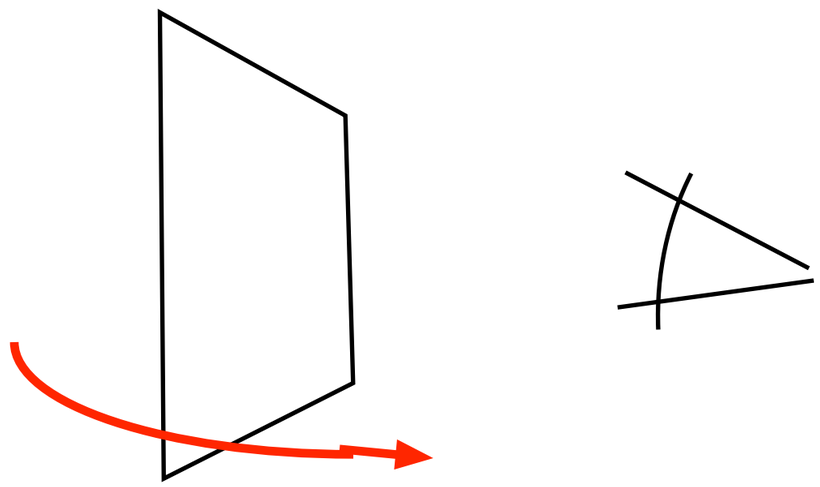
Particles in 3D? You need:

Billboards

A texture mapped polygon, which always faces the viewer



Billboards

2D images placed on surfaces that are always facing the camera

“Sprites” in older action games

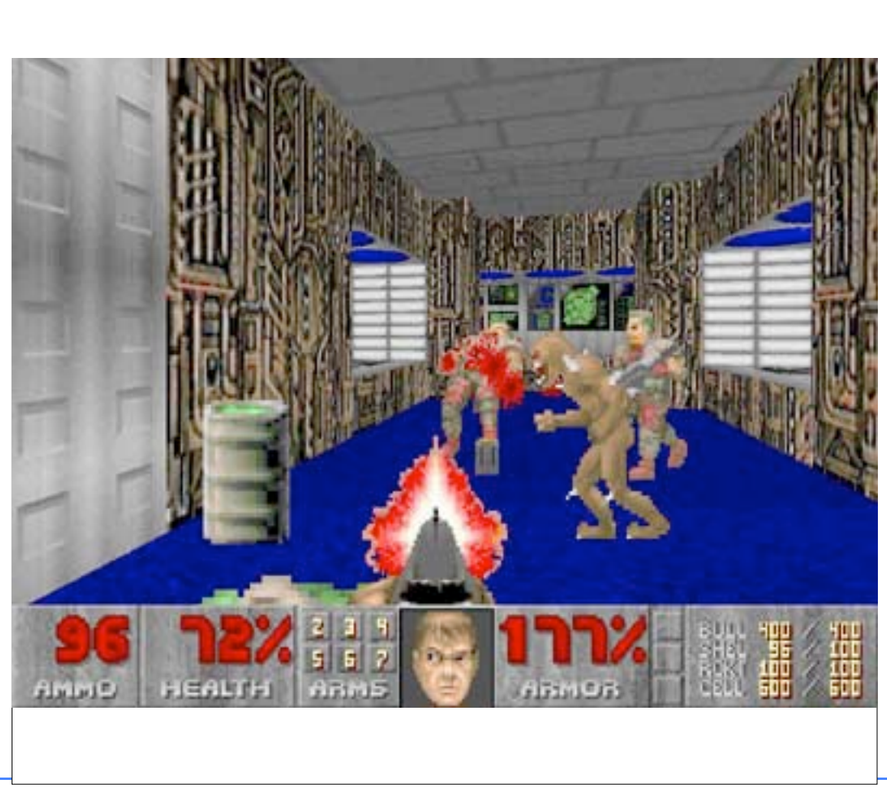
Advanced version: “impostors”

Often used for complex objects even today

Example: forests

Classic example: Doom

Big billboards. No 3D models.



Implementing billboards

- Billboards generally need transparency!

=> Special care is needed to avoid problems with Z-buffering!

Transparency

1) Sort by distance (Painter's algorithm)

- Perfect results
- Supports semi-transparency
- Required sorting and a separate stage

2) Discard fragments

- No semi-transparency
- Some artifacts at edges
- Very easy

3) For some cases: Additive blending

discard; as cheap solution to transparency

```
#version 150

out vec4 outColor;

in vec2 texCoord;
uniform sampler2D tex;

void main(void)
{
    vec4 t = texture(tex, texCoord);
    if (t.a < 0.01) discard;
    else
        outColor = t;
}
```

Blending = transparency

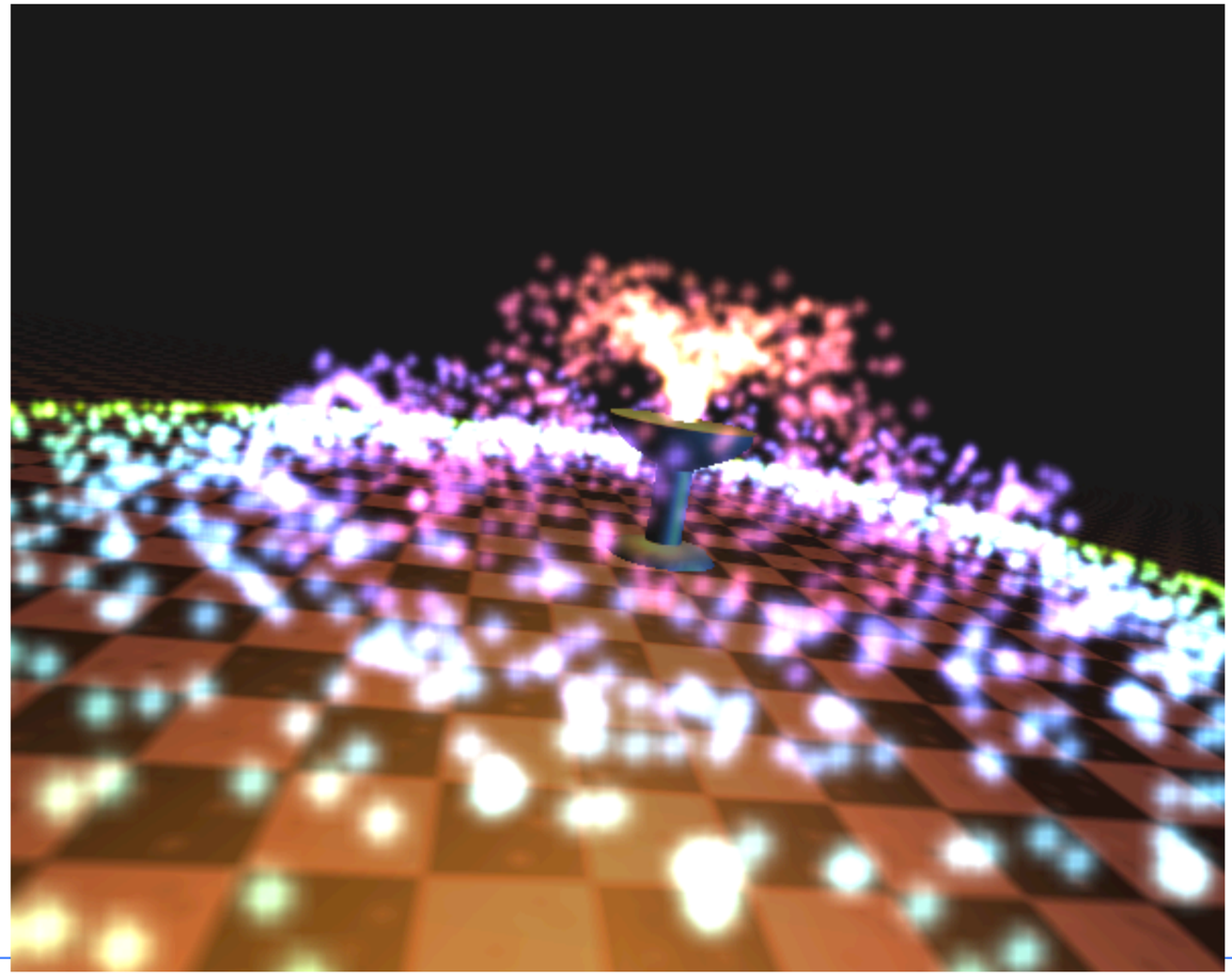
Usually

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,
      GL_ONE_MINUS_SRC_ALPHA);
```

but for *additive blending* it is

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE);
```

Old example with additive blending



Billboard texture formats

• Texture files must support transparency!

File formats:

TGA
PNG

TGA: simple format, good for demos

Libraries for PNG:
libPNG
PNGlite by Daniel Karling

Trick without transparency: Indicate with some unusual color.

Implementing billboards

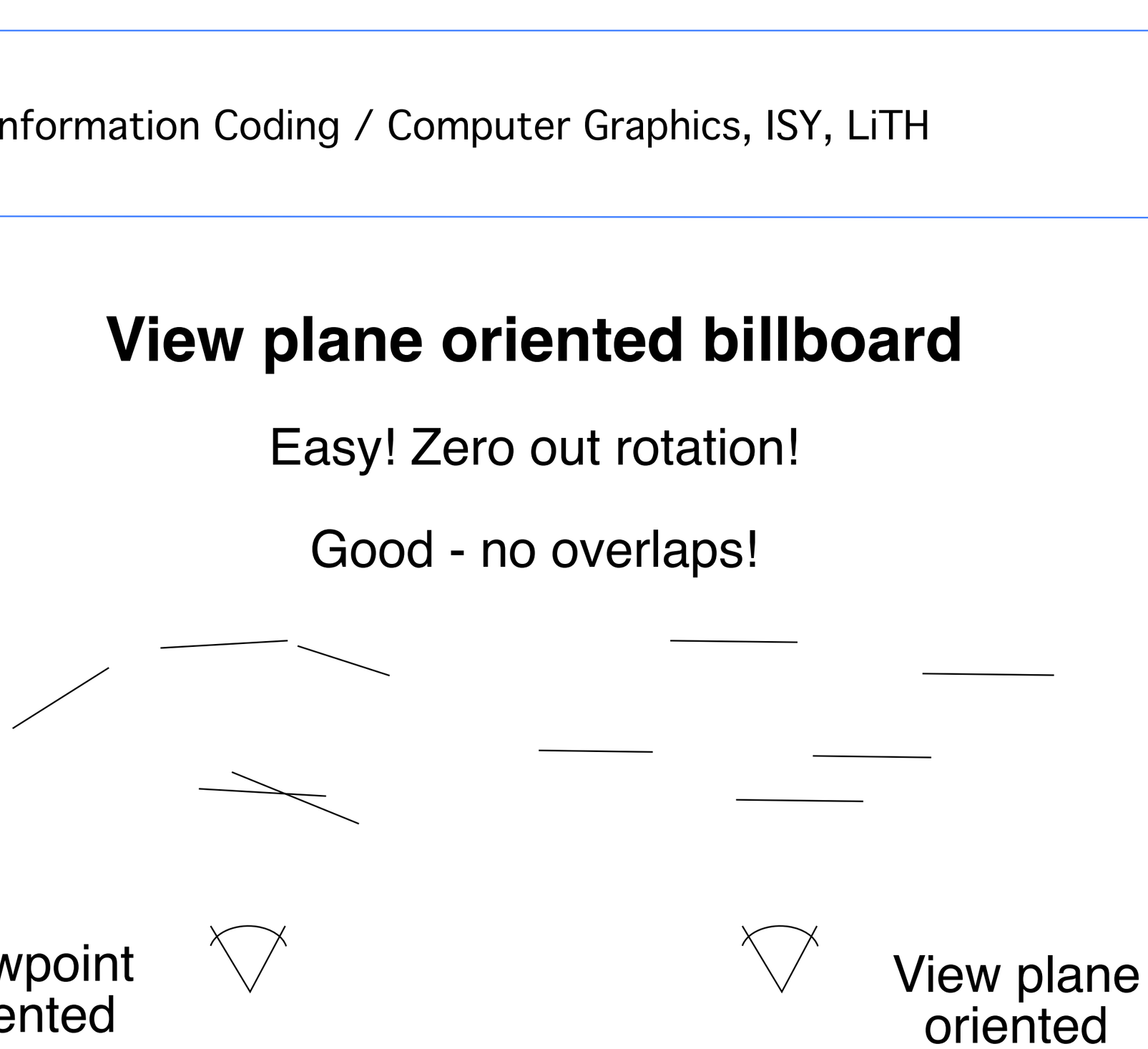
Variants:

- World oriented billboard
- Viewpoint oriented billboard (face the camera in full 3D)
- Axial (viewpoint) billboard
- View plane oriented billboard
- View plane oriented axial billboard

View plane oriented billboard

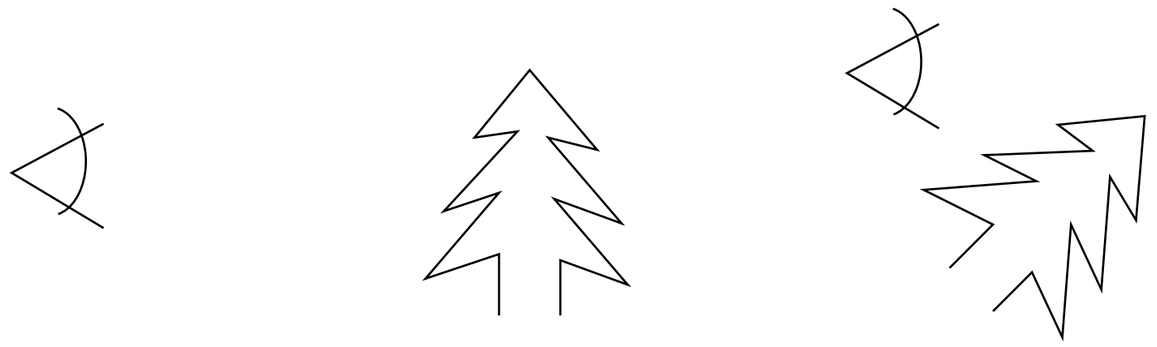
Easy! Zero out rotation!

Good - no overlaps!

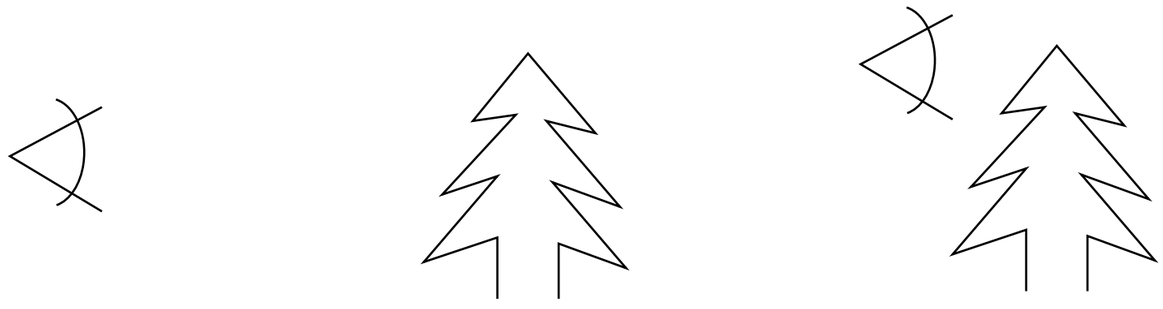


Axial billboard

Rotate around Y



Non-axial



Axial
Suitable for
trees etc

Implementing axial billboards

Project forward/eye vector on the XZ plane, find sin and cos from the normalized vector.

Same method for both, just select vector.

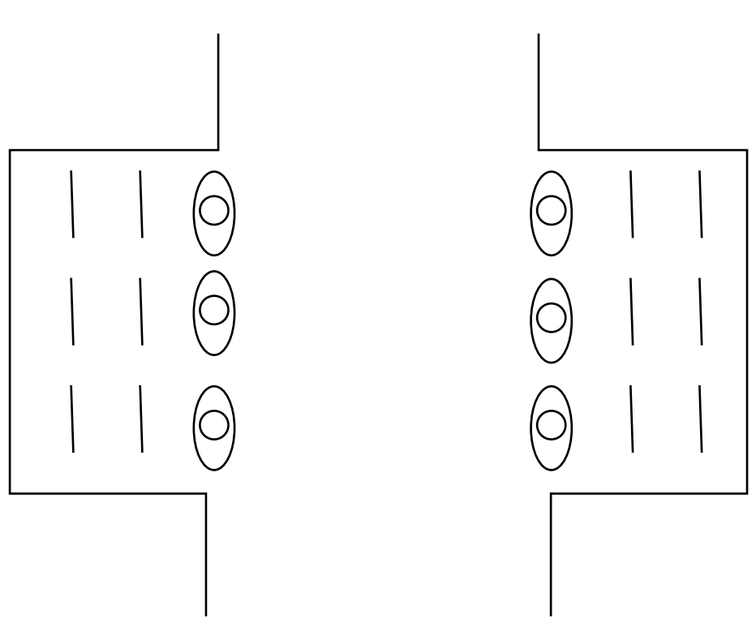
Full 3D viewpoint oriented billboard

Change of basis solution

Z vector from viewpoint, pick an up vector (usually Y axis), form basis with cross products

World oriented billboard

No camera dependent rotation



Example: Tomb Raider 2 Terracotta warriors (Temple of Xian)

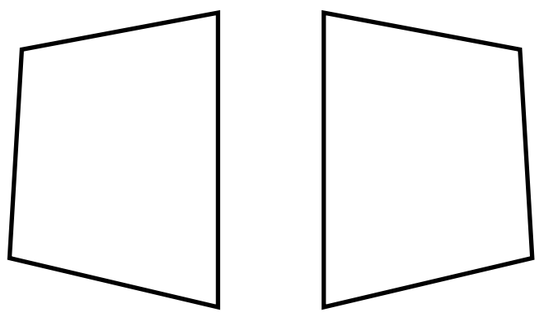
A grid of billboards

	Non-axial	Axial
Viewpoint oriented	Construct basis	Clear rotations
View plane oriented	Construct basis based on view plane	Construct axial rotation

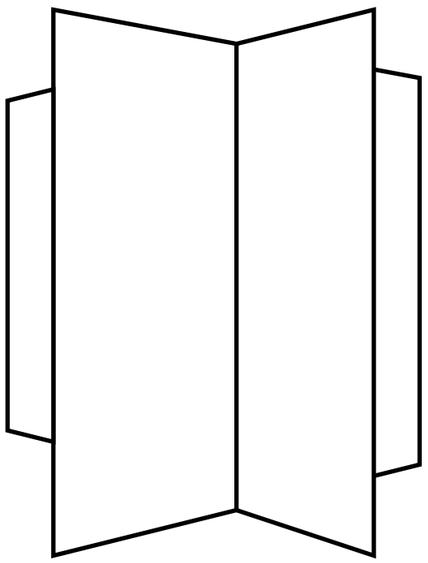
World oriented billboard not in grid

Billboard variants

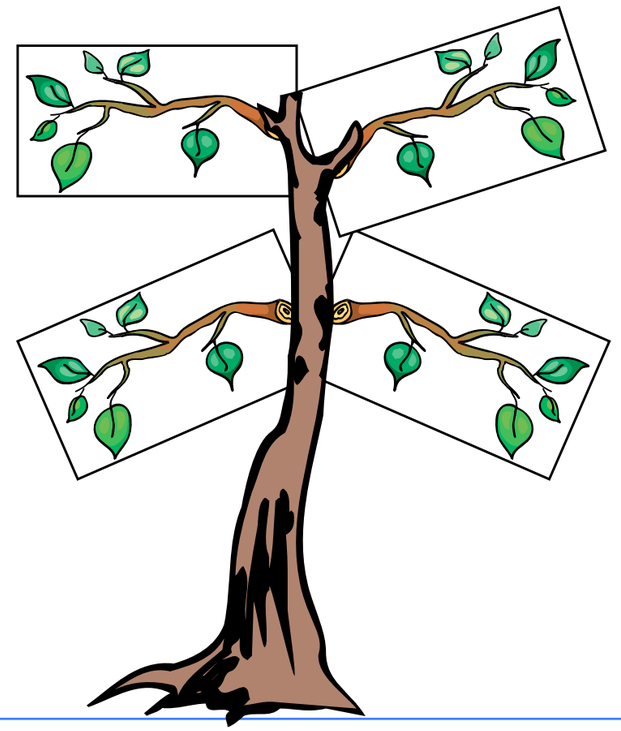
1) Always facing the camera.
One triangle or quad is enough!



2) A few polygons.
Good for world oriented billboards.



3) Multiple billboards.
Simplify complex objects om moderate distance.



Temple of Xian

Statues behind first row are billboards!

Billboard variants

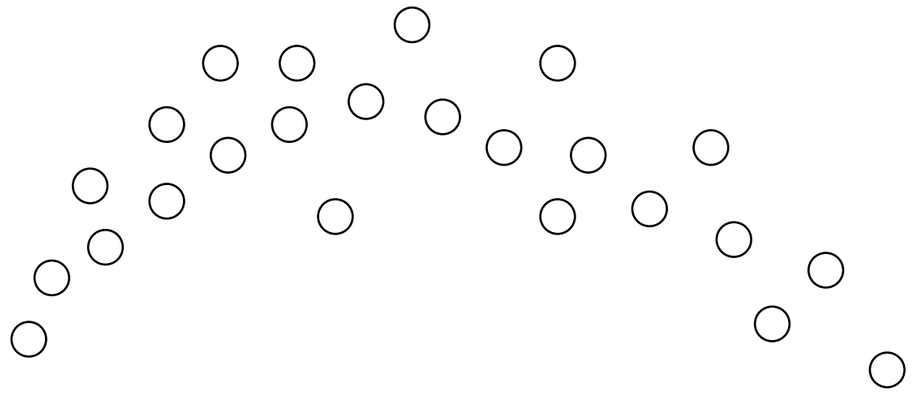
The objects on the plate are billboards - some 2-poly!

Application: Particle systems

Explosions, rain, fountains, smoke...

Excellent billboarding application

Many small objects - good opportunities to ”cheat” with transparency problems



More about this in the Animation part!

Impostors

”Live” billboards

Render to texture, update sometimes

Render as other billboards

Decide when to update

Instancing

Draw a large number of the same model!

Each instance has an index, the instance number.

glDrawArraysInstanced(GL_TRIANGLES, 0, 3, 10);

draws a triangle 10 times!

gl_InstanceID tells the shader which instance we have.
Use for affecting position.

Billboard instancing demo

One single call to glDrawArraysInstanced

Position trivially affected by gl_Instance ID

```
#version 150

in  vec3 in_Position;
uniform mat4 myMatrix;
uniform float angle;
uniform float slope;
out vec2 texCoord;

void main(void)
{
	mat4 r;
	float a = angle + gl_InstanceID * 0.5;
	float rr = 1.0 - slope * gl_InstanceID * 0.01;
	r[0] = rr*vec4(cos(a), -sin(a), 0, 0);
	r[1] = rr*vec4(sin(a), cos(a), 0, 0);
	r[2] = vec4(0, 0, 1, 0);
	r[3] = vec4(0, 0, 0, 1);
	texCoord.s = in_Position.x+0.5;
	texCoord.t = in_Position.y+0.5;
	gl_Position = r * myMatrix *  vec4(in_Position,
1.0);
}
```

Instancing complex models

Less significant; A more complex model puts enough load on the system to hide the impact of instancing.