# Exporting 3D scenes from Maya to WebGL using Clang and LLVM

Jochen Wilhelmy

# Overview

- What ist WebGL?
- Principles of Illumination
- Inka3D Maya to WebGL exporter
- Examples
- Technical
- Demo

# What is WebGL?

## Short introduction

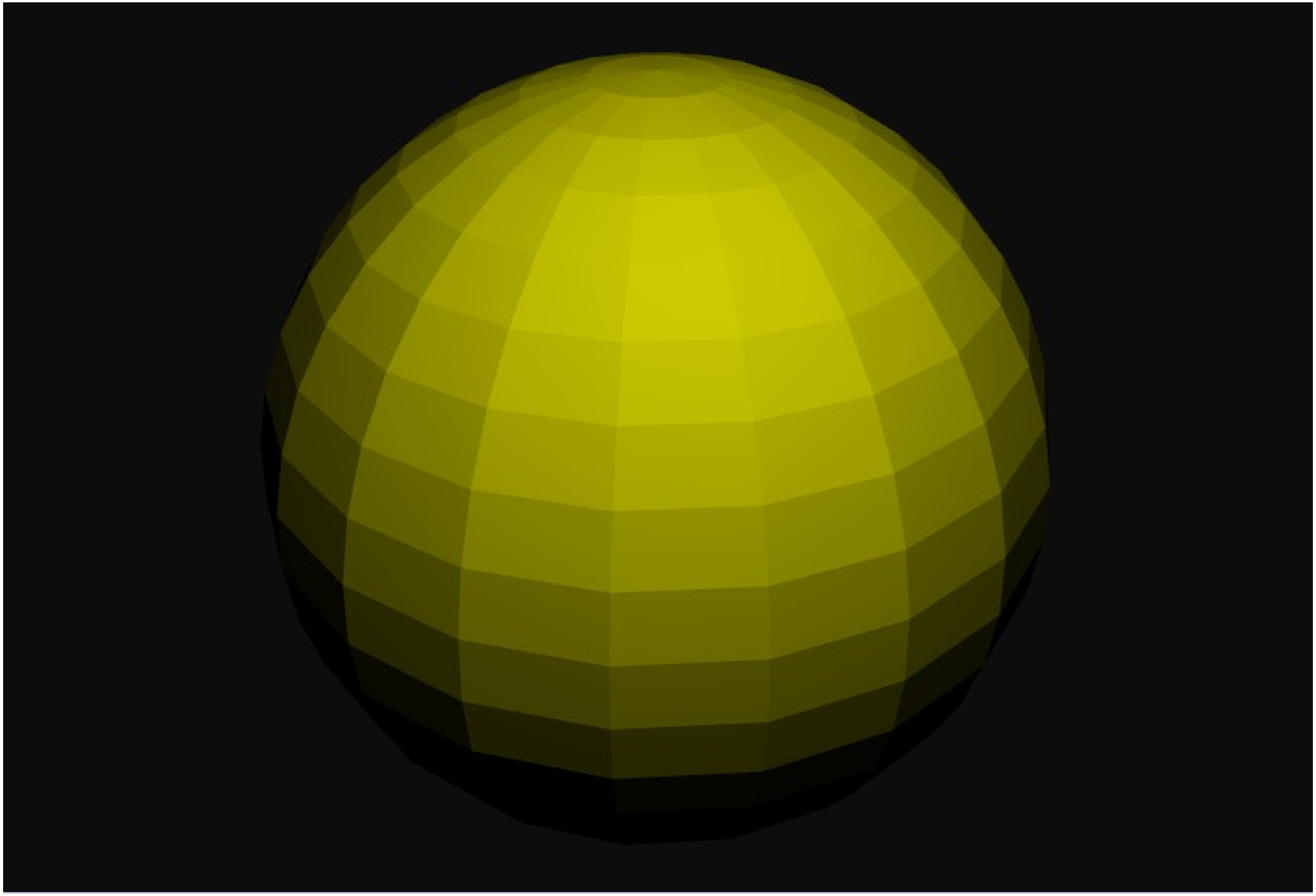
# What is WebGL?

- OpenGL for the web browser
- Specification completed in march 2011
- Programmed using JavaScript
- Legacy free and restriced to the necessary
- Can use DirectX on Windows using ANGLE (Almost Native Graphics Layer Engine)

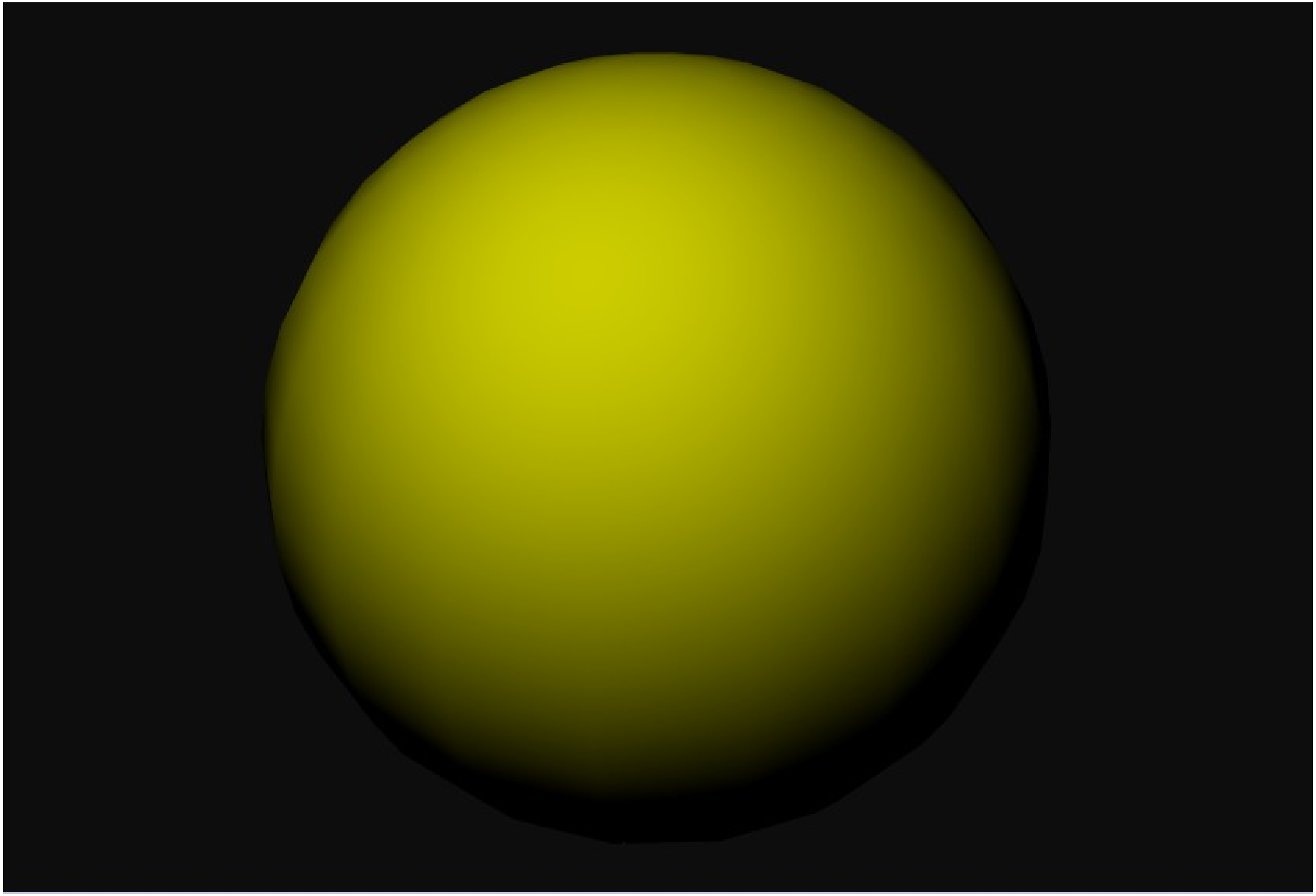
# Principles of Illumination

Illusory real images...

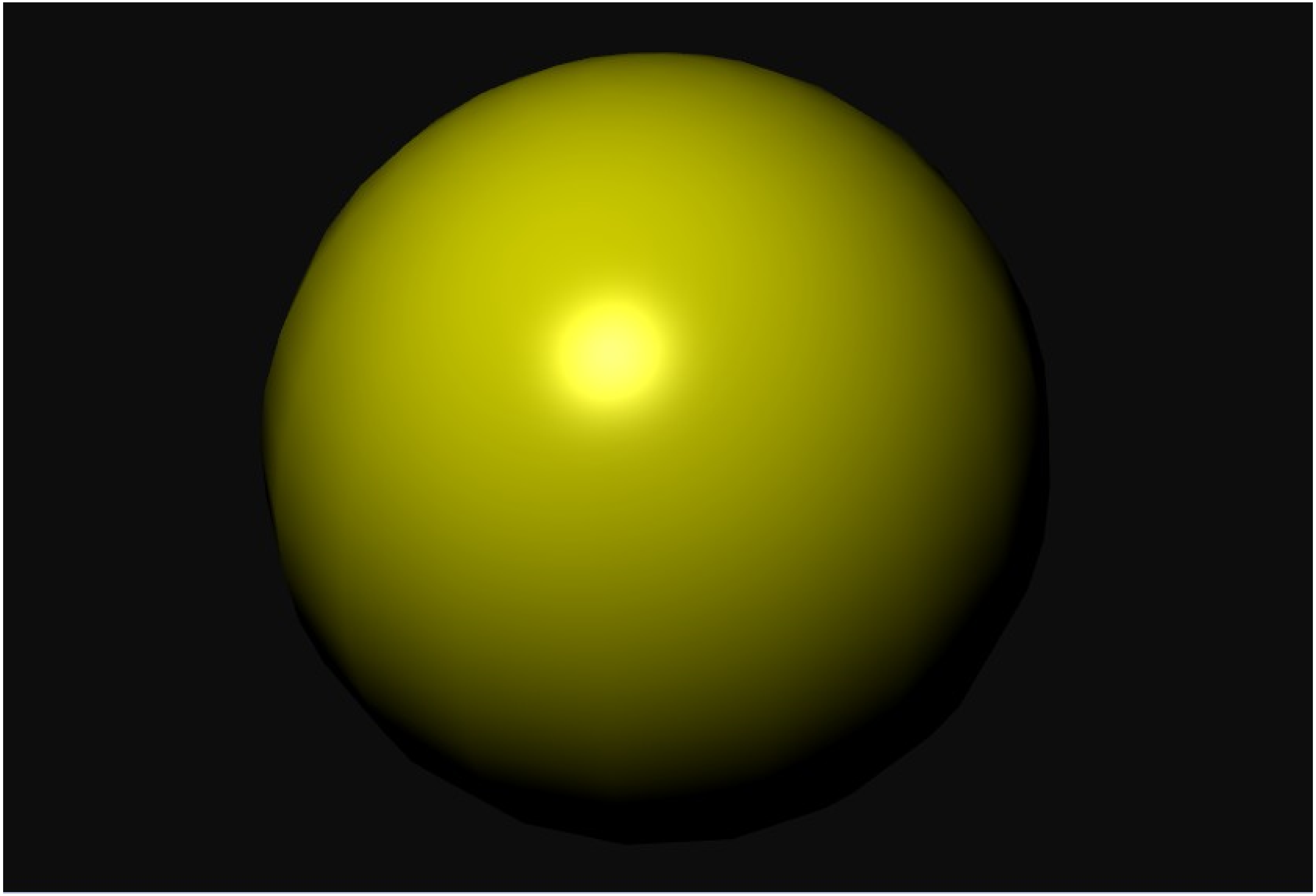
# Flat shading, Lambert illumination



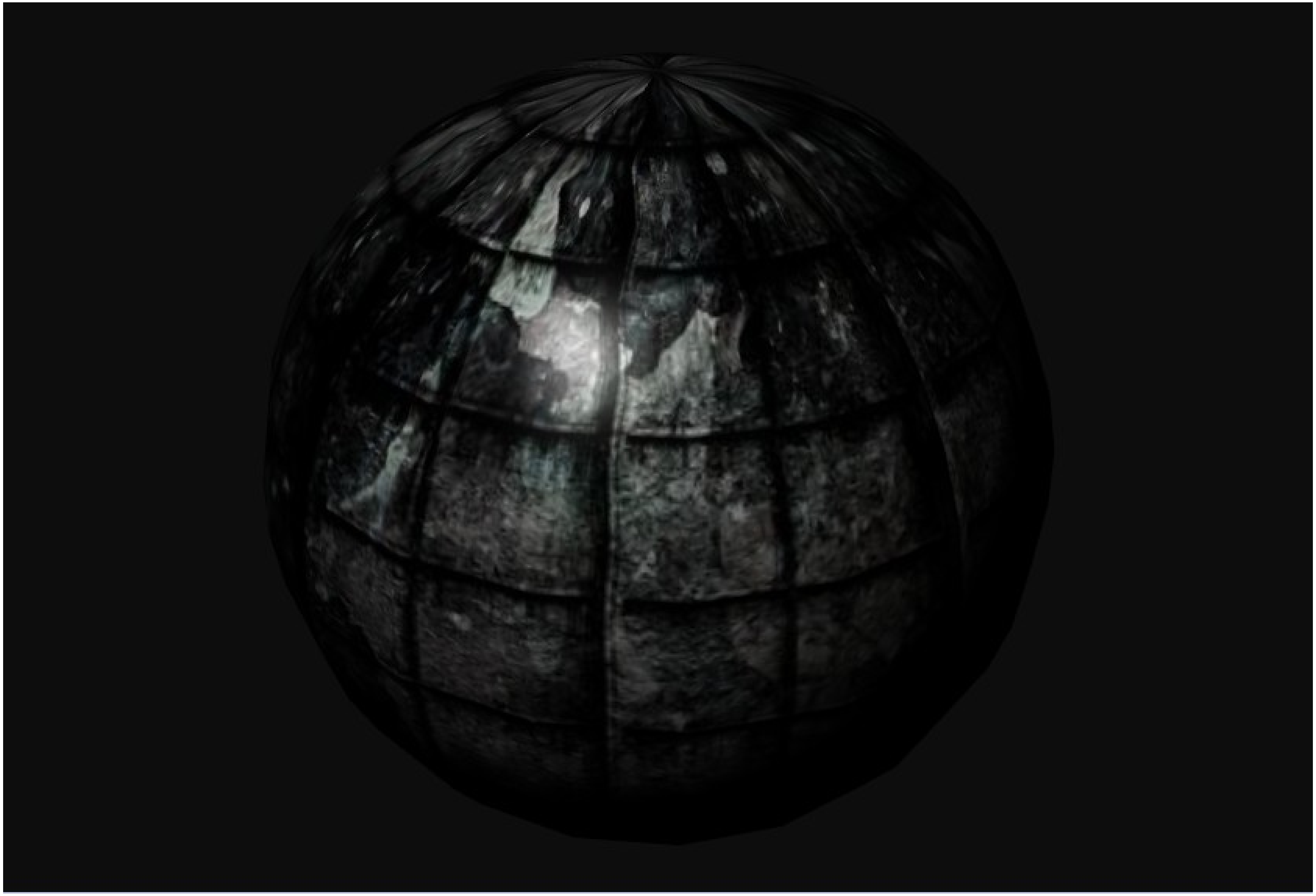
# Phong shading, Lambert illumination



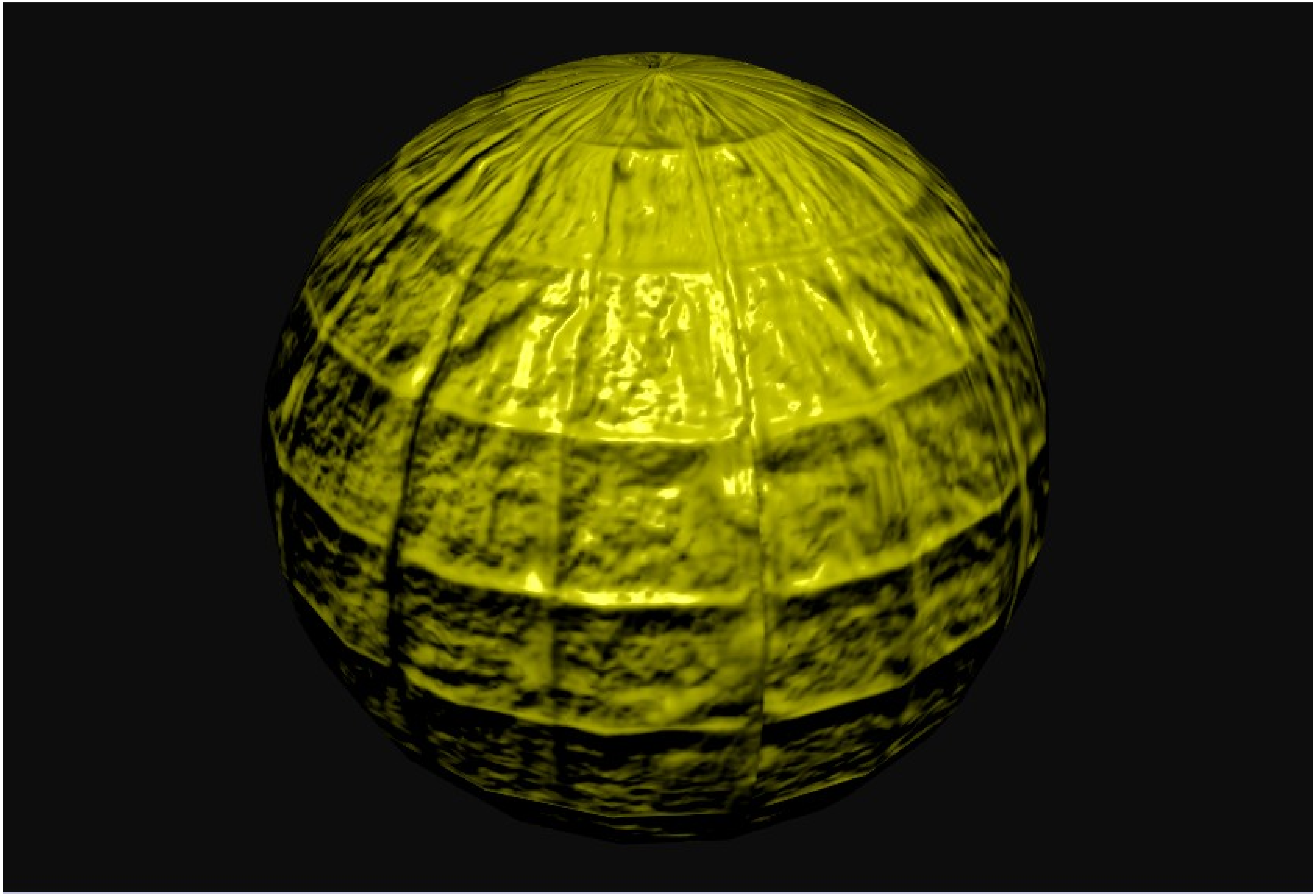
# Phong shading, Phong illumination



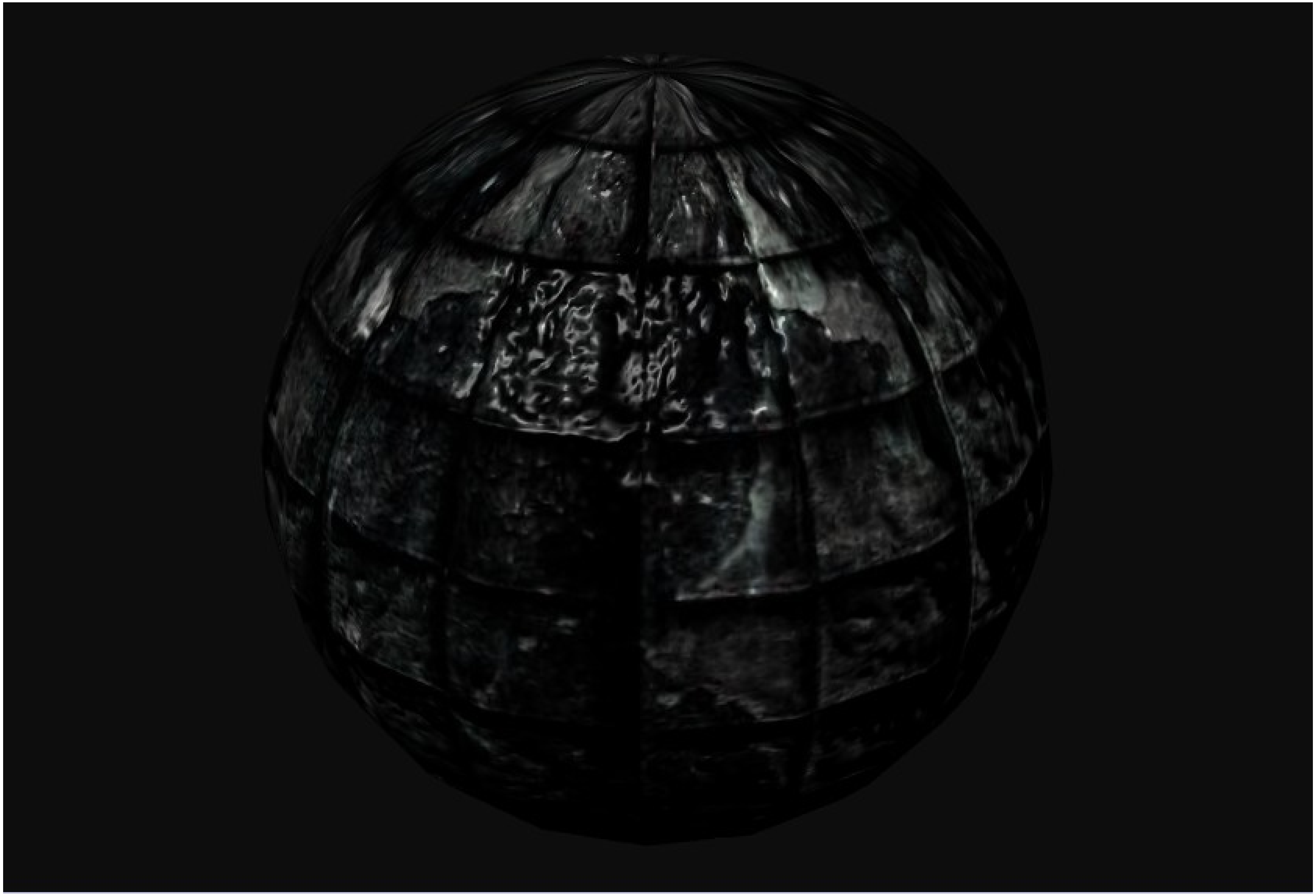
# Texture mapping



# Tangent space normal mapping



# Texture and normal mapping



# Inka3D Maya to WebGL exporter

Autodesk Maya → WebGL

# Motivation

- Artists have their favorite Tools, e.g. Max, Maya, Softimage
- Directly support them by
  - Every attribute animatable
  - Scripted expressions
  - Vertex deformers
  - Shading networks
  - Particle systems
- → Compiler based approach: Translation to code

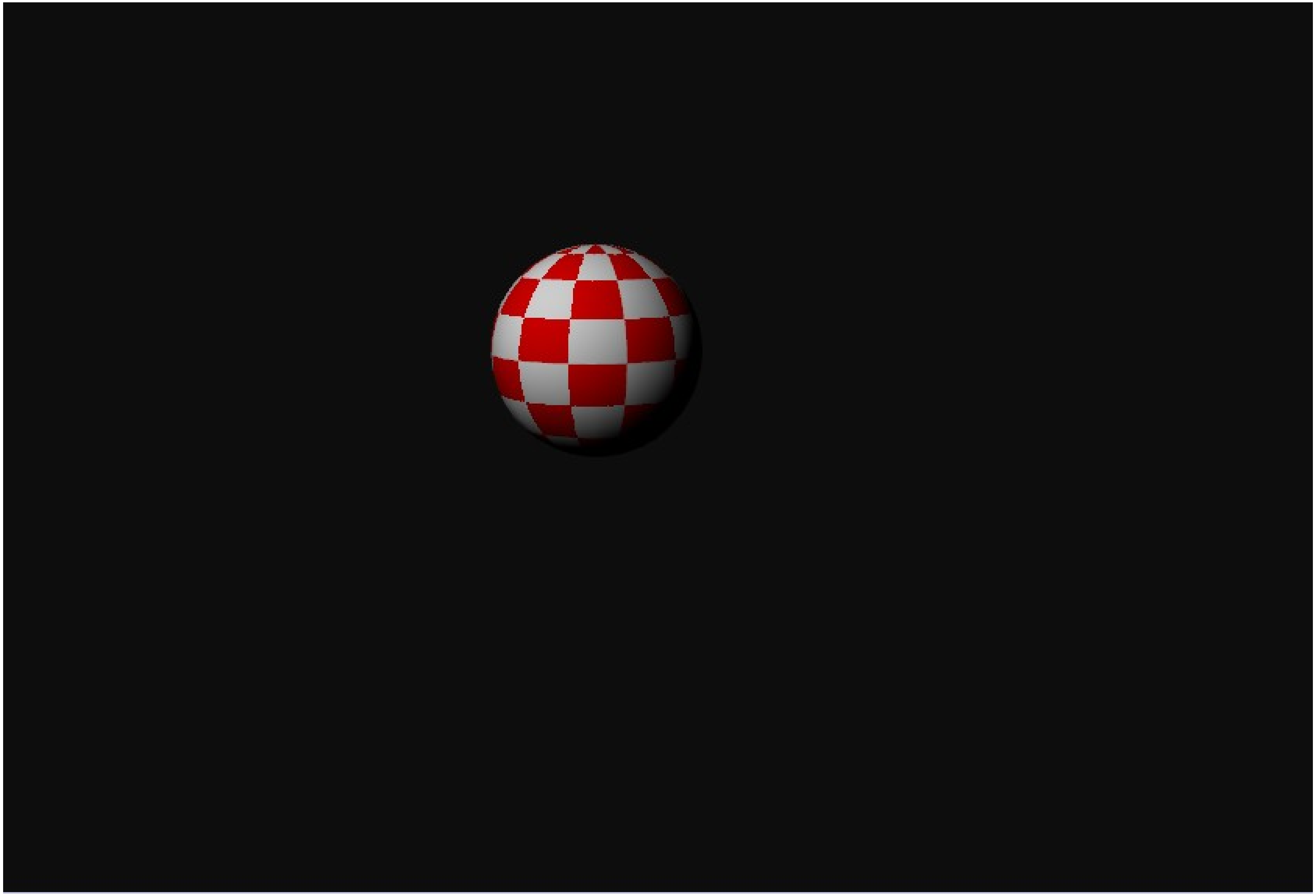
# History

- 2005: First works on compiler based engine
- 2008: Restarted project using LLVM
- 2009: First working version for OpenGL/CgFX
- 2010: Removed need for CgFX, using Clang and LLVM JIT
- 2011: Support for WebGL

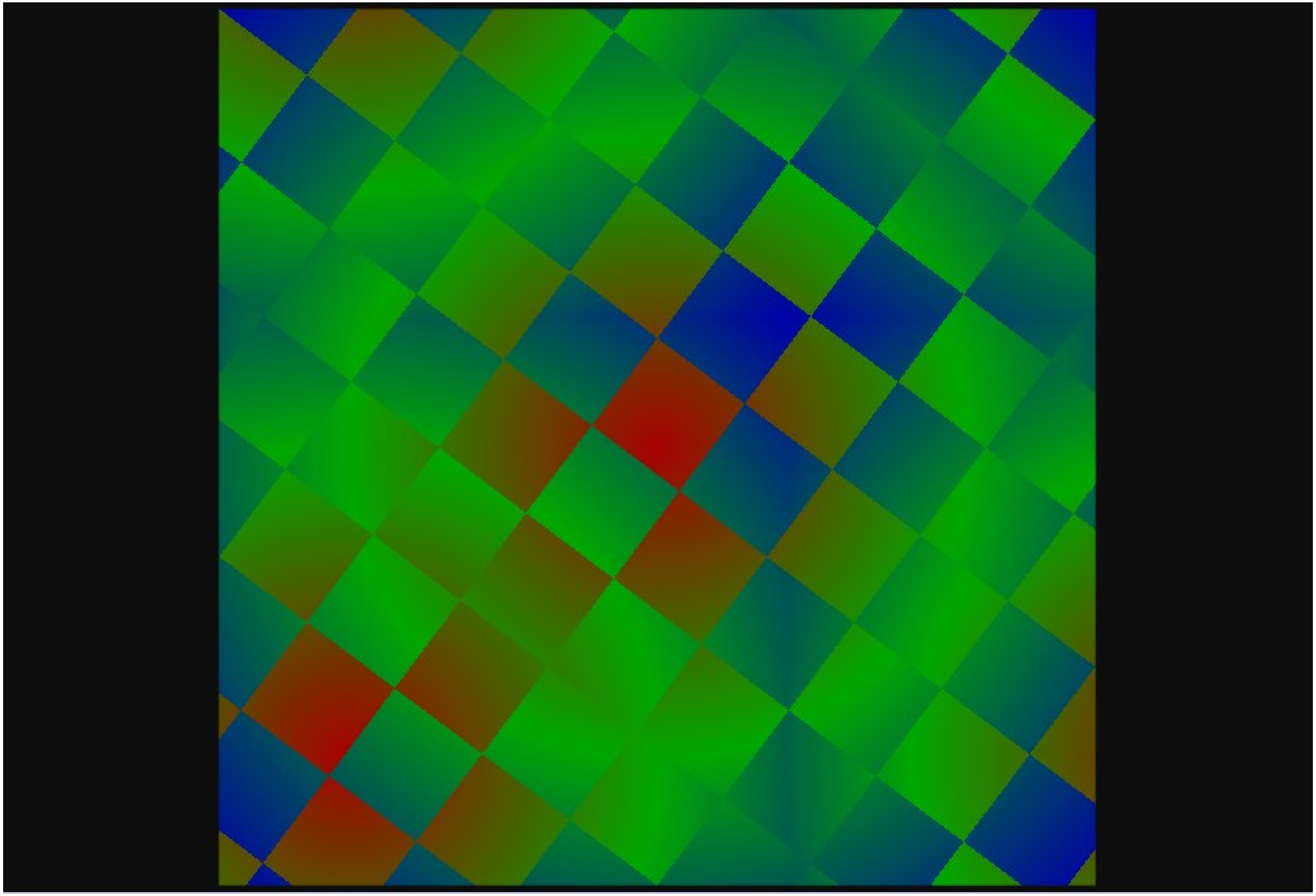
# Examples

What you can do as a coder ;-)

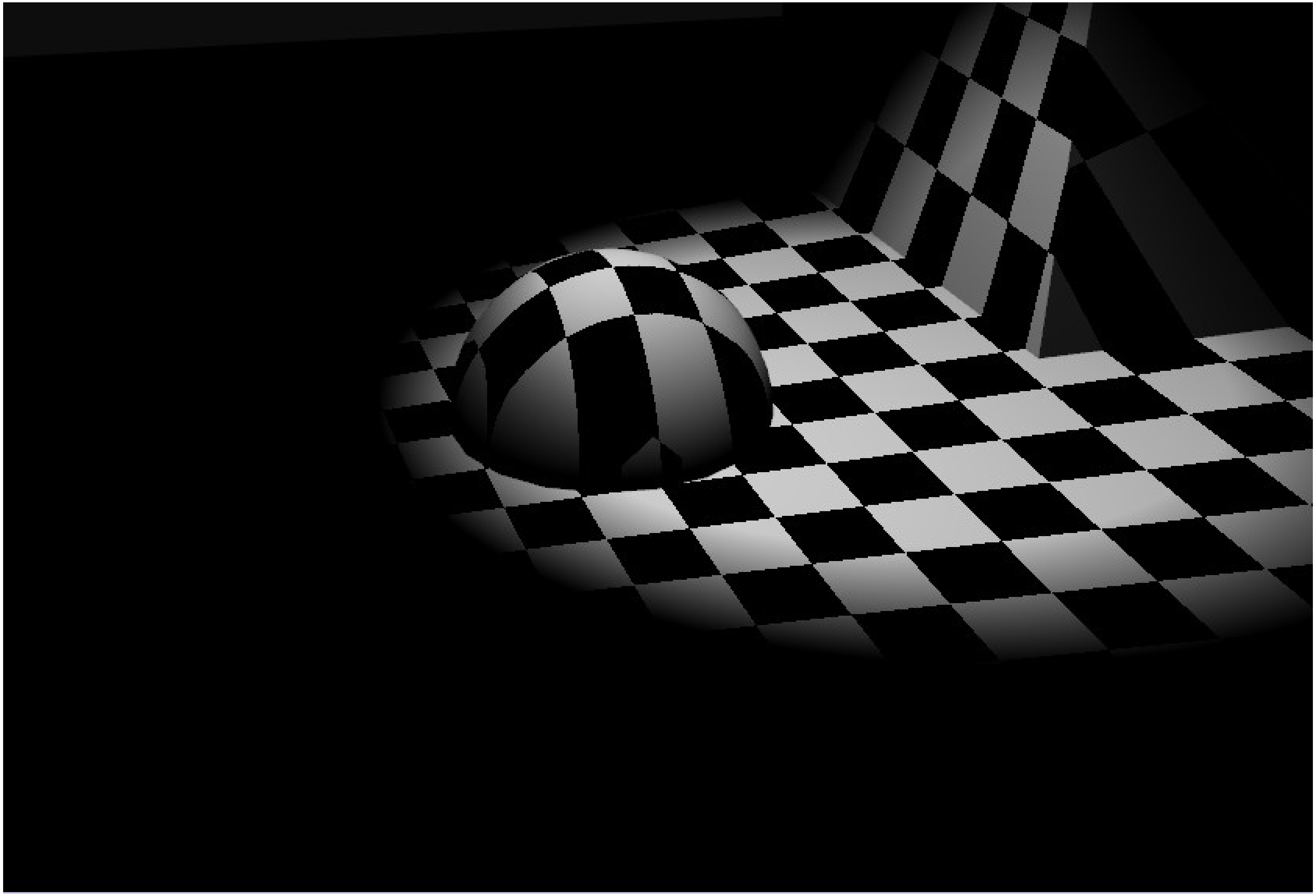
# Only AMIGA makes it possible ;-)



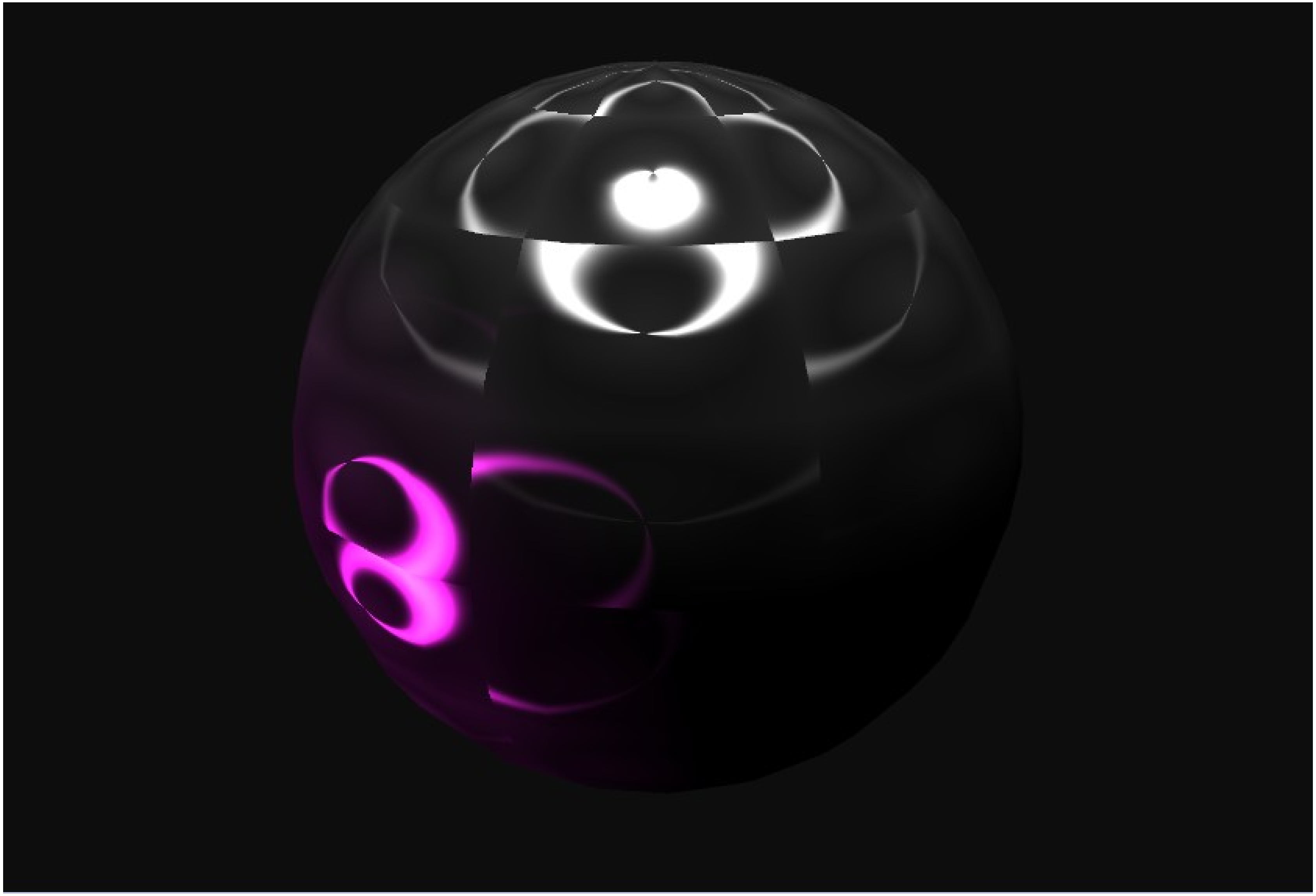
# 2D-Effect: texture displacement



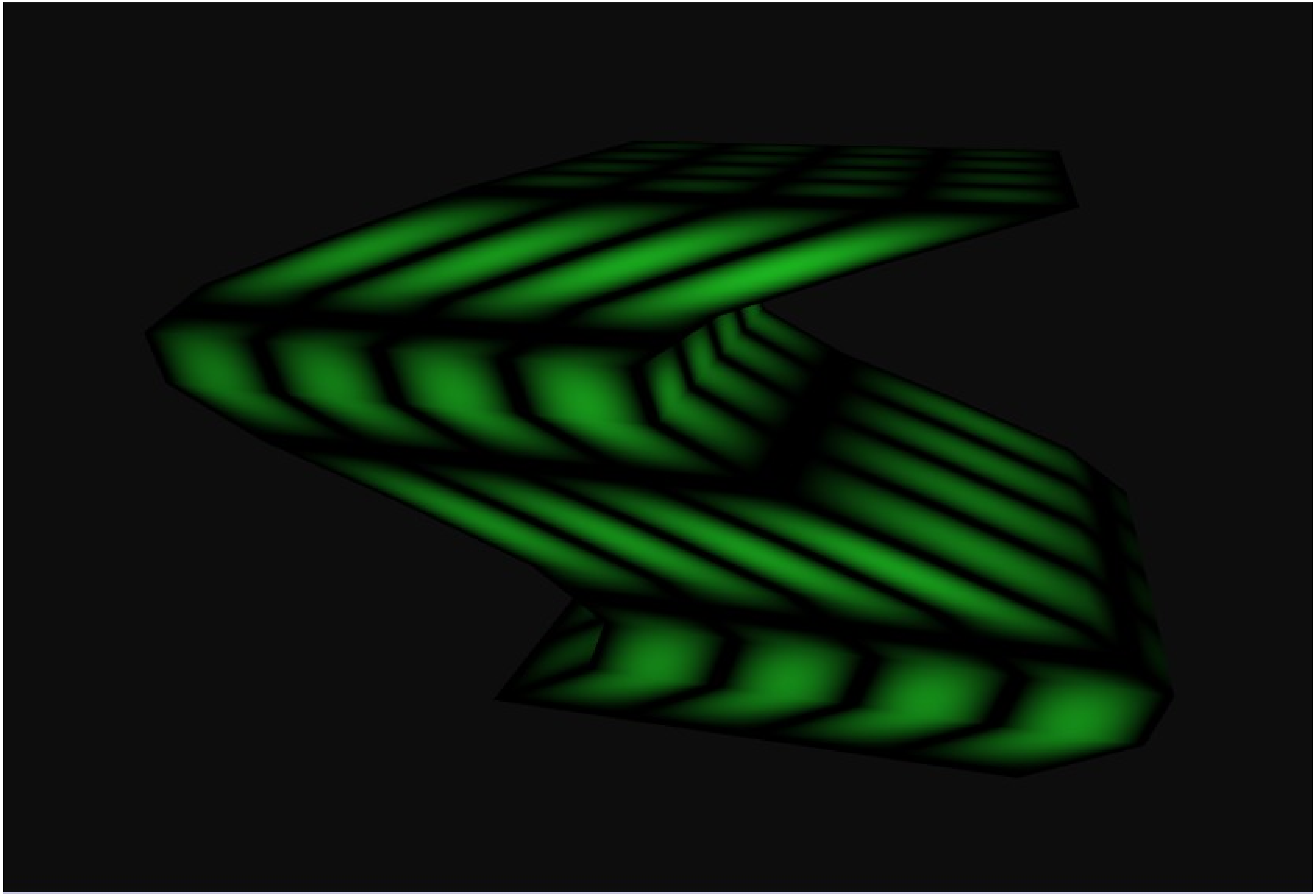
# Projector



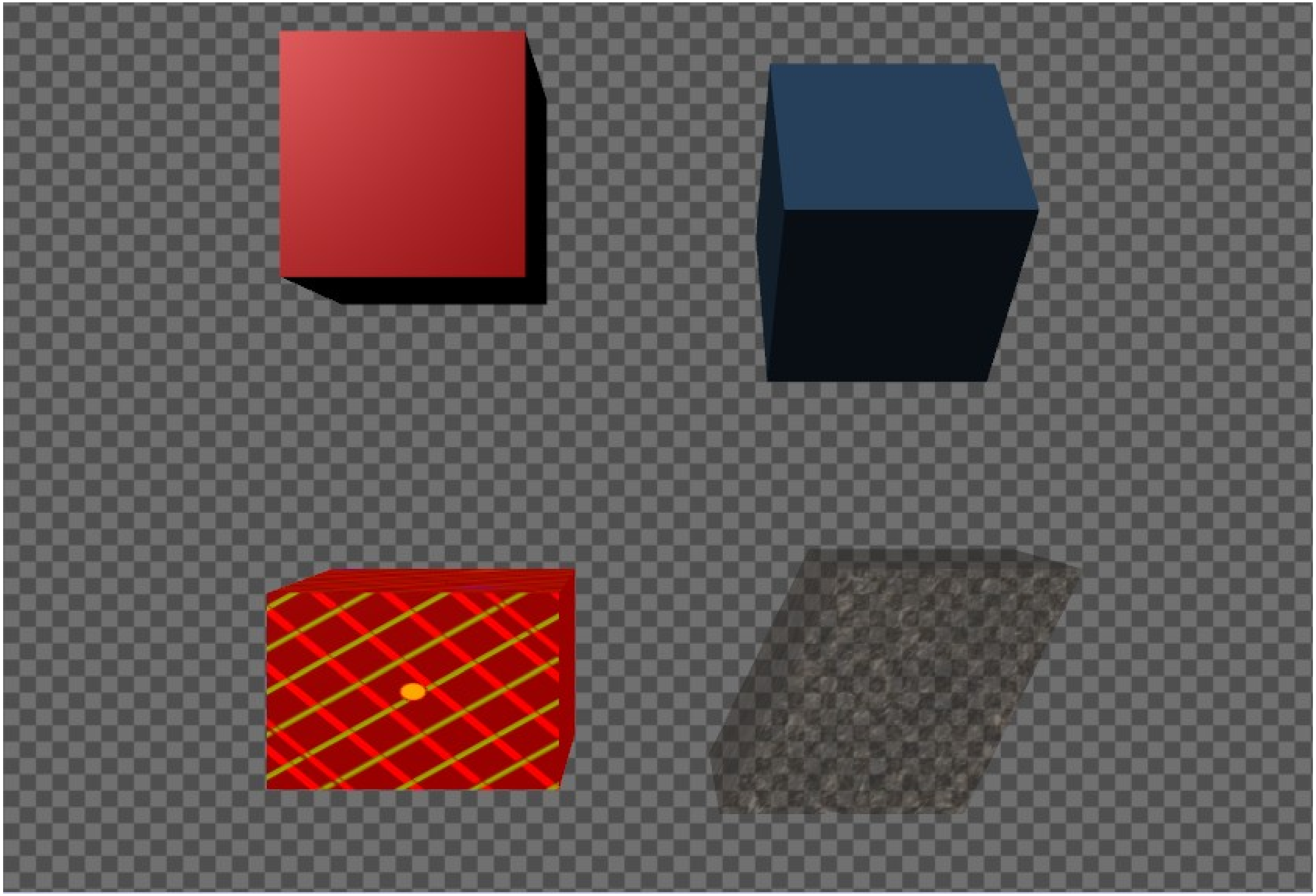
# Brushed Metal (anisotropic shader)



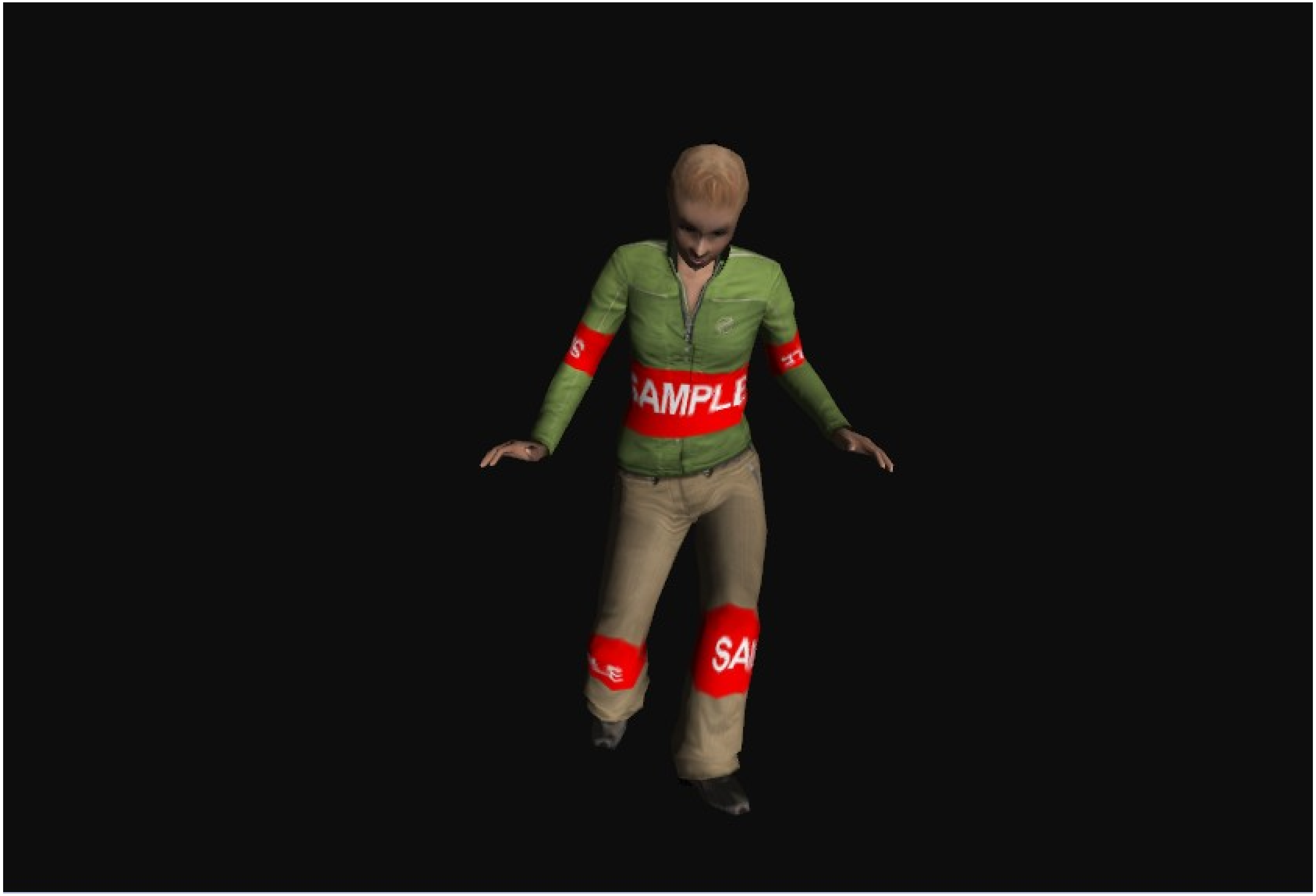
# Sine deformer



# Picking on the GPU



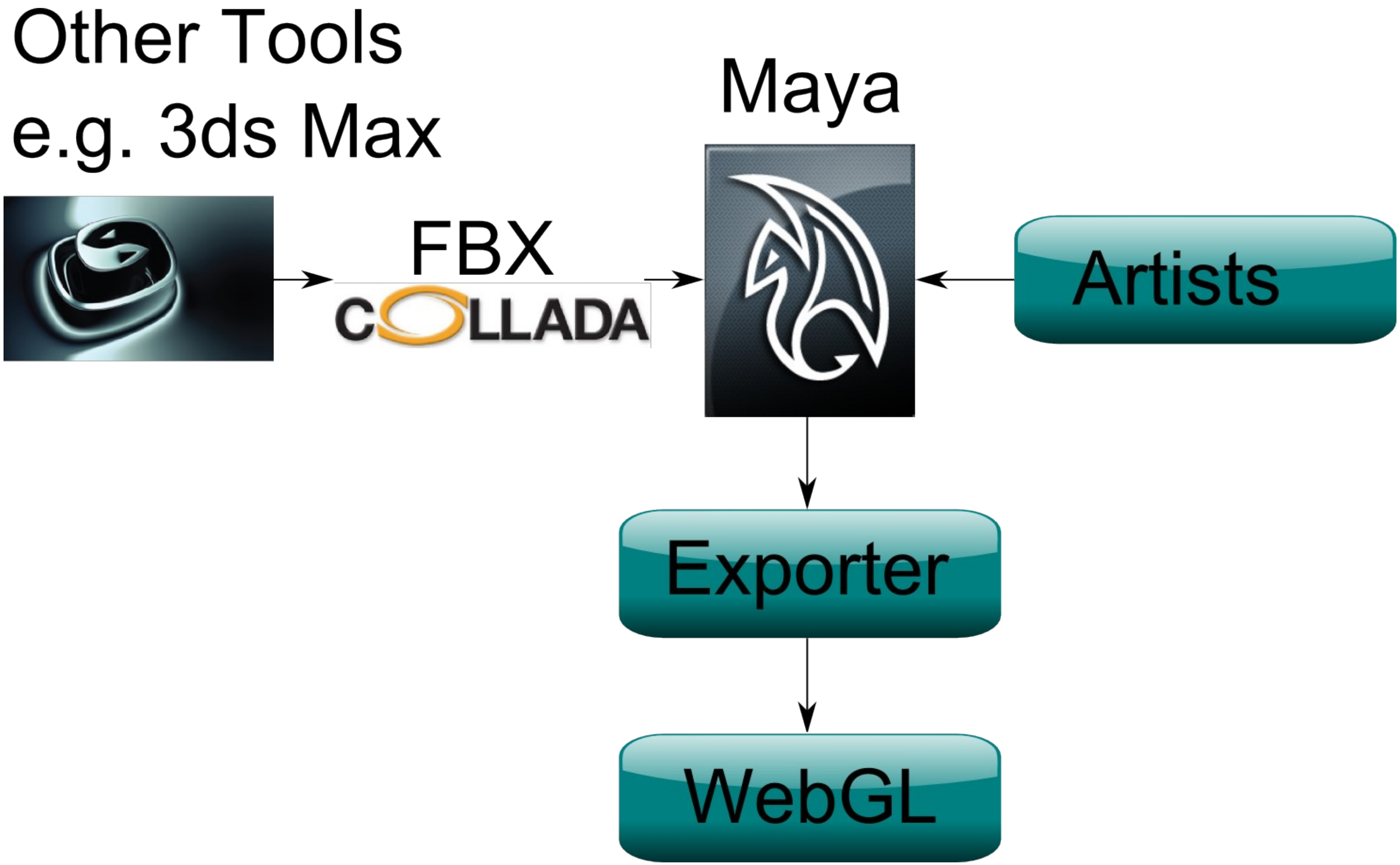
# Character



# Technical

## How does it work?

# Workflow (theory)



Other Tools
e.g. 3ds Max

FBX

COLLADA

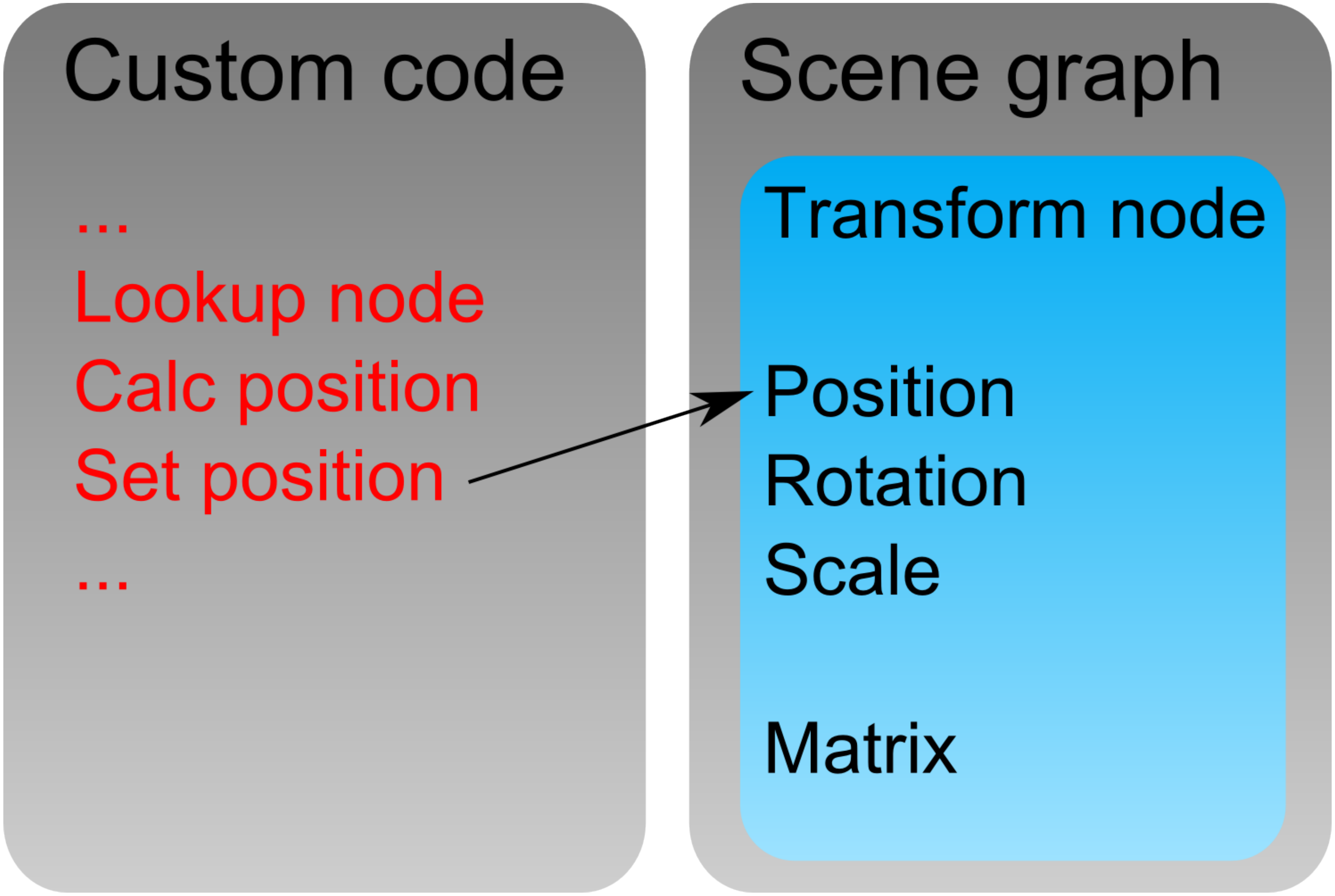
Maya

Artists

Exporter

WebGL

# Classical Implementation



Custom code

...

Lookup node

Calc position

Set position

...

Scene graph

Transform node

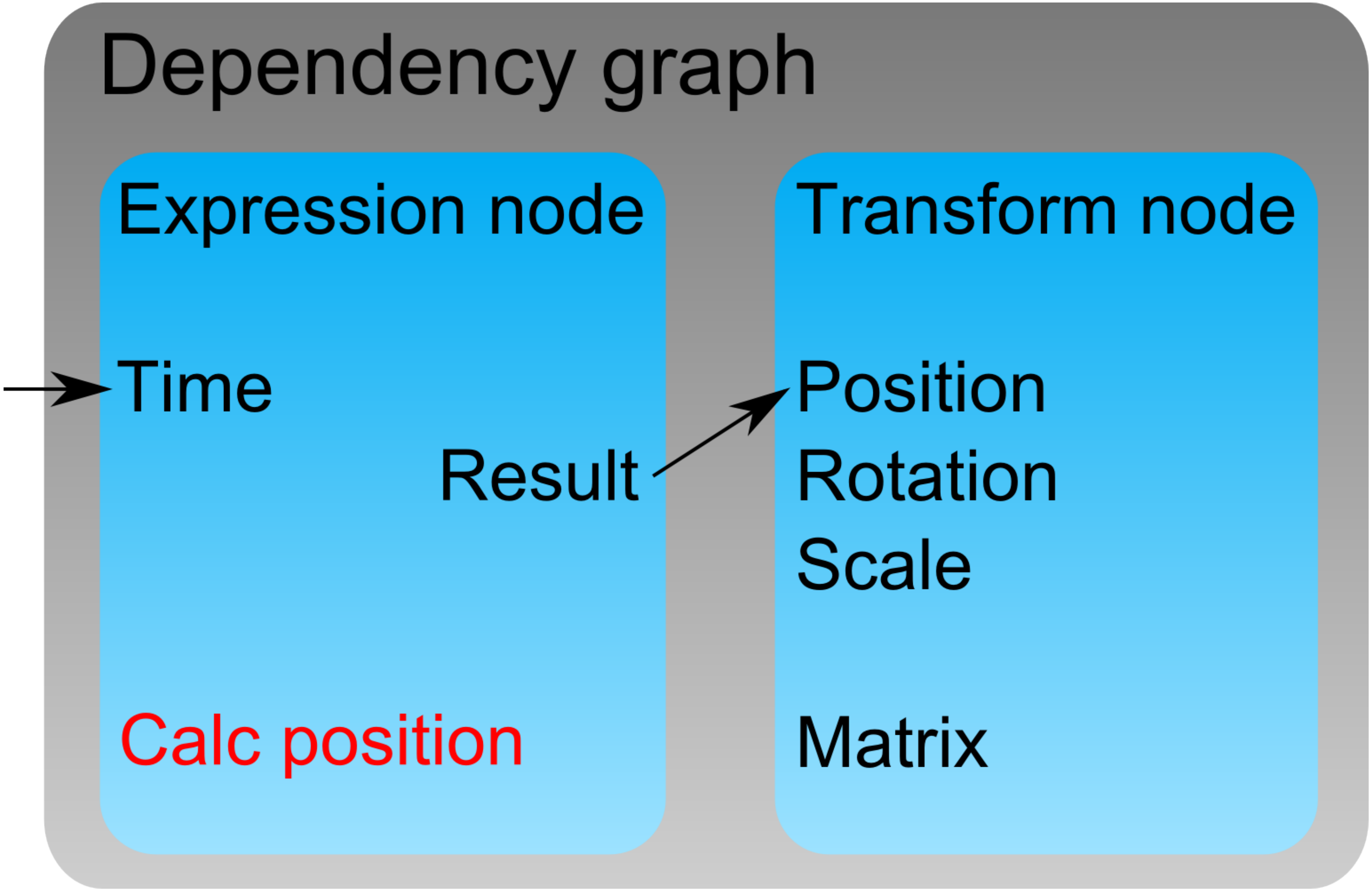
Position

Rotation

Scale

Matrix

# Compiler Based Implementation



Dependency graph

Expression node

Time

Result

Calc position

Transform node

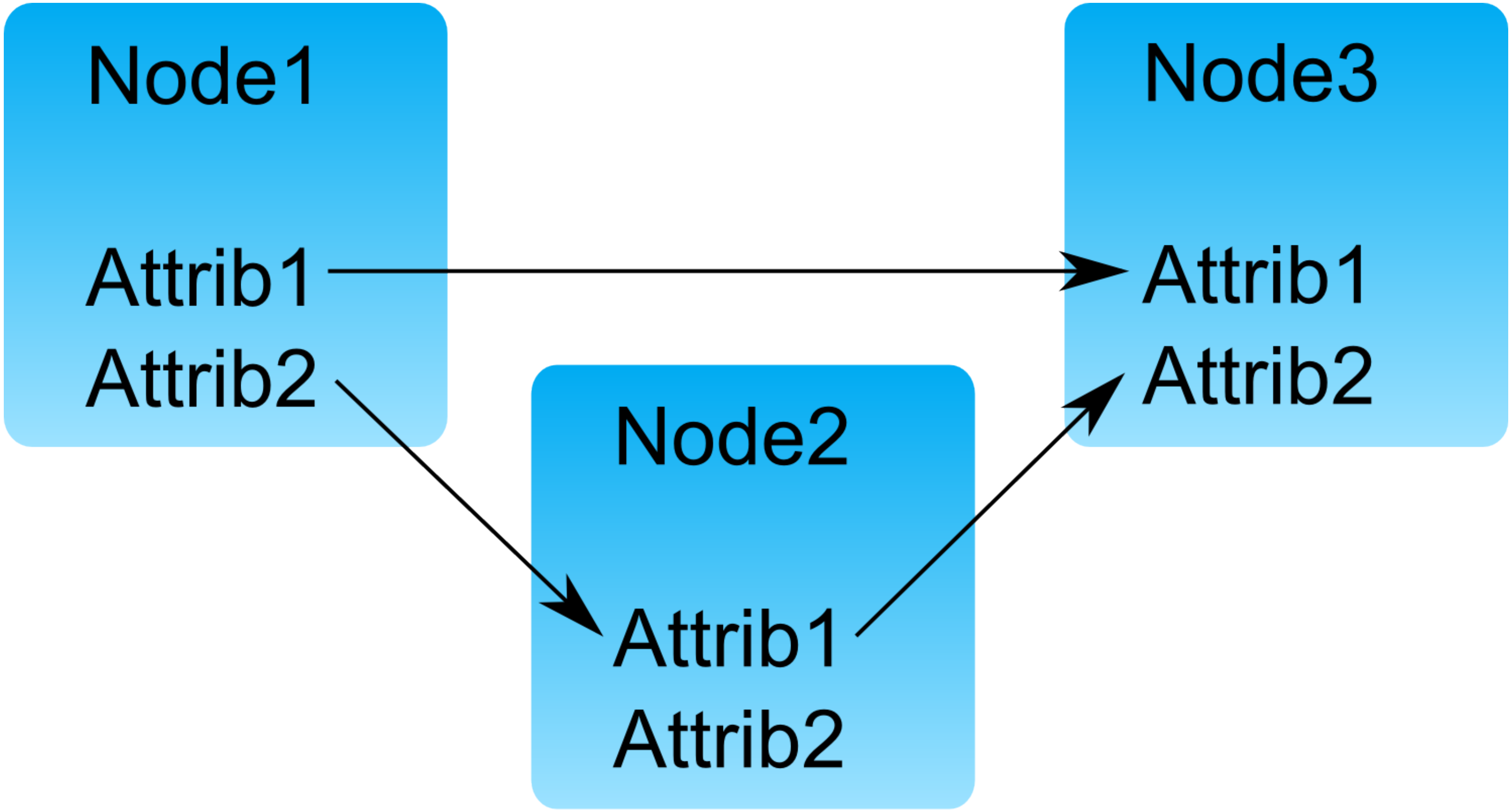
Position

Rotation

Scale

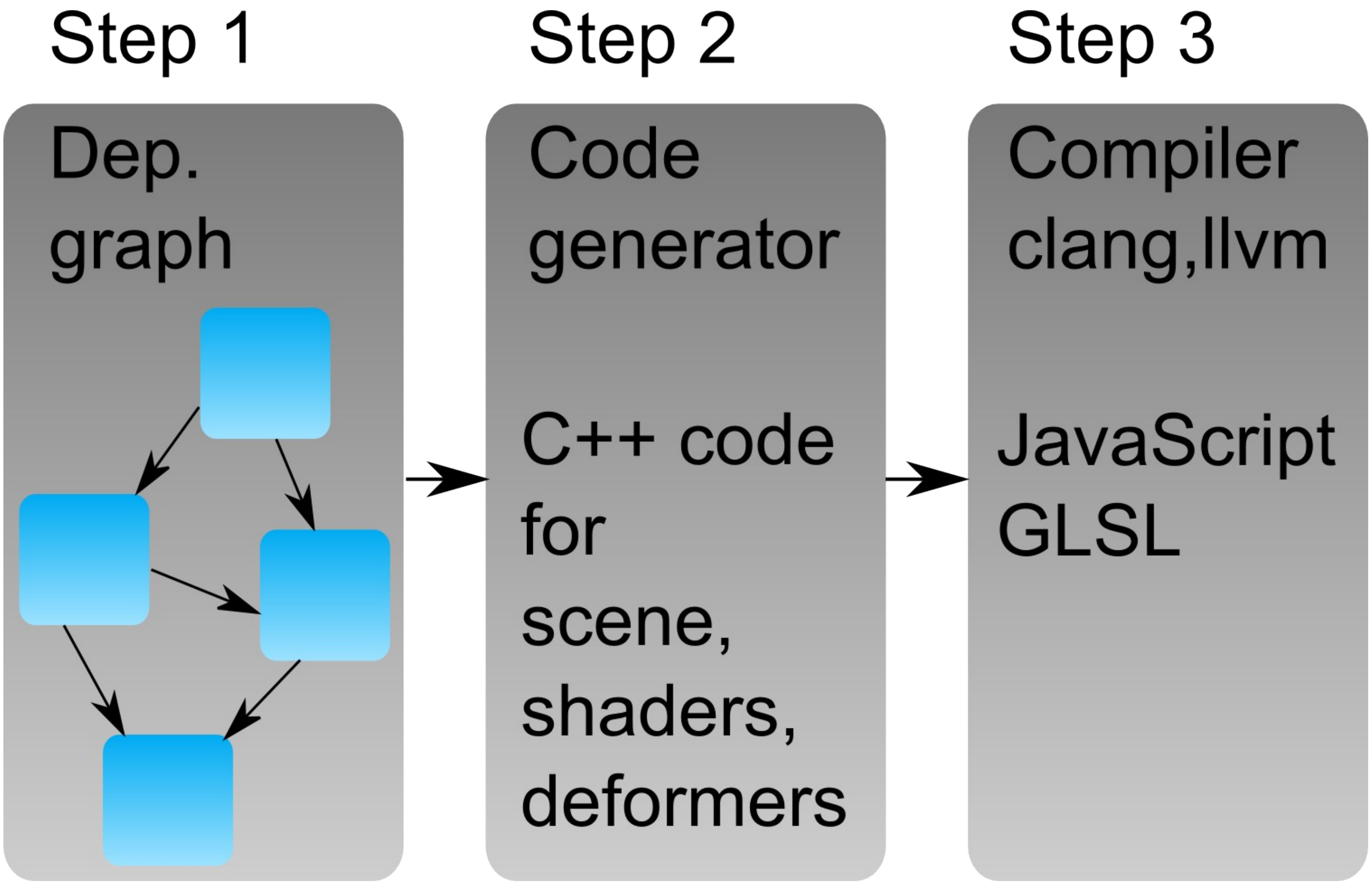
Matrix

# Dependency Graph



A Loop free dependency graph can be transformed to sequential code

# Pipeline



Step 1

Dep. graph

Step 2

Code generator

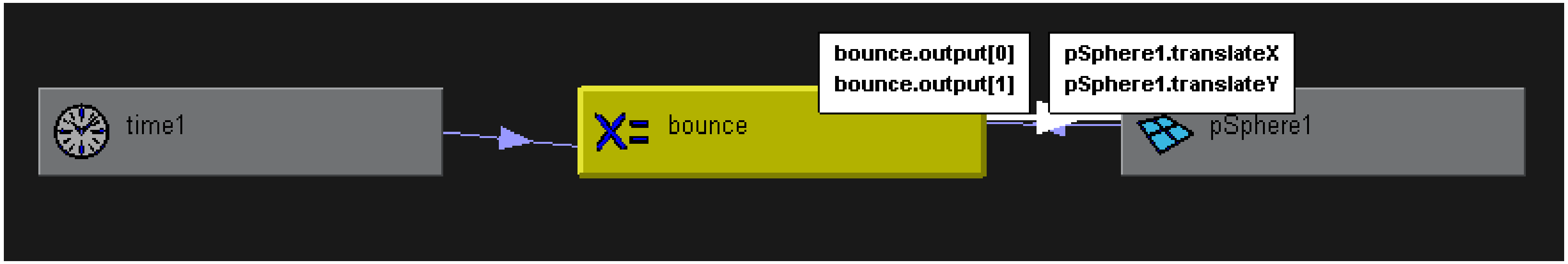
C++ code for scene, shaders, deformers

Step 3

Compiler clang,llvm

JavaScript GLSL

# Step 1: Dependency graph



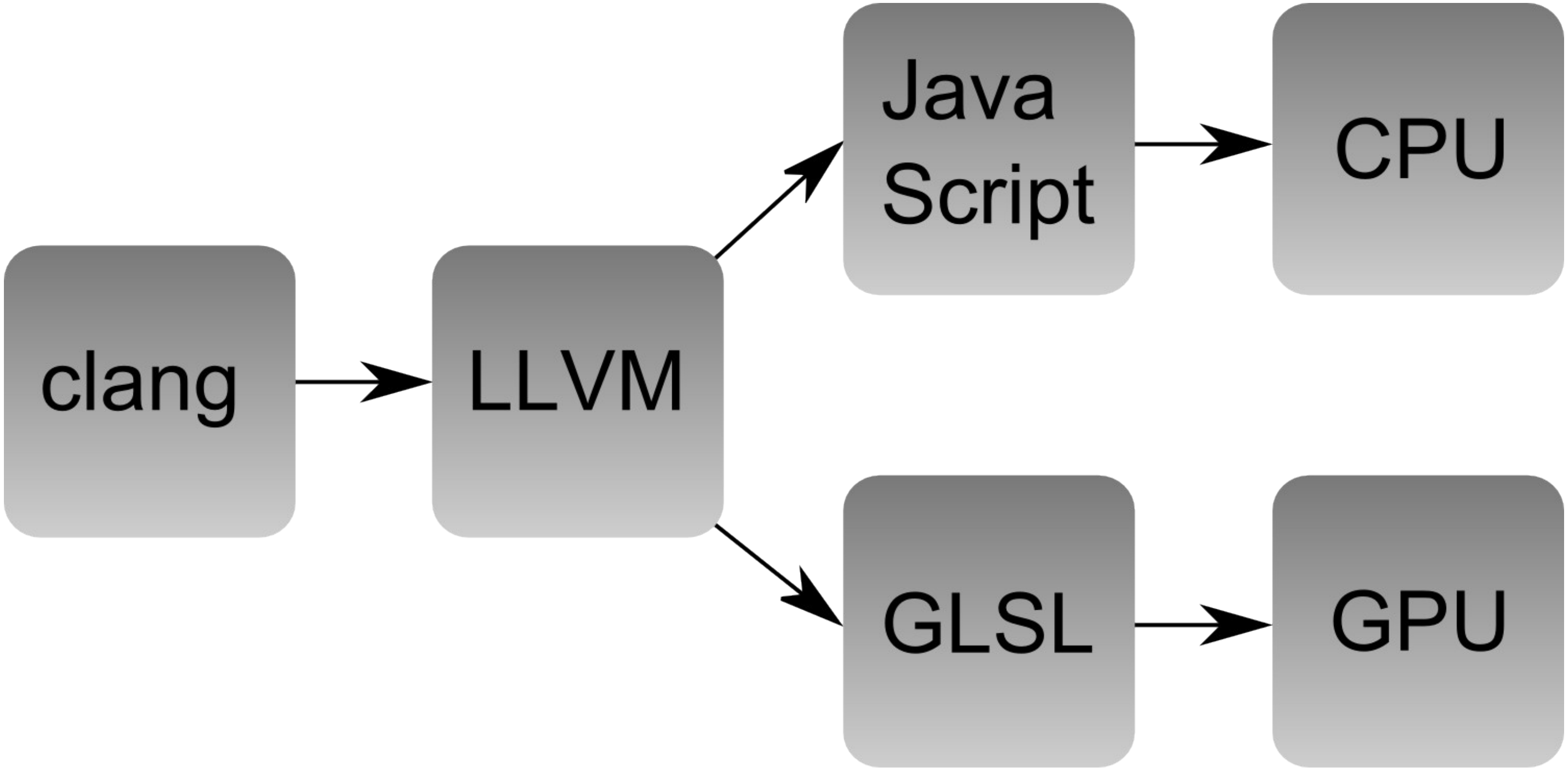
bounce.output[0]
bounce.output[1]

pSphere1.translateX
pSphere1.translateY

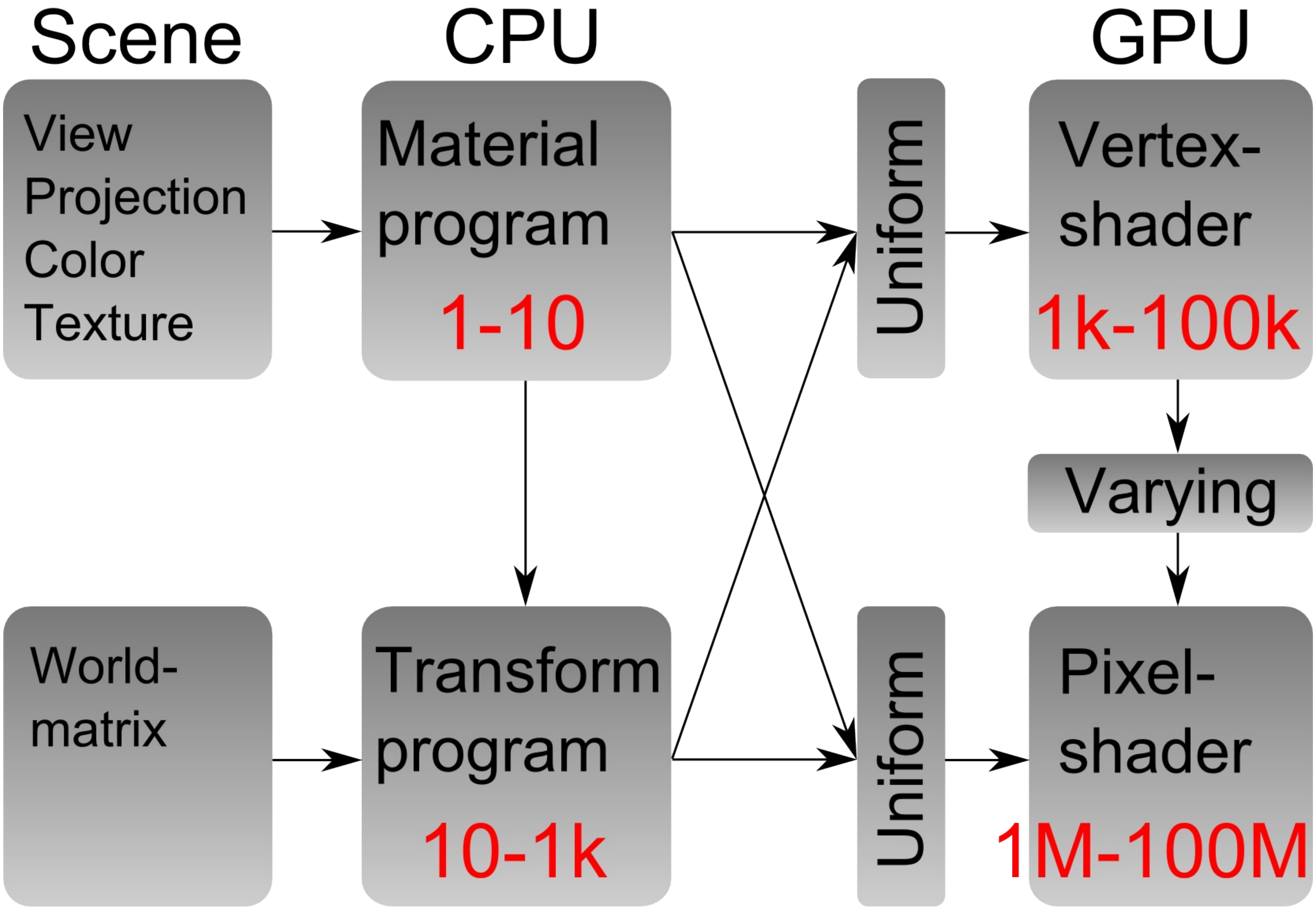
# Step 2: Code

```
// 'bounce' (ScriptNode)
{
  bounce._output0 = time * 3.0f;
  float t = mod(time * 2.0f, 2.0f) - 1.0f;
  bounce.output1 = 6.0f - 5.0f * t * t;
}

pSphere1.translate.x = bounce.output0;
pSphere1.translate.y = bounce.output1;

// 'pSphere1' (TransformNode)
{
  float4x4 matrix = matrix4x4PositionRotation
    (pSphere1.translate, ...
}
```

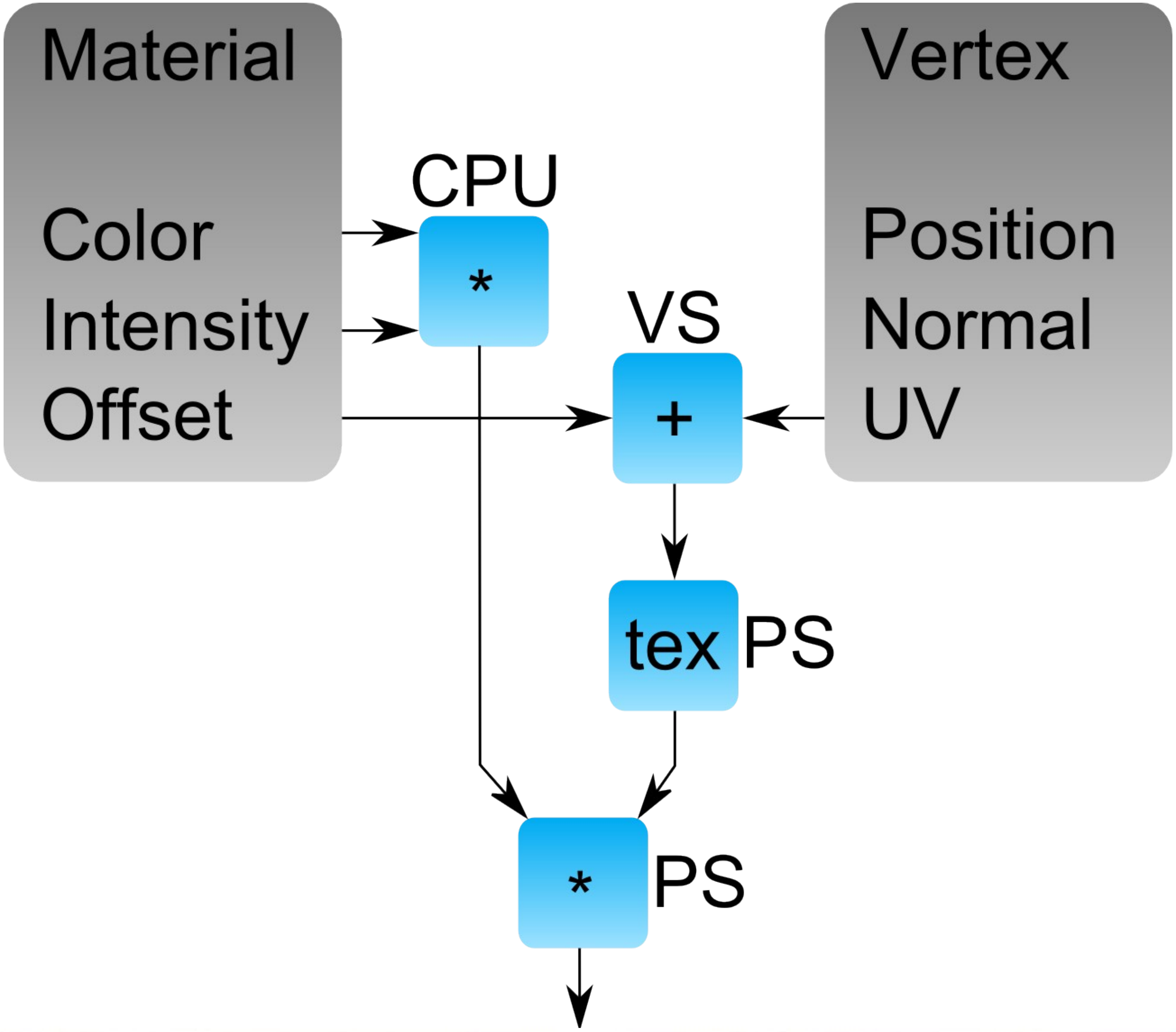
# Step 3: Compiler



# Shader Splitting



# Instruction classification



Demo

# "Azathioprine" WebGL Demo

- Demo group "Alcatraz"
- 2nd place on Evoke 2011 in Cologne/Germany
- 2nd place at Mozilla online demo competition
- Watch demo

Public beta at www.inka3d.com

Thank you for your attention