

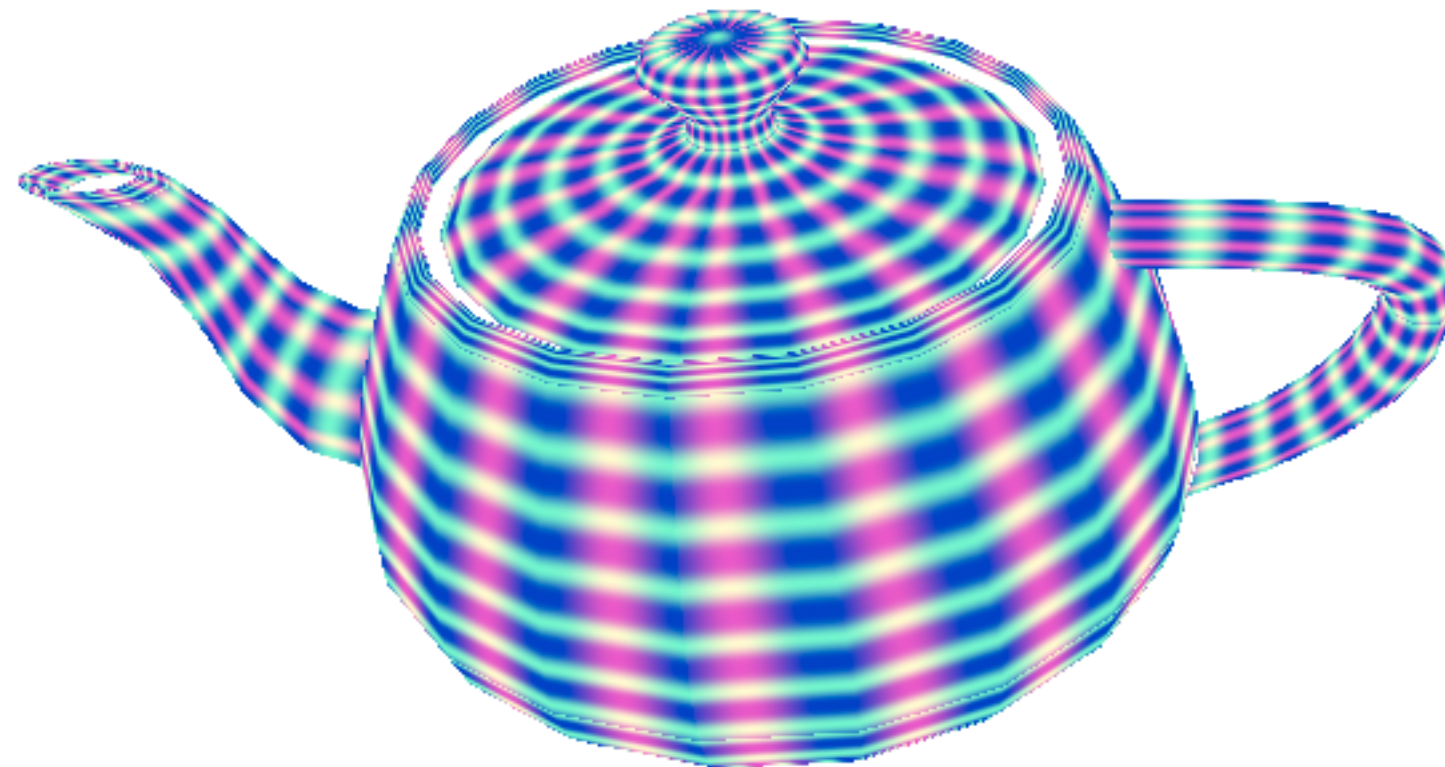


Information Coding / Computer Graphics, ISY, LiTH

TNM084

Procedural images

Ingemar Ragnemalm, ISY





Information Coding / Computer Graphics, ISY, LiTH

Lecture 8

More on terrains

Geometry shaders

Tessellation shaders

Grass



Lab 3

We lost some time, but I think most are more or less done with the tree generation.

Using GLUGG seems to work just fine. The link to the archive was broken and it was hard to unpack .tgz files. All this is fixed.

You needed to figure out how to deal with recursion as well as using a matrix stack.



Extensions on Lab 3

What can you do beyond the lab?

Extended terrains: Bigger terrains, more features for the terrains

Extended tree generation: Tropism, organic growth, more parameters, different tree types



Information Coding / Computer Graphics, ISY, LiTH

Projects

It is time to submit project proposals!

I hope I have given feedback on all proposals.



More on multi-patch terrains

Frustum culling

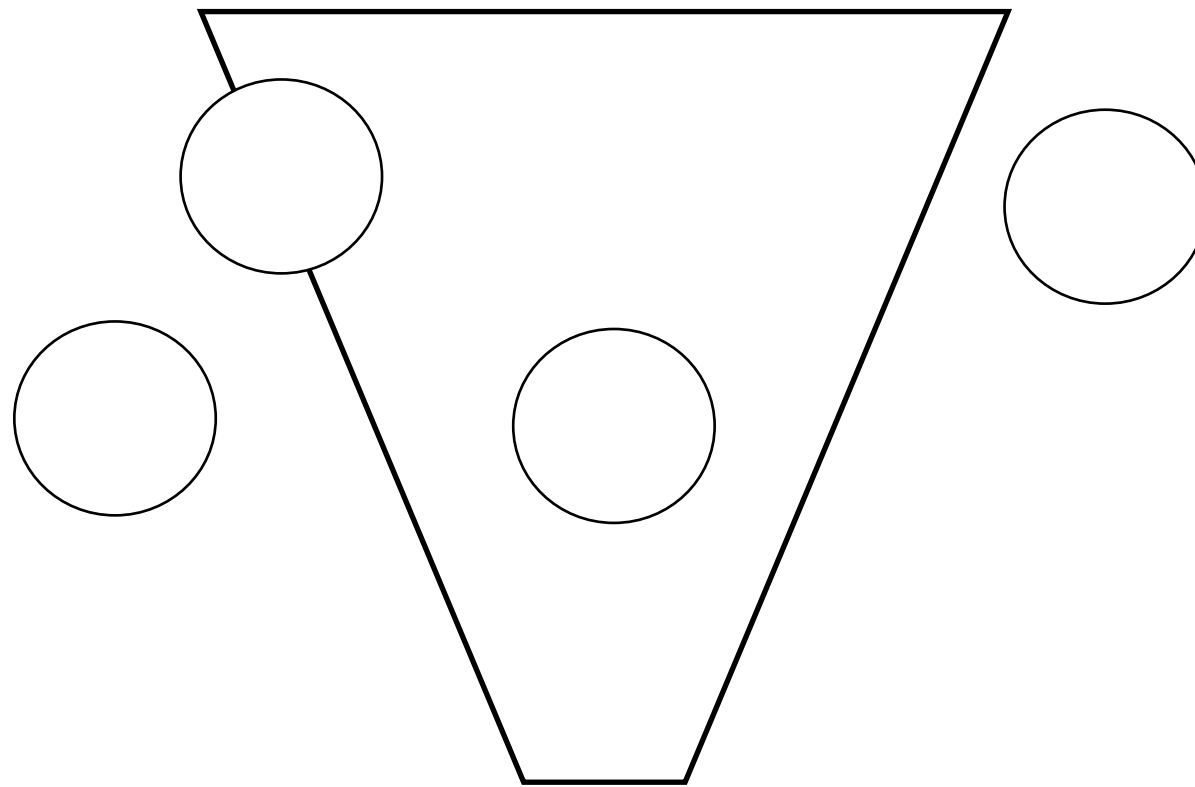
Level-of-detail

Using a repeating patch to fake it

Generating patches on the fly



Step 1. Frustum culling (View volume culling) **What polygons are inside the frustum?**



Principle: Make a subdivision of the scene, so tests can be done on groups, e.g. separate objects or limited parts of the scene.

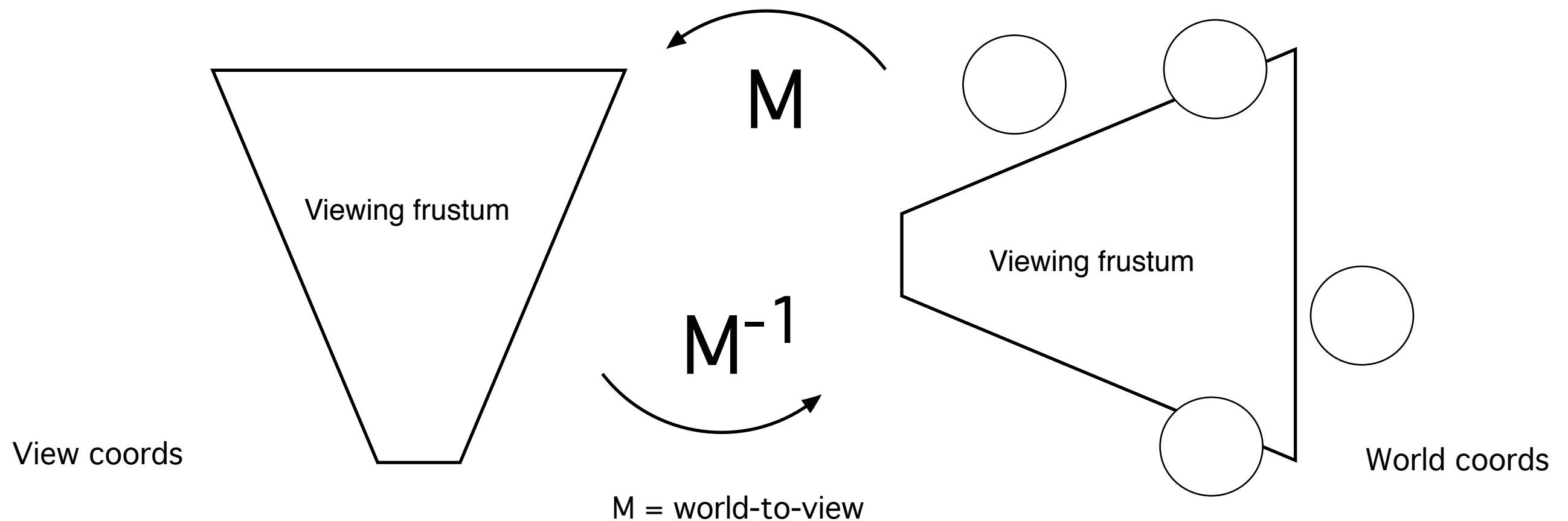


Frustum culling

Create plane equations for each frustum side

Transform from view to world coordinates

Test against bounding spheres of objects





The frustum culling test against a sphere

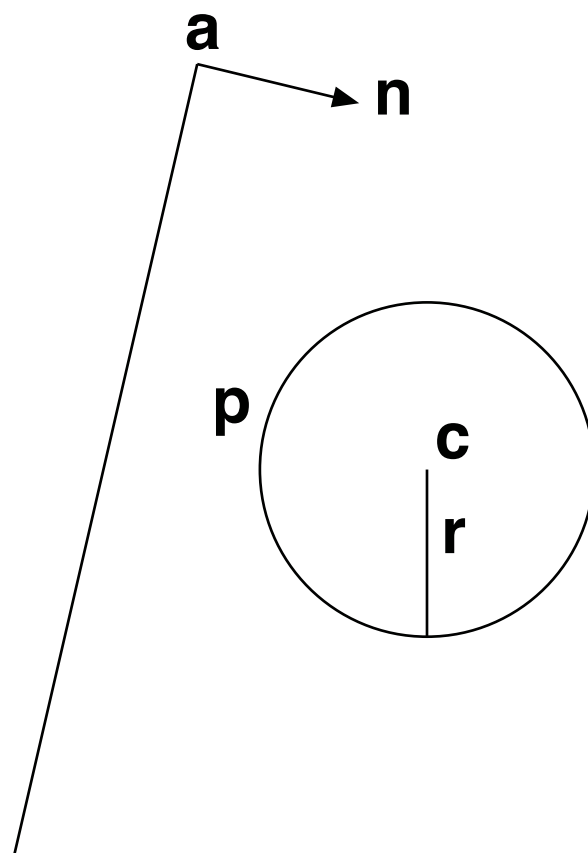
You need

center of sphere \mathbf{c}

radius of sphere r

normalized normal of plane \mathbf{n}

A point in the plane \mathbf{a}



Point \mathbf{p} nearest plane = $\mathbf{c} - \mathbf{n} \cdot r$

Project \mathbf{p} on \mathbf{n} by $\mathbf{p} \cdot \mathbf{n}$

Project \mathbf{a} on \mathbf{n} by $\mathbf{a} \cdot \mathbf{n}$

Check sign of $\mathbf{a} \cdot \mathbf{n} - \mathbf{p} \cdot \mathbf{n}$

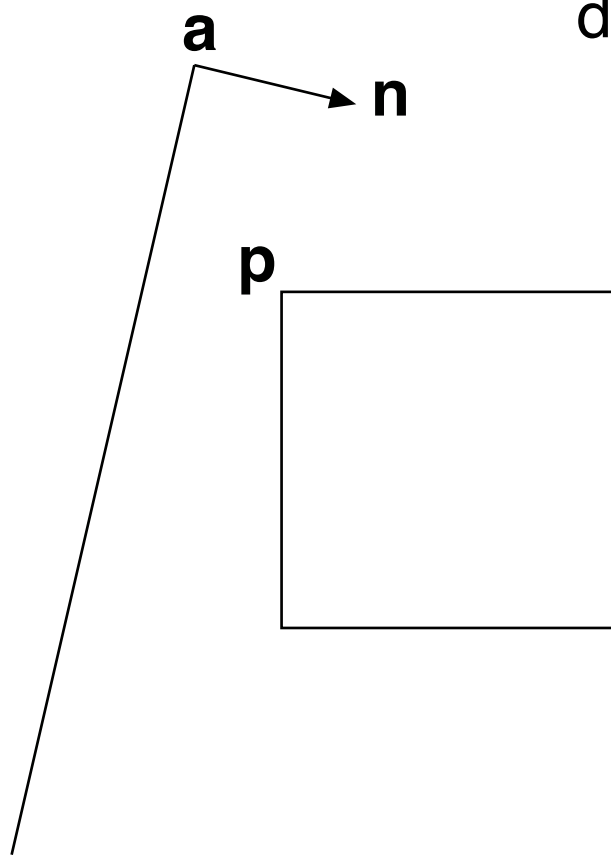


The frustum culling test against an AABB

Relevant for testing square patches!

You need

dimensions of cube, top, left, bottom, right, near, far
normalized normal of plane **n**
A point in the plane **a**



Testing all corners works, but:

Find relevant corner (farthest along $-n$) from signs of the normal!

Test that point with dot product like before:
Check sign of $\mathbf{a} \cdot \mathbf{n} - \mathbf{p} \cdot \mathbf{n}$

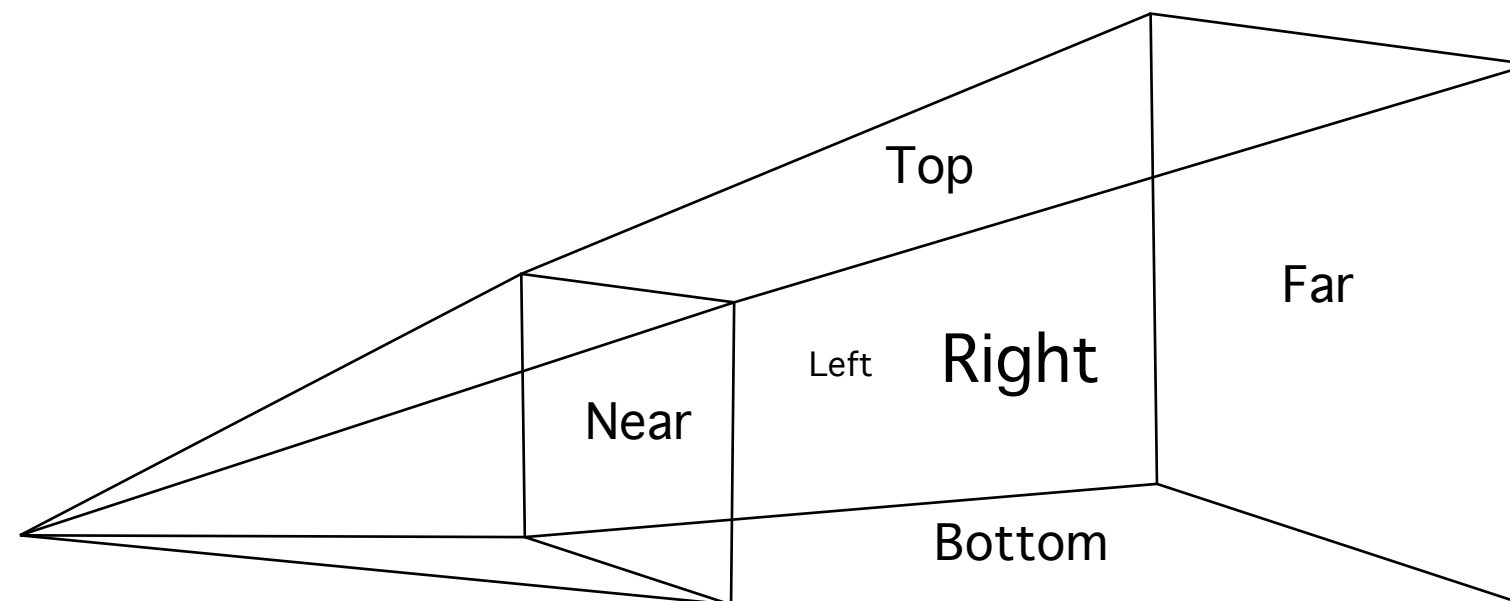


What planes to test?

You never need to test the near plane! (Why?)

You usually need right, left and far.

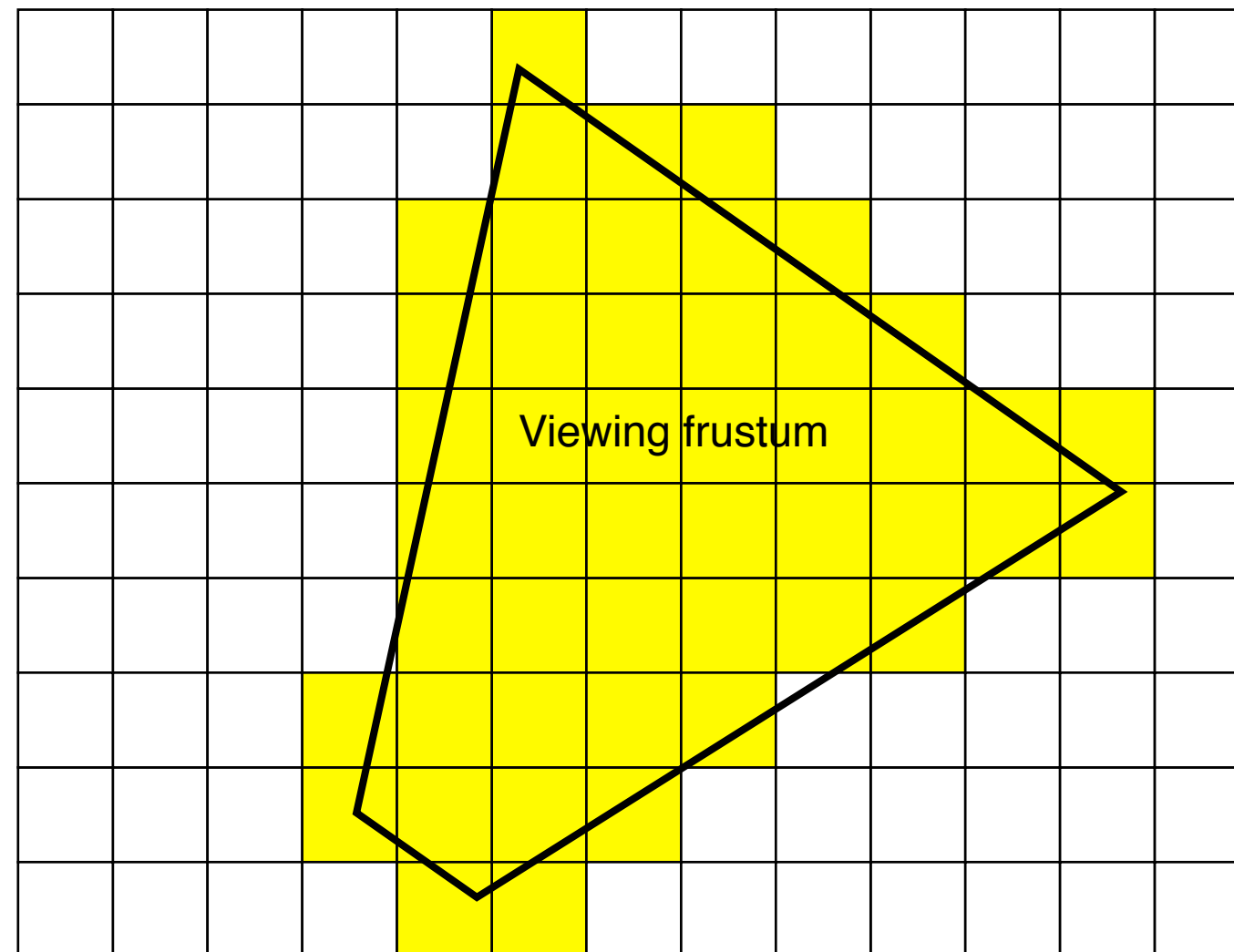
Top and bottom are often unnecessary - especially when discussing terrains!





Uniform space subdivision

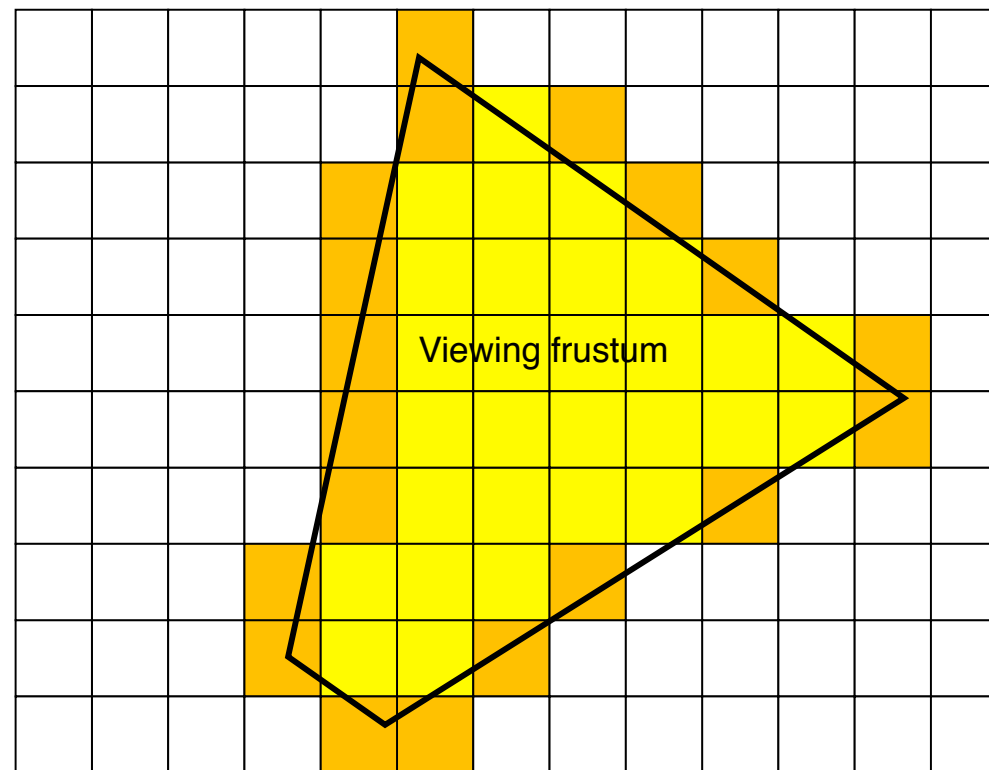
Simple common case: Terrain defined as multiple patches of same size; forms a regular grid





Map the frustum edges to the grid coordinates

Draw all patches between edges



Cheap quick hack version:

Find the bounding box of the frustum. Gives a simple 2D rectangle with grid spaces to draw. Up to 50% unnecessary polygons.



Level of detail

Fewer polygons

Terrains = Special case

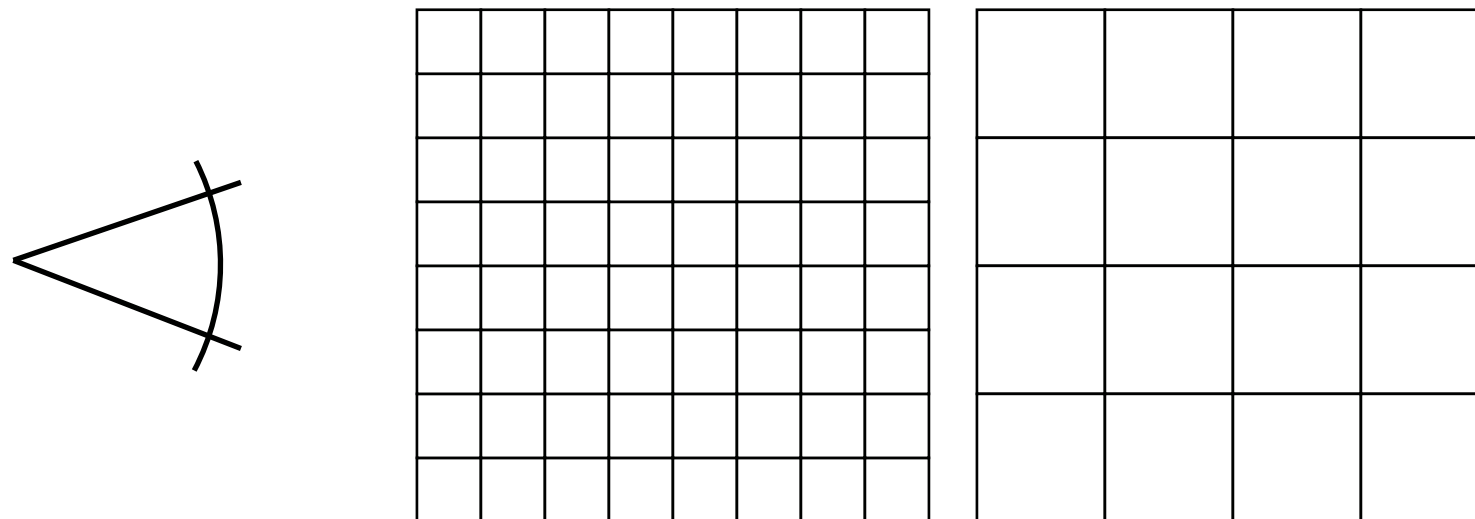
Coarser patch on a distance

"Geometrical mip-mapping"



Coarser detail on distance

Decide a level to keep constant resolution



Problem: Gaps in edges!



Avoiding gaps

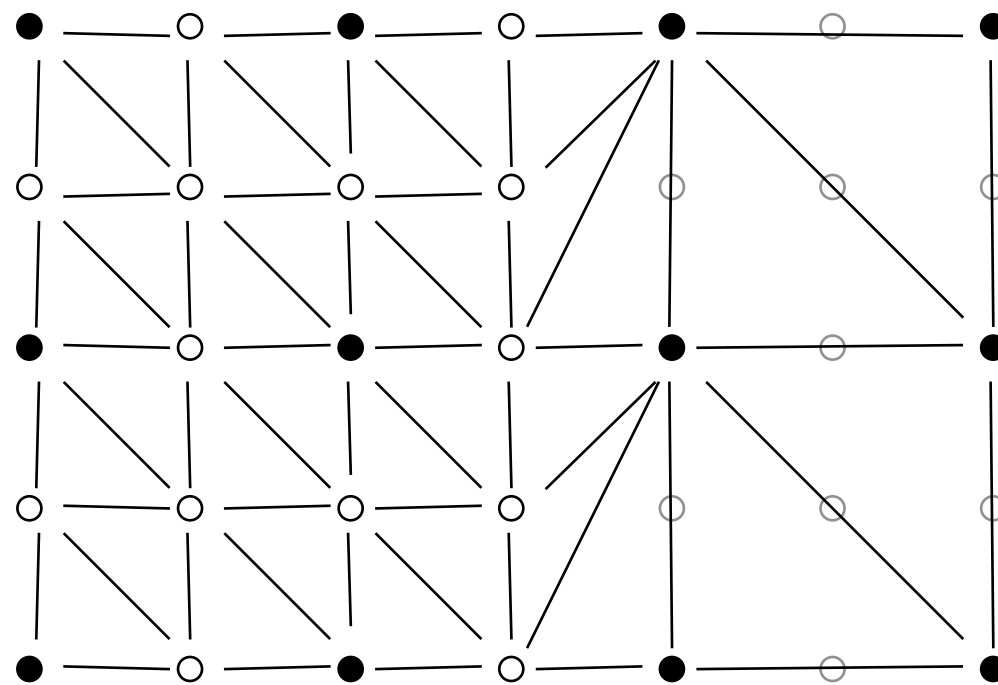
Two methods:

- Skirts
- Adjusting the terrain near the edge



Patching edges between different levels

Modify the geometry at the edge to fit the neighbor



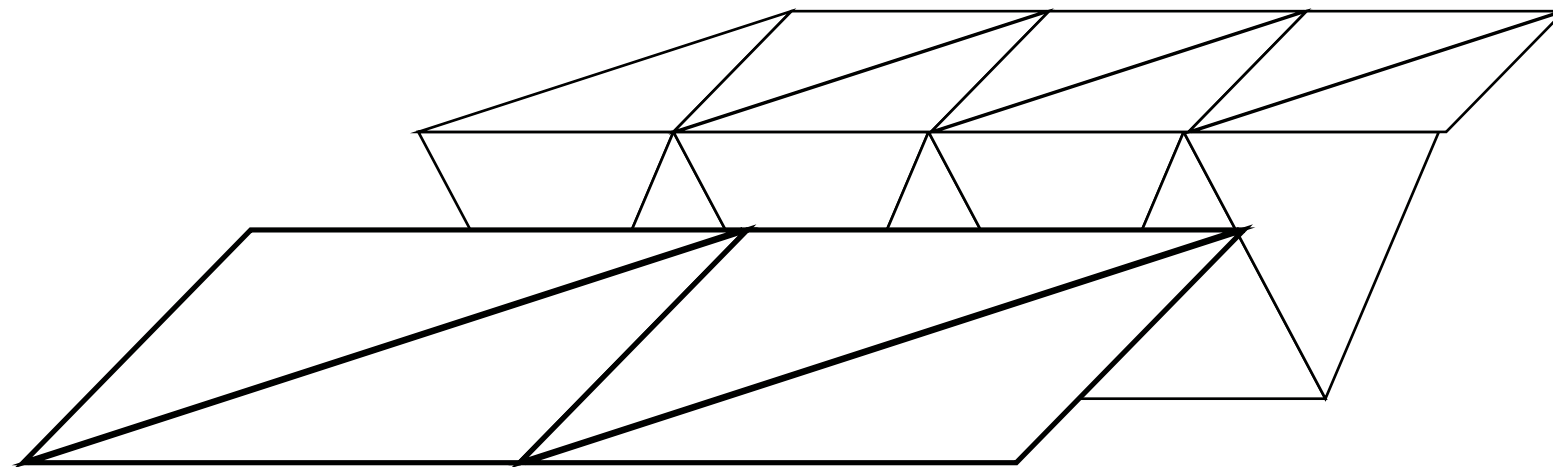
Decisions best taken at edges between patches!



Patching edges between different levels

Second approach: "Skirts". Insert extra polygons at the edge in a way that will fill the gap.

More visible than adapting resolution, but lets you work with separate patches at different resolutions





Limitless terrain

Open world-games need to expand arbitrarily far.

Two approaches:

- Generate new patches
 - Repeat



Information Coding / Computer Graphics, ISY, LiTH

Generate new patches on the fly

Player moved out of the existing area

Generate new patches

Disposing left ones may be desirable for memory management



Information Coding / Computer Graphics, ISY, LiTH

Or use a large repeating terrain

Make the world so big that the player is unlikely to notice

or

make the world explicitly wraparound (sphere or torus)

You still want to split the world into patches!