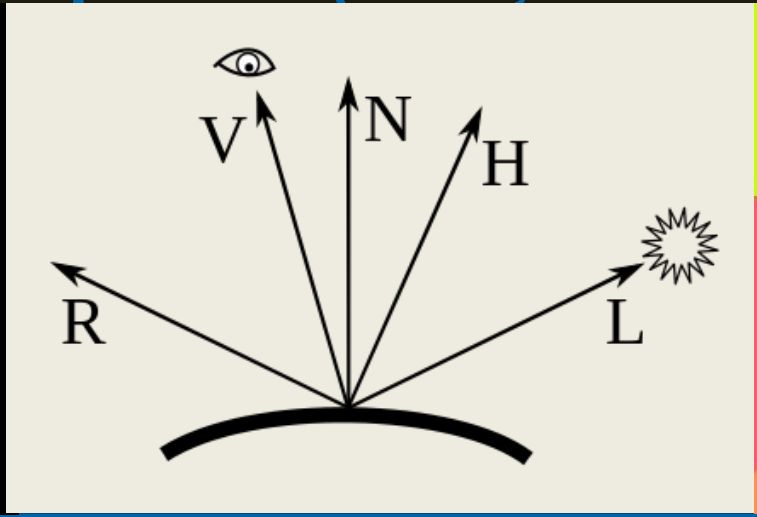
TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

Multimedia – OpenGL (VBO, Uniforms)



Jonas Treumer / Ben Lorenz

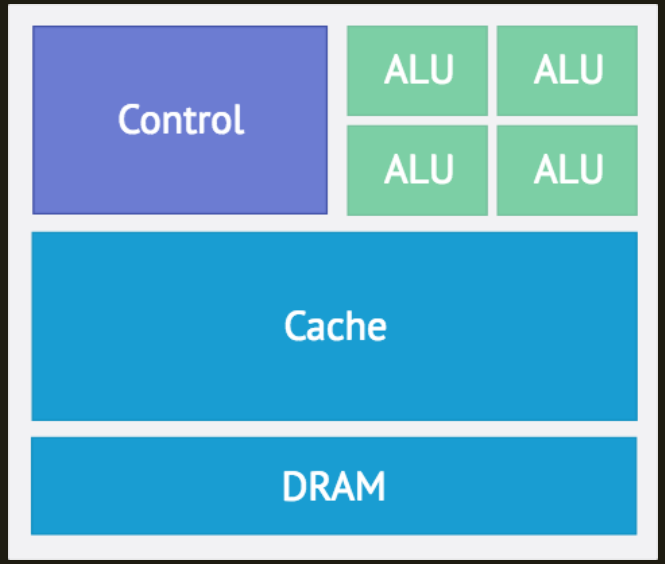
Agenda

- CPU vs GPU Hardware
- Shader Performance

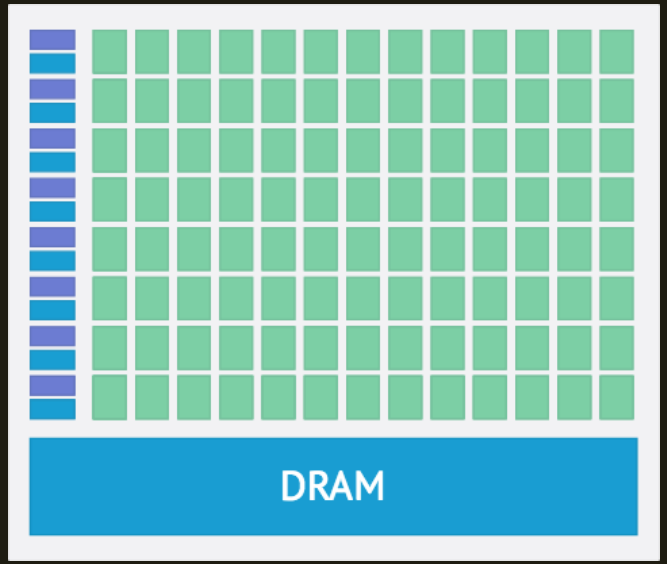
- CPU/GPU Communication
- Vertex Buffer Objects

- Uniforms

CPU vs GPU Hardware



CPU



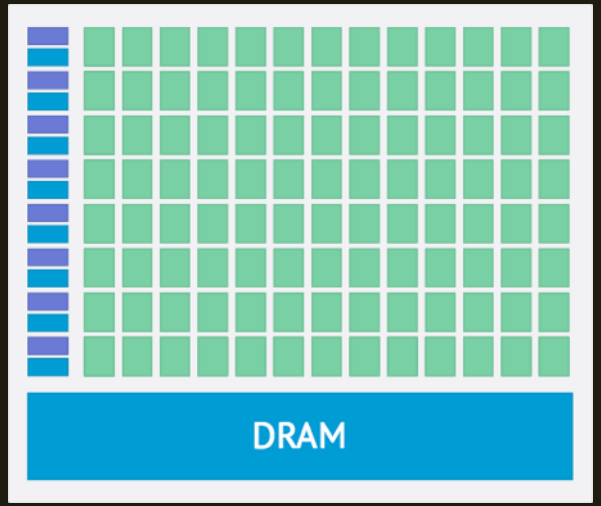
GPU

CPU vs GPU Hardware

- Geforce RTX 3080: 8704 Shader Processors
- RX 6800 XT: 4608 Shader Processors

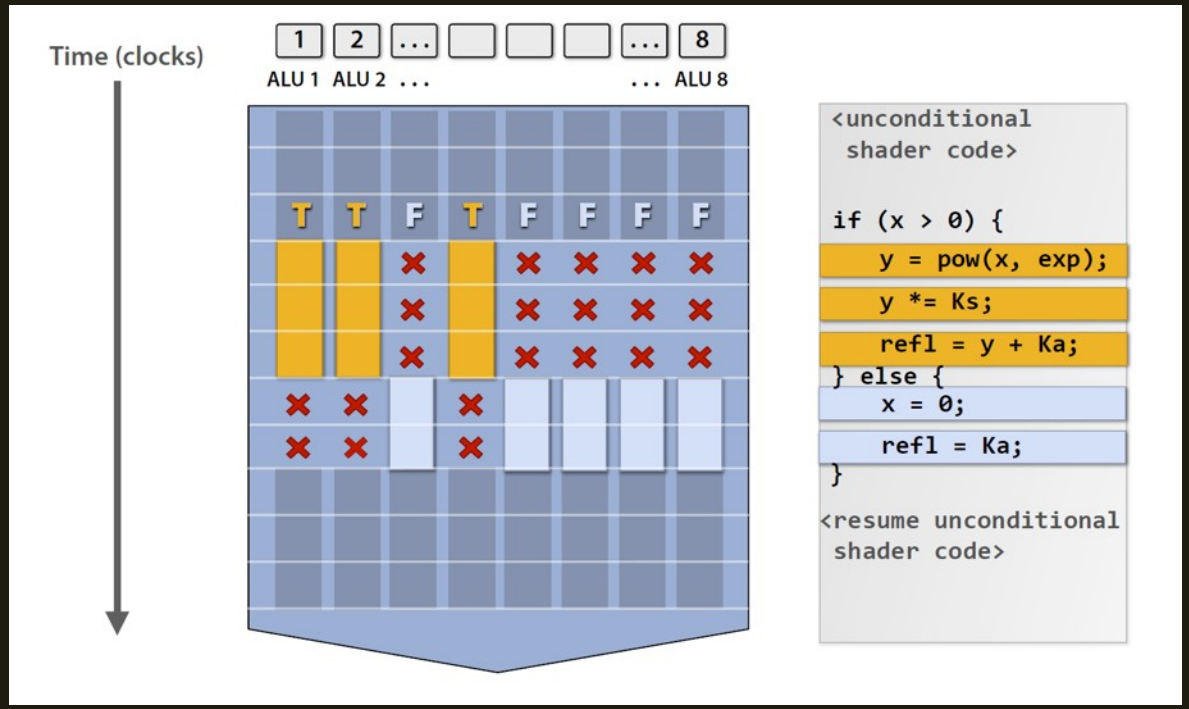
- simplified logic
    - no out-of-order execution
    - no branch prediction

- **all cores do the same thing at the same time**



GPU

Shader Performance



Shader Performance

```
//fragment shader
#version 300 es
precision mediump float;
out vec4 outColor;
in lowp vec4 fColor;

void main() {
    if (fColor.a < 0.5) {
        fColor.a = 0.0;
    }else{
        fColor.a = 1.0;
    }
    outColor = fColor;
}
```

http://www.shaderific.com/glsl-functions

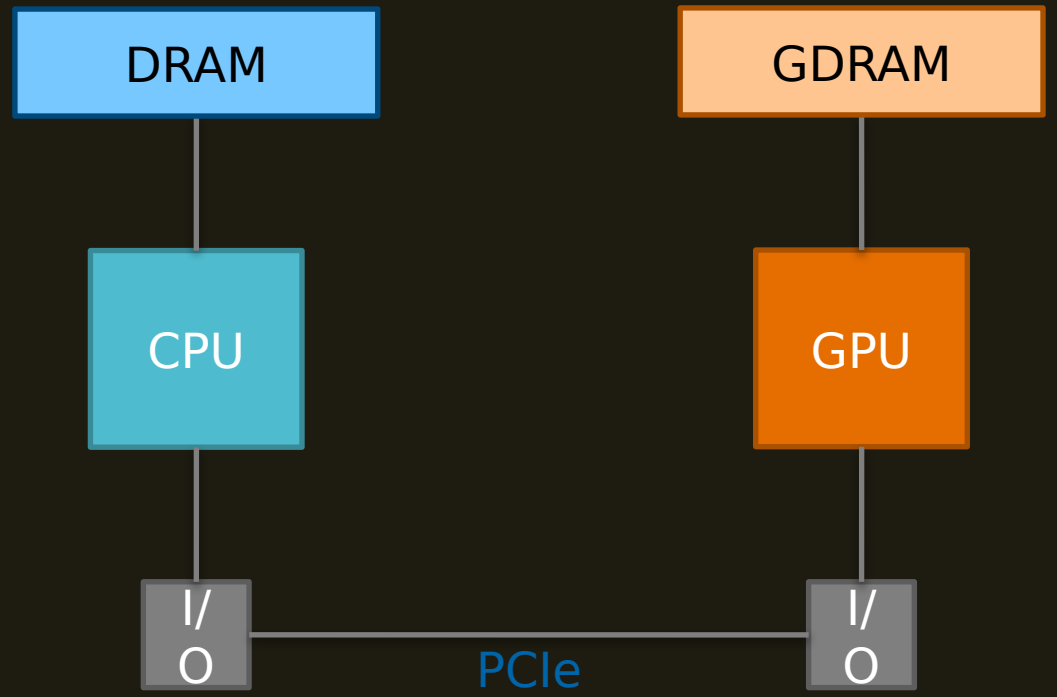
Shader Performance

```
//fragment shader
#version 300 es
precision mediump float;
out vec4 outColor;
in lowp vec4 fColor;

void main() {
    if (fColor.a < 0.5) {
        fColor.a = 0.0;
    }else{
        fColor.a = 1.0;
    }
    outColor = fColor;
}
```

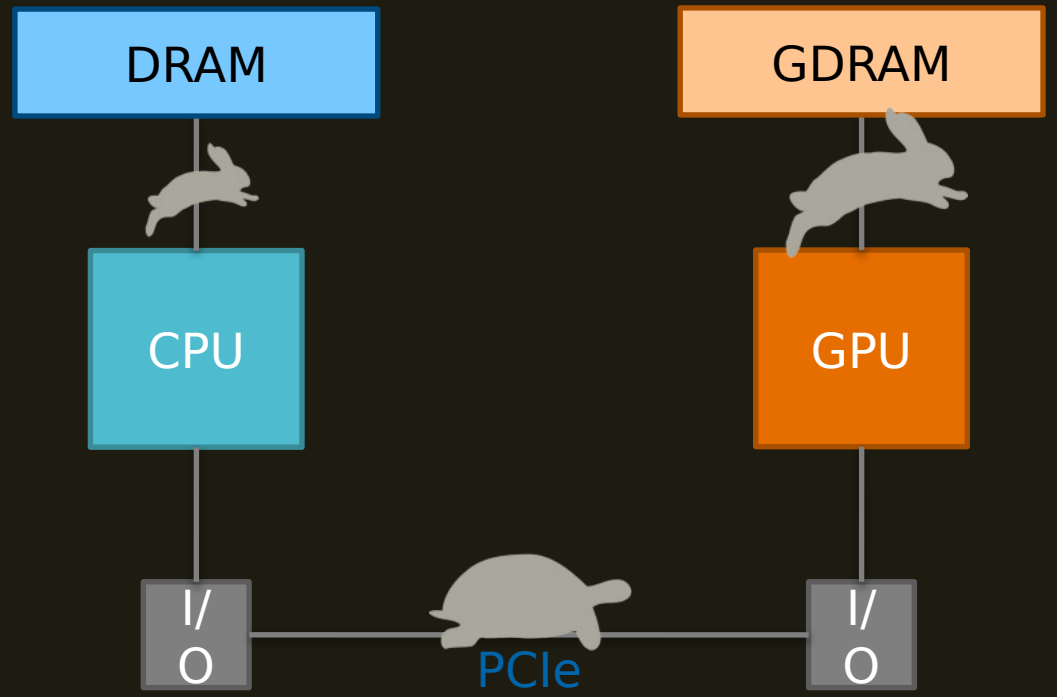
```
//fragment shader
#version 300 es
precision mediump float;
out vec4 outColor;
in lowp vec4 fColor;

void main() {
    fColor.a = step(0.5, fColor.a);
    outColor = fColor;
}
```

http://www.shaderific.com/glsl-functions

CPU/GPU Communication

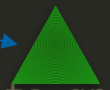


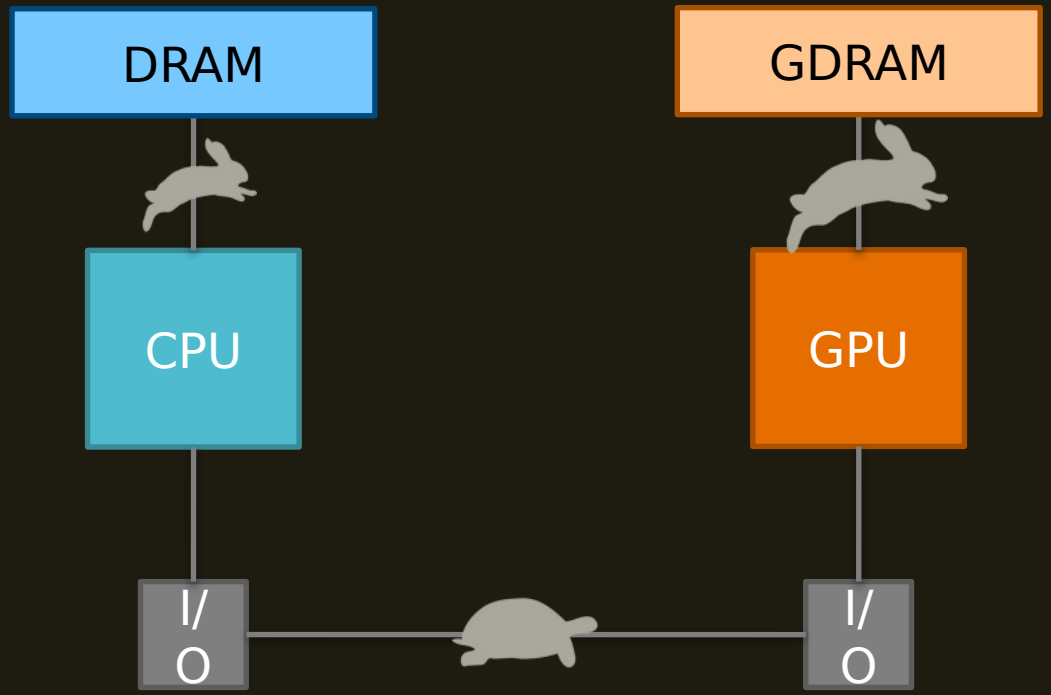
CPU/GPU Communication



CPU/GPU Communication

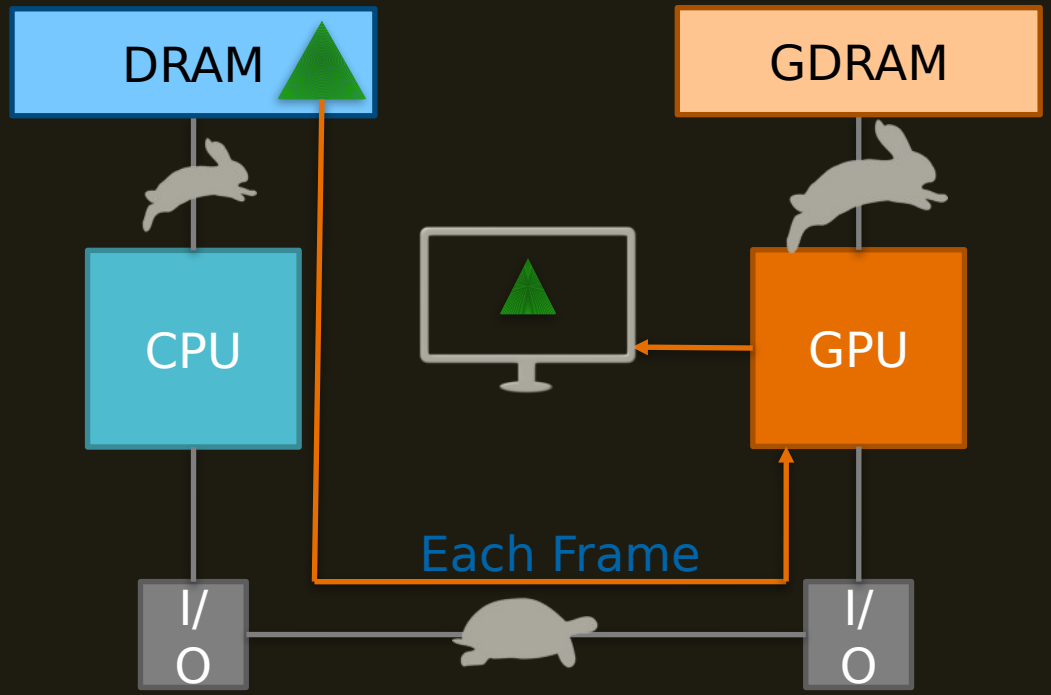
```
void draw(GLFWwindow* window){
    //create/manipulate vertex data
    VertexData vertexData[3];
    ...

    //load vertex data to gpu
    glBufferData(...);

    //draw vertex data
    glDrawArrays(...);
}
```





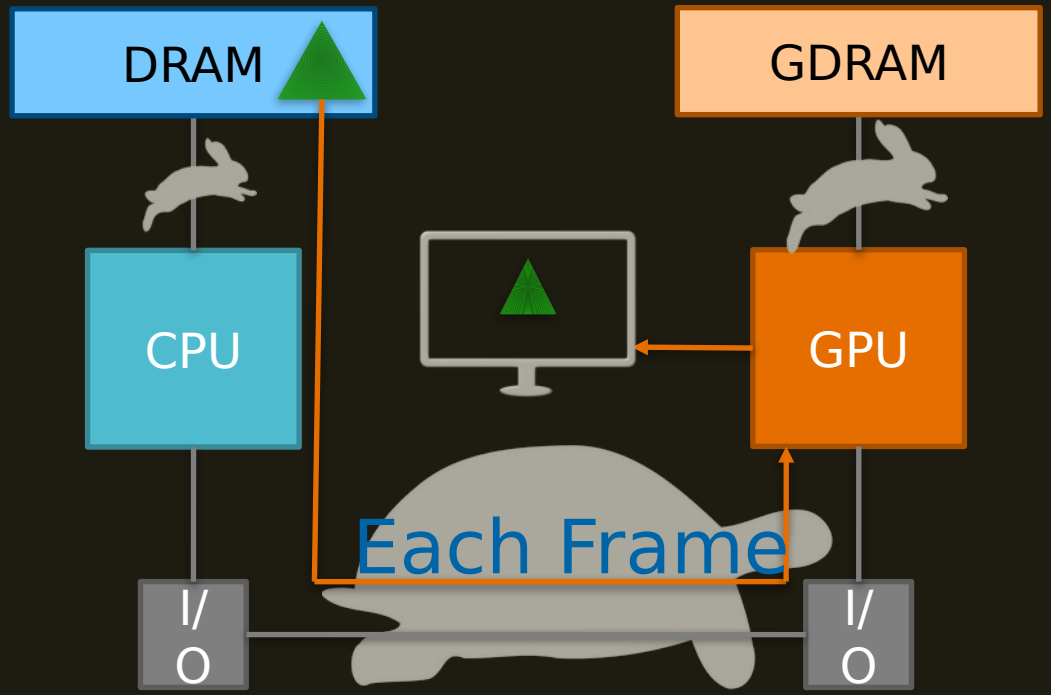
CPU/GPU Communication

```
void draw(GLFWwindow* window){
    //create/manipulate vertex data
    VertexData vertexData[3];
    ...

    //load vertex data to gpu
    glBufferData(...);

    //draw vertex data
    glDrawArrays(...);
}
```



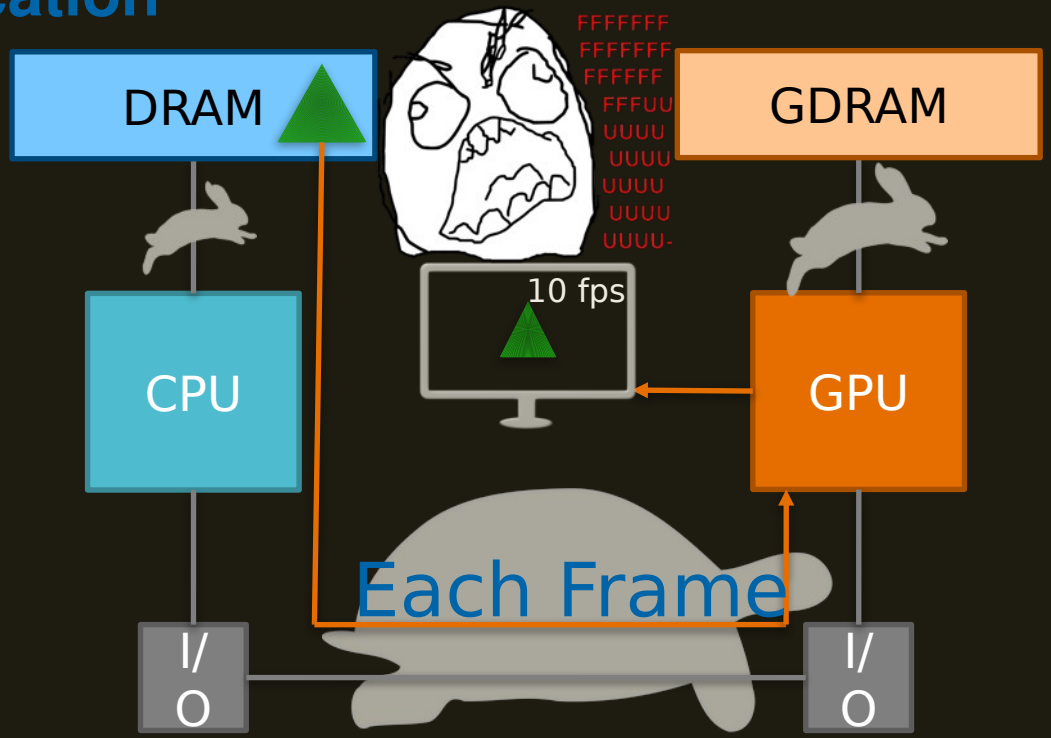
CPU/GPU Communication

```
void draw(GLFWwindow* window){
    //create/manipulate vertex data
    VertexData vertexData[3];
    ...

    //load vertex data to gpu
    glBufferData(...);

    //draw vertex data
    glDrawArrays(...);
}
```

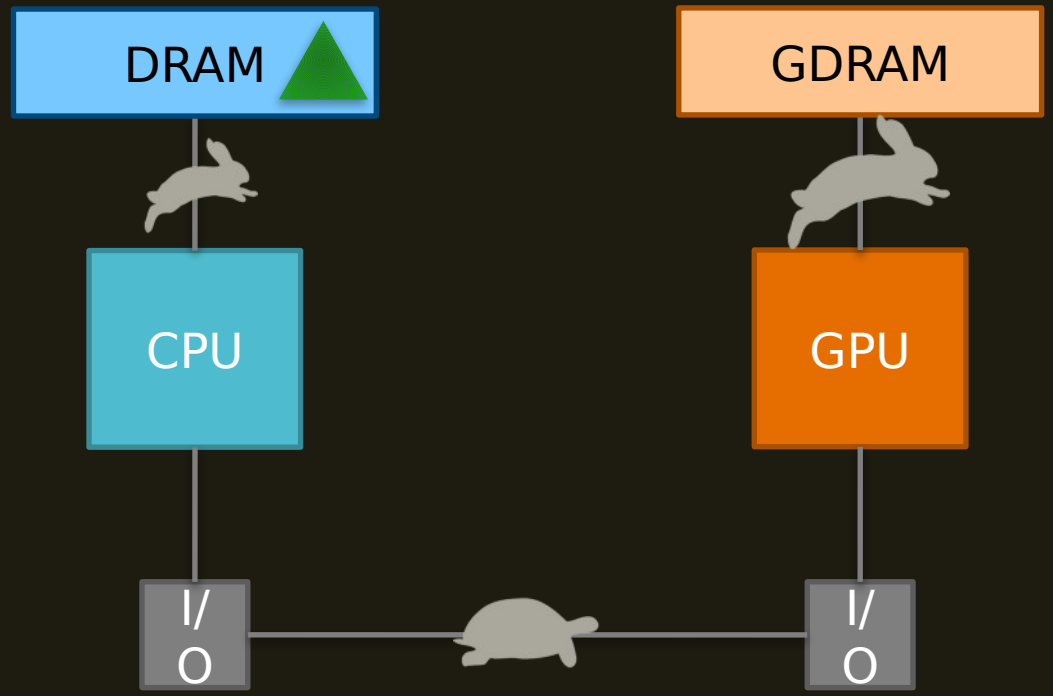


CPU/GPU Communication

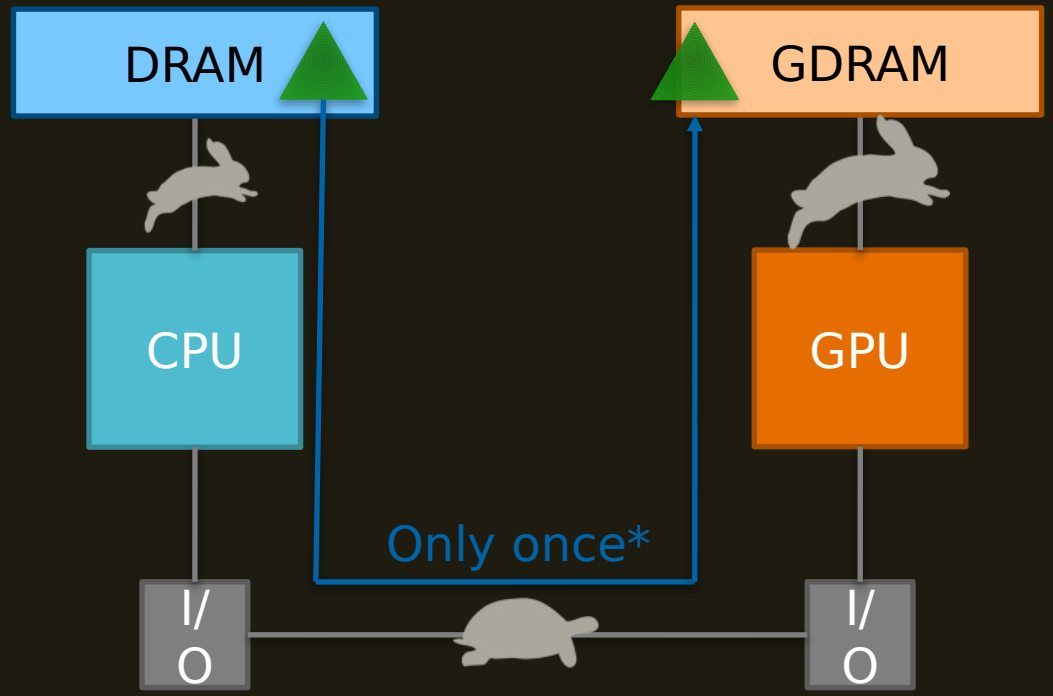
```
void draw(GLFWwindow* window){
    //create/manipulate vertex data
    VertexData vertexData[3];
    ...

    //load vertex data to gpu
    glBufferData(...);

    //draw vertex data
    glDrawArrays(...);
}
```



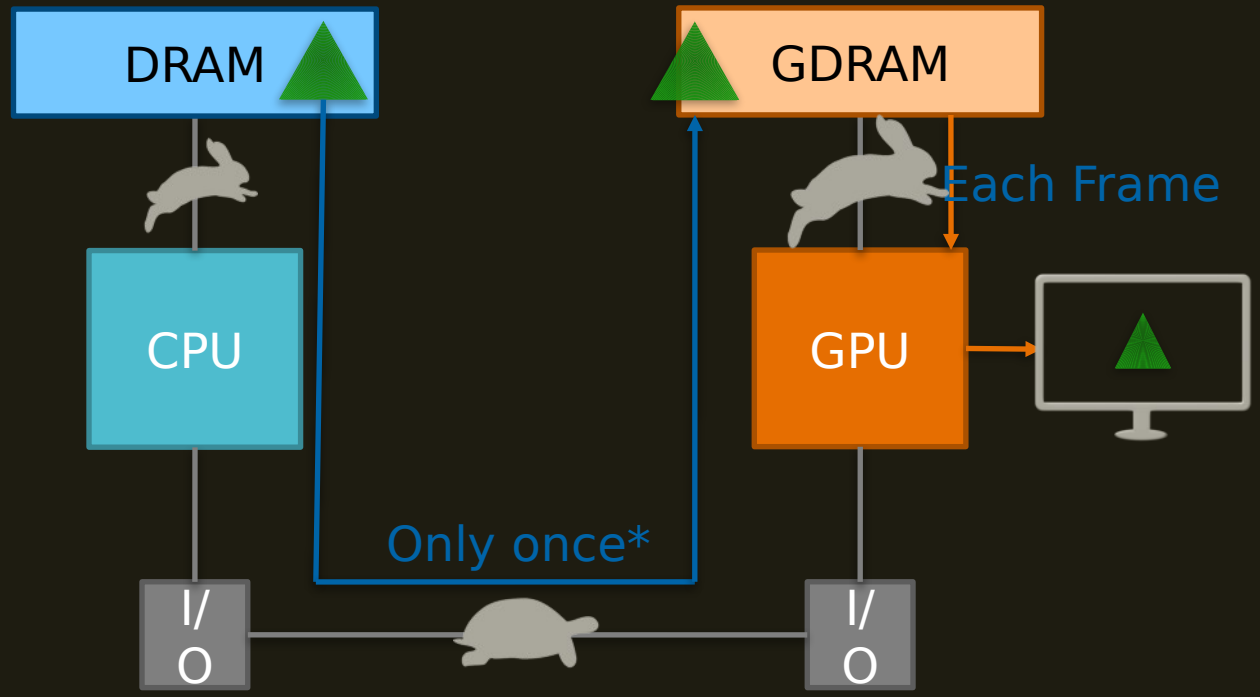
CPU/GPU Communication



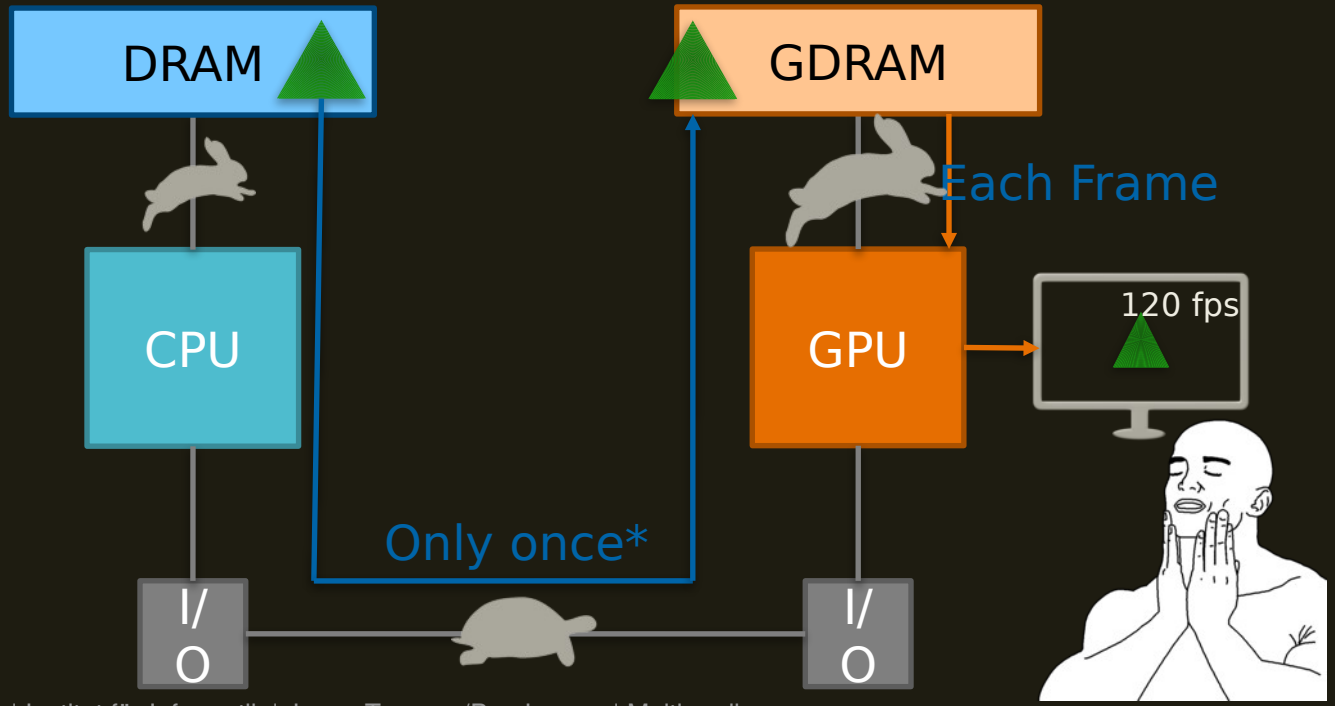
CPU/GPU Communication



CPU/GPU Communication



CPU/GPU Communication



Vertex Buffer Objects - VBO

- User defined chunks of graphics memory for vertex data

- Functions:
  - **glGenBuffers**(sizei n, uint *buffers)
  - **glBindBuffer**(enum target, uint buffer)
  - **glBufferData**(enum target, sizeiptrARB size, const void *data, enum usage)
  - **glDeleteBuffers**(sizei n, const uint *buffers)

https://www.khronos.org/registry/OpenGL-Refpages/

Vertex Array Objects - VAO

- stores layout/format of VBO

- Functions:
  - void **glGenVertexArrays**(GLsizei n, GLuint *arrays);
  - void **glBindVertexArray**(GLuint array);
  - void **glDeleteVertexArrays**(GLsizei n, const GLuint *arrays);
  - For all attributes:
    - void **glVertexAttribPointer**(GLuint index, GLint size, GLenum type, GLboolean normalized, GLsizei stride, const void * pointer);
    - void **gl<Enable/Disable>VertexAttribArray**(GLuint index);

```c
#define ATTRIB_POSITION 0
#define ATTRIB_COLOR 1

typedef struct{
    GLfloat position[3];
    GLubyte color[3];
} VertexData;

VertexData vertexData[3] = {/* ... */};
/* ... */
int vbo_handle;
glGenBuffers(1, &vbo_handle);
glBindBuffer(GL_ARRAY_BUFFER, vbo_handle);
glBufferData(GL_ARRAY_BUFFER, 3 * sizeof(VertexData), (GLvoid*)vertexData, GL_STATIC_DRAW);
/* ... */
int vao_handle;
glGenVertexArrays(1, &vao_handle);
glBindVertexArray(vao_handle);
glVertexAttribPointer(ATTRIB_POSITION, 3, GL_FLOAT, GL_FALSE,
                      sizeof(VertexData), (GLvoid*)offsetof(VertexData, position));
glVertexAttribPointer(ATTRIB_COLOR, 3, GL_UNSIGNED_BYTE, GL_TRUE,
                      sizeof(VertexData), (GLvoid*)offsetof(VertexData, color));
glEnableVertexAttribArray(ATTRIB_POSITION);
glEnableVertexAttribArray(ATTRIB_COLOR);
/* ... */
glDeleteBuffers(1, &vbo_handle);
glDeleteVertexArrays(1, &vao_handle);
```

```glsl
#version 410
layout(location = 0) in vec4 vPosition;
layout(location = 1) in vec4 vColor;

out vec4 fColor;

//Uniforms:
uniform float angle_x;
uniform float angle_y;

void main()
{
    fColor = vColor;
```

Uniforms

- constant global variable in program object
- each shader in a program shares the uniform
- can be of any (GLSL) type or struct of (GLSL) types

- beware of shader optimization!

Uniforms

C Code

```
glUseProgram(programObject);

// only once
// target, name of variable in shader
GLint mpvUniformLocation =
    glGetUniformLocation(programObject, "MVPmatrix");

float MVPmatrix[16] = {1.0, 1.0 ......, 0.0};

//...

// each frame or if there are changes
// target, count, transpose, data pointer
glUniformMatrix4fv(mpvUniformLocation, 1, GL_FALSE, MVPmatrix);
```

```
//vertex shader
#version 300 es
layout(location = 0) in vec3 vertex;

uniform mat4 MVPmatrix;

void main() {
    gl_Position =  MVP * vec4(vertex,1);
}
```

Shader Code

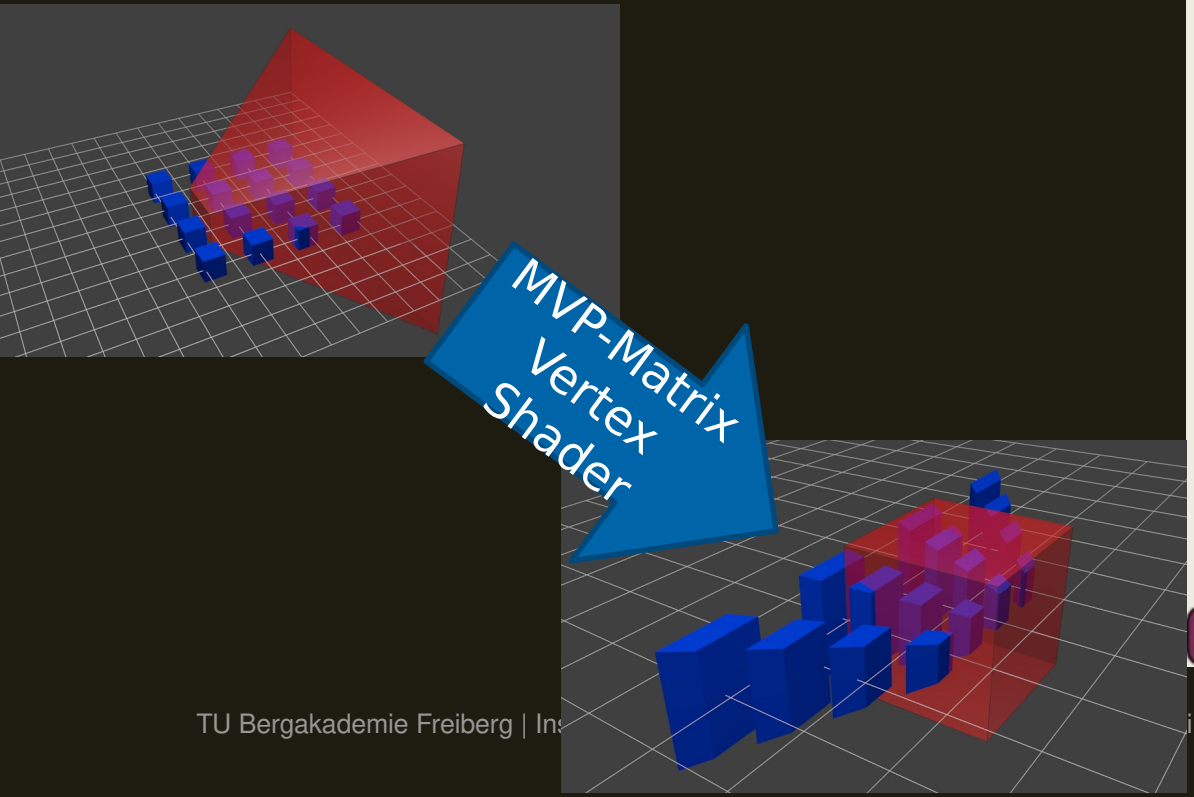
https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glUniform.xhtml

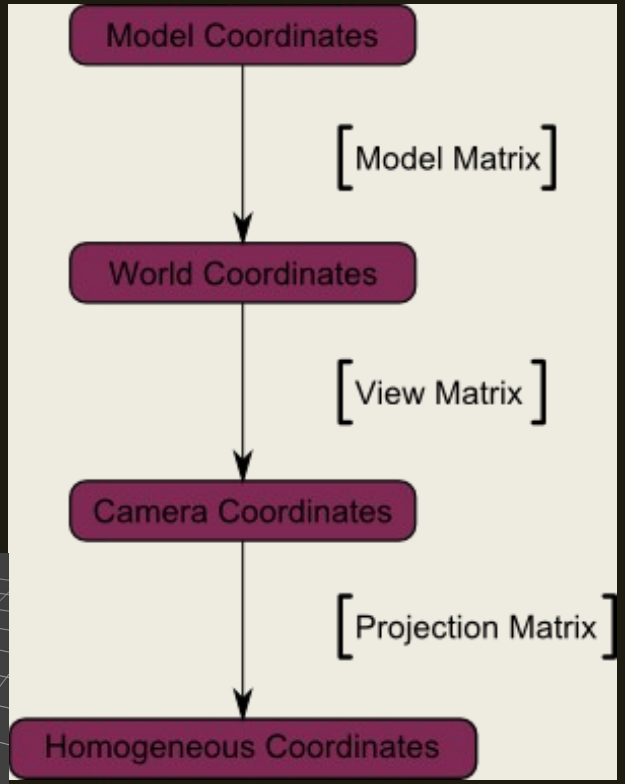
Wavefront .obj

- Geometry definition (+ material in associated mtl-File)
- Plain ASCII (comments allowed)
- Line based
  - Vertecies (v)
  - Texture Coordinates (vt)
  - Vertex Normals (vn)
  - Parameters (vp)
  - Faces (f)
  - Line (l)
  - Object name (o), Group name (g)

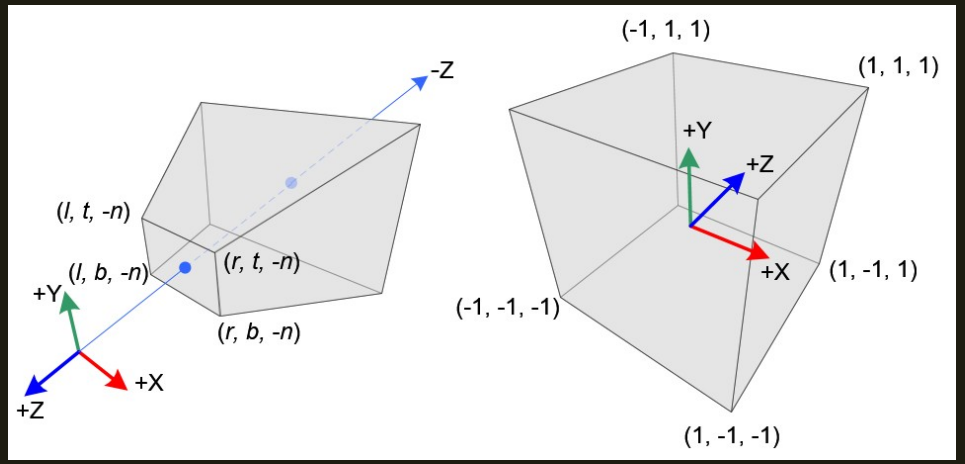
```
# cube.obj
#

g cube

v  0.0  0.0  0.0
v  0.0  0.0  1.0
v  0.0  1.0  0.0
v  0.0  1.0  1.0
v  1.0  0.0  0.0
v  1.0  0.0  1.0
v  1.0  1.0  0.0
v  1.0  1.0  1.0

vn  0.0  0.0  1.0
vn  0.0  0.0 -1.0
vn  0.0  1.0  0.0
vn  0.0 -1.0  0.0
vn  1.0  0.0  0.0
vn -1.0  0.0  0.0

f  1//2  7//2  5//2
f  1//2  3//2  7//2
f  1//6  4//6  3//6
f  1//6  2//6  4//6
f  3//3  8//3  7//3
f  3//3  4//3  8//3
f  5//5  7//5  8//5
f  5//5  8//5  6//5
f  1//4  5//4  6//4
f  1//4  6//4  2//4
f  2//1  6//1  8//1
f  2//1  8//1  4//1
```

MVP-Matrix


MVP-Matrix
Vertex
Shader

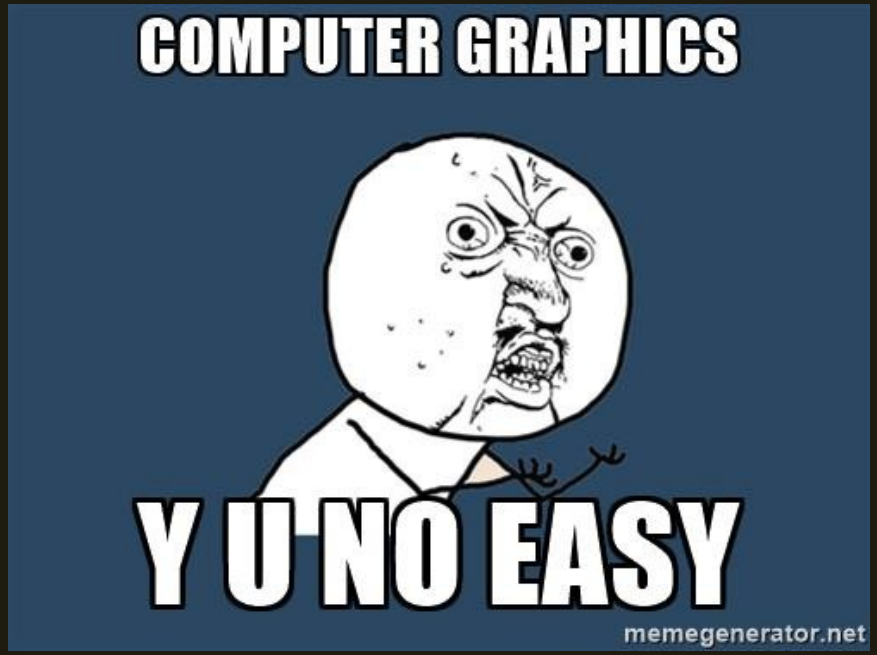

Model Coordinates
[Model Matrix]
World Coordinates
[View Matrix]
Camera Coordinates
[Projection Matrix]
Homogeneous Coordinates

P-Matrix



Beware: Transpose matrix in GLSL!

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$


COMPUTER GRAPHICS
Y U NO EASY
memegenerator.net