[aps]

{abertay /*PROGRAMMING*/ society;}

Computer Graphics

Meeting 6 - 17/09/2019

Moving to 4506

We’re thinking of moving the meetings to 4506

Opinions?

Computer Graphics

By the end of this talk you should have learned

- What are computer graphics
- How do they work
- Why you should never get me started on computer graphics

What are Computer Graphics?

What are computer graphics?

Images and videos created by computers.

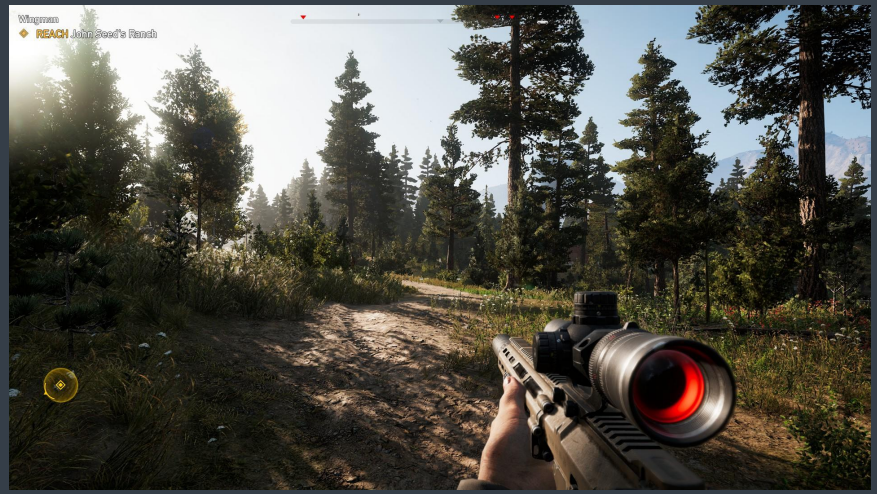
What are computer graphics?

Images and videos created by computers.

They have many applications, such as...

Video games

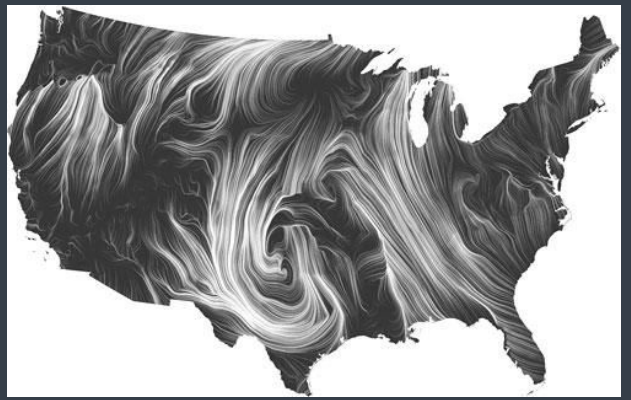

CONTRACT: DEADLY DELIGHTS
Collect your reward from the guard


Wingman
REACH John Seed's Ranch


Equus Oils

Movies and TV

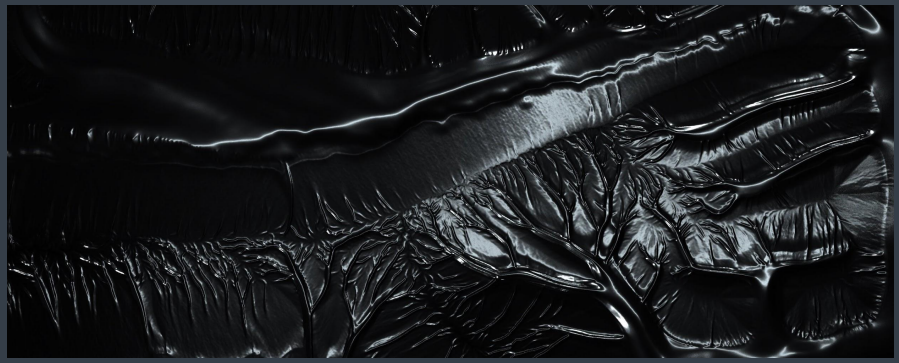
Data Visualization

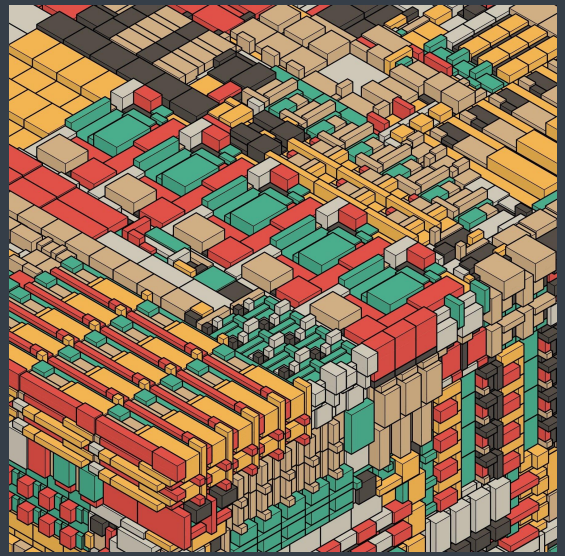


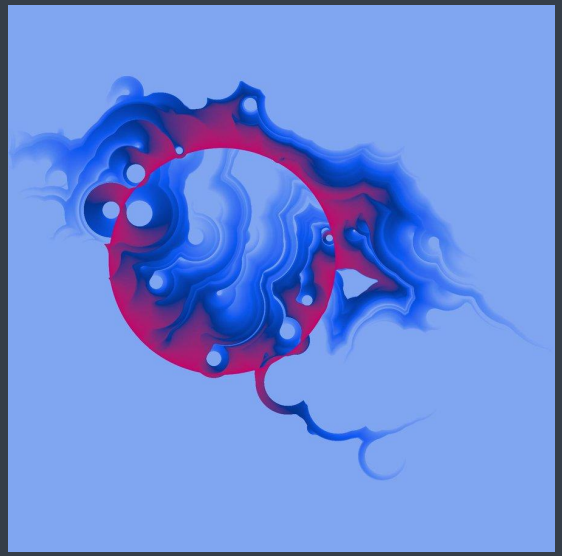
Algorithmic art

Usually uses randomization

Often animated or interactive

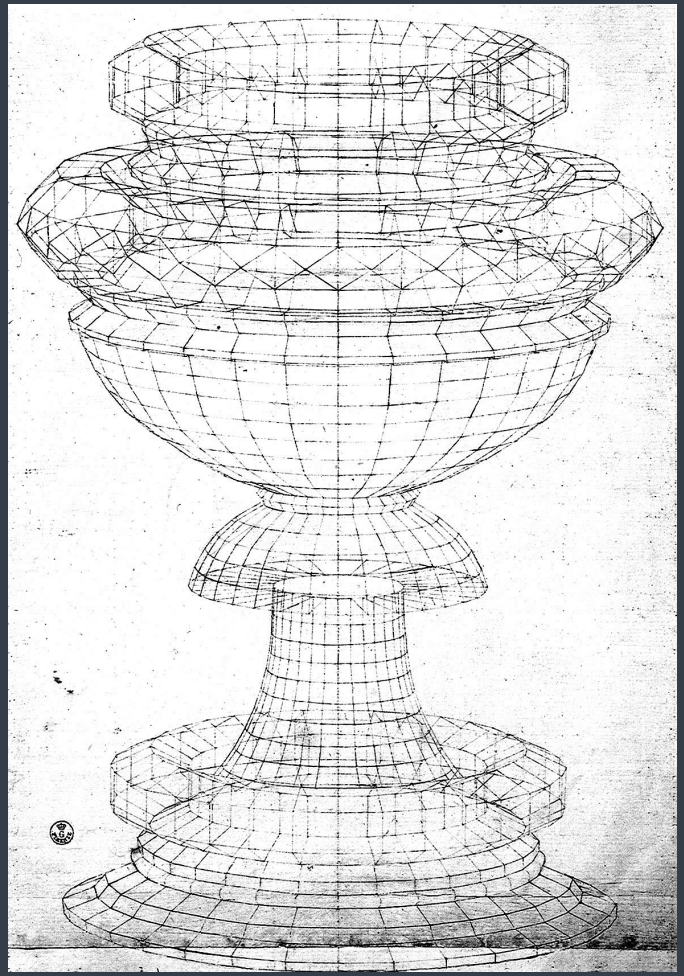


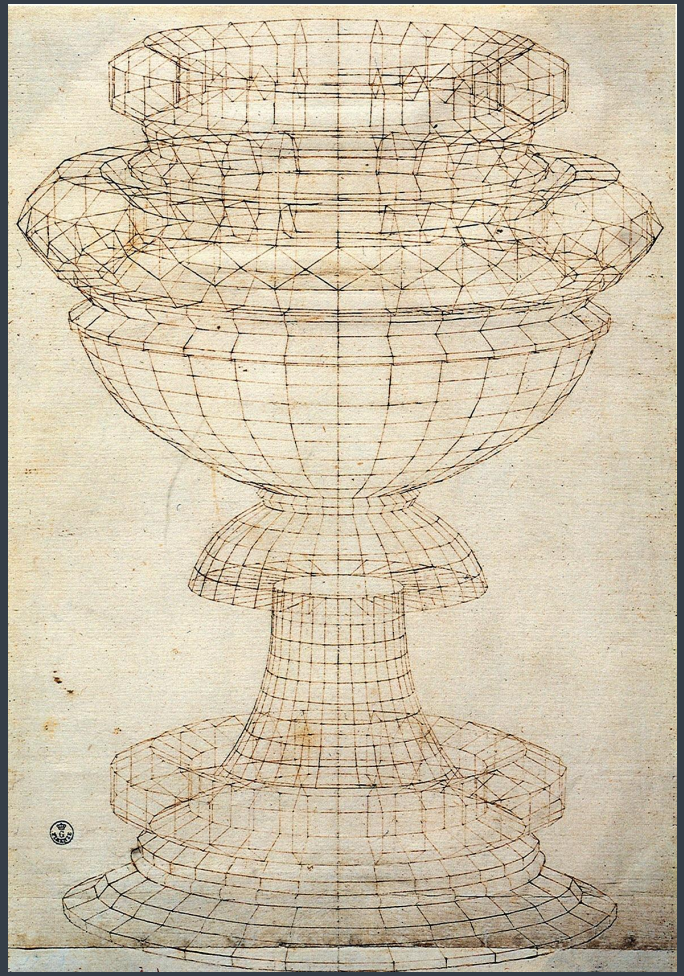




CW from top: Sage Jenson, Thomas Lin Pedersen, Kjetil Golid, Daniel Travers

A brief history...



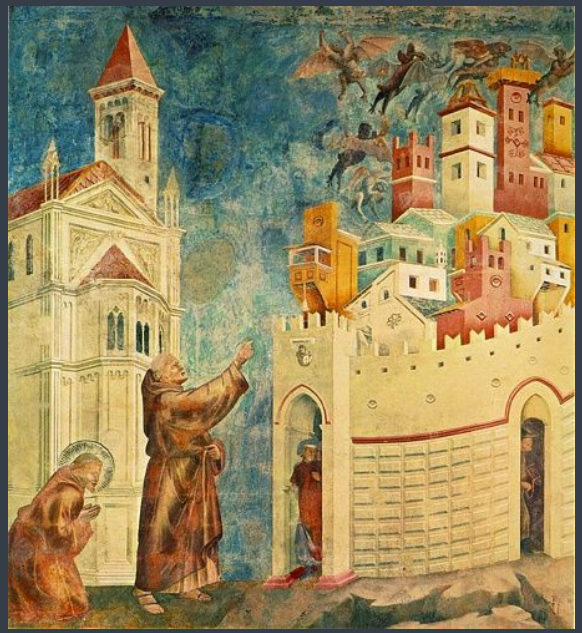


Vase In Perspective
Paolo Uccello
15th Century

Many of the fundamentals of computer graphics were developed centuries before the first computers!

For example, Albrecht Dürer is credited with the invention of ray tracing in the 16th century

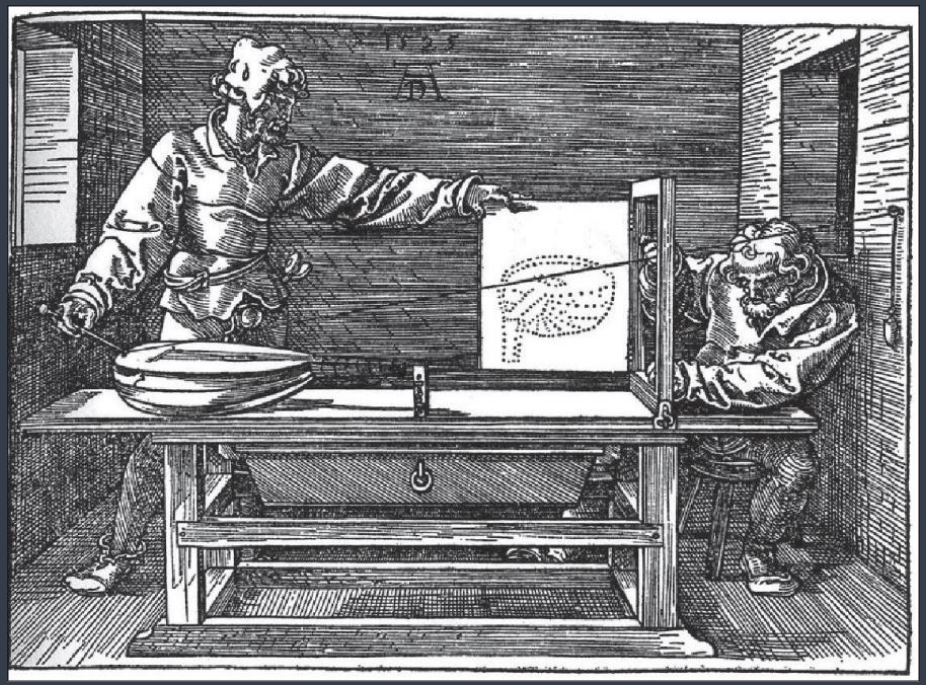
The invention of perspective

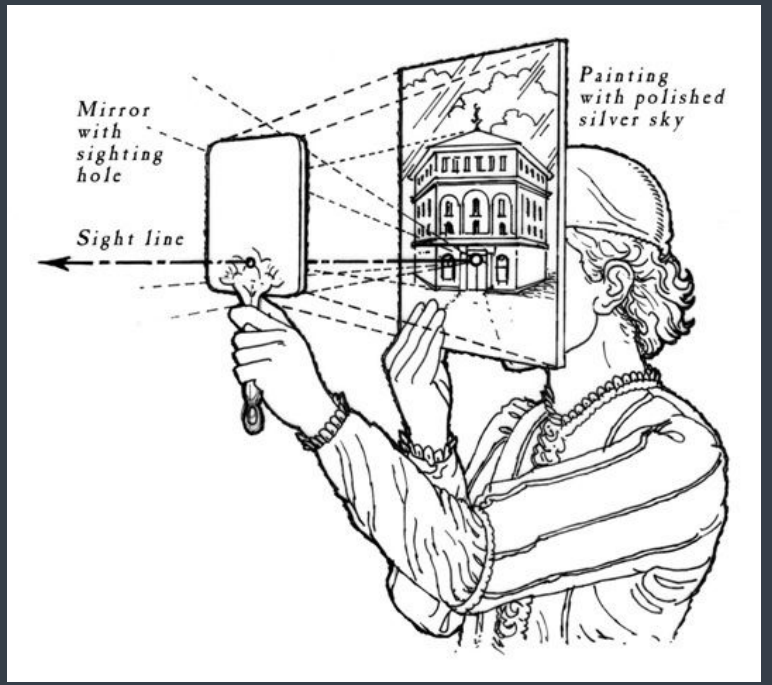


Before perspective

After perspective

Perspective was developed in the Renaissance by Filippo Brunelleschi and other artists, architects and mathematicians

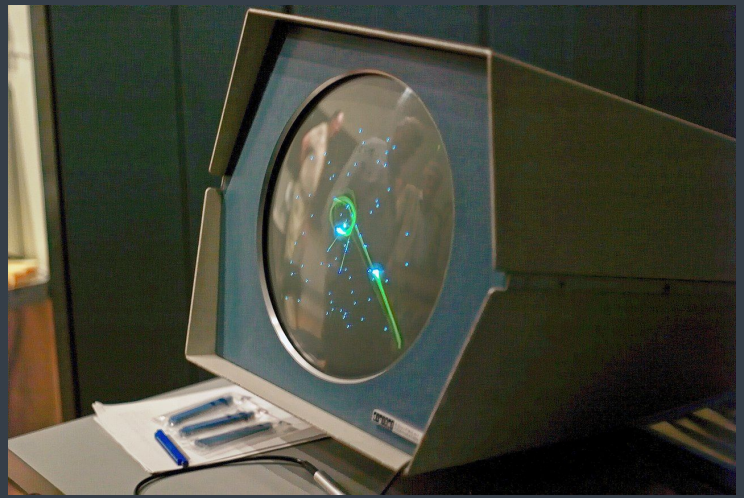




Closer to the present...

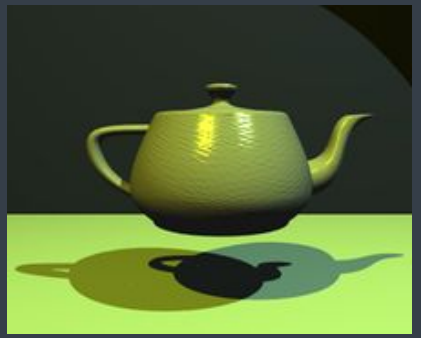
The term “computer graphics” was coined in the 60’s

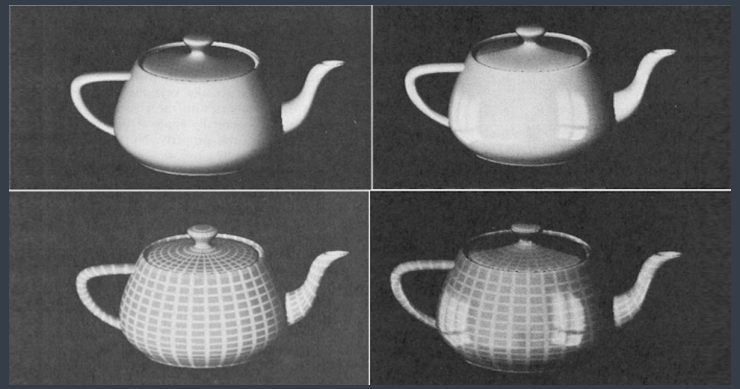
But it was mostly just lines on oscilloscopes

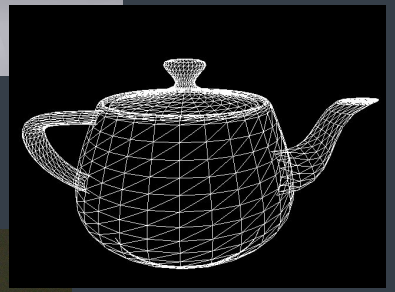


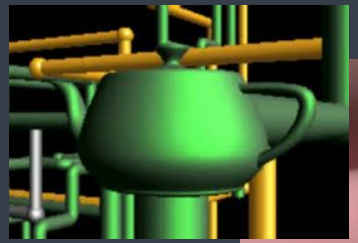
3D computer graphics began to emerge in the 70’s

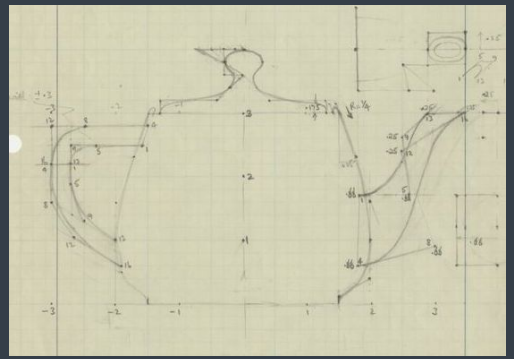
And they needed a test object...



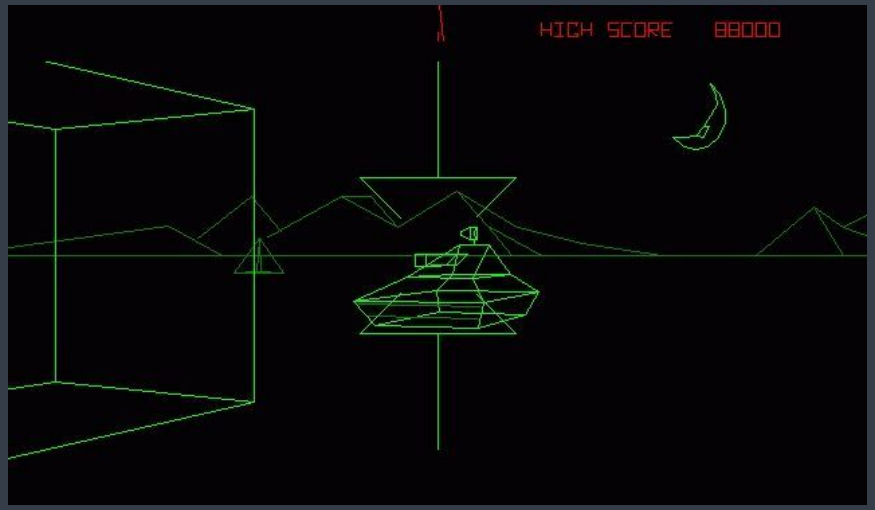








80’s Games



Battlezone - 1980

Elite - 1984

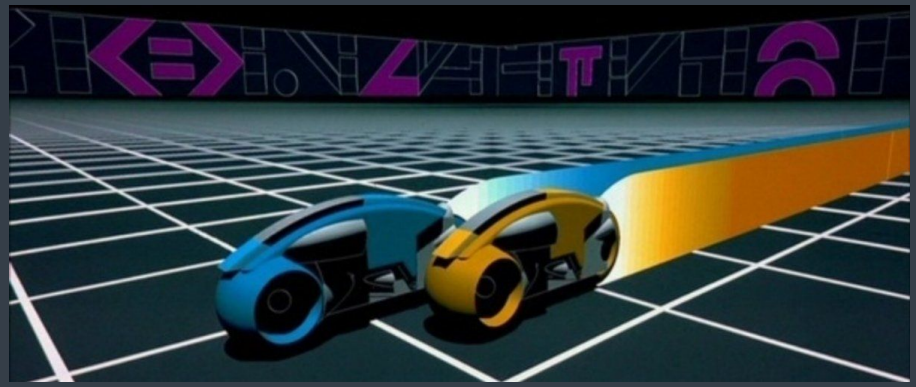
90’s Games

Wolfenstein 3D - 1992

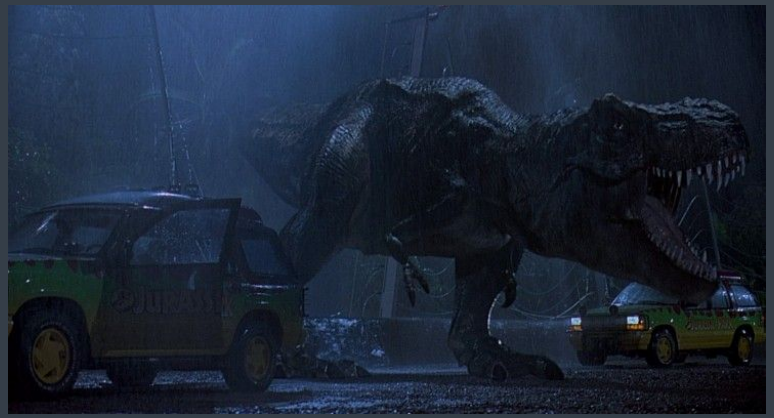
Doom - 1993

Quake - 1996

Movies



Tron - 1982



Jurassic Park - 1993

Toy Story - 1995

How do they work?

Several common approaches

Ray tracing

Ray marching

Rasterization

Ray tracing

For each pixel, cast a ray away from the viewer and see what surface it hits.

Lighting - trace a ray from the surface to a light source

Reflections - trace another ray bouncing off the surface

Path tracing

A more realistic type of ray tracing

Rough surfaces scatter light. This can be simulated by ray tracing lots of times and bouncing off in a random direction each time. The results are then combined to get a single value.

Path tracing

Very computationally expensive!

Good for handling real light properties such as refraction and depth of field

Used for CGI in movies

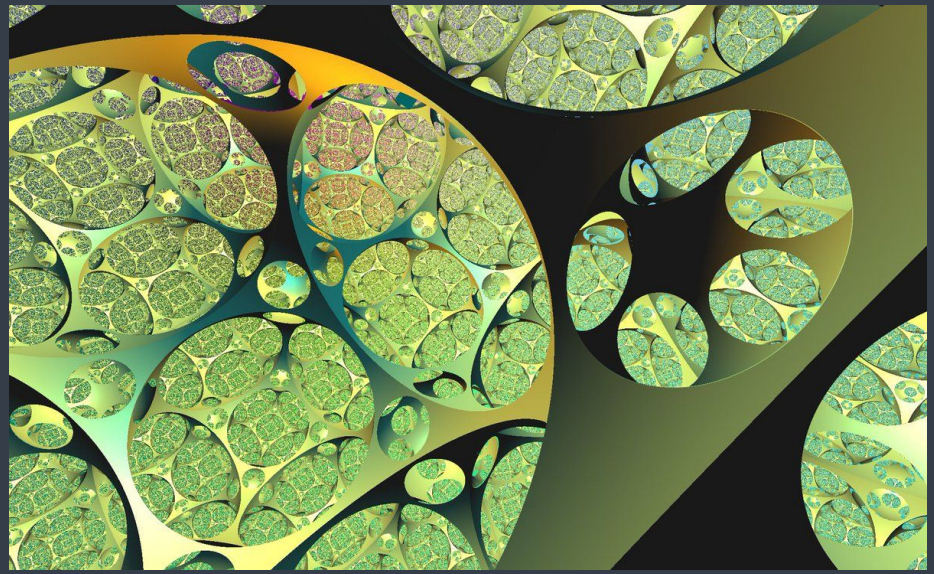
Ray marching

Like ray tracing but you “march” along the ray, sampling at regular intervals

Good for when:

- ray intersection calculations aren’t possible
- Volumetric data is required

Ray marching



Rasterisation

Objects are made of vertices and triangles

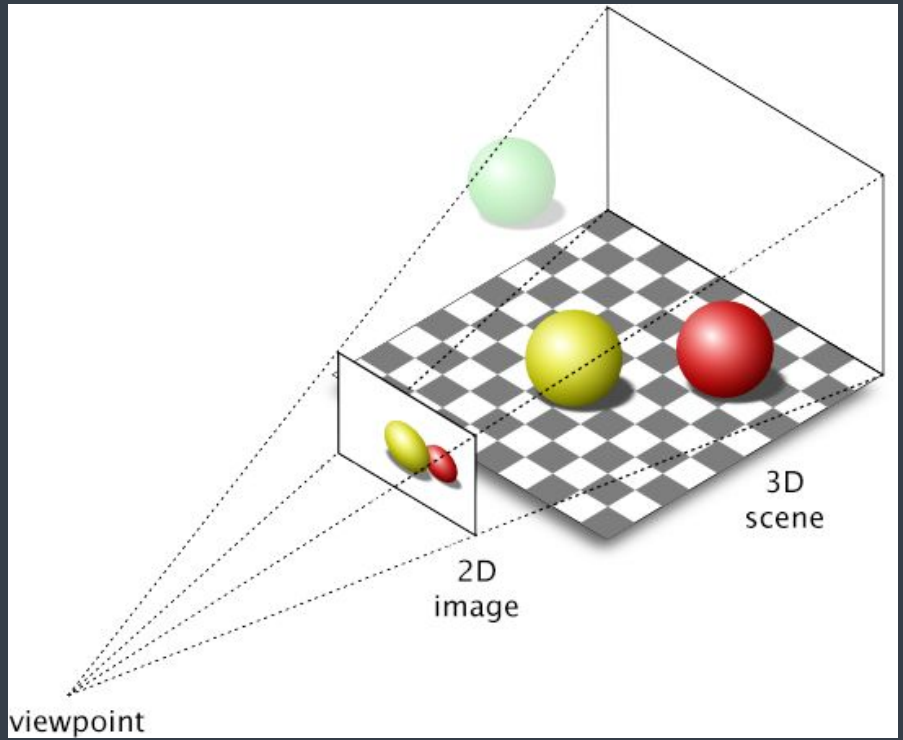
Vertex positions are projected onto the screen

Each triangle is drawn onto all the pixels it covers (unless it’s behind another triangle)

Rasterisation

The vertices are projected by multiplying them by a projection matrix.

Basically it distorts the view frustum into a cube, which is then flattened.



Rasterisation

Really fast!

Used for real-time applications like video games

Doesn’t simulate light properties like reflections so they have to be faked

Some techniques

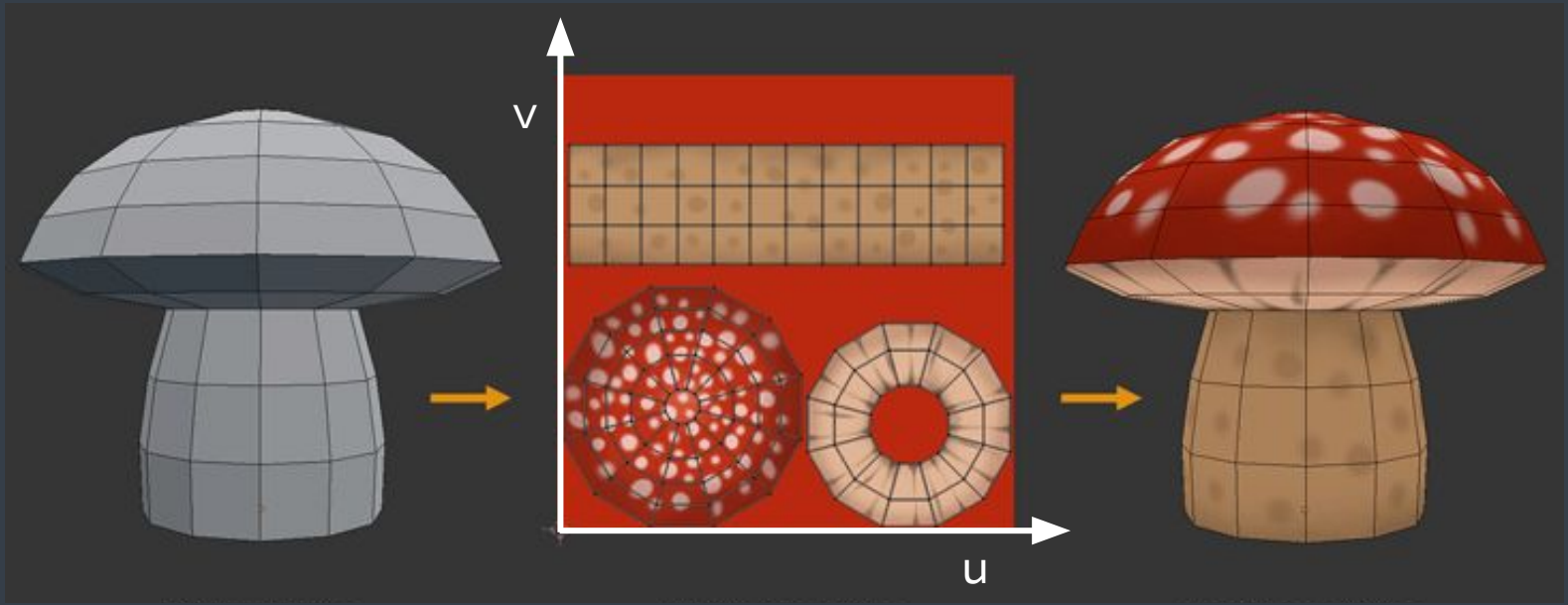
UV mapping

Each vertex has a 2D texture coordinate

u and v instead of x and y

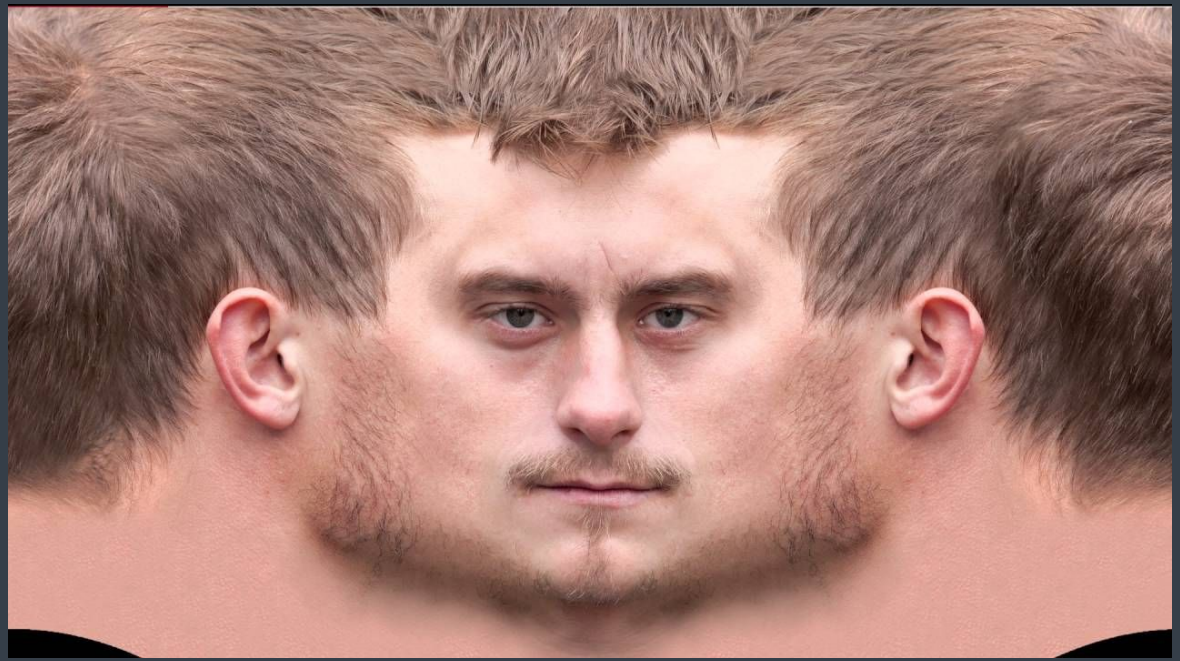
Ranging from (0, 0) to (1, 1)

UV mapping

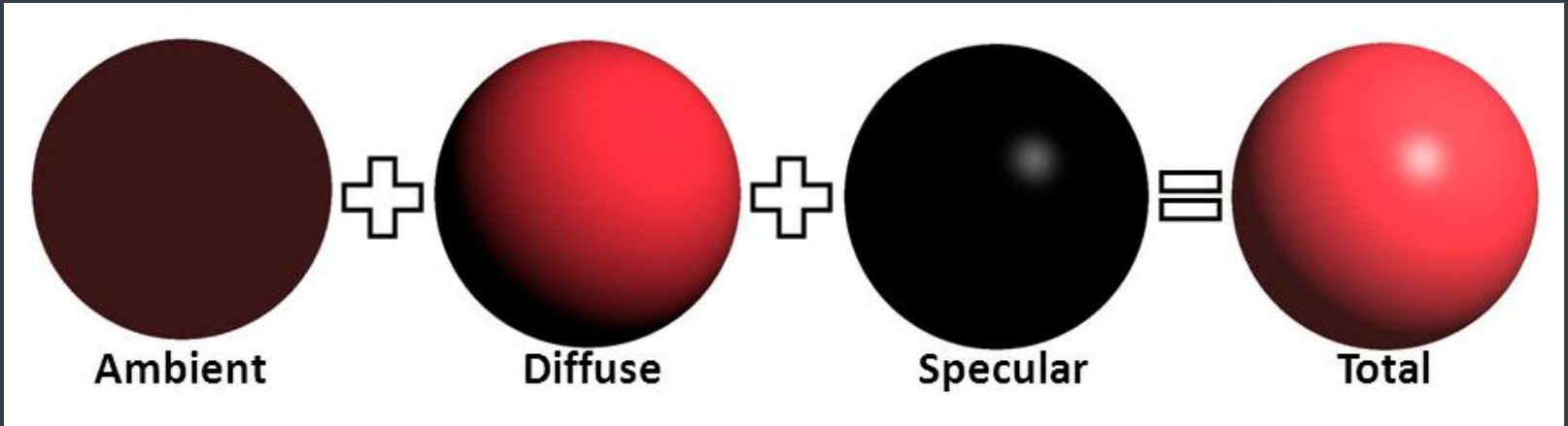


UV mapping

Character texture files are terrifying



Lighting



Lighting

Real lighting is incredibly complex!

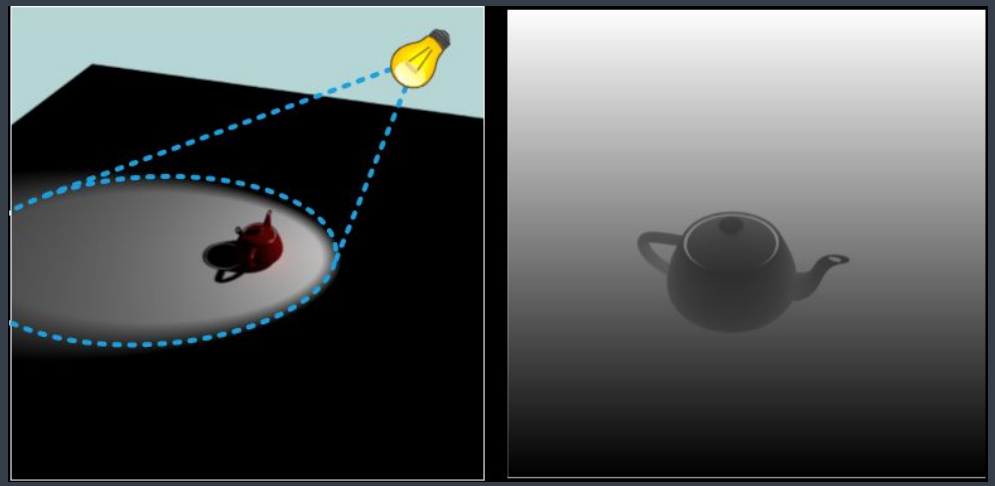
Accurate simulation is too expensive

Instead we fake a bunch of its properties

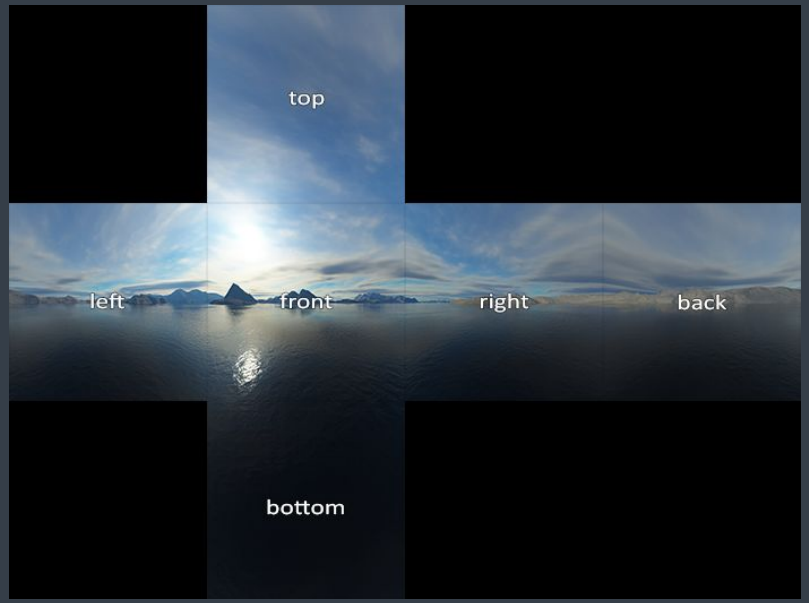
Shadow mapping

Render the scene from the point of view of the light source

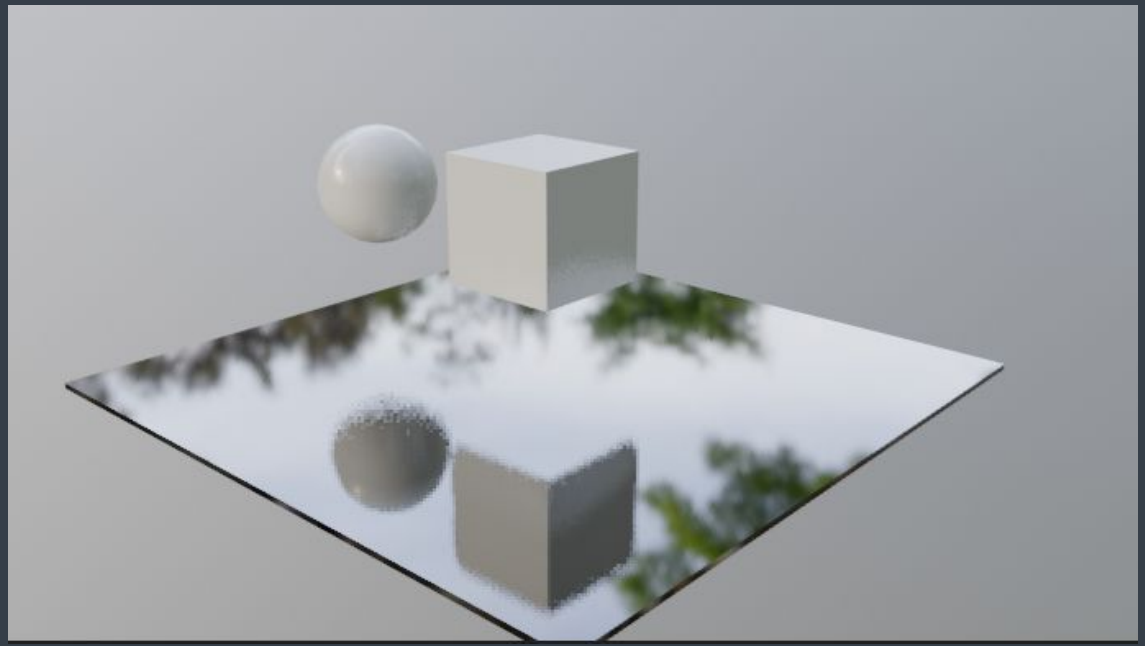
Only light pixels the light source can see



Cubemap reflections



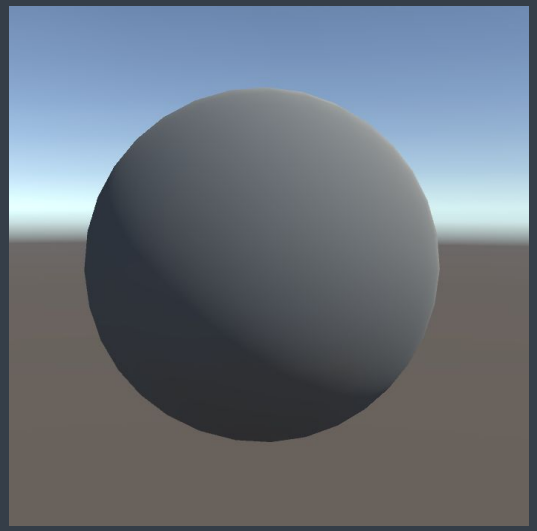
Screen space reflections



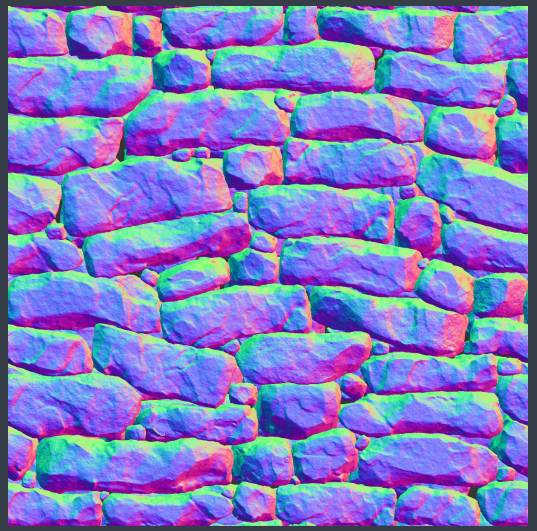
Ambient occlusion

Can be baked (precomputed) or generated from the depth buffer

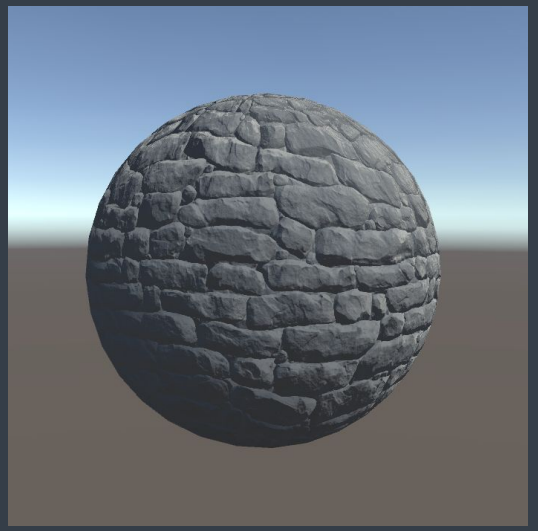
Normal maps (aka bump maps)



+



=



Same mesh!

Shaders

What are shaders?

Little programs that run concurrently on the GPU

Various types:

- Fragment shaders - executed once per pixel of a material or the whole screen
- Vertex shaders - executed once per vertex of a model
- Geometry shaders - used to create additional geometry
- Compute shaders - used for arbitrary data processing

What are shaders?

For example, a material’s fragment shader might receive:

- UV coordinates
- Normal direction
- Vertex position
- Vertex colour

And output:

- Pixel colour

What are shaders?

Modern games use shaders to create a huge variety of materials and image effects. Almost every cool visual effect in a game is created using shaders. Some examples:

- Rippling water
- Cel shading
- Motion blur
- Heat haze over fire
- Sunbeams
- Bloom

Shadertoy

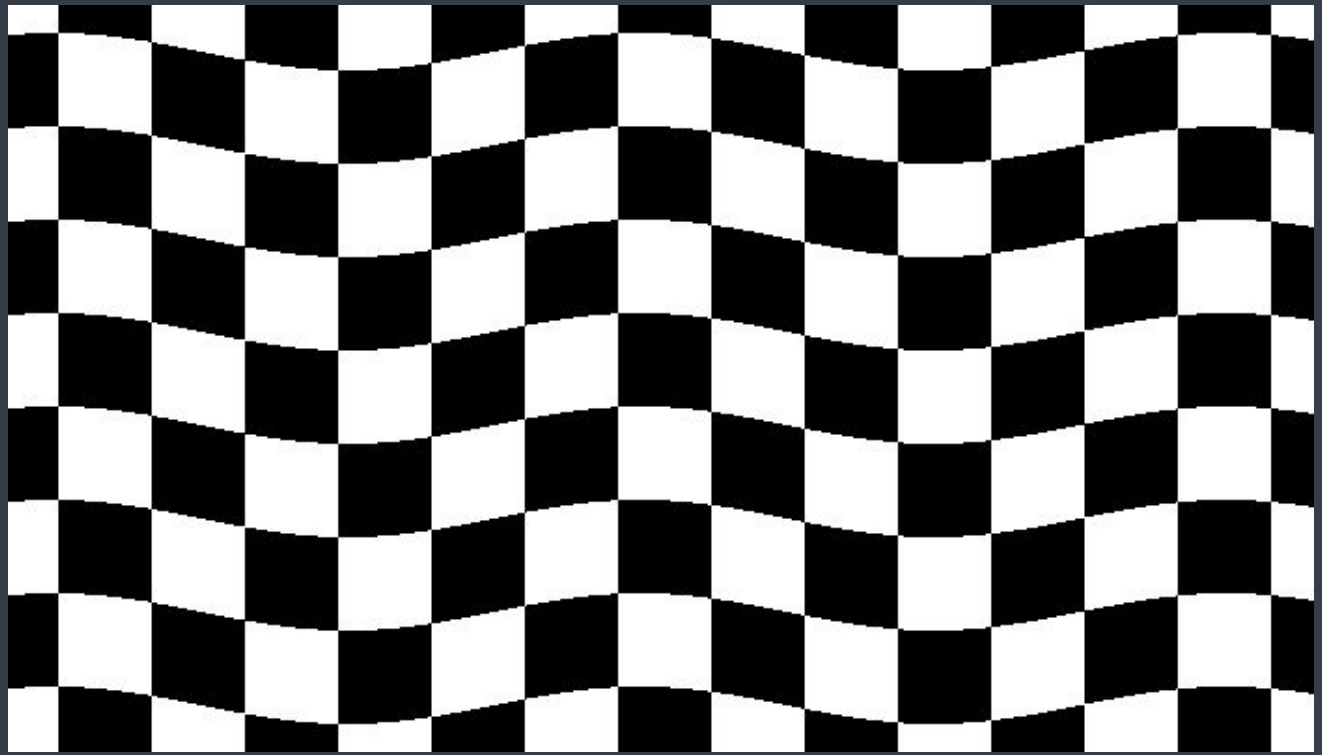
Shadertoy

An online tool for writing shaders with GLSL (OpenGL Shading Language)

You don’t need an account unless you want to save or share your shaders

Go to https://www.shadertoy.com/ and click “New”

Let’s write a shader!



Let's write a shader!

```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    float waveAmount = 10.0 * sin(fragCoord.x / 50.0 - iTime * 5.0);

    bool xb = mod(fragCoord.x, 100.0) > 50.0;
    bool yb = mod(fragCoord.y + waveAmount, 100.0) > 50.0;

    vec3 col = (xb == yb) ? vec3(1, 1, 1) : vec3(0, 0, 0);

    fragColor = vec4(col, 1);
}
```

Try recreating one of these: