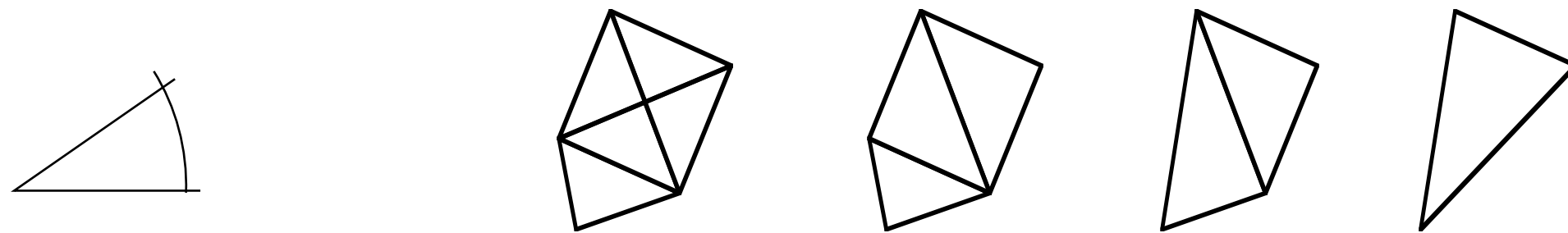




# Level-of-detail LOD

**Multiresolution representations  
Reducing the polygon count for distant  
objects**





## Example: Stanford bunny

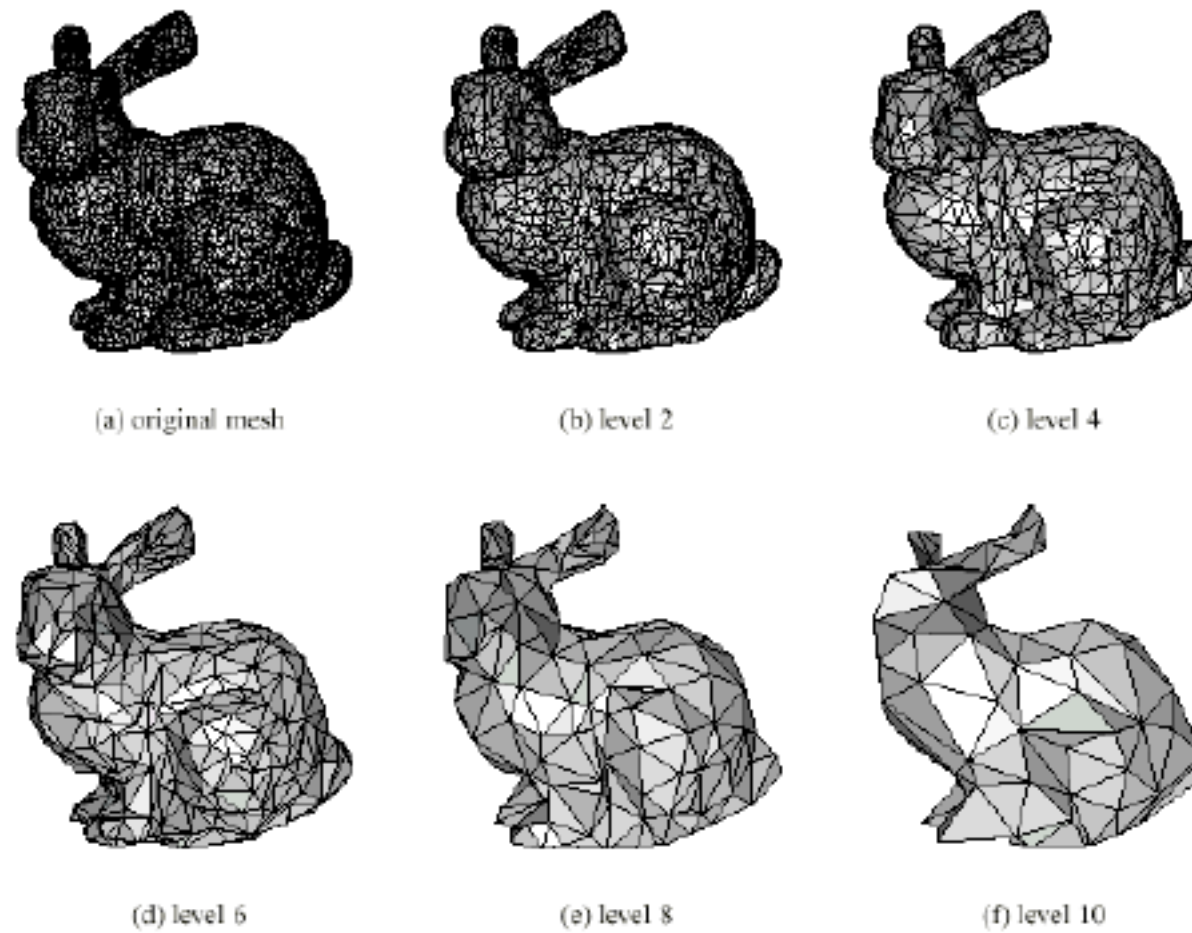


Figure 8. Stanford Bunny. Simplified meshes with 10000, 4577, 2106, 988, 463, 245 triangles



## **Level-of-detail LOD for models**

### **1. Pre-generate in different detail**

**Risk for noticable “popping” when switching model**

### **2. Progressive mesh**

**Continuous deformation, no “popping”**

**Non-trivial to select the polygons to reduce**

**At very low resolutions, we may switch to impostors  
(billboards)**



## **Reduction methods**

**Collapse edges**

**Insert new vertex, remove neighbors, re-triangulate**

**Remove vertices**

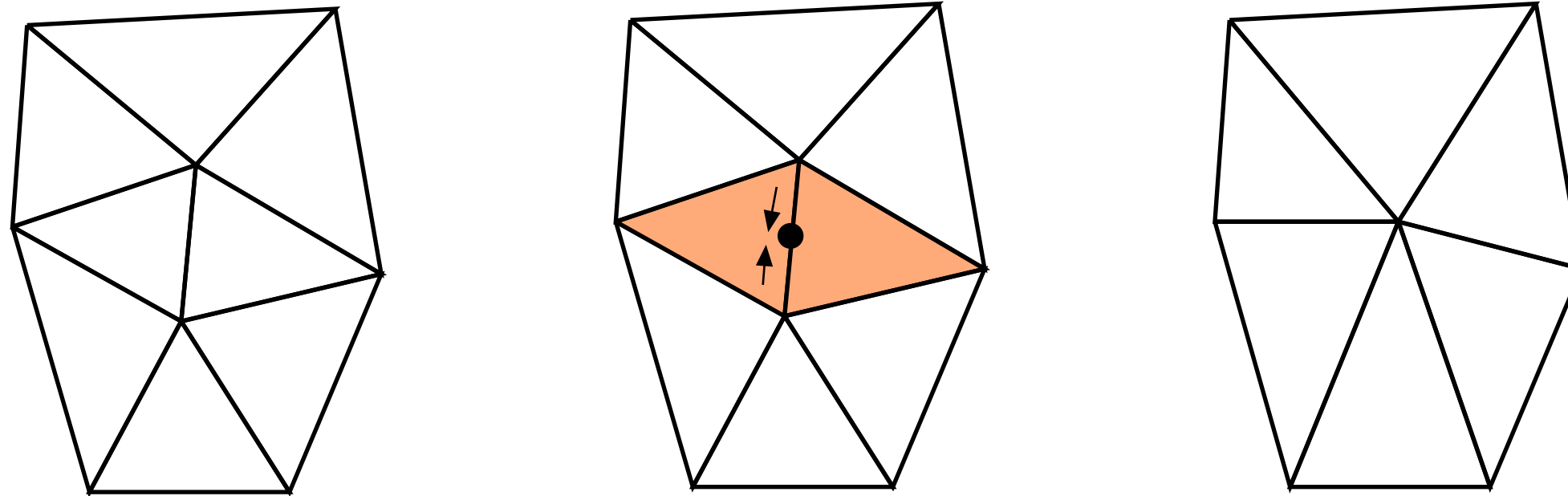
**Remove vertices, re-triangulate  
(similar)**

**Find neighbor polygons in the same plane (or near), and  
merge them.**

**Note that only some can be progressive!**



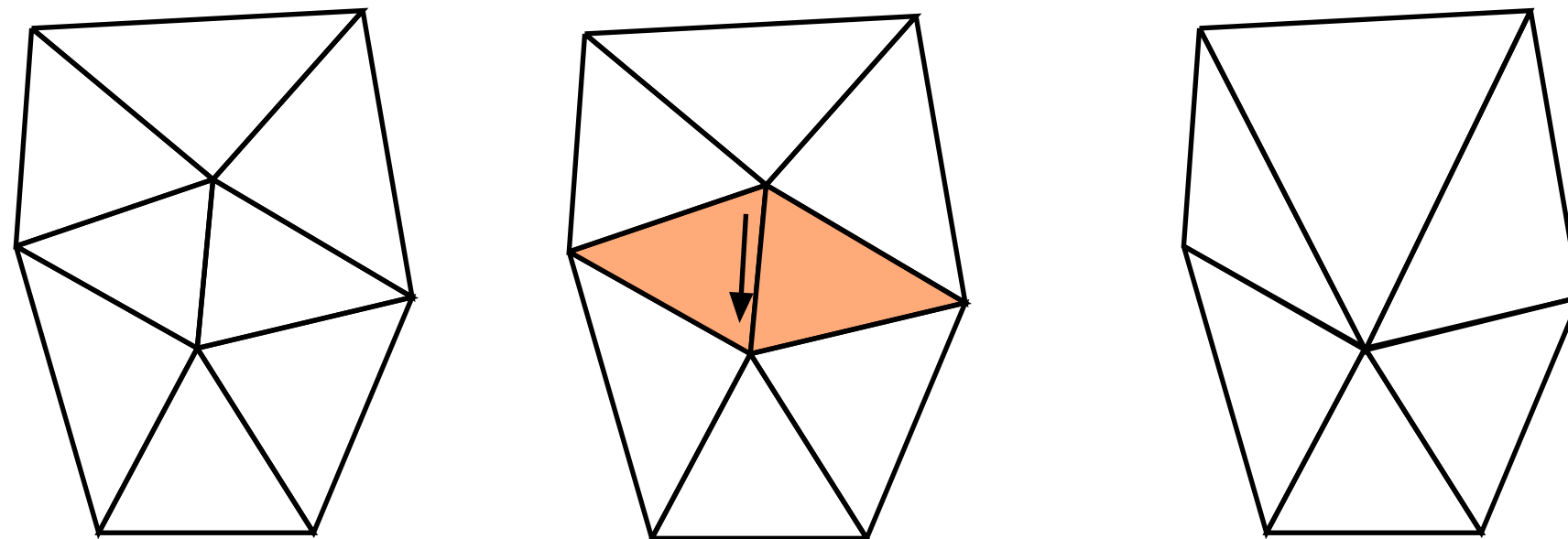
## Edge collapsing



**Simple - but vertex attributes (normals, texture coords) must be recalculated**



## Vertex removal



**Simple - no recalculation of vertex attributes**



## **Problem in LOD: volume reduction**

**The mesh is a sampling of a continuous surface**

**Careless removal or interpolation will cause errors**





# **Level-of-detail LOD for terrains**

## **Geometrical mip-mapping**

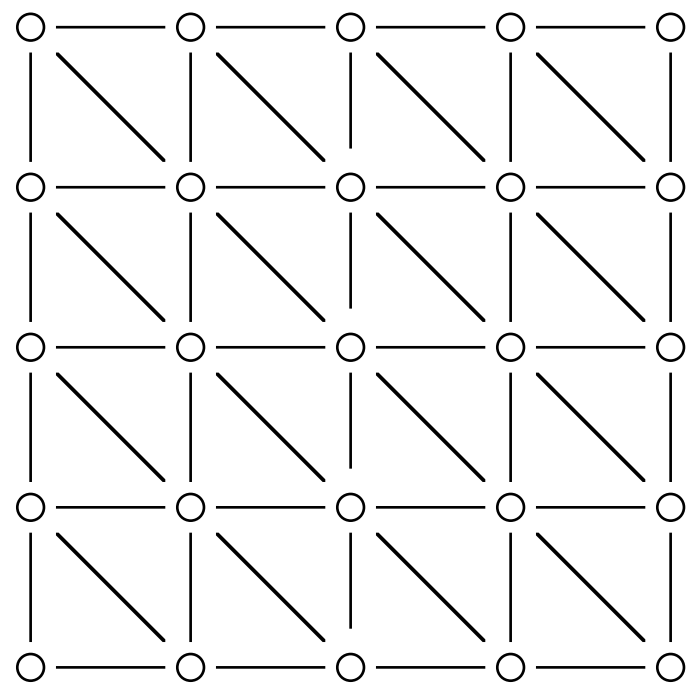
**Produces a polygon terrain with approximately constant polygon size in screen coordinates**

**Reduces the polygon count effectively to what is actually needed.**

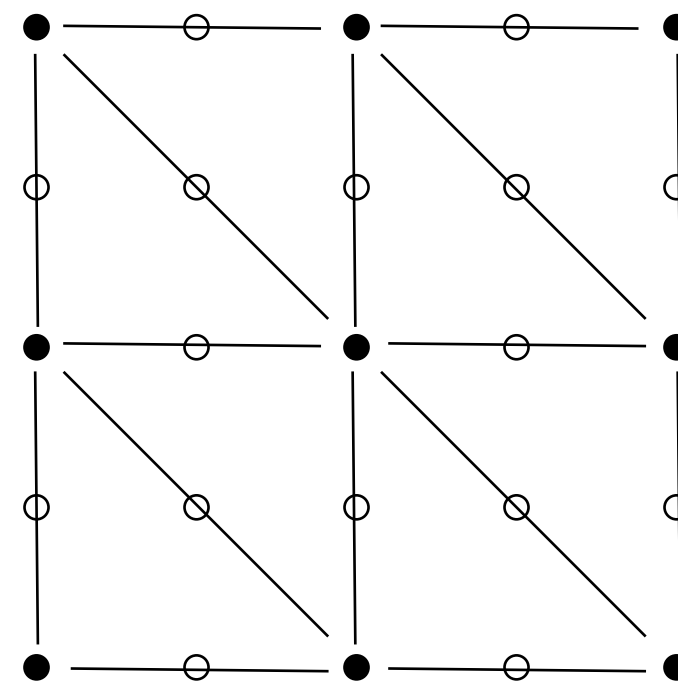




# Geometrical mip-mapping



Level 0 - full resolution

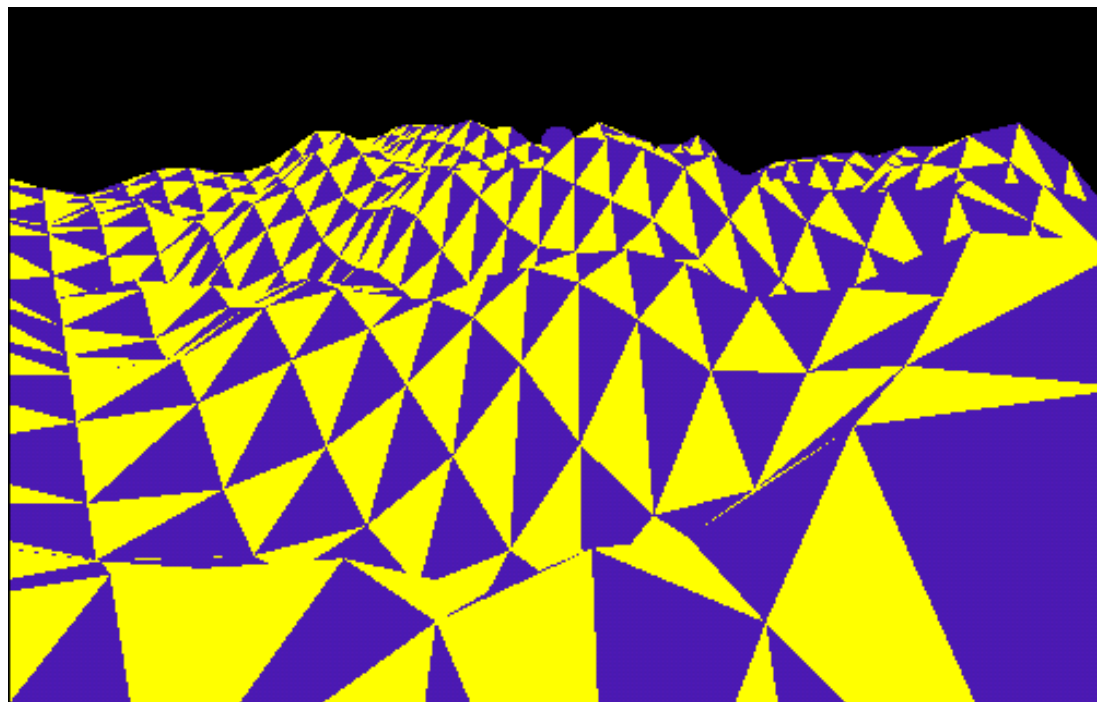


Level 1

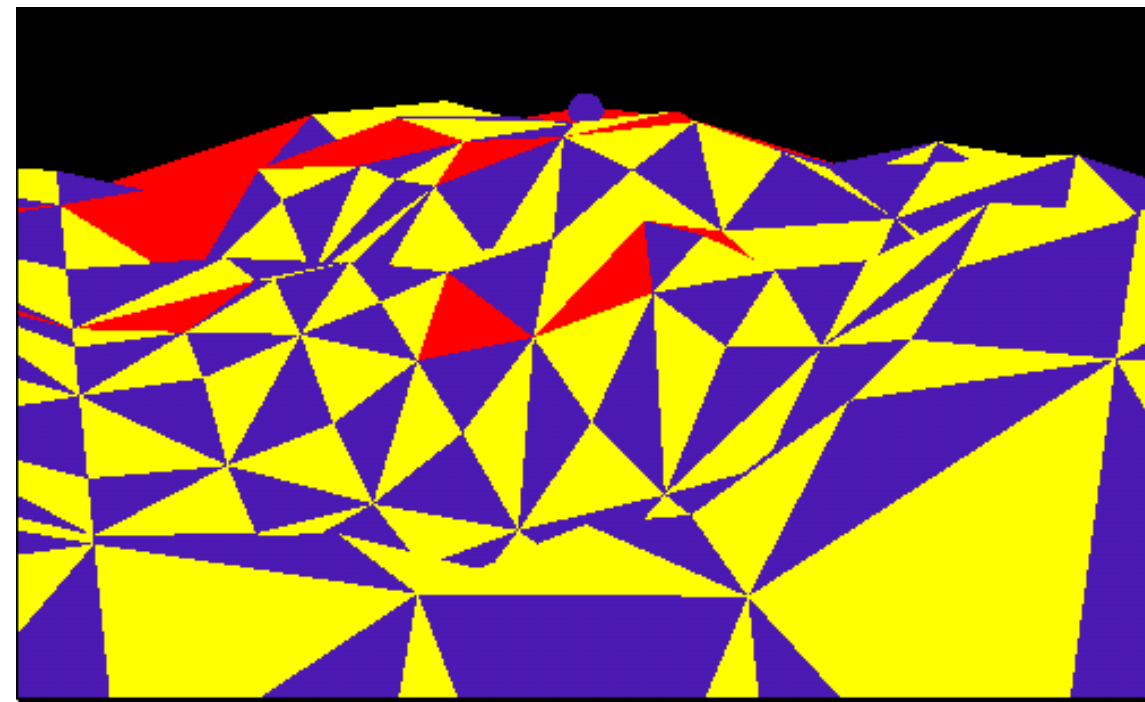


# Geometrical mip-mapping

**No geomipmapping - polygon density grows with distance**



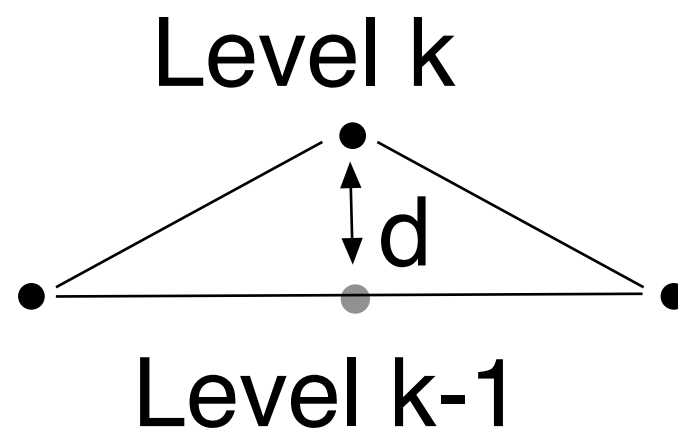
**With geomipmapping - polygon density similar on all distances**





## Decide resolution level

- **distance**
- **screen-space error measures**



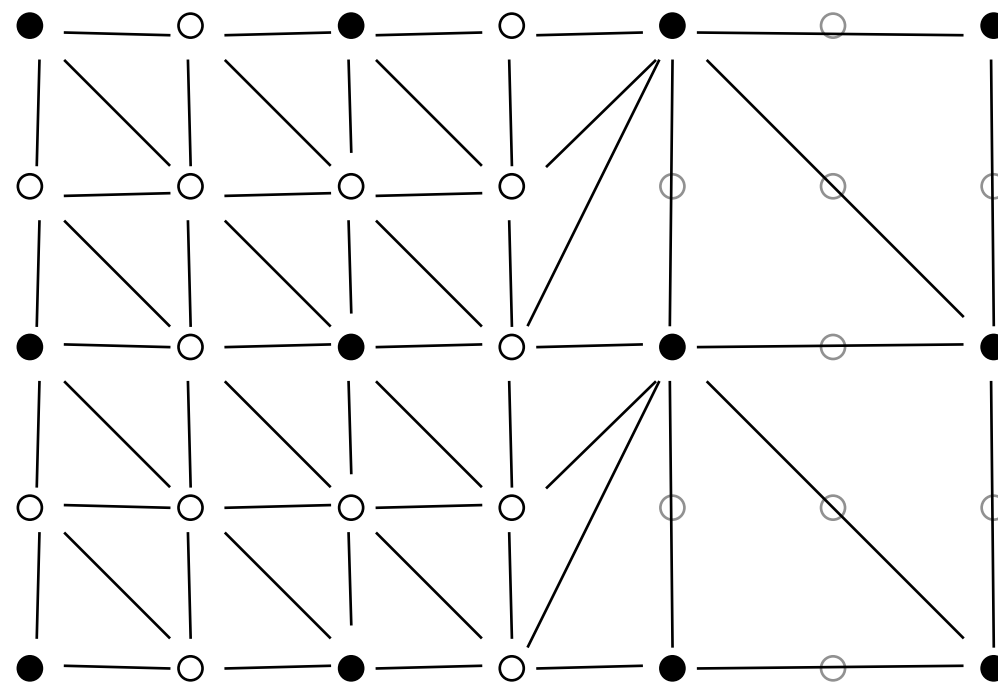


## **Problems to solve in geomipmapping**

- **Popping**
- **Gaps**
- **Sliding textures bug**



## Patching edges between different levels



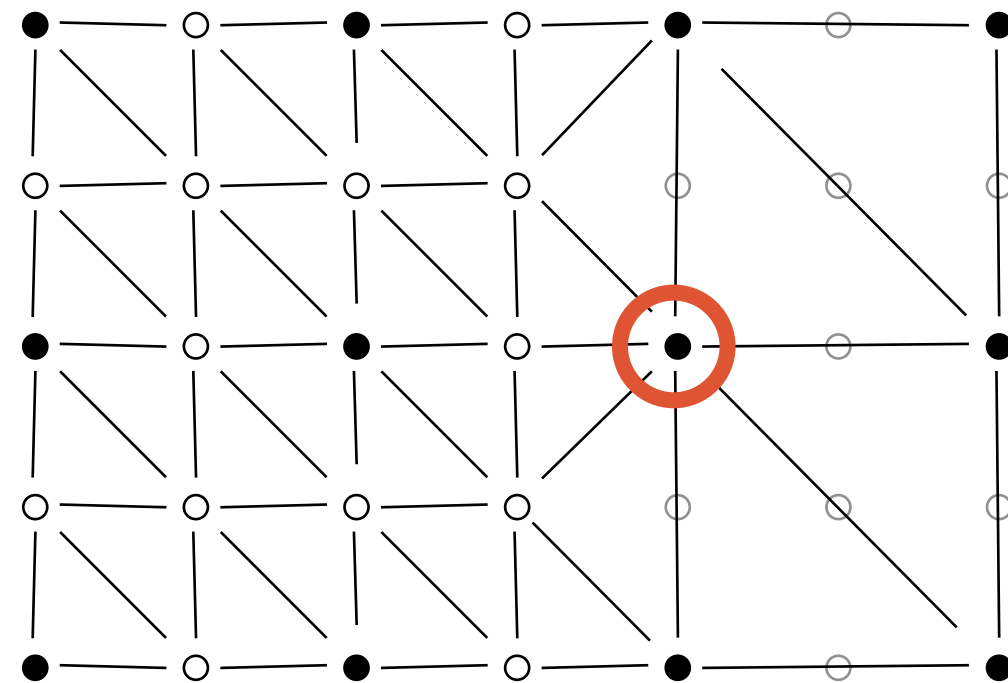
**Decisions best taken at edges between patches!**



## Why take decisions at edges?

**Both patches will take the same decision at that particular edge. What happens along other edges does not matter.**

**Tessellation shaders are designed on this principle.**

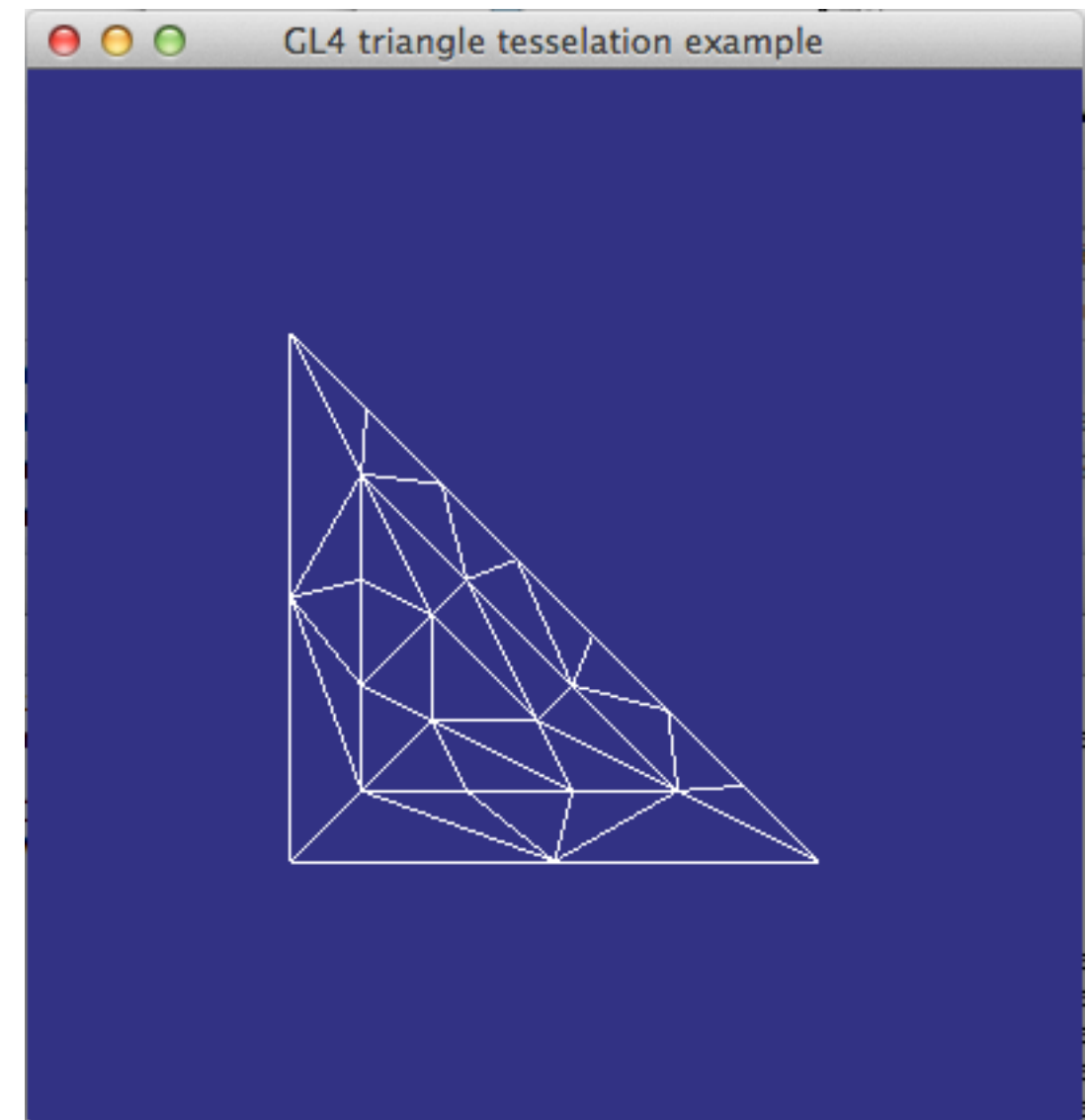




## **Tessellation shaders (advanced):**

**Produces a grid of any density.**

**Designed for decisions by edge.  
The edges may have different  
resolution.**

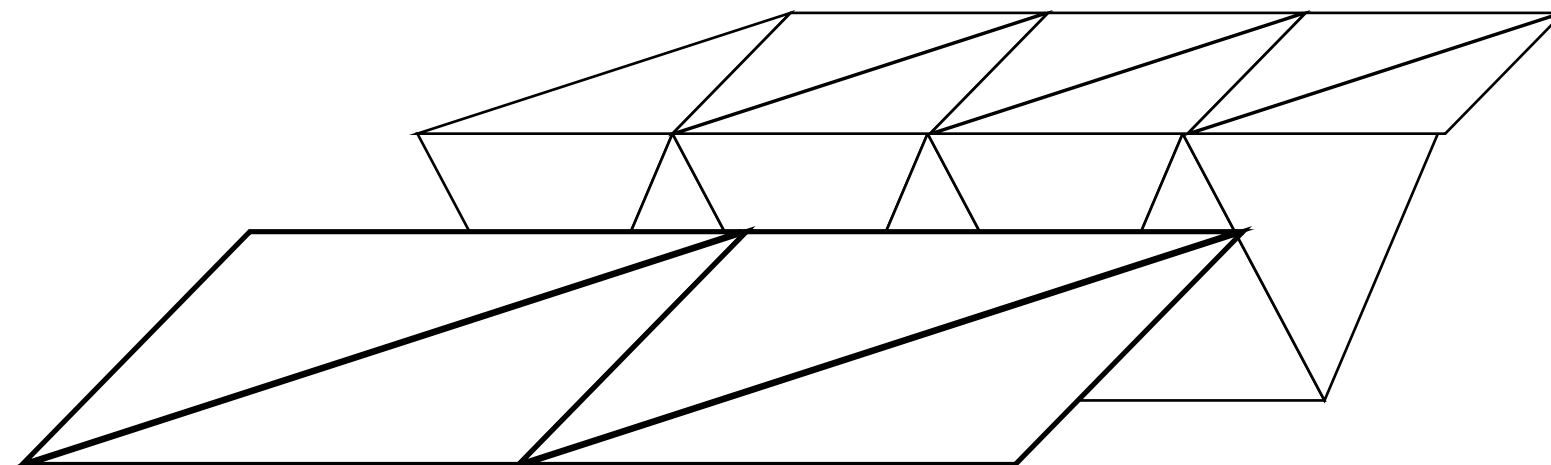




# Patching edges between different levels

**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**



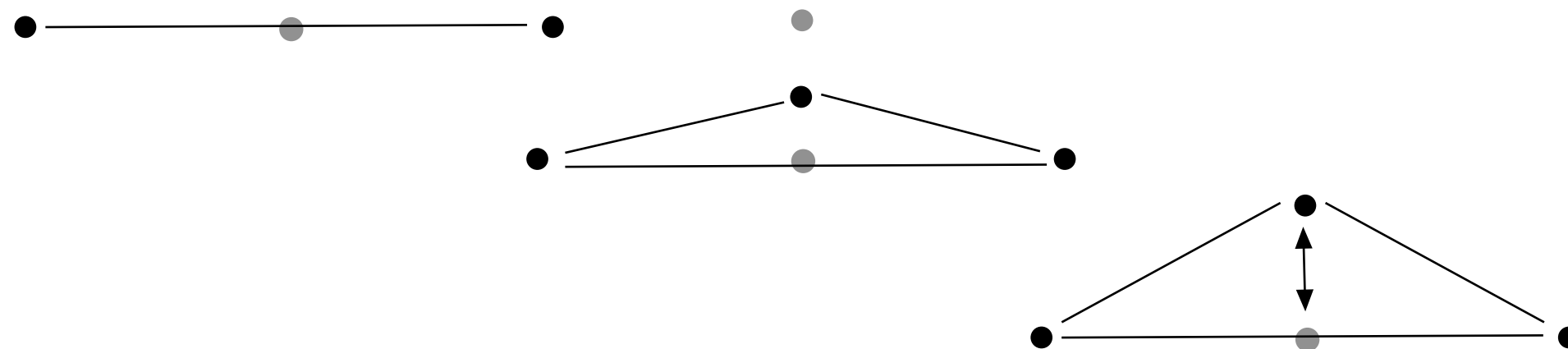




## Avoid popping

**Popping is solved by “morphing” between levels.**

**Interpolate vertices that are close to removal with the average between neighbors**



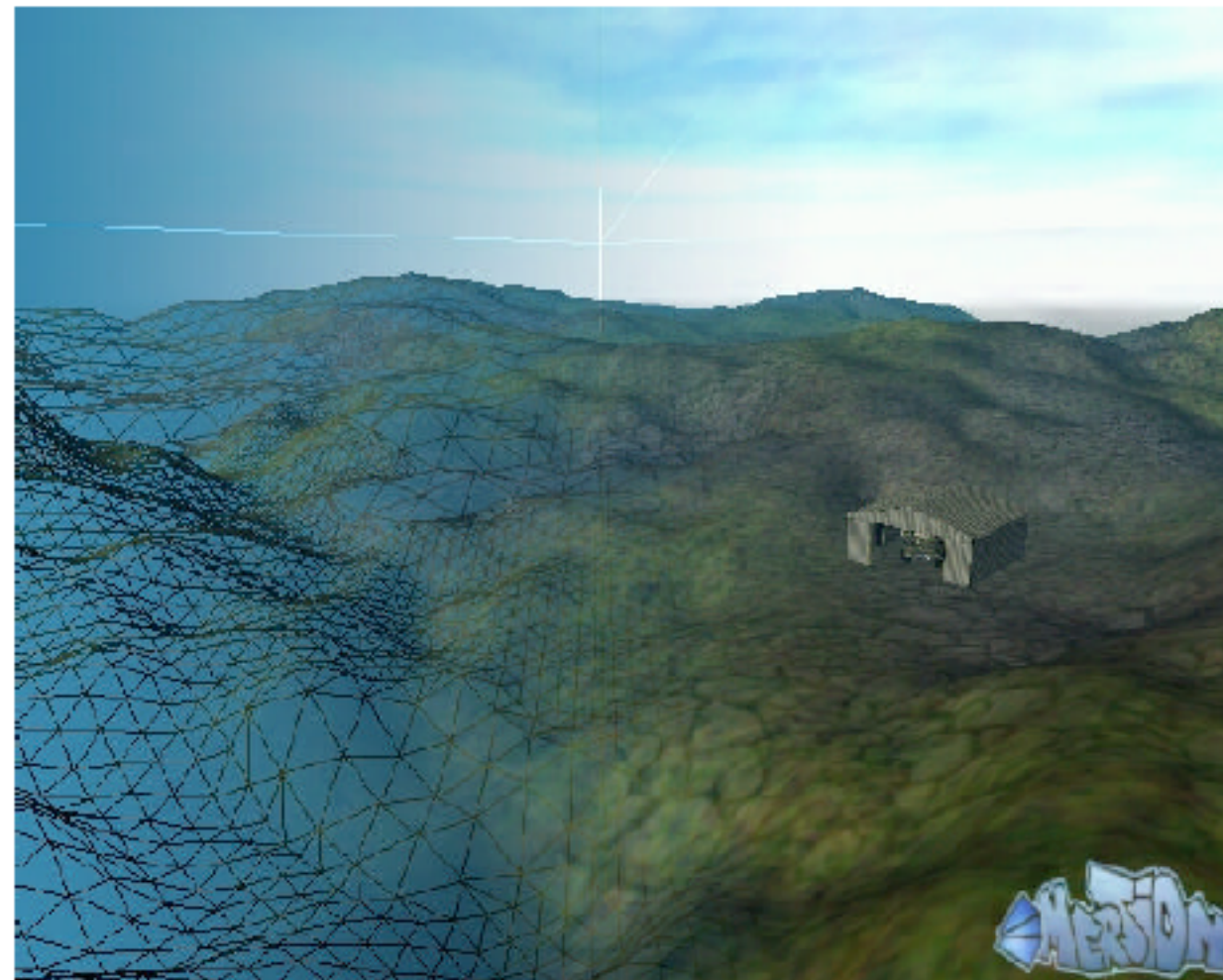


# Geomipmapping

- **should produce polygons with roughly the same size on all distances**
- **will greatly reduce polygon count on very large terrains with large “far” distance**



# Geometrical mipmapping, example (from paper by de Boer)





# **Geomipmapping with hardware support**

**can be implemented in the geometry  
stage, in tessellation shaders**



## **Level of detail - conclusions**

**Saves vertex/polygon processing time**

**Can be a significant speed booster**

**Relatively hard to make "perfect" result,  
you must fade or morph between detail  
levels**