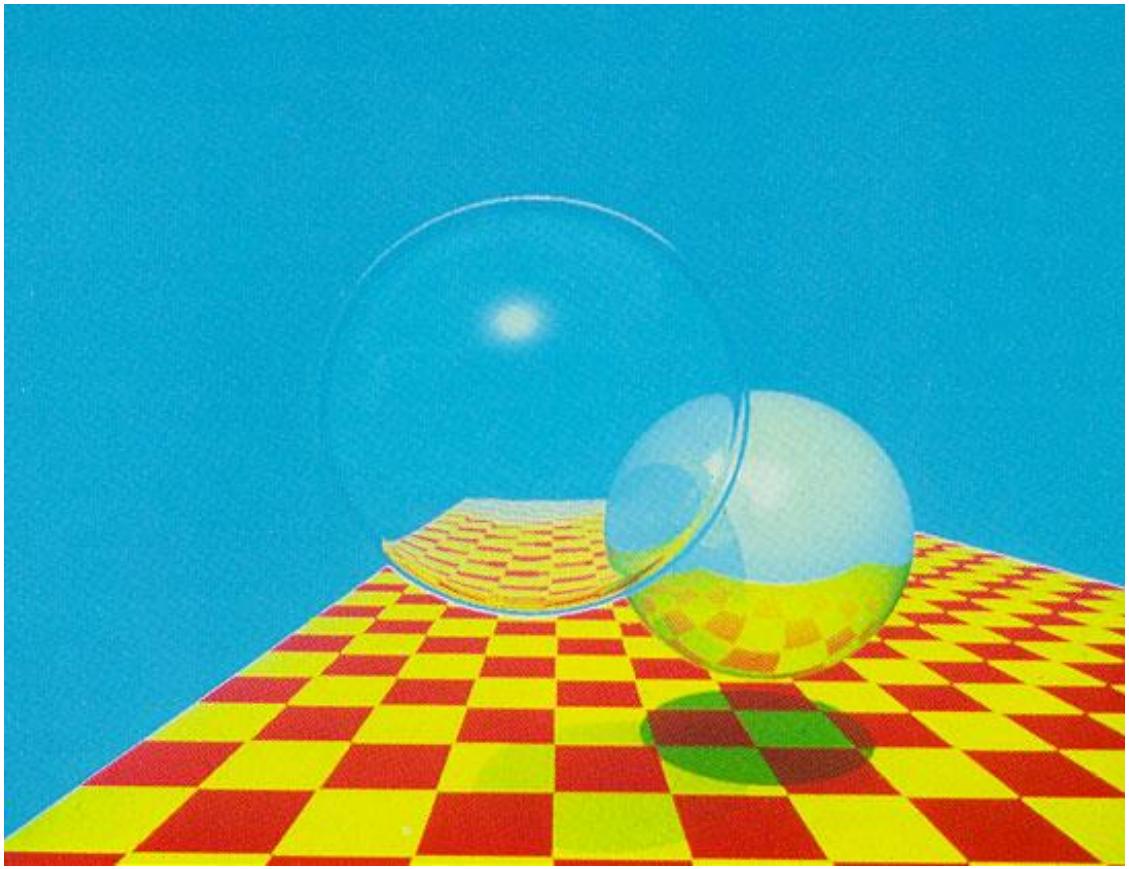


Raytracing

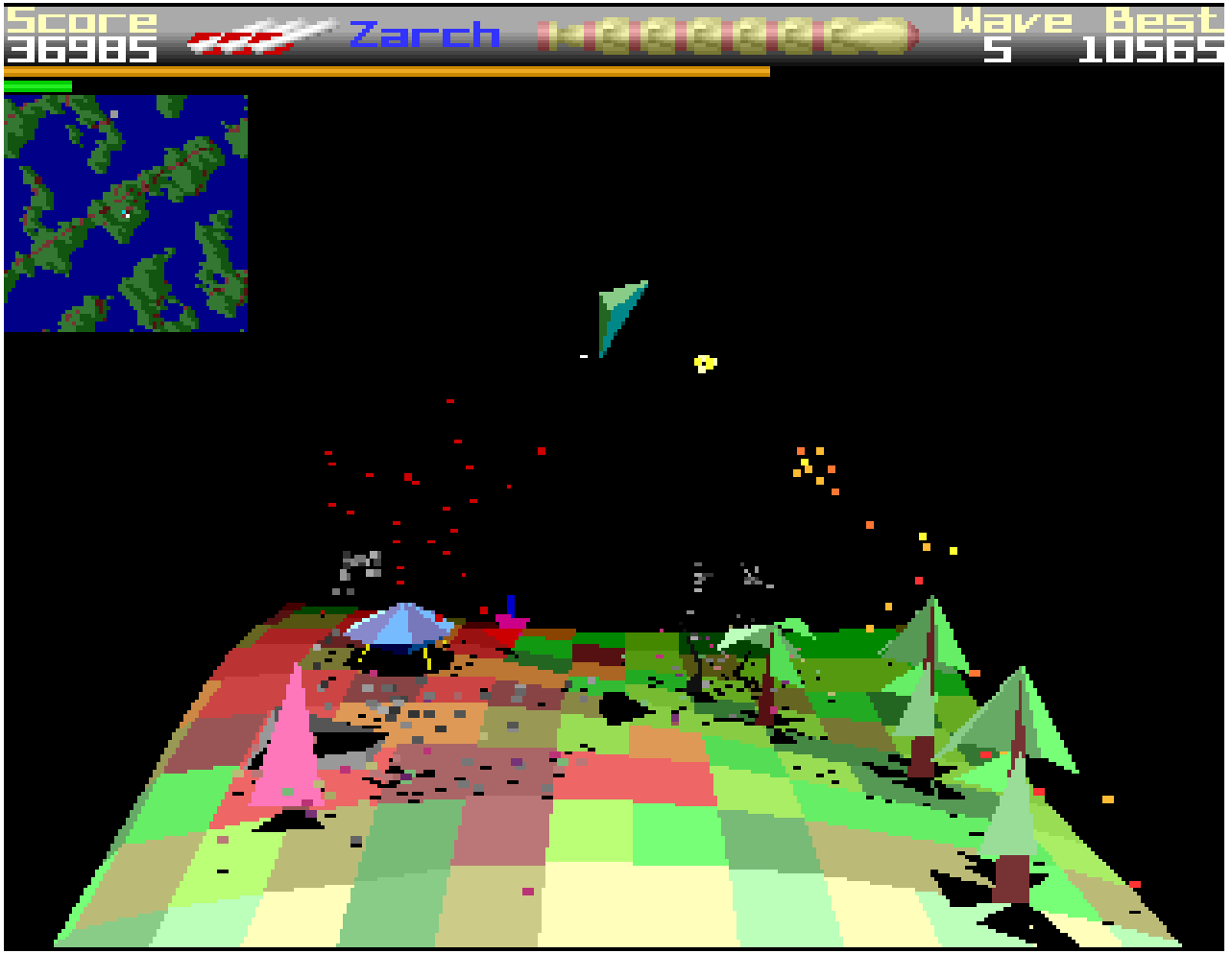


Raytracing

TECHNISCHE UNIVERSITÄT DARMSTADT



Rasterisierung



Rasterisation

Rasterisation & Raytracing

Rasterisation

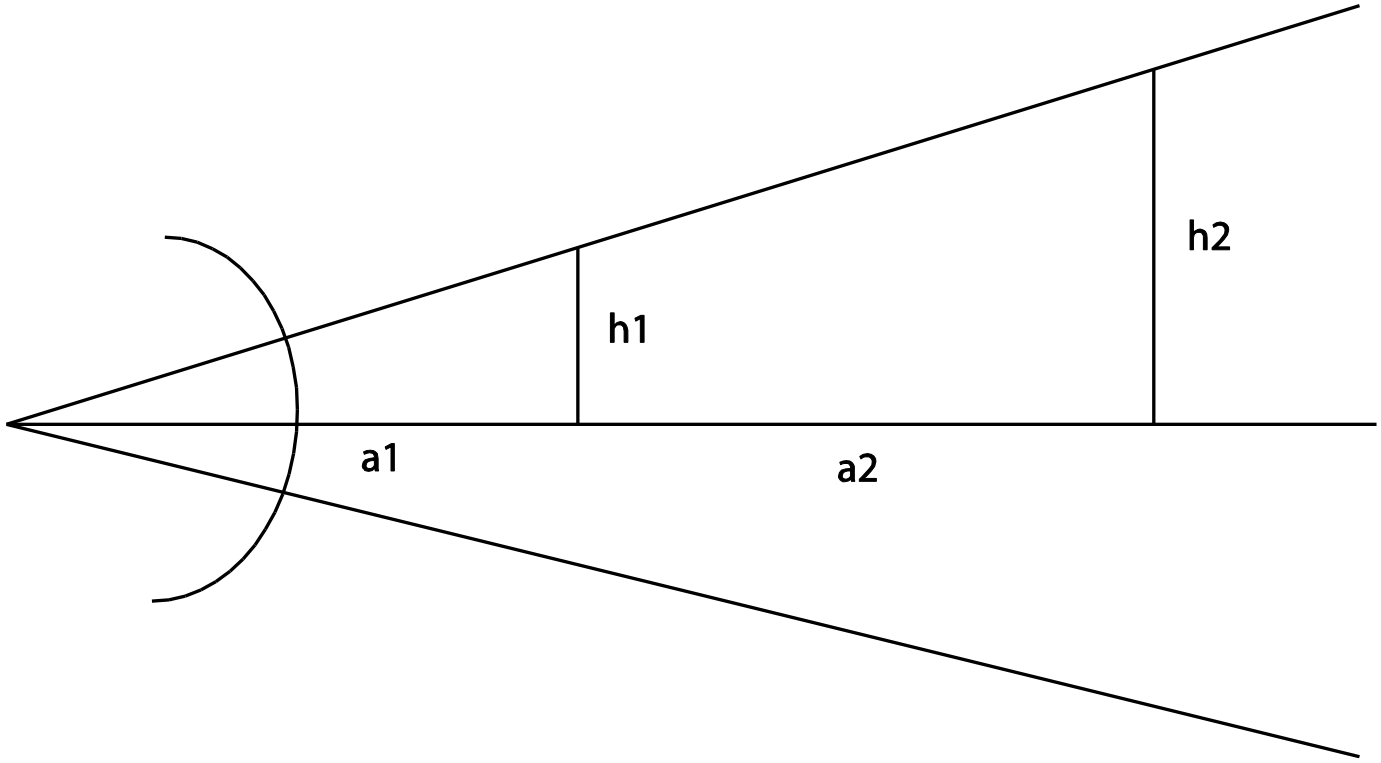
TECHNISCHE UNIVERSITÄT DARMSTADT

- **World defined by triangles**
  - Each triangle -> 3 3D points
- **foreach (tri in world) {**

```
Point p1 = transform(tri._1);
Point p2 = transform(tri._2);
Point p3 = transform(tri._3);
drawTriangle(p1, p2, p3); // 2D operation
```

}

Perspective



h1 / a1 = h2 / (a1 + a2)

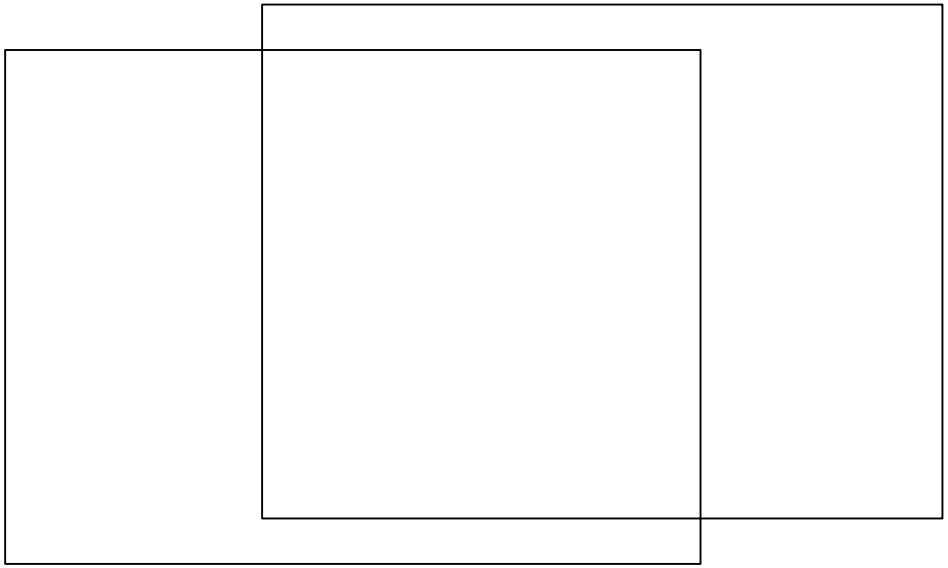
Doubled distance + doubled size -> same projection

Xp = (zmin / distance) * X

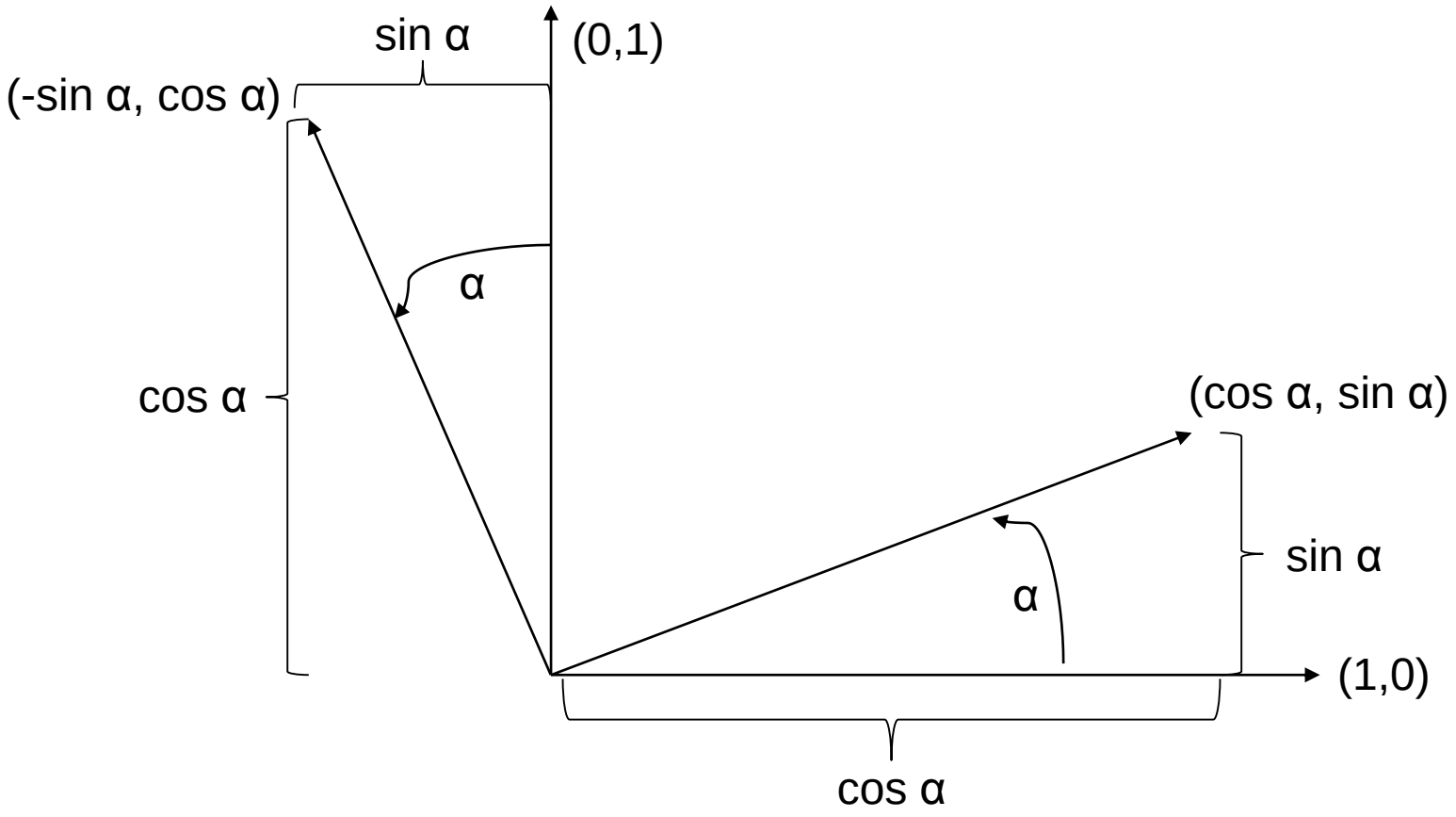
Offset from camera

Xp = (zmin / distance) * (X - camera.x) + screenWidth / 2

Yp = …



Camera Rotations



Camera Rotations

TECHNISCHE UNIVERSITÄT DARMSTADT

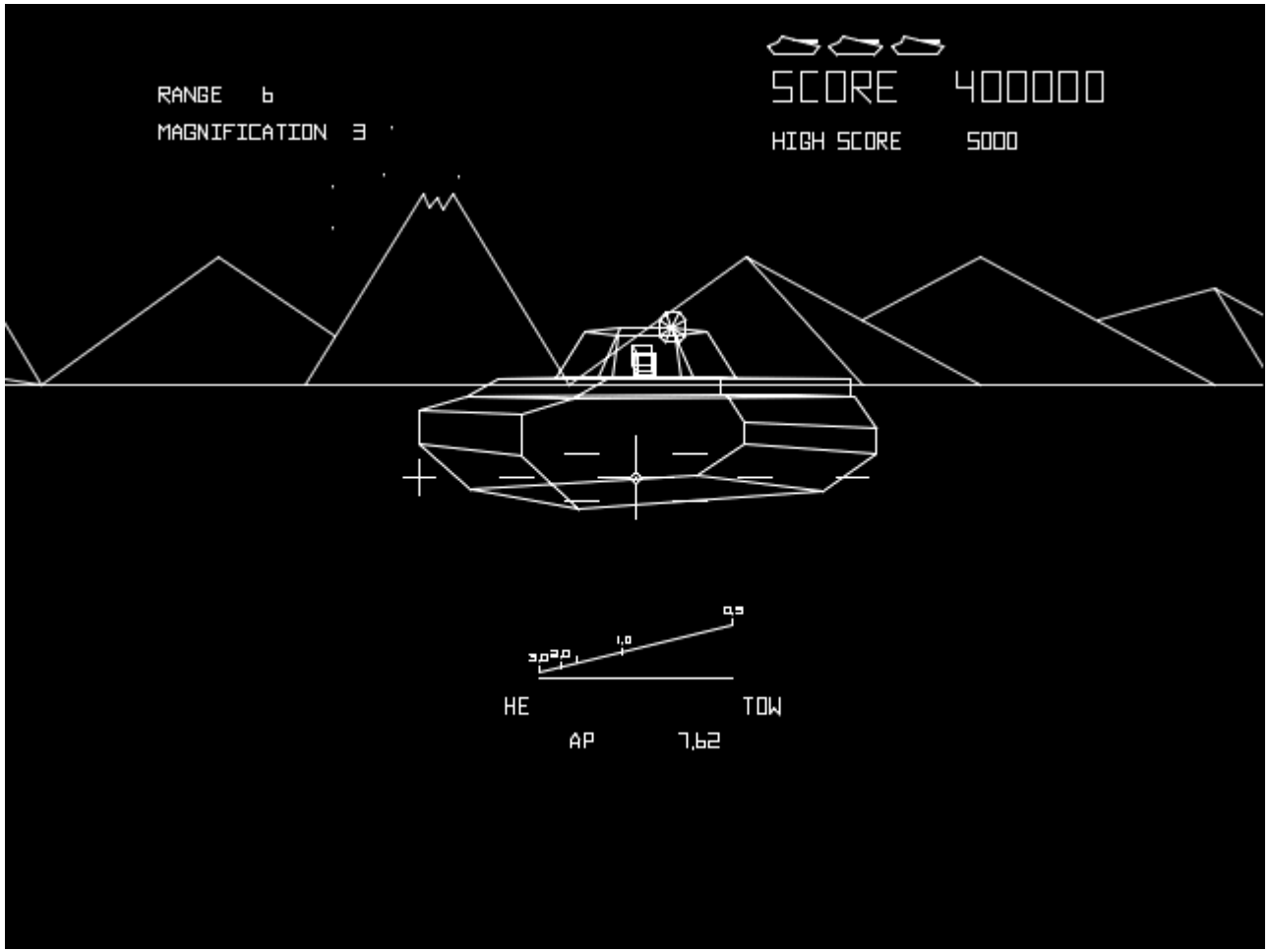
- **Old Point**
  - (x,y) = x(1,0) + y(0,1)
- **New Point**
  - R(x,y,α) = x(cos α, sin α) + y(-sin α, cos α)
  - = (x cos α, x sin α) + (-y sin α, y cos α)
  - = (x cos α - y sin α, x sin α + y cos α)

Camera Rotations

TECHNISCHE UNIVERSITÄT DARMSTADT

```
float dx = X - camera.x;
float dy = Y - camera.y;
float dz = Z - camera.z;
float d1x = cos(camera.ry) * dx + sin(camera.ry) * dz;
float d1y = dy;
float d1z = cos(camera.ry) * dz + sin(camera.ry) * dx;
float d2x = d1x;
float d2y = cos(camera.rx) * d1y - sin(camera.rx) * d1z;
float d2z = cos(camera.rx) * d1z + sin(camera.rx) * d1y;
float d3x = cos(camera.rz) * d2x + sin(camera.rz) * d2y;
float d3y = cos(camera.rz) * d2y – sin(camera.rz) * d2x;
float d3z = d2z;
Xp = (zmin / d3z) * d3x + screenWidth / 2
```

Lines


RANGE 6
MAGNIFICATION 3
SCORE 400000
HIGH SCORE 5000
HE
TOW
AP
7,62

Digital differential analyzer

TECHNISCHE UNIVERSITÄT DARMSTADT

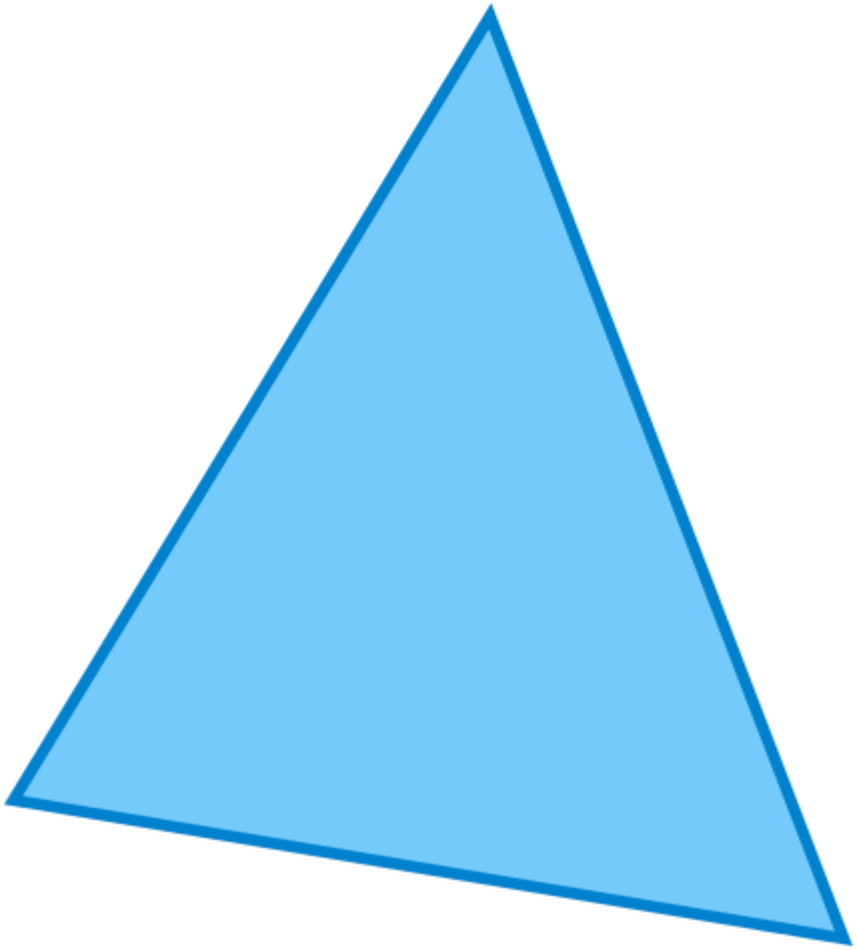
```
dx = x2 - x1
dy = y2 - y1
for x from x1 to x2 {
 y = y1 + dy * (x - x1) / dx
 plot(x, y)
}
```

Bresenham

TECHNISCHE UNIVERSITÄT DARMSTADT

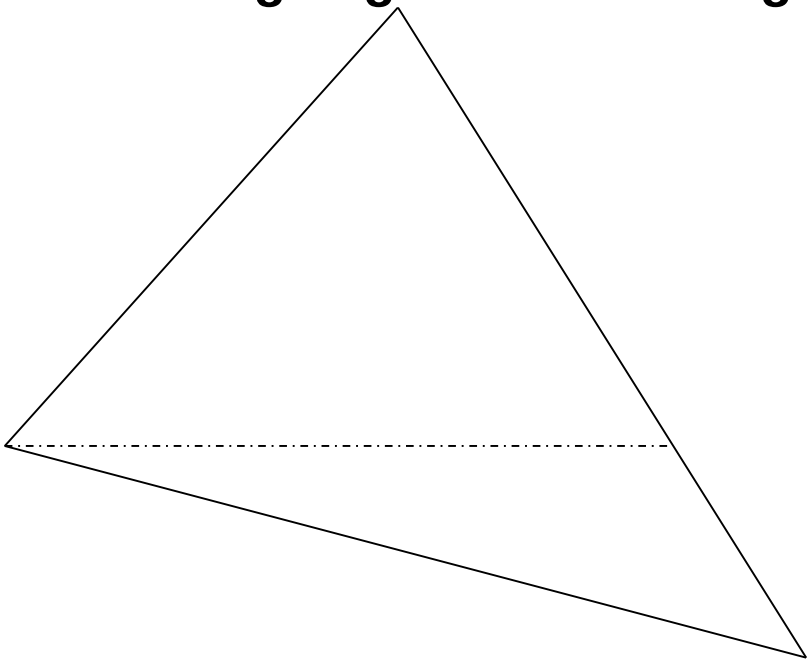
```
plotLine(x0,y0, x1,y1)
  dx=x1-x0
  dy=y1-y0

  D = 2*dy - dx
  plot(x0,y0)
  y=y0

  for x from x0+1 to x1
    if D > 0
      y = y+1
      plot(x,y)
      D = D + (2*dy-2*dx)
    else
      plot(x,y)
      D = D + (2*dy)
```

Triangles



Triangle Rasterisation

- **Find edge longest with biggest ydif**
- **Fill lines between long edge and other edge 1**
- **Fill lines between long edge and other edge 2**



Mesh structure

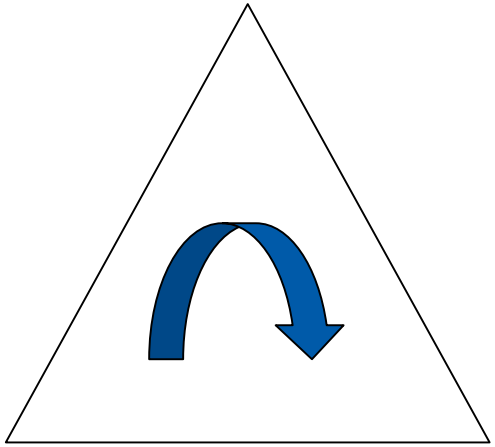
TECHNISCHE UNIVERSITÄT DARMSTADT

- **Array of 3D positions**
  - Often additional data
- **Array of indices**
  - Three indices -> triangle

Backface Culling

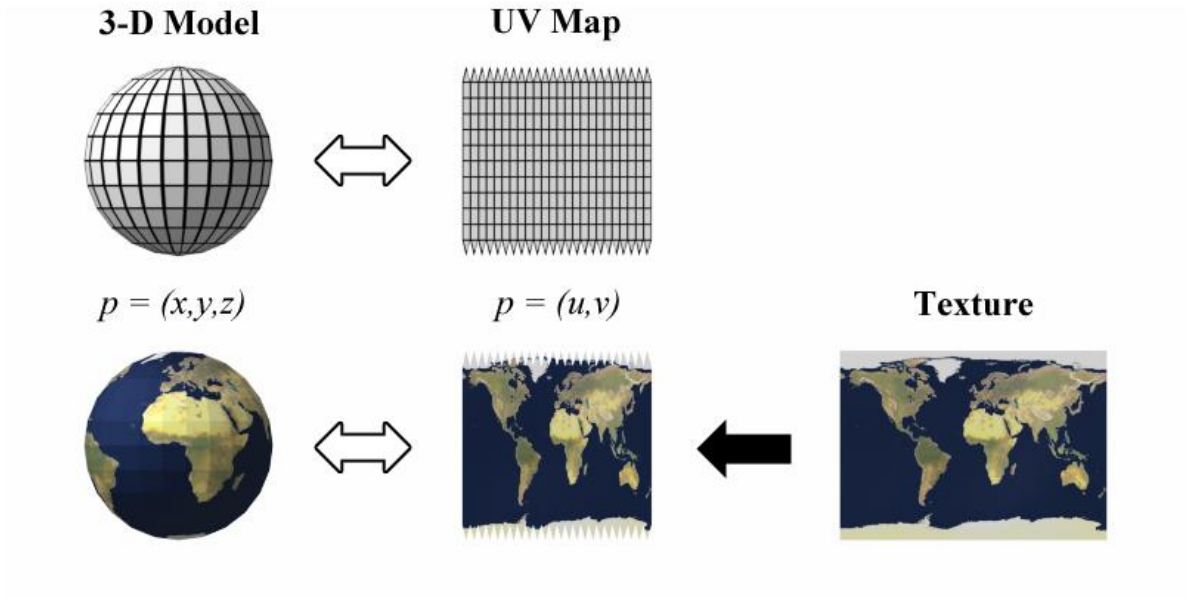
TECHNISCHE UNIVERSITÄT DARMSTADT

- **Remove tris showing away from camera**
- **Use tri winding**
- **cross(b - a, c - a)**



Textures

- **Add texture coordinates (uv) to mesh positions**
- **Interpolate coordinates during rasterisation**
- **Sample texture at interpolated coordinates**

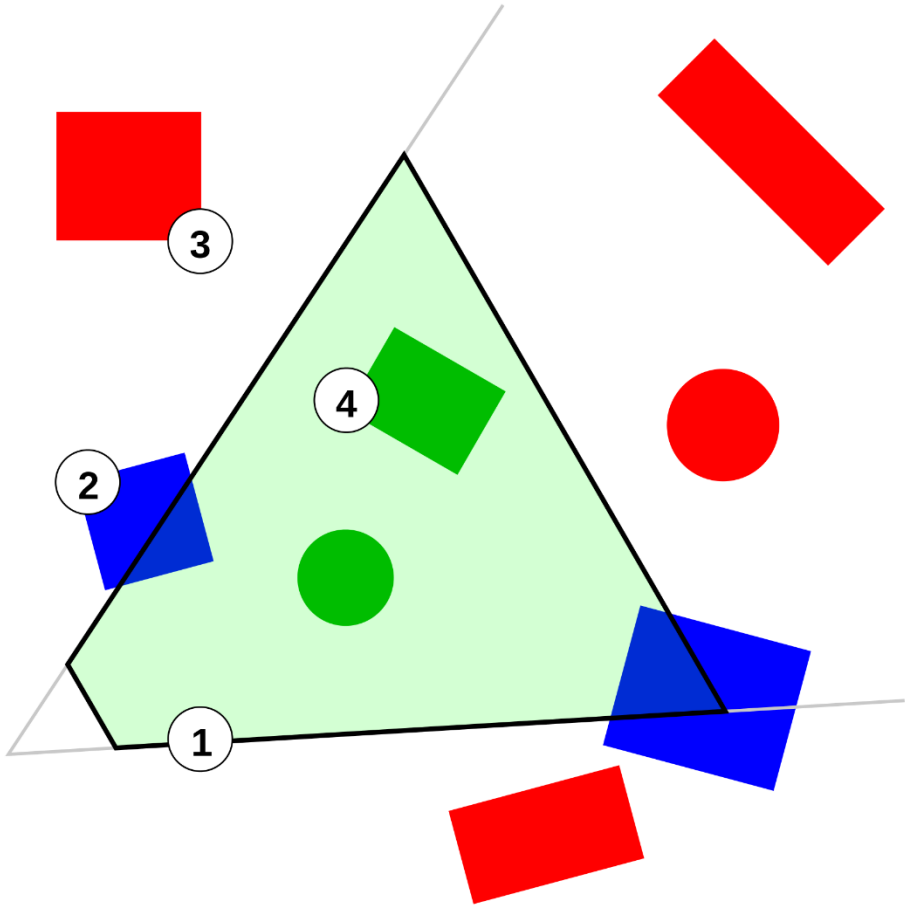


Depth Sorting

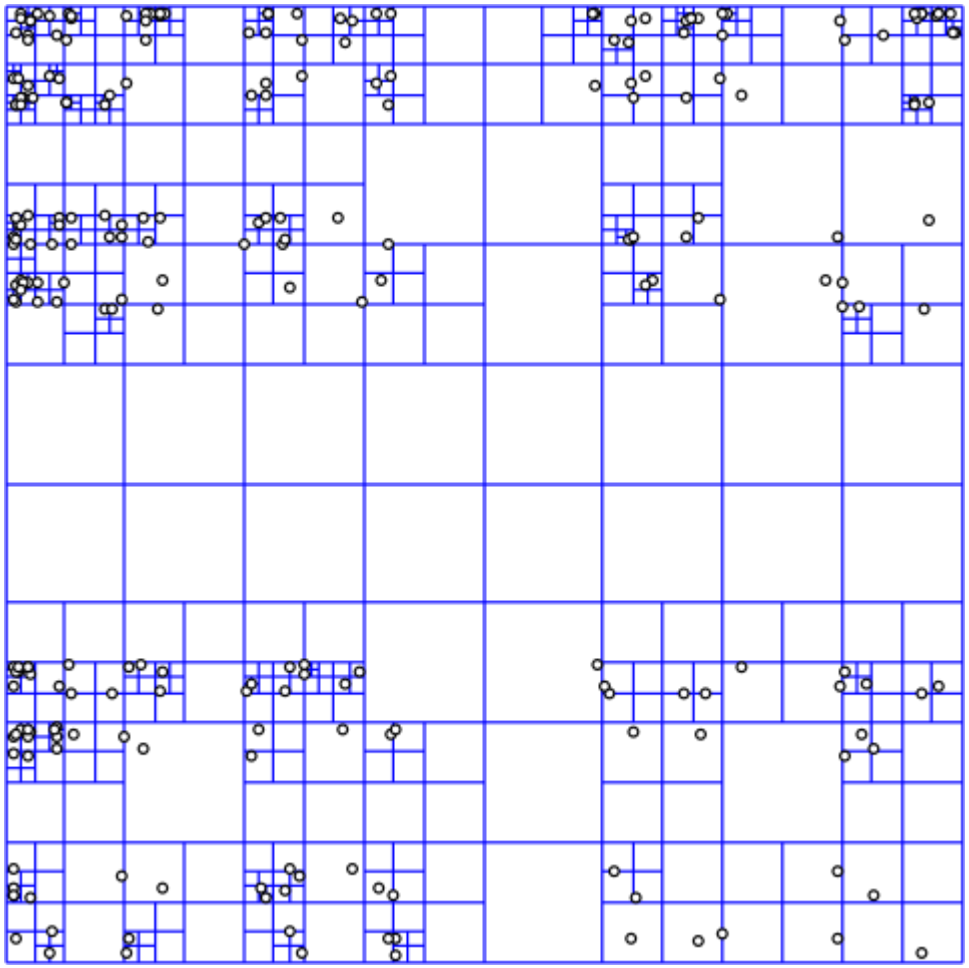
TECHNISCHE UNIVERSITÄT DARMSTADT

- **Sort only complete meshes**
  - For performance reasons
- **Sort by distance from camera to object**
- **Draw nearest objects last**

Frustum Culling



Hierarchical Scenes



Quake

Optimization

TECHNISCHE UNIVERSITÄT DARMSTADT

```
foreach (object in world) {
  if (infrustum(object)) {
        render(object);
  }
}
```

- **Hierarchical structures slow on modern CPUs**
  - Cache misses

Occlusion Culling

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Computations can be expensive**

- **Precompute**
  - Unity
- **Rasterize in low resolution**
  - Software Occlusion Culling from intel
- **Occulsion Query**
  - Hardware feature
- **Deferred Rendering**
  - Hardware feature

- **Remember: Games need steady performance**

Operator overloading

TECHNISCHE UNIVERSITÄT DARMSTADT

```
class Number {
  …
};

Number operator+(Number a, Number b) {
  return …
}

Number a;
Number b;
Number c = a + b;
```

Operator overloading

TECHNISCHE UNIVERSITÄT DARMSTADT

```
class Number {
  Number operator++(); // ++num;
  Number operator++(int); // num++;
};

for (Number i = 0; i < 10; i++) { }
for (Number i = 0; i < 10; ++i) { } // maybe faster when ++ is overloaded
```

References

- New name for existing variable
- Same syntax for access

```
int a = 3;
Int& b = a;
b = 4;
// a == 4
```

References

TECHNISCHE UNIVERSITÄT DARMSTADT

```
void reference_test(int& b) {
  b = 4;
}

int a = 3;
reference_test(a);
// a == 4
```

References

TECHNISCHE UNIVERSITÄT DARMSTADT

- **In theory just an unchangeable reference**
  - Not a hardware level concept
  - Can often be removed by the optimizer
- **In practice works like restricted pointers**

- **Added to support map implementations**

```
class Map {
  int& operator[](int index);
};
```

Constness

TECHNISCHE UNIVERSITÄT DARMSTADT

const int a = 3;

a = 4; 

Constness

TECHNISCHE UNIVERSITÄT DARMSTADT

```
void bla(const int a) {
  a = 4;
}
```

Constness

TECHNISCHE UNIVERSITÄT DARMSTADT

```
const char* bla1 = „bla“;

char* const bla2 = „bla“;

bla1 = „blub“;

bla2[0] = ‚g‘;
```

Constness

```
void bla(const int& a) {
  …
}
```

- **Hint for the compiler: Do what you want**
  - Copying a value can be faster than using a pointer

Constness

TECHNISCHE UNIVERSITÄT DARMSTADT

```
class A {
  void method1() const {
        a = 3;
  }

  void method2() const {
        b = 3;
  }

  int a;
  mutable int b;
};
```

Constness

TECHNISCHE UNIVERSITÄT DARMSTADT

```
class A {
  void method1() const;
  void method1();
};

A a;
a.method1();

const A b;
b.method1();
```

Templates

TECHNISCHE UNIVERSITÄT DARMSTADT

```
template<class T> class A {
  void method1() {

  }

  void method2();
};

A<int> a;
```

Templates

TECHNISCHE UNIVERSITÄT DARMSTADT

```
template<int i> void bla() { … }

bla<3>();
```

static

TECHNISCHE UNIVERSITÄT DARMSTADT

```
// functions.h

void func1() { }

static void func2() { }

inline void func3() { }

namespace {
  void func4() { }
}
```