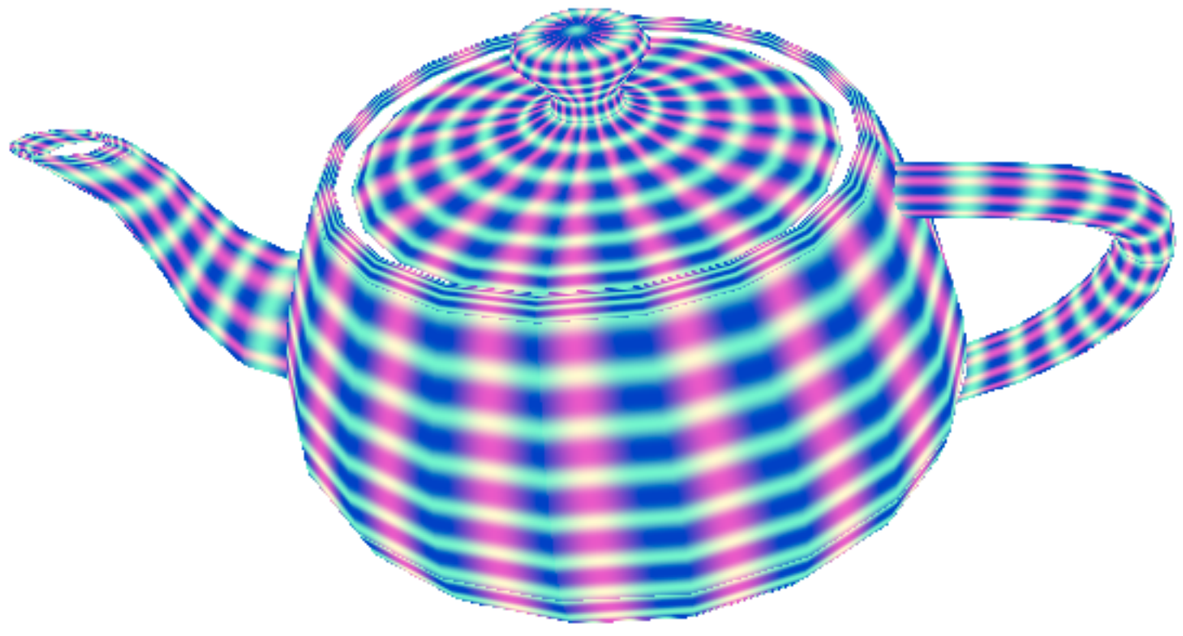
Information Coding / Computer Graphics, ISY, LiTH

TNM084
Procedural images

Ingemar Ragnemalm, ISY



Lecture 8

More on terrains

Geometry shaders

Tesselation shaders

Grass

Lab 3

I think most are done with the tree generation.

Using GLUGG seems to work just fine.

Unexpected problem: How to deal with recursion.

Extensions on Lab 3

Extended terrains: Bigger terrains, more features for the terrains

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

Projects

Most (all?) have submitted project proposals.

Some of you have already started working on the projects. (Good idea!)

I have not had time to reply to the recent proposals (only, I hope).

Many nice proposals. I will keep processing them after this lecture.

More on multi-patch terrains

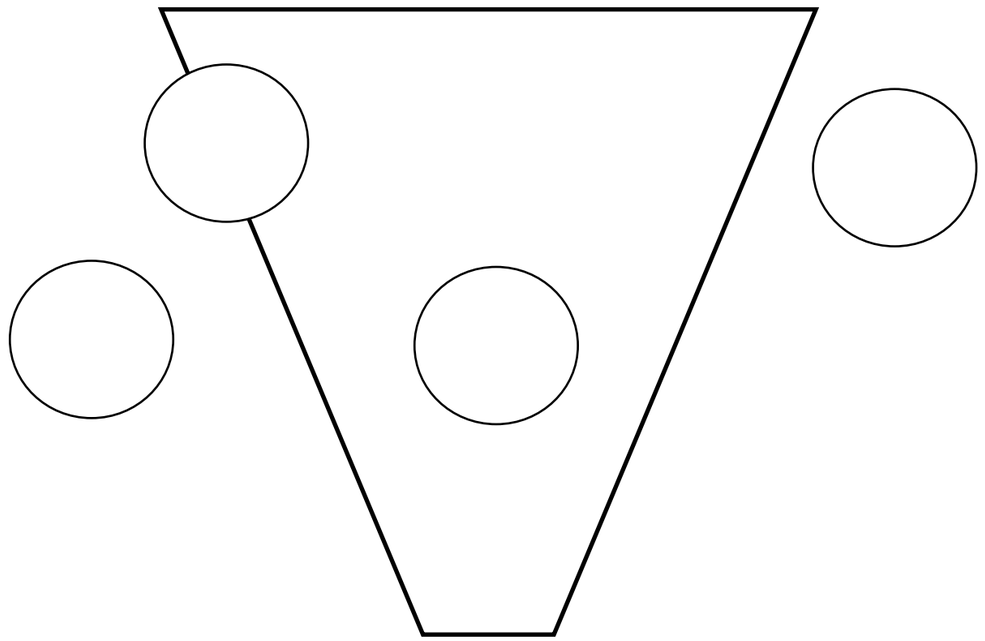
Frustum colling

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?



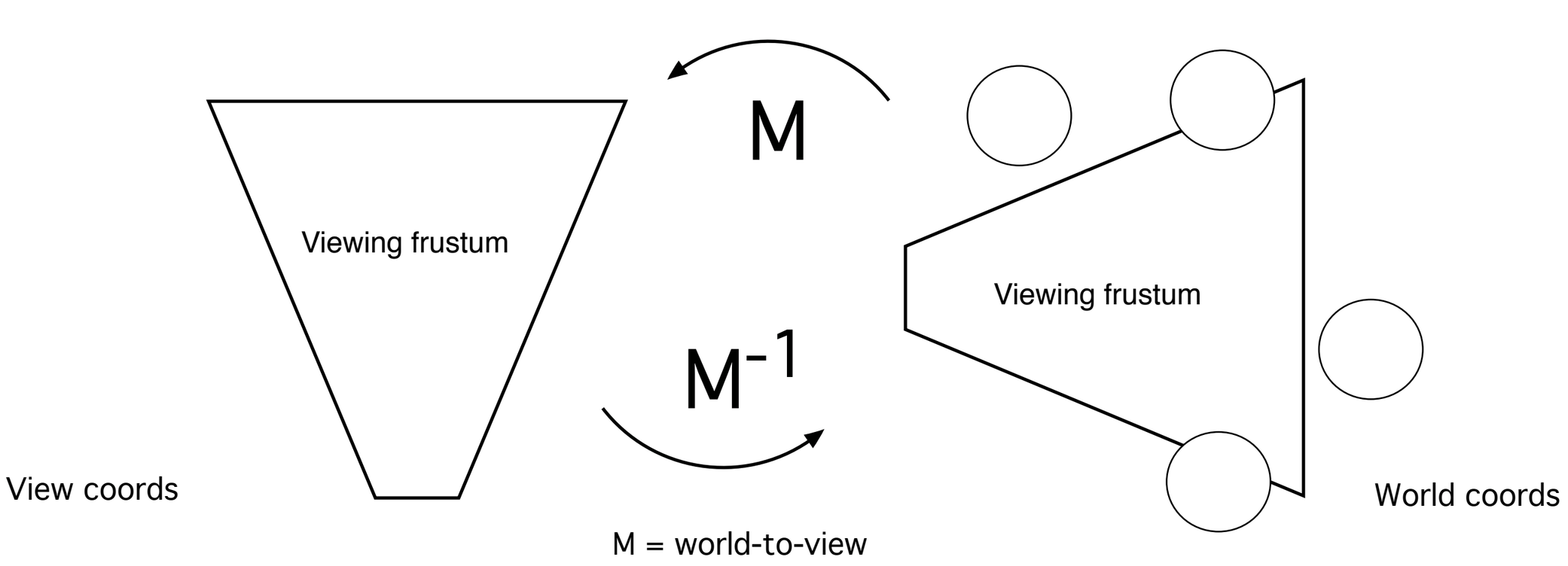
Principle: Make a subdivision of the scene, so tests can be done on groups, e.g. separate objects or limited parts of the scene.

Frustum culling

Create plane equations for each frustum side

Transform from view to world coordinates

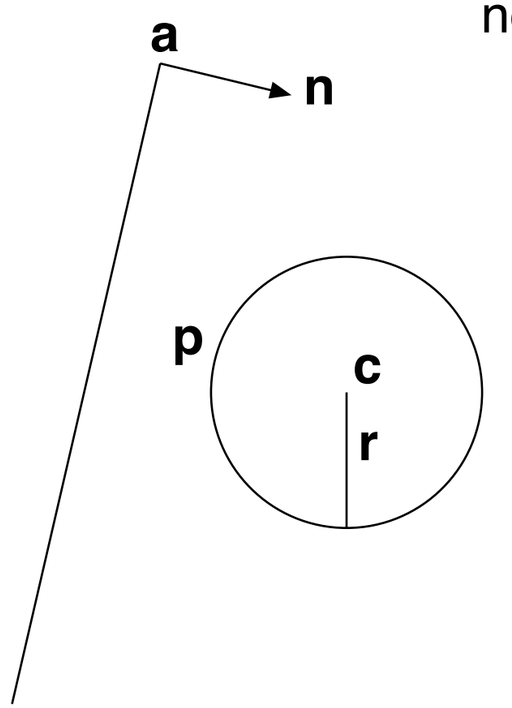
Test against bounding spheres of objects



The frustum culling test against a sphere

You need

center of sphere **c**
radius of sphere r
nomralized normal of plane **n**
A point in the plane **a**



Point p nearest plane = **c** - **n**·r

Project **p** on n by **p**•**n**

Project **a** on **n** by **a**•**n**

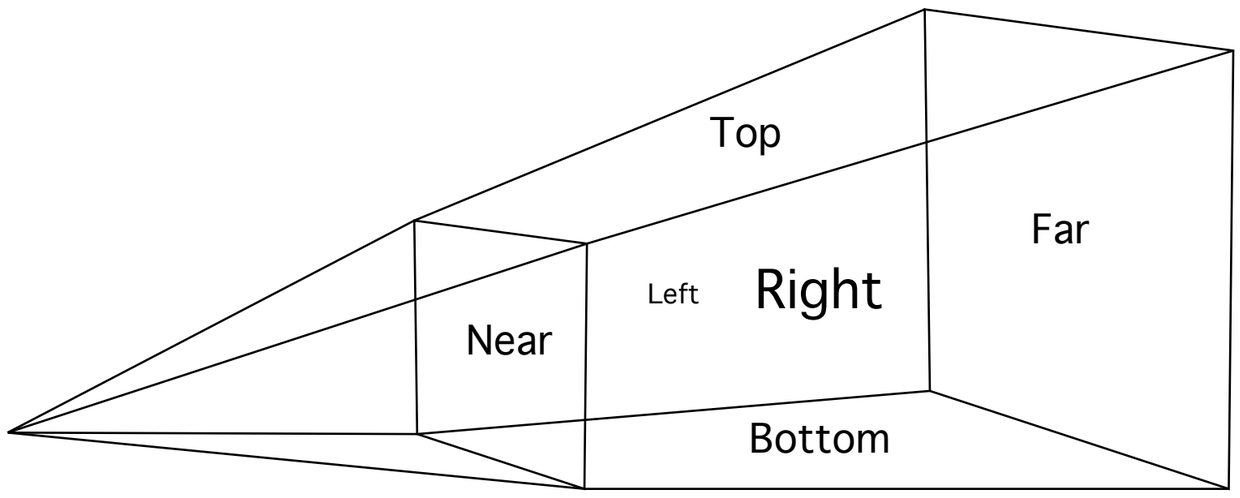
Check sign of **a**•**n** - **p**•**n**

What planes to test?

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

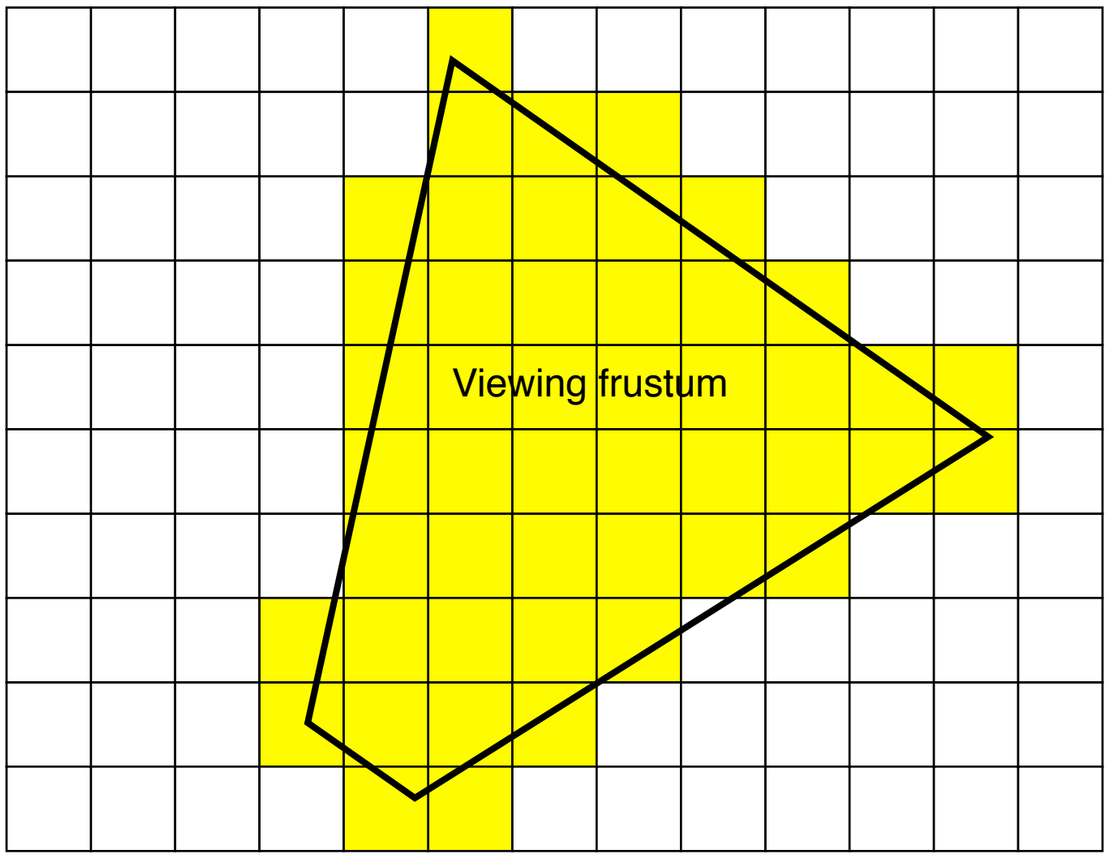
You usually need right, left and far.

Top and bottom are often unnecessary - not least when discussing terrains!



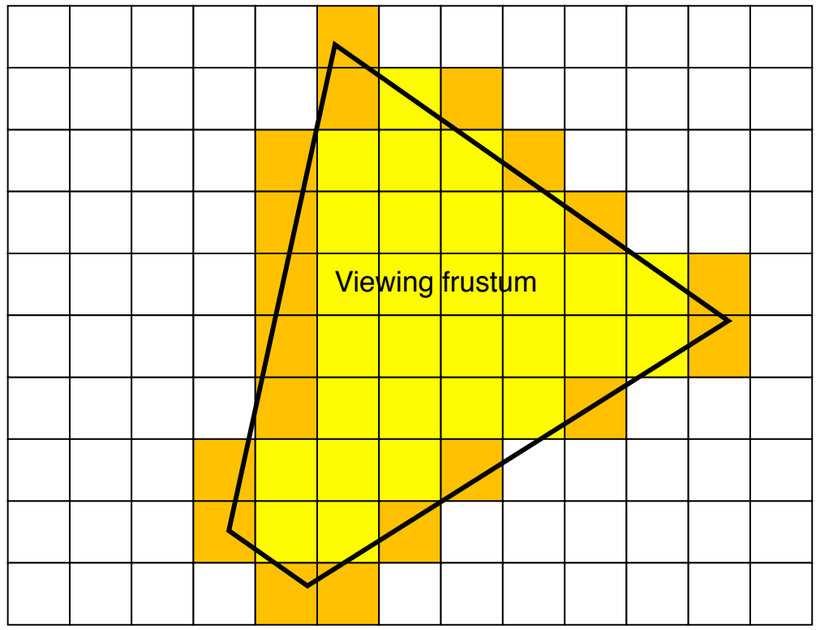
Uniform space subdivision

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

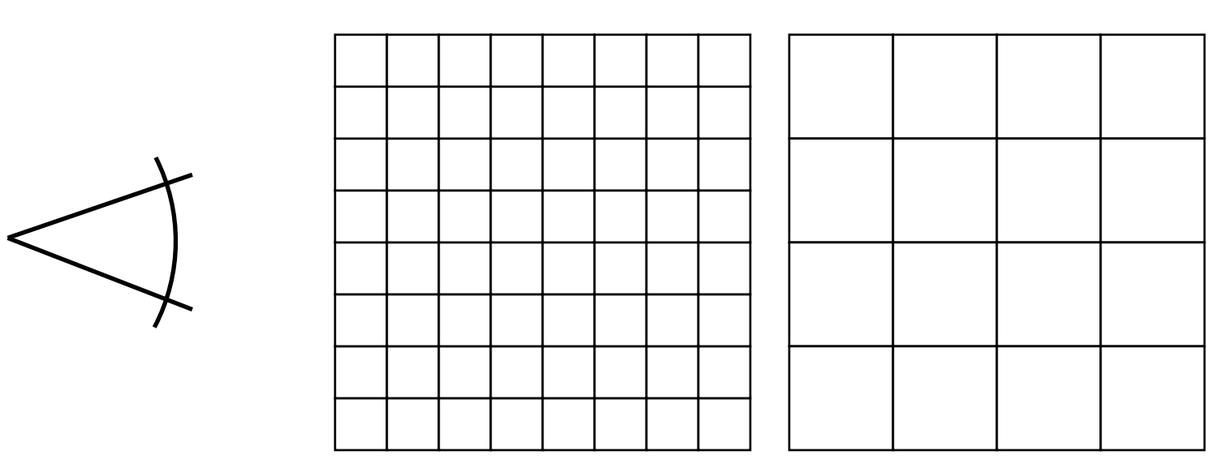
Terrains = Special case

Coarser patch on a distance

"Geometrical mip-mapping"

Coarser detail on distance

Decide a level to keep constant resolution



Problem: Gaps in edges!

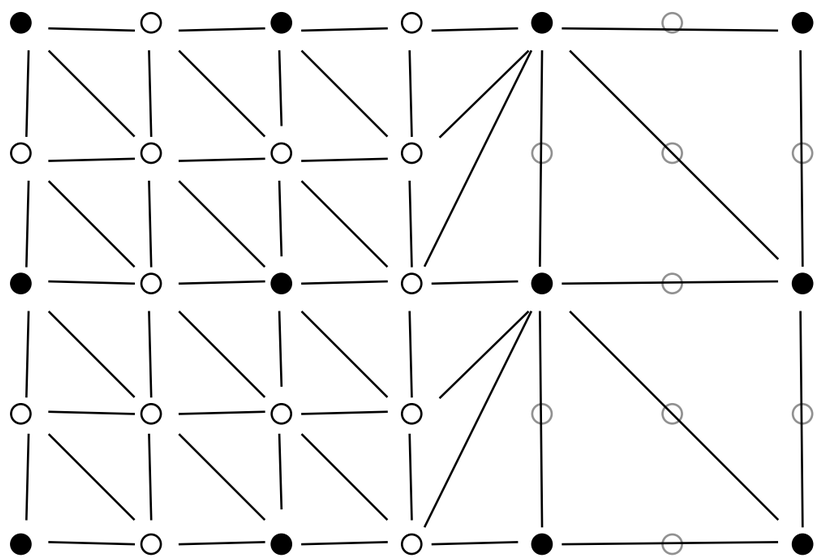
Avoiding gaps

Two methods:

- Skirts
- Adjusting the terrain near the edge

Patching edges between different levels

Modify the geometry at the edge to fit the neighbor

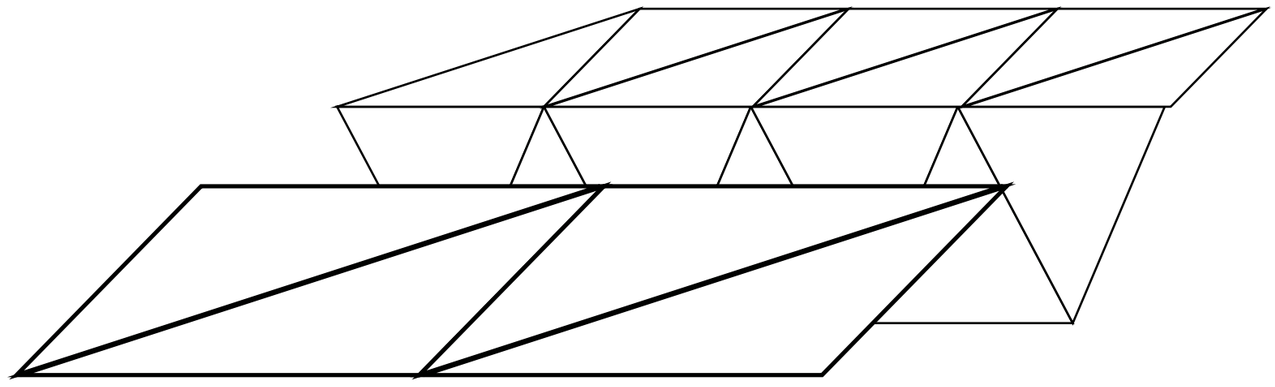


Decisions best taken at edges between patches!

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

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

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!