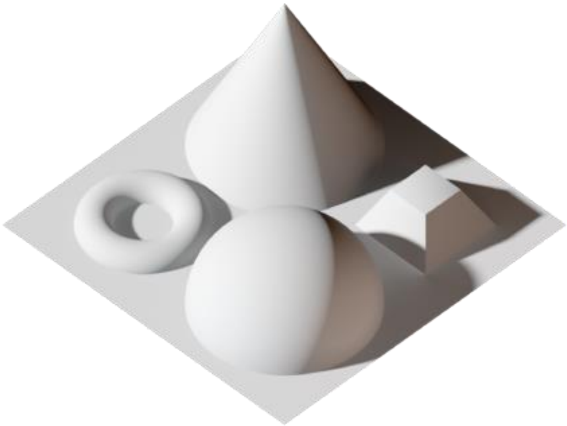
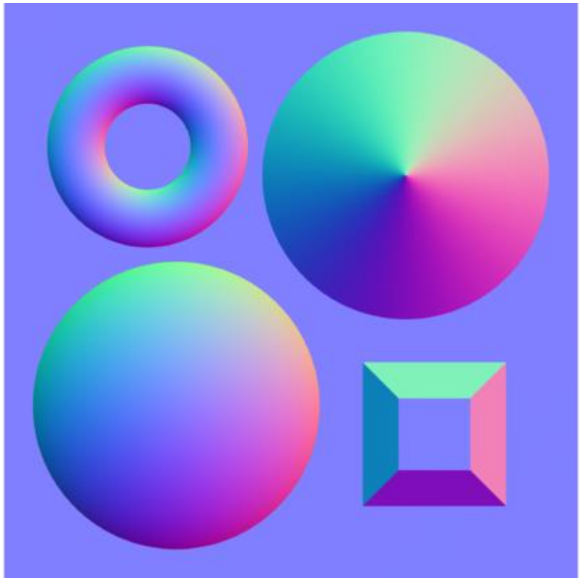
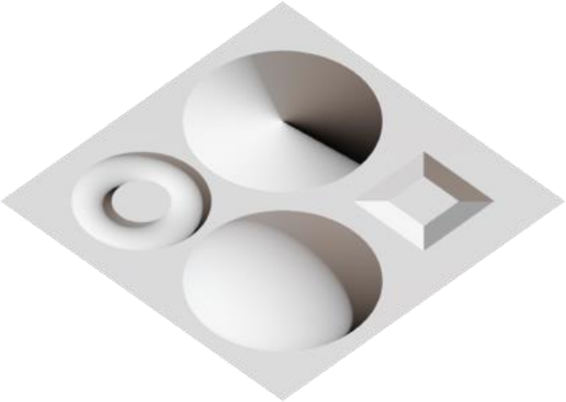
Game Technology

Lecture 6 – 28.11.2015
Bumps and Animations







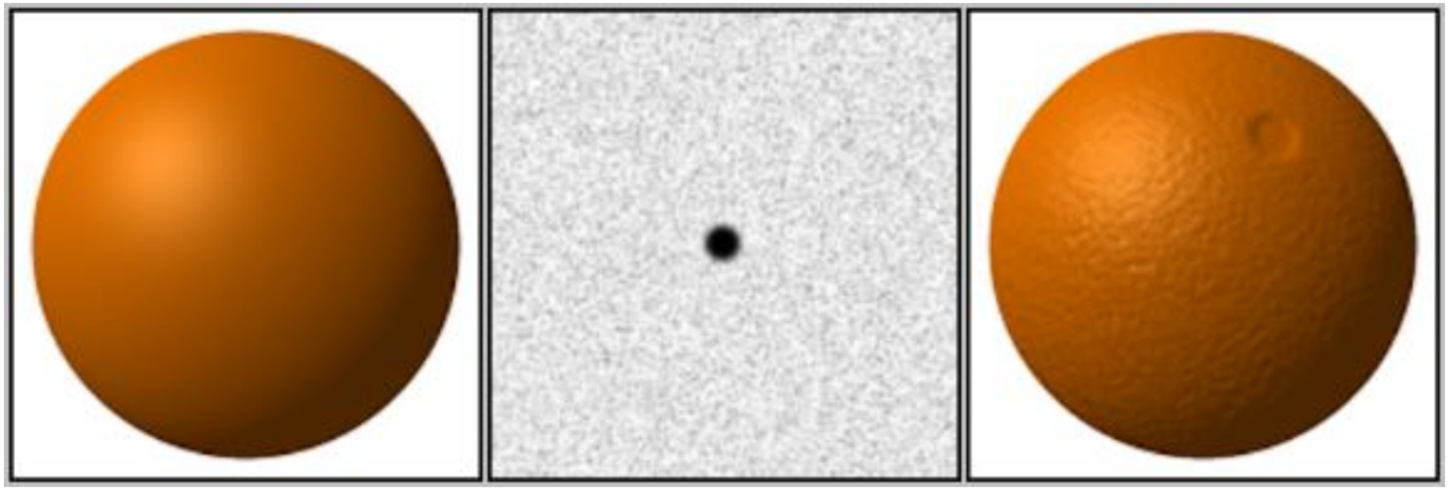
Dipl-Inf. Robert Konrad
Dr.-Ing. Florian Mehm

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

© author(s) of these slides including research results from the KOM research network and TU Darmstadt; otherwise it is specified at the respective slide
Template all v.3.4

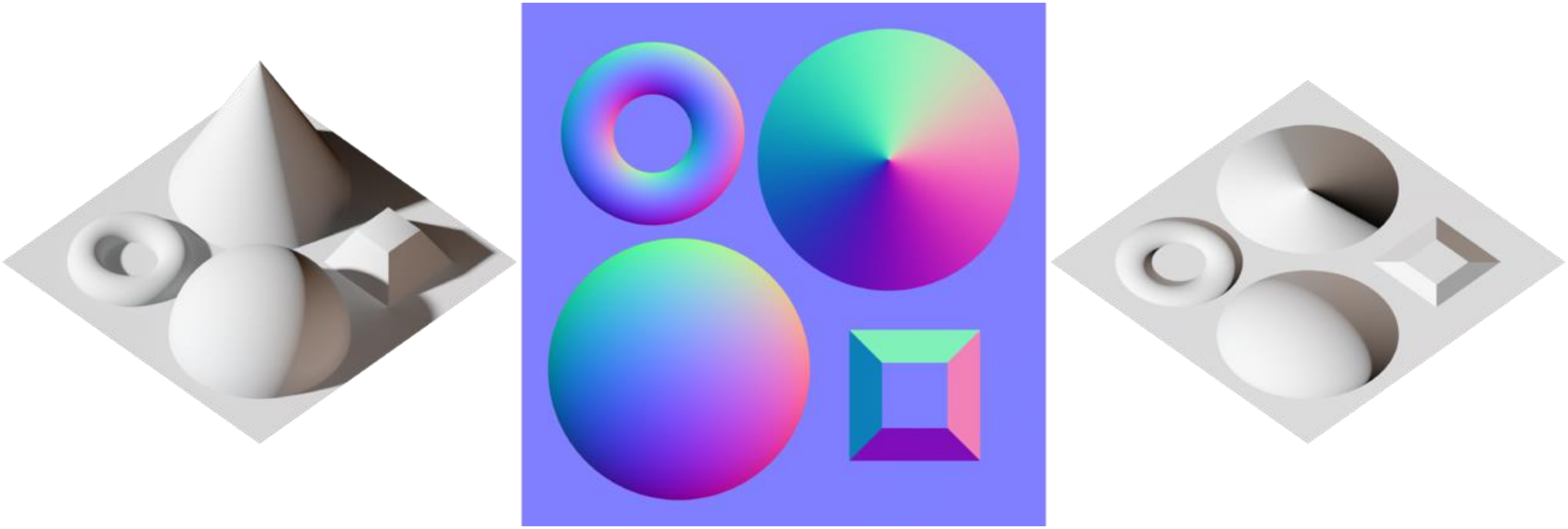
Bump Mapping

- Encode information about the surface
- Use during shading to simulate more detail than there is



Normal Maps

- Encode normals in the mesh
- Bake from high-poly mesh



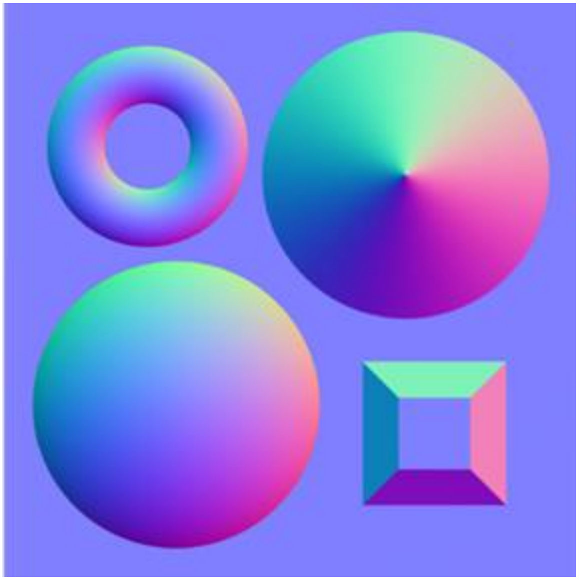
Normal Maps

Use a normal texture to encode the map

normal = 2 * color - 1;

Default color is blueish

- (128, 128, 255)
- Geometric interpretation: Perpendicular to the x-y-plane



Tangent Space

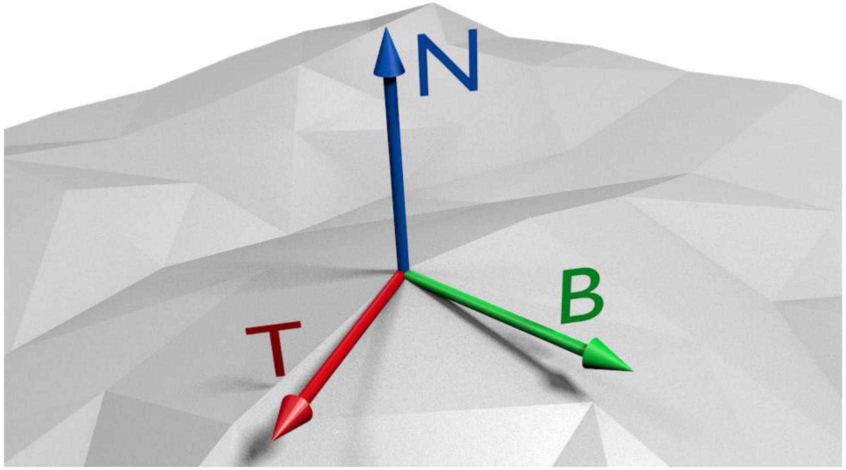
Defines coordinate systems orthogonal to the surface

Reuse texture coordinates:

deltaPos1 = deltaU1 * T + deltaV1 * B

deltaPos2 = deltaU2 * T + deltaV2 * B

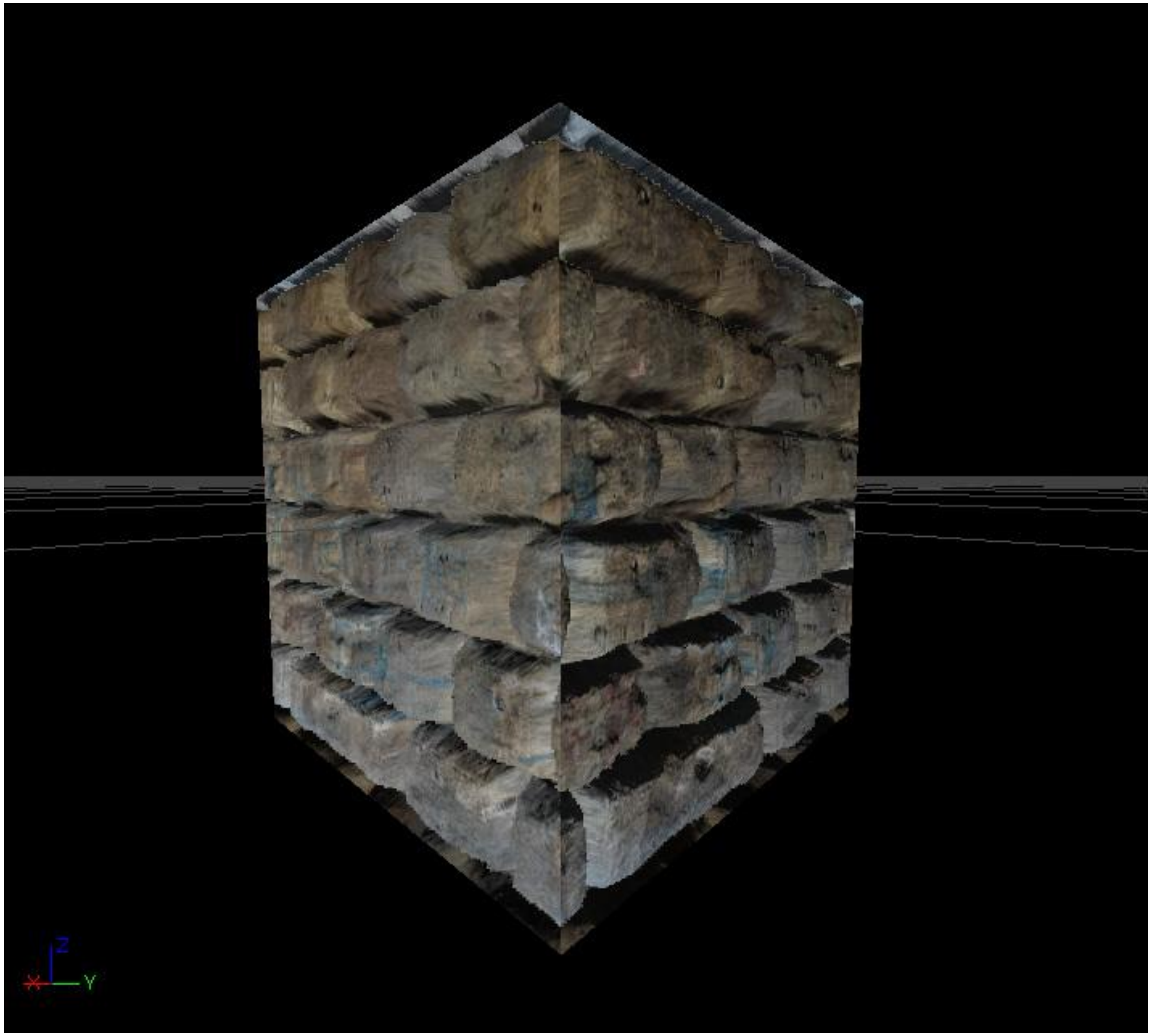
$$\begin{pmatrix} T & B & N \\ T & B & N \\ T & B & N \end{pmatrix}$$



Normal Mapping

Doom 3, 2004

Parallax Occlusion Mapping

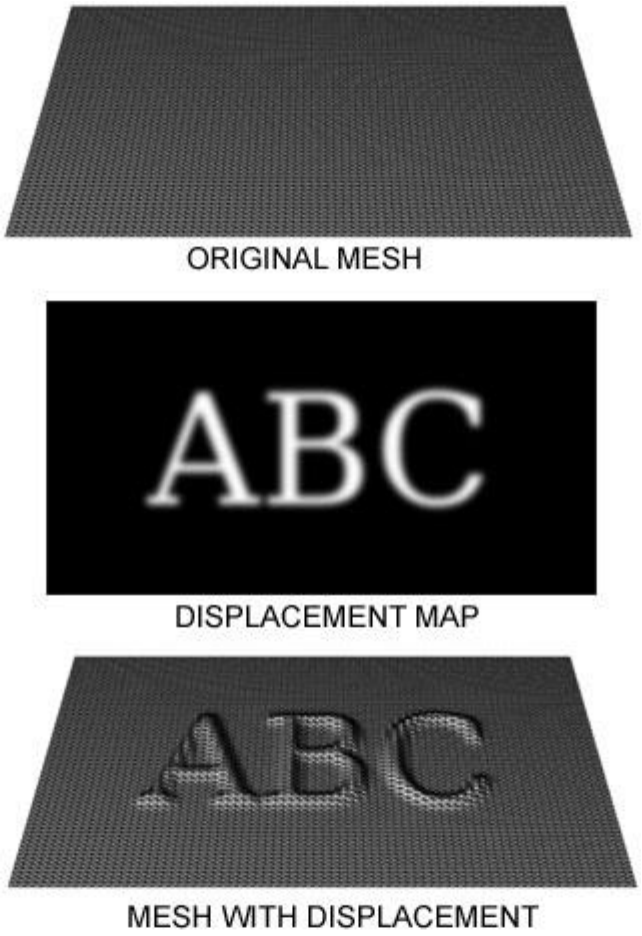


Displacement Mapping

Bump/normal mapping add the illusion of depth during shading

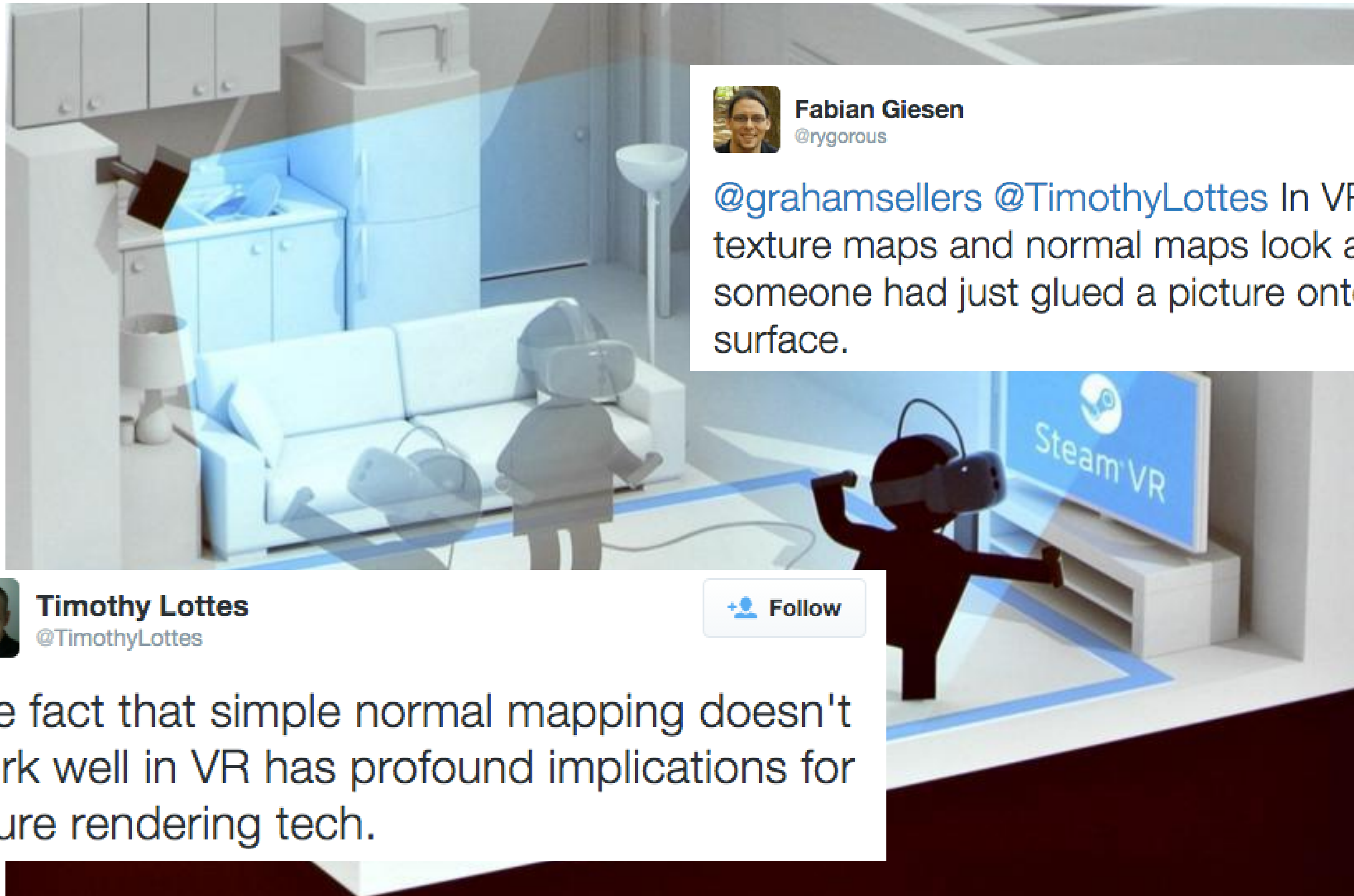
Displacement mapping adds actual geometry

Really useful if GPU supports it



VR - The death of normal maps?

TECHNISCHE UNIVERSITÄT DARMSTADT



Fabian Giesen @rygorous — Follow

@grahamsellers @TimothyLottes In VR, texture maps and normal maps look as if someone had just glued a picture onto a flat surface.

Timothy Lottes @TimothyLottes — Follow

The fact that simple normal mapping doesn't work well in VR has profound implications for future rendering tech.

VR - The death of normal maps?

Normals maps don't supply real height differences

- No parallax

User can get close to most surfaces, can test for parallax with head movements

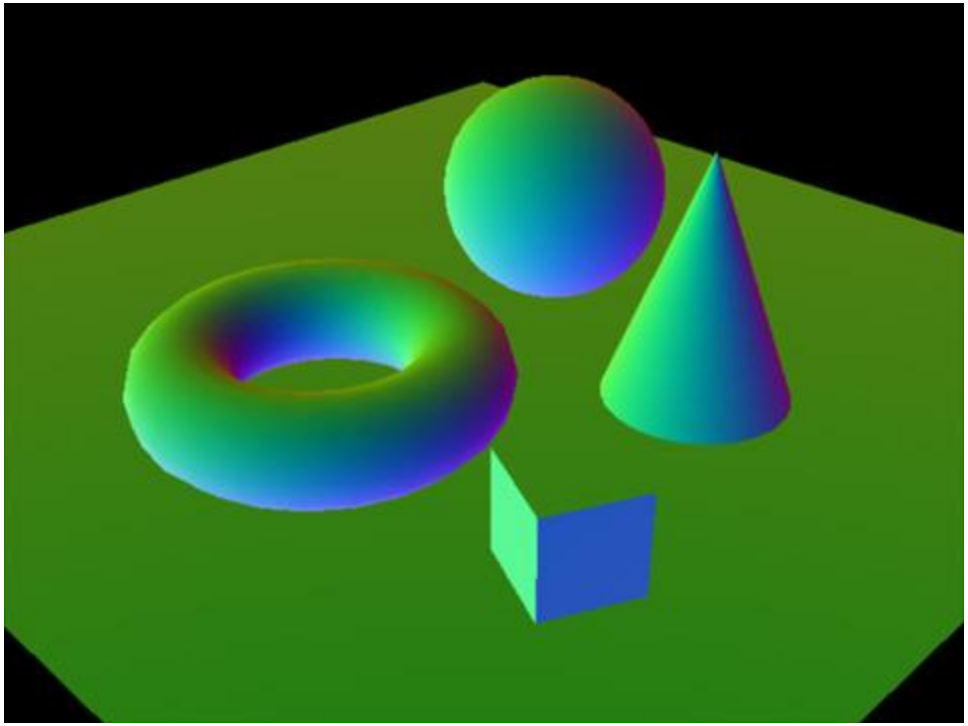
Solutions

- Use displacement or higher resolution meshes for everything that is close
- Use normal maps for fine details and relatively far-away surfaces

Deferred Shading

TECHNISCHE UNIVERSITÄT DARMSTADT

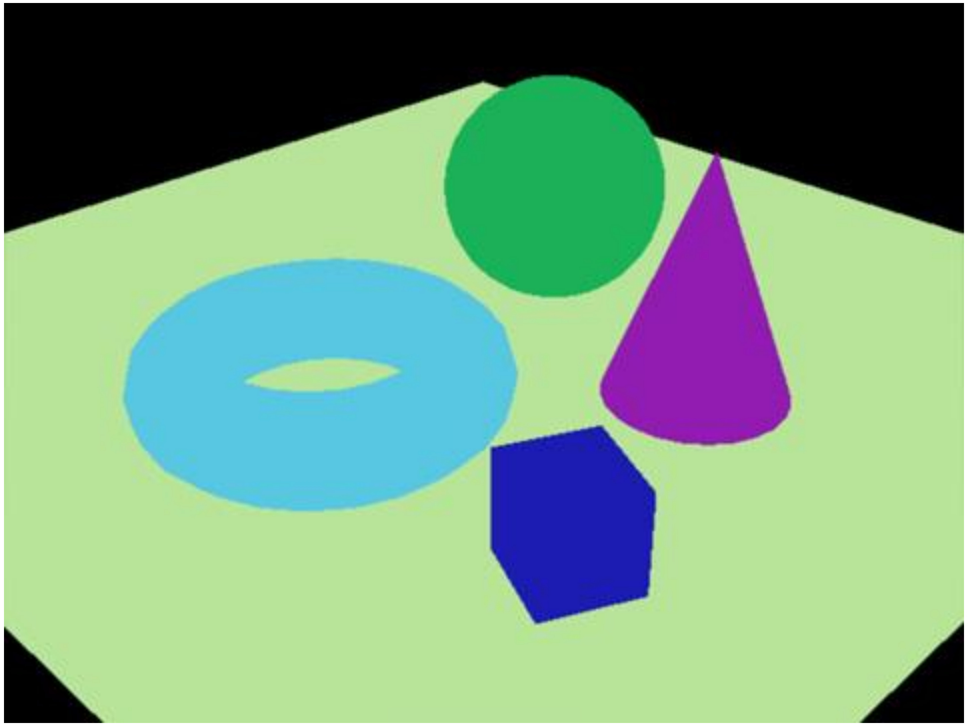
Buffer for normals



Deferred Shading

TECHNISCHE UNIVERSITÄT DARMSTADT

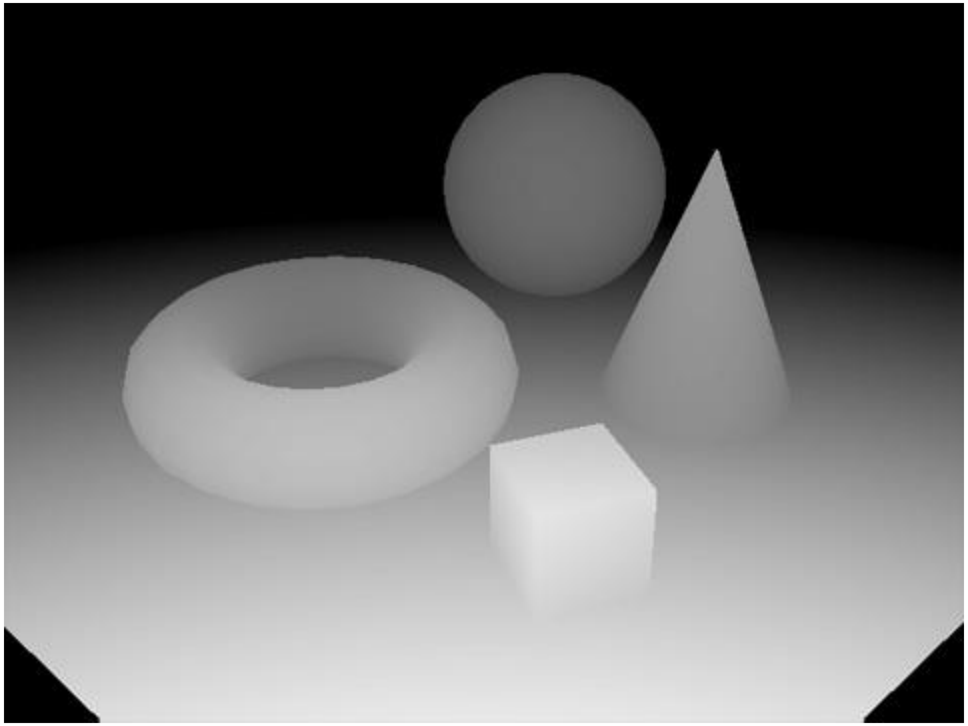
Buffer for different objects



Deferred Shading

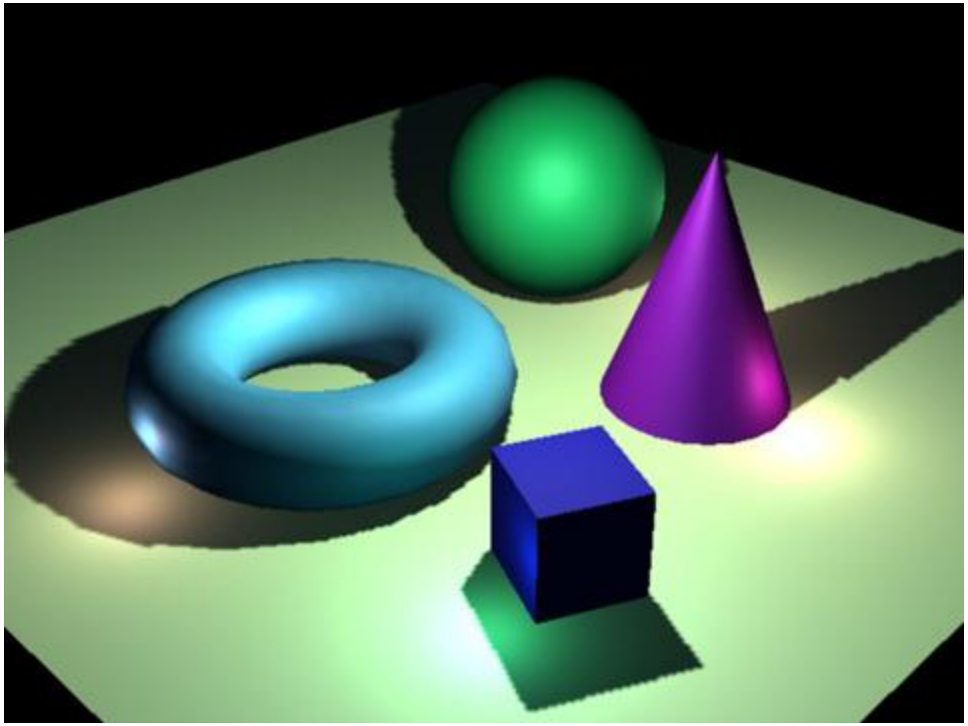
TECHNISCHE UNIVERSITÄT DARMSTADT

Depth Buffer



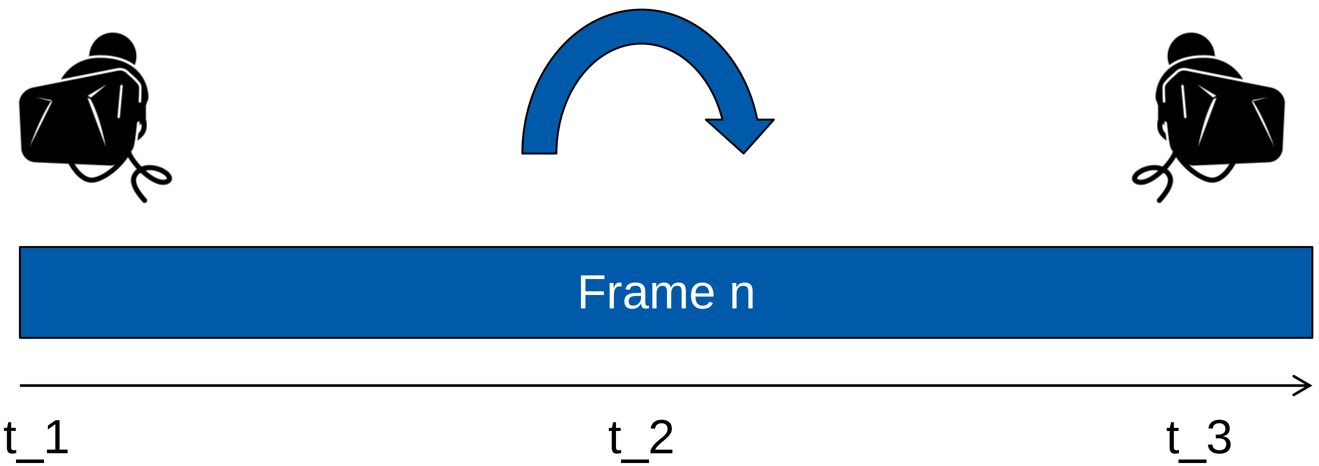
Deferred Shading

Carry out lighting calculations on the buffers



Virtual Reality Frame Time

Which head position to use?



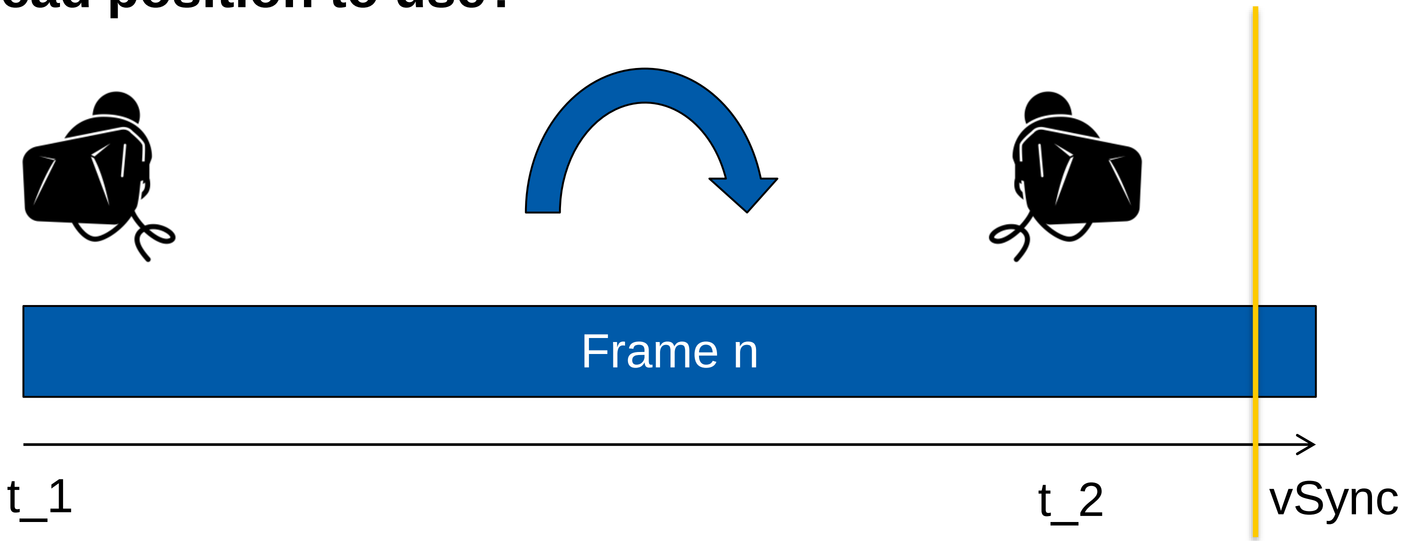
Future positions often predicted by HMD

- E.g. using the measured acceleration, physiological models
- Can use timewarp mechanism → will look at this in a later lecture

Icon by Hans Gerhard Meier, Noun Project

VR Frame Time: Time Warp

Which head position to use?



t1: Render image including depth buffer

t2: Update head position, reproject image

Icon by Hans Gerhard Meier, Noun Project

Time warp

Render to texture

Project back from 2D to 3D

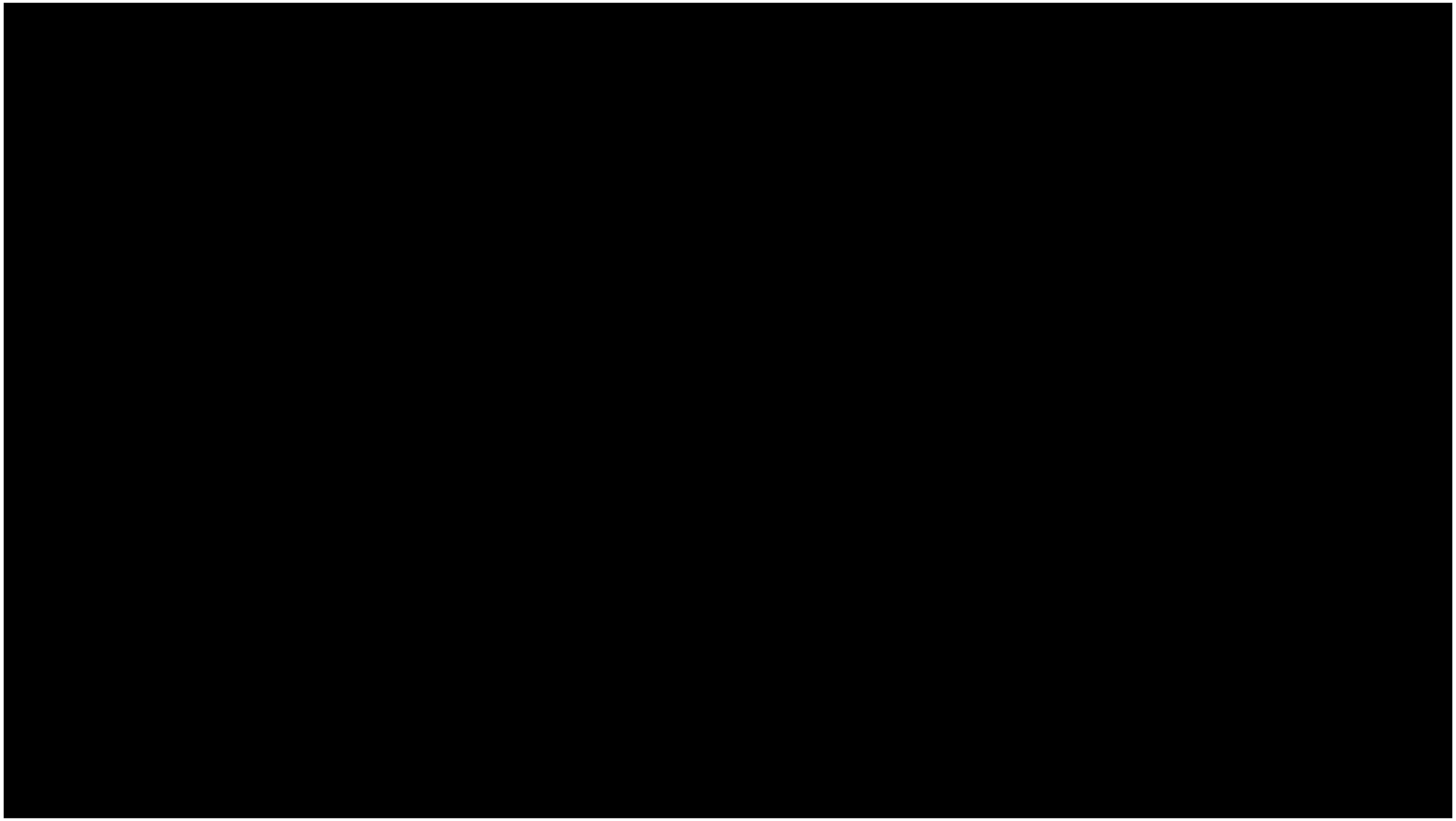
Apply new camera rotation (ideally only rotation)

Re-project to 3D

"Pulling in black"

- We only have a 2D image as the reference
- Pixels that are occluded are not in the image – "shadowed"
- If we move too fast or don't use pure rotation: We have nothing to interpolate with
  - Display black
  - Display blend of nearby colors
  - …

Time warp explanation



https://www.youtube.com/watch?v=WvtEXMlQQtI

Deferred Shading

TECHNISCHE UNIVERSITÄT DARMSTADT

Render the whole geometry into a (set of) buffer(s) (G-buffer), including

- Normals
- Colors
- Texture coordinates
- …

Calculate the shading, for each pixel once and only for the lights that influence the pixel

--> Main difference to forward rendering

No need to render everything for each new light

MVP

projection * view * model

Animate the model matrix to animate an object

Animate the view matrix to change the camera‘s viewpoint

Animate the projection matrix for FOV changes (scopes, binoculars)

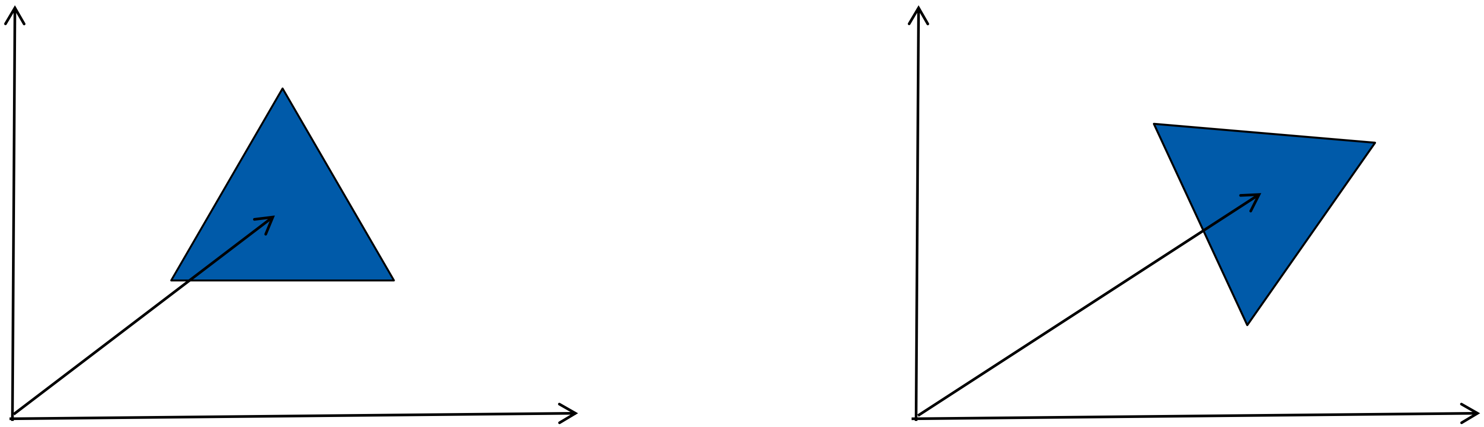
Be careful about the order

→ Can be reversed depending on matrix layout

Rotation Off-Center

model = (translate to end position) * rotation
*** (translate rotation center to 0)**

Needed when the object is to be moved off-center (pivot point not at the model's origin)



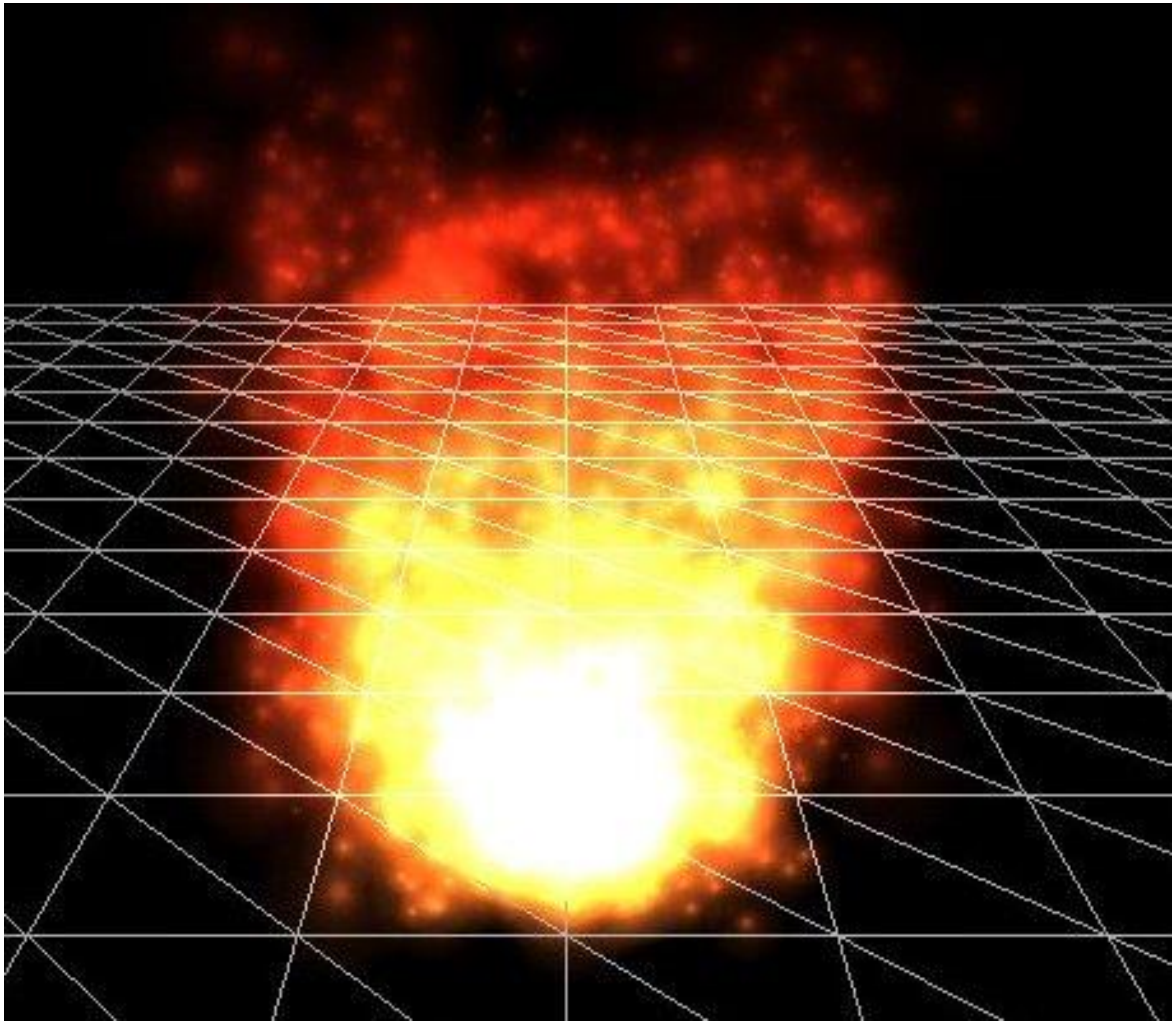
Scale

Super Mario 64, 2004

Shear

Motor Toon Grand Prix, 1994

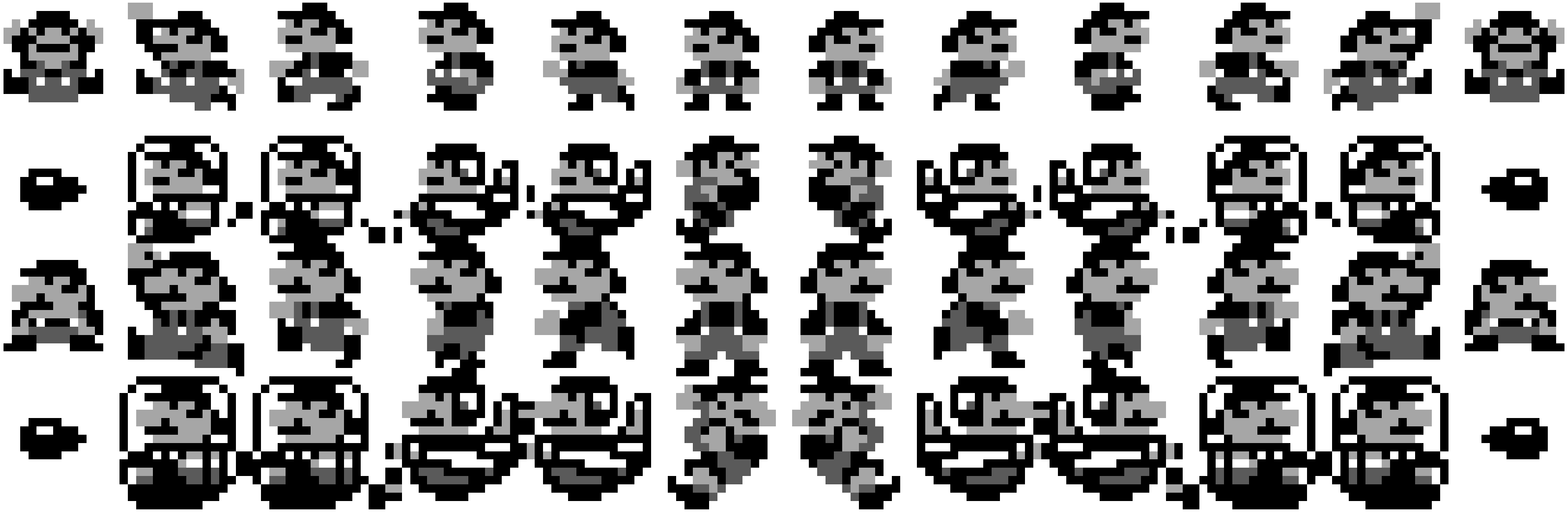
Particle Systems (more in 2 lectures)



Fluids

https://www.youtube.com/watch?v=7q8s7DMOOD4

Characters - Sprite Sheets



Vertex Animations

Quake, 1996

Vertex Animations

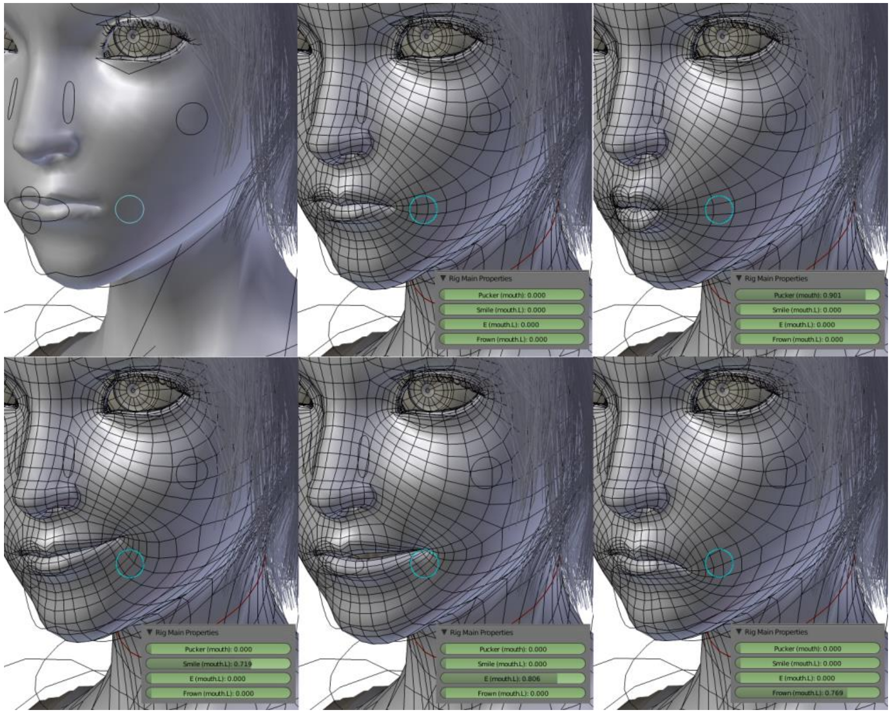
100 frames * 100000 vertices = lots of data

Blend Vertex Positions

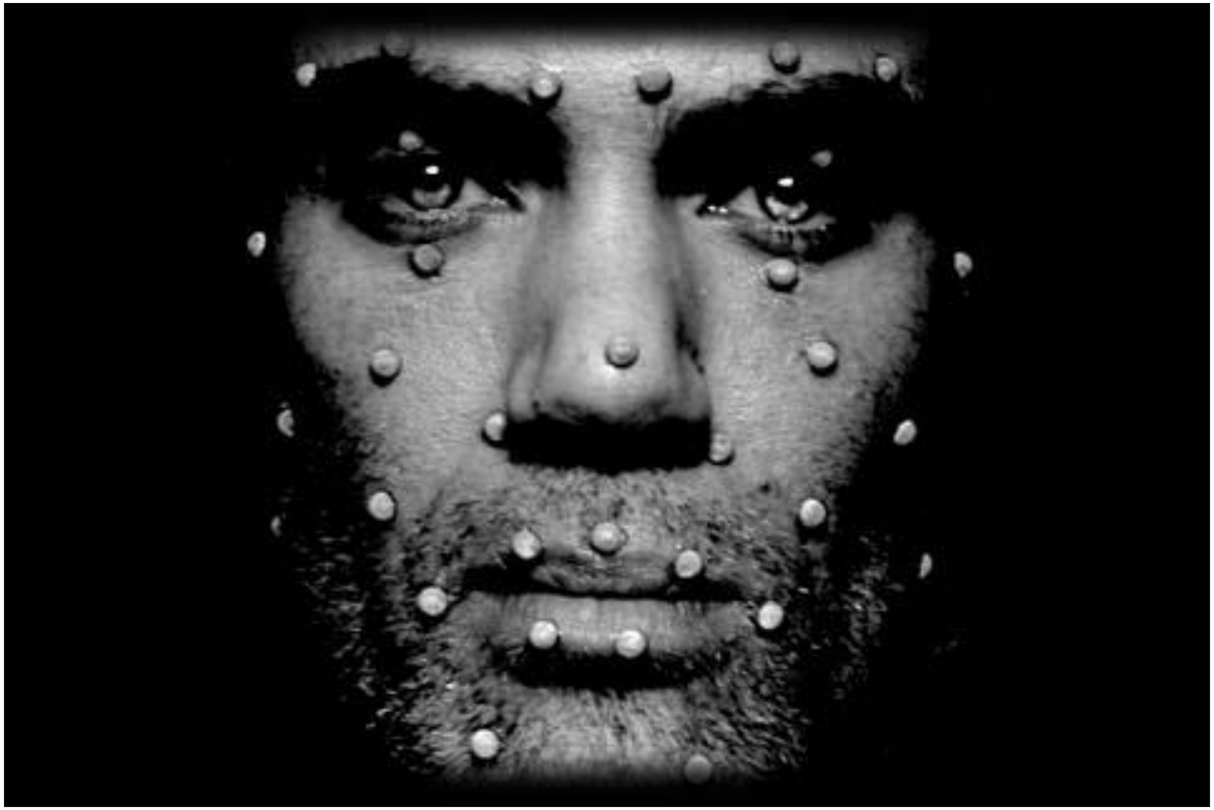
Dragon Quest 8, 2004

Faces

Morph target animation



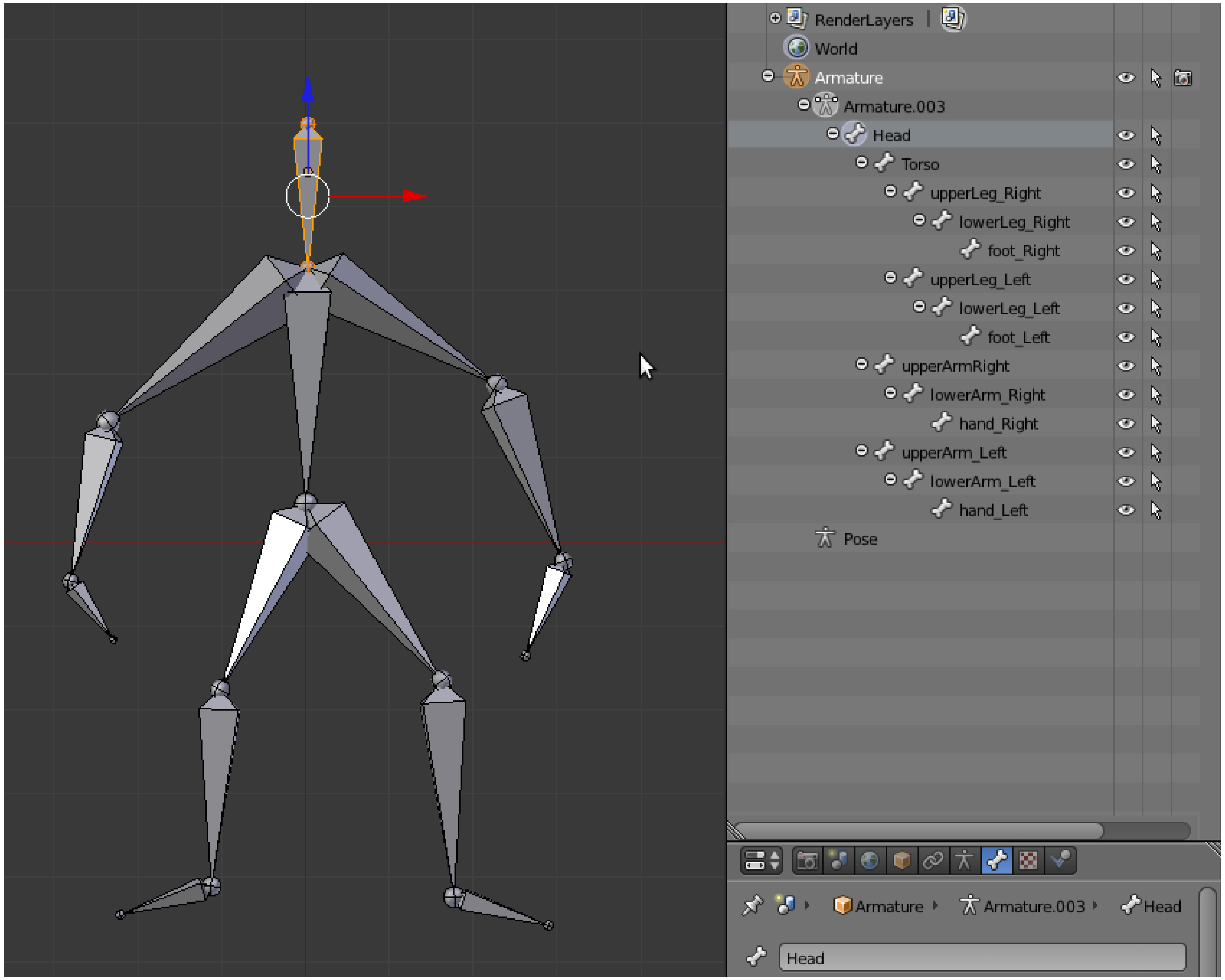
Performance Capture



Skeletal Animation

Skeletal Animation

Skeletal Animation



Skeletal Animation

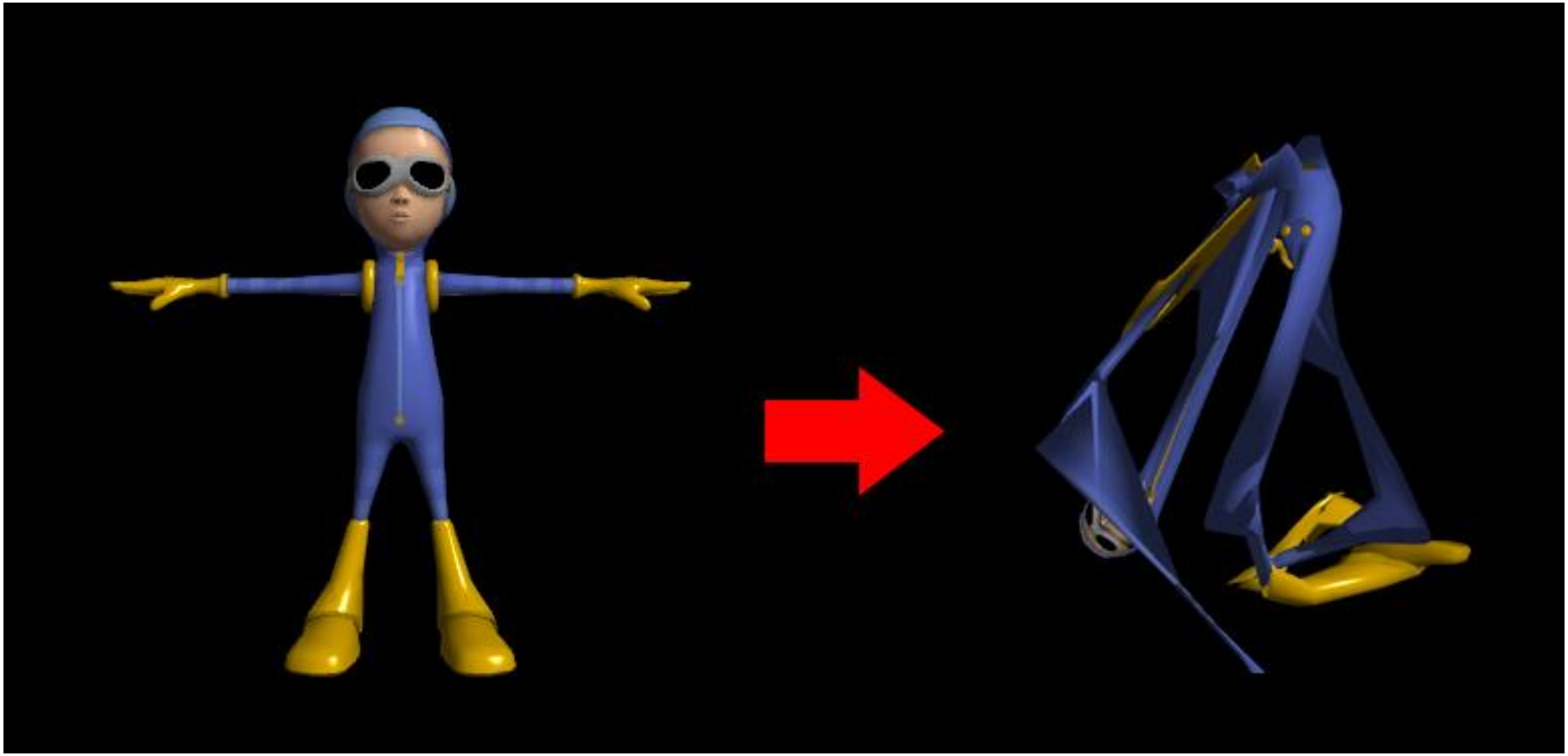
One bone – One Transformation matrix

- Or just a rotation
  - Depends on your gfx tool

Animation

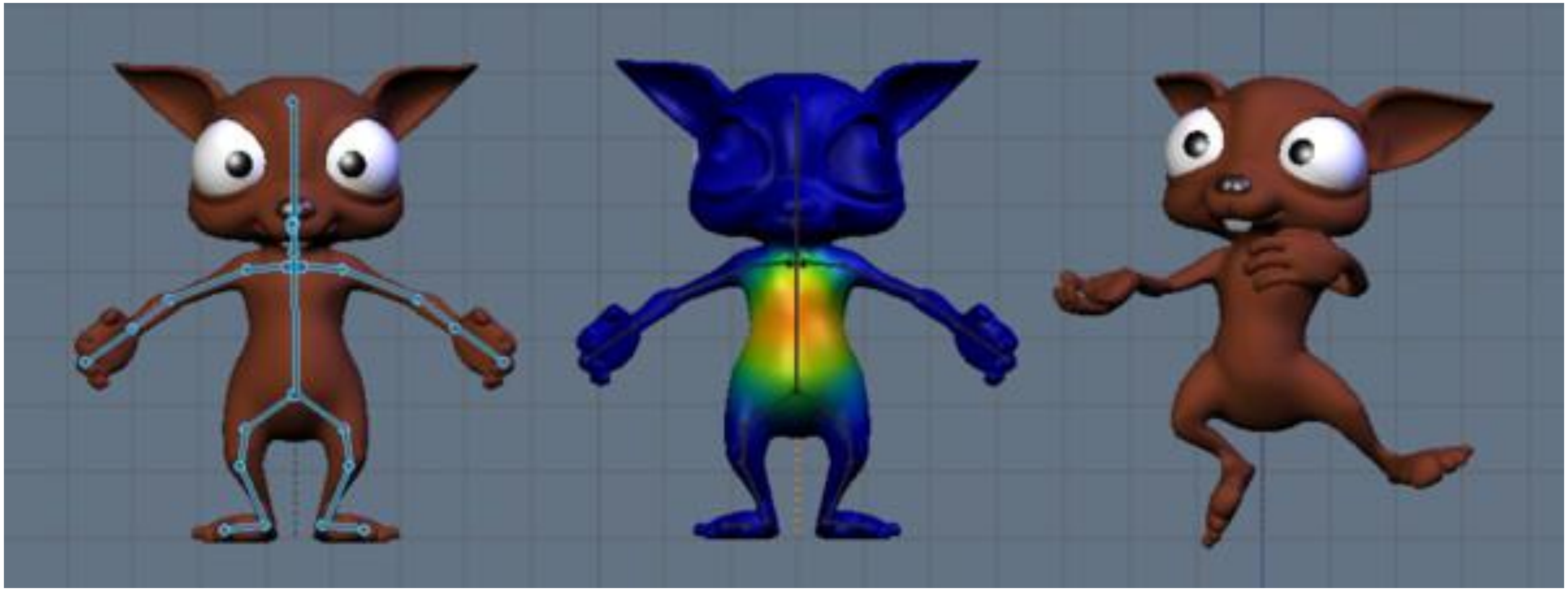
- Just an array of small transformation matrix arrays
- Framerate can be low
  - Interpolation works fine

Skinning



Skinning

TECHNISCHE UNIVERSITÄT DARMSTADT



Skinning

For each vertex

- Array of (weight, index)

At start

- Compute inverse of every bone transform matrix

For animation step

- Compute new transform matrices
- For each bone compute new transform * old inverse

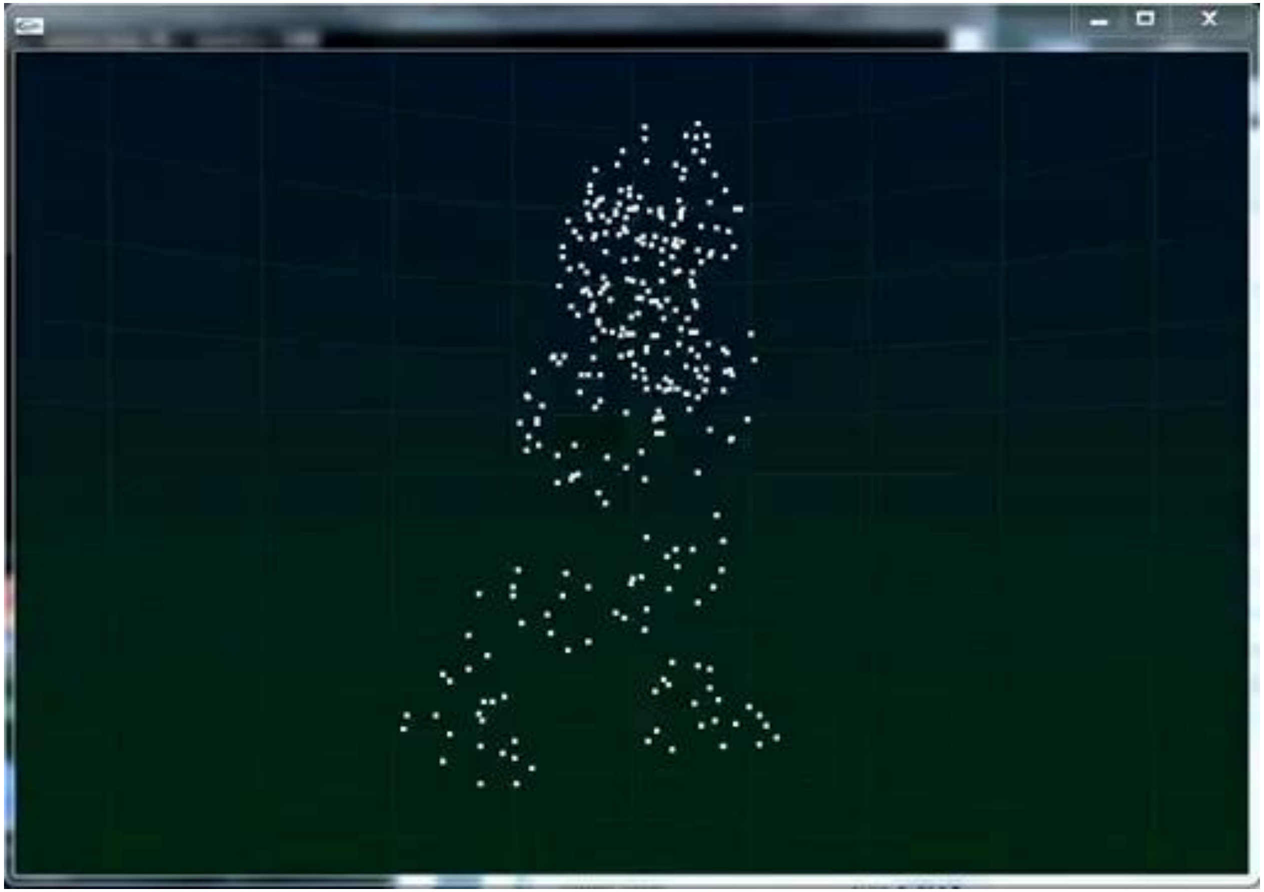
For each vertex

- For each weight
  - Compute (new transform * old inverse * vertex) * weight
- Sum it up

Quiz: Which animation?

https://www.youtube.com/watch?v=J8JPVj-AYTw

Quiz: Which animation?



https://www.youtube.com/watch?v=AxEdZiQlSOA

Root motion

Variant 1: Save motion of root bone during animation

- Motion is "hard-coded“
- Can be fine-tuned by the designers, e.g. different speeds at different points

Variant 2: No root motion, character stays in one place

- Can be blended easier
- Can be used more versatile
- Problems
  - Footskating
  - Accelerations
  - ...

Motion Capturing

Motion Retargeting

https://www.youtube.com/watch?v=Vn-vVzMGgec

Inverse Kinematics

Forward Kinematics

Input: Bone rotations

Output: Final positions

Inverse Kinematics

Input: Final positions

Output: Bone rotations

Inverse Kinematics

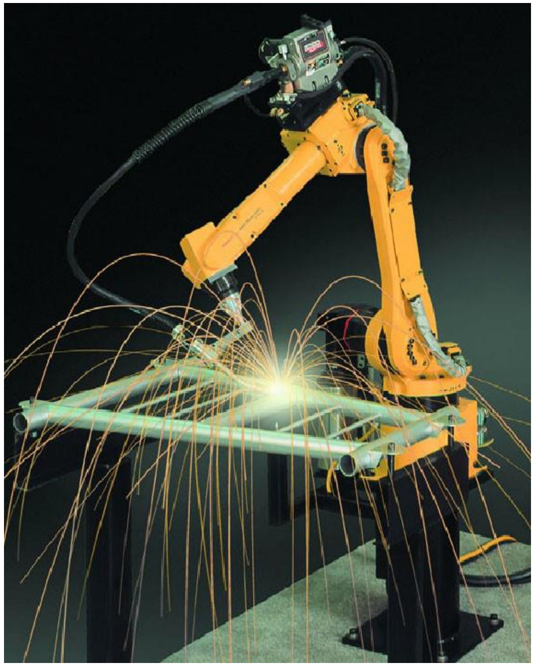
Super Mario Sunshine, 2002

Inverse Kinematics

TECHNISCHE UNIVERSITÄT DARMSTADT

Numerical, iterative solution using Jacobi Matrix

- See Robotics Lectures



Unexpected Deformations

„Achselhölle“

Spherical Skinning

- http://www.crytek.com/download/izfrey_siggraph2011.pdf

Dual Quaternion Skinning

- https://www.youtube.com/watch?v=4e_ToPH-I5o

Muscles

Muscles

Physical Animations

Goat Simulator, 2014

Hair, Cloth,…

Tomb Raider, 2013

Rag Dolls

QWOP (2008)

Rag Doll ←→ Skeletal Animation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Player hit → rag doll simulation
- Wait
- Blend from current positions to nearest known animation state
- Play animation

Mixture between forwards and physically based

During regular animation

→ Driven by forward animation

Physical Interactions

- On becoming unconscious
- On stumbling
- → Switch to ragdoll behaviour

On regaining control

→ Blend to the forward kinematics again

Summary

Normals maps, bump mapping

- Increase the visual quality without increasing vertex count
- Bake from higher-poly version or paint/generate

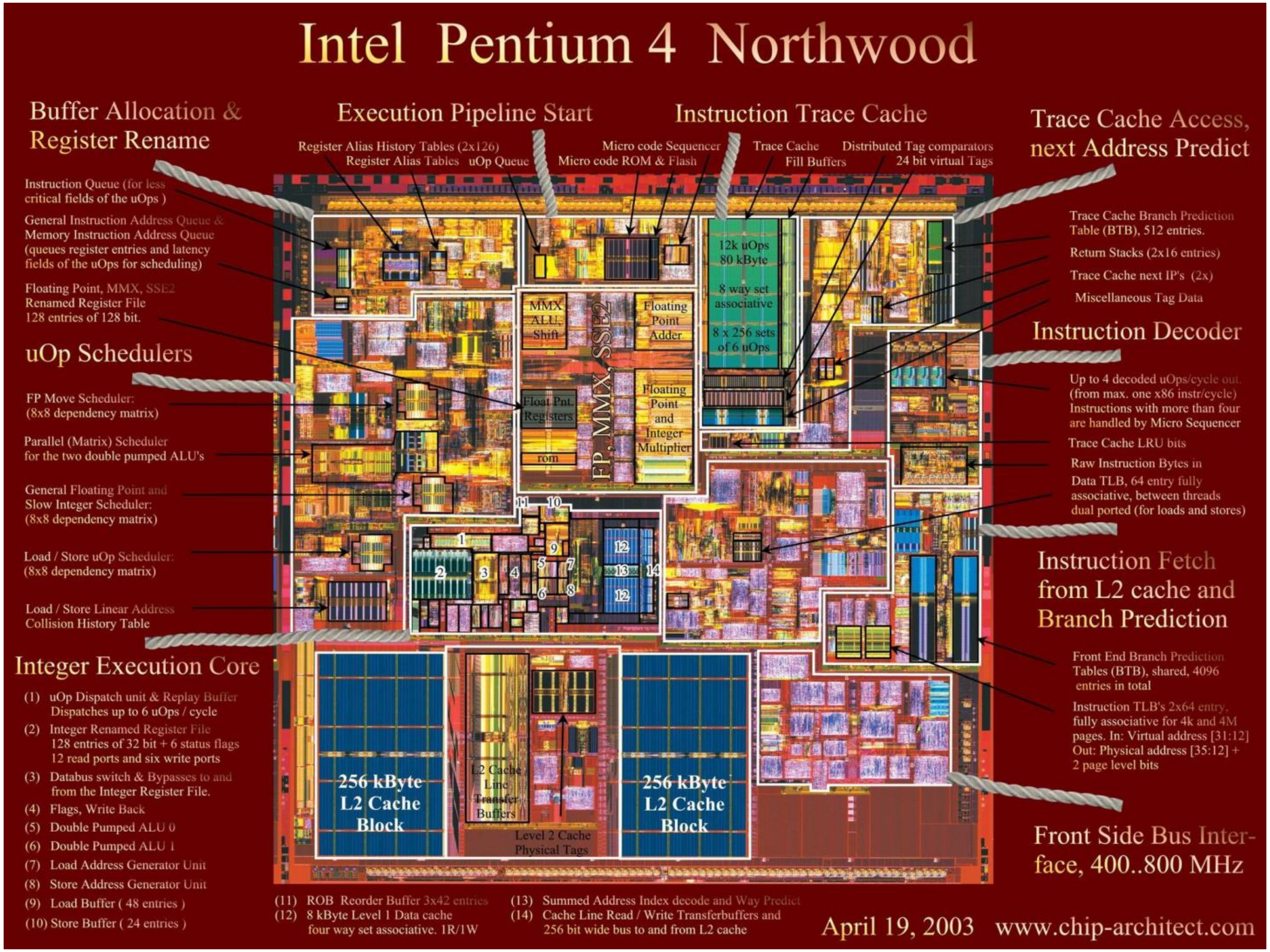
Displacement maps

- Increase visual quality by increasing vertex count
  - But our badass GPU does it for us

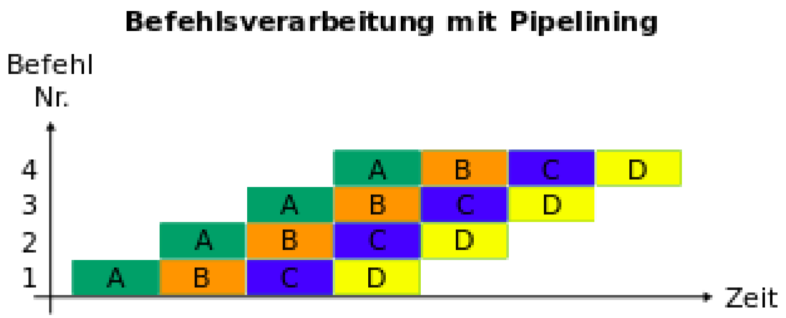
Animation techniques

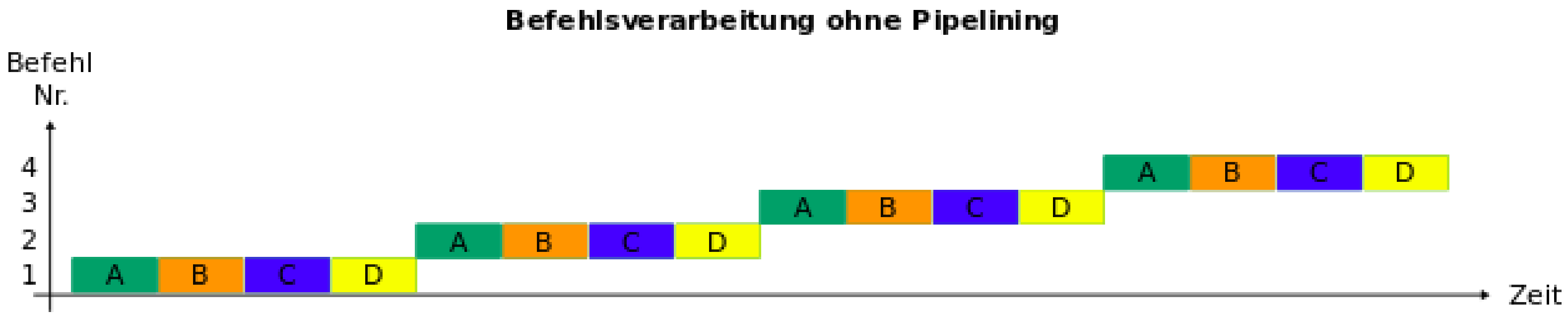
- Morph Targets
- Skeletal animation

CPU internals



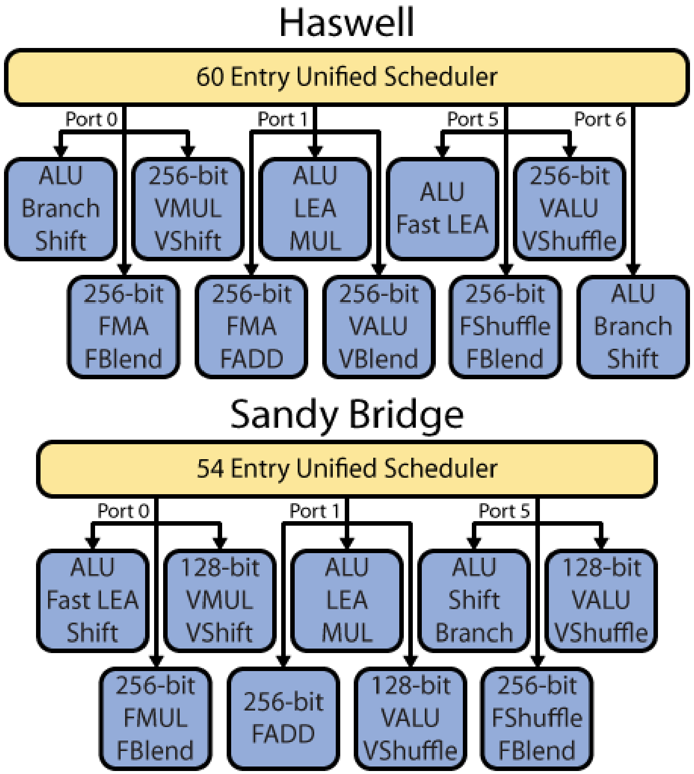
Pipelining


Befehlsverarbeitung mit Pipelining
Befehl Nr.
4
3
2
1
A
B
C
D
Zeit


Befehlsverarbeitung ohne Pipelining
Befehl Nr.
4
3
2
1
A
B
C
D
Zeit

Multiple Execution Units

- „ Note that Figure 3 does not show every execution unit, due to space limitations.“ (from http://www.realworldtech.com/haswell-cpu/4/)



Hazards

Structural Hazards

- Out of hardware

Data Hazards

- Data dependencies

Control Hazards

- Dynamic branching

Structural Hazards

Example

- One command is in the fetch state and wants to read memory
- One command wants to write to the memory

Modern CPUs add more ALUs

Already at a very high level

Data Hazards

TECHNISCHE UNIVERSITÄT DARMSTADT

Sometimes just register uses, but not real data dependencies

#	Instruction
1	R1 = M[1024]
2	R1 = R1 + 2
3	M[1032] = R1
4	R1 = M[2048]
5	R1 = R1 + 4
6	M[2056] = R1

→

#	Instruction	#	Instruction
1	R1 = M[1024]	4	R2 = M[2048]
2	R1 = R1 + 2	5	R2 = R2 + 4
3	M[1032] = R1	6	M[2056] = R2

→ Register renaming

- CPU uses more registers internally than can be directly addressed

Data Hazards

Compiler can help

- Reorder instructions
- Depends highly on CPU

Out-of-Order CPUs

- Can reorder instructions themselves
- Can incorporate current situation in decisions
- All current x86 CPUs are out-of-order
- More and more ARM CPUs are out-of-order
- PS360 are in-order

Control Hazards

TECHNISCHE UNIVERSITÄT DARMSTADT

Speculative execution

- Branch Prediction more and more sophisticated

Branch prediction example

```
int main()
{
    // generate data
    const unsigned arraySize = 32768;
    int data[arraySize];

    for (unsigned c = 0; c < arraySize; ++c)
        data[c] = std::rand() % 256;


    // !!! with this, the next loop runs faster
    std::sort(data, data + arraySize);


    // test
    clock_t start = clock();
    long long sum = 0;

    for (unsigned i = 0; i < 100000; ++i)
    {
        // primary loop
        for (unsigned c = 0; c < arraySize; ++c)
        {
            if (data[c] >= 128)
                sum += data[c];
        }
    }

    double elapsedTime = static_cast<double>(clock() - start) / CLOCKS_PER_SEC;

    std::cout << elapsedTime << std::endl;
    std::cout << "sum = " << sum << std::endl;
}
```

Memory Access

Cache Hierarchy critical for performance

L1 cache ~ KiloBytes

L2 cache ~ MegaBytes

Main memory ~ GigaBytes

L1 cache ~ 0.5 ns

L2 cache ~ 7 ns

Main memory ~ 100 ns

Memory Access

Access pattern prediction

- Works best when data is reused or for sequential data reads

Cache Lines

- Memory read in blocks
- ~ 64 Bytes
- Proper data alignment can help

POD

„Plain old data"

```
struct Data {
 int a;
 float b;
};
```

Predictable data structures

No constructor calls during array allocation

No additional data for virtual function pointers

Data data[64];

Linear data of 64*sizeof(Data) bytes

Memory alignment

TECHNISCHE UNIVERSITÄT DARMSTADT

Add unused data

Use system specific things

- posix_memalign(..)

Use alignas in C++ 11

```
struct alignas(16) Data {
 int a;
 float b;
};
```

alignas(128) char cacheline[128];

Packed structures

```
struct InsufficientParticle //total size 44 bytes
{
  bool visible;   //31 bits of padding
  Texture* texture;      //pointer to texture
  int alpha;             //only needs 0 to 256
  int type;              //enumeration – 4 possible types
  Vec3 position;
  Vec3 velocity;
}
```

Steve Rabin: Game Programming Gems 8: Game Optimization through the Lens of Memory of Data Access

Packed structures

```
struct Efficient particle //total size 30 bytes
{
  Vec3 position;
  Vec3 velocity;
  unsigned char alpha;//saved 3 bytes (0-256)
  unsigned char rotation; //saved 3 bytes (0-255 degrees)
  unsigned texture:4;   //saved 28 bits (texture index)
  unsigned type:2;      //saved 29 bits (enumeration)
  unsigned visible:1;   //saved 31 bits(single bit)
}
```

Cache efficience

Order from largest to smallest members to reduce padding

- sizeof(MyStruct) gives you the size including padding

Separate hot and cold data

- Keep hot data (often used) close together
- Watch out for gaps between hot data

Prefetch data

- Available on some platforms
- Make sure data is available on time

Lock the cache

- Some platforms, e.g. Wii, allow parts of the cache to be locked and managed by the application

Summary

CPU Internals

Hazards

- Structural Hazards
- Data Hazards
- Control Hazards

Memory access

Memory alignment