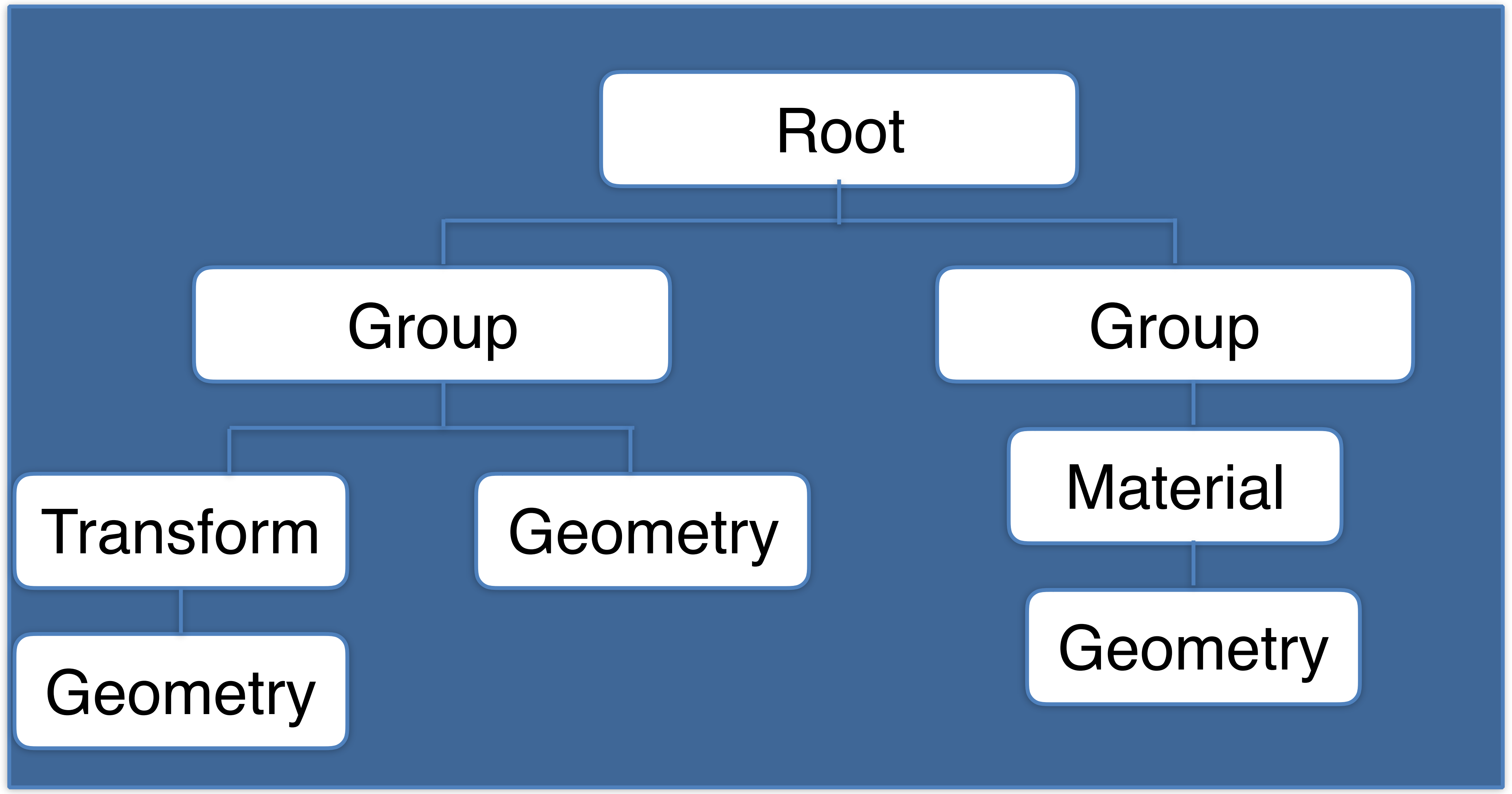
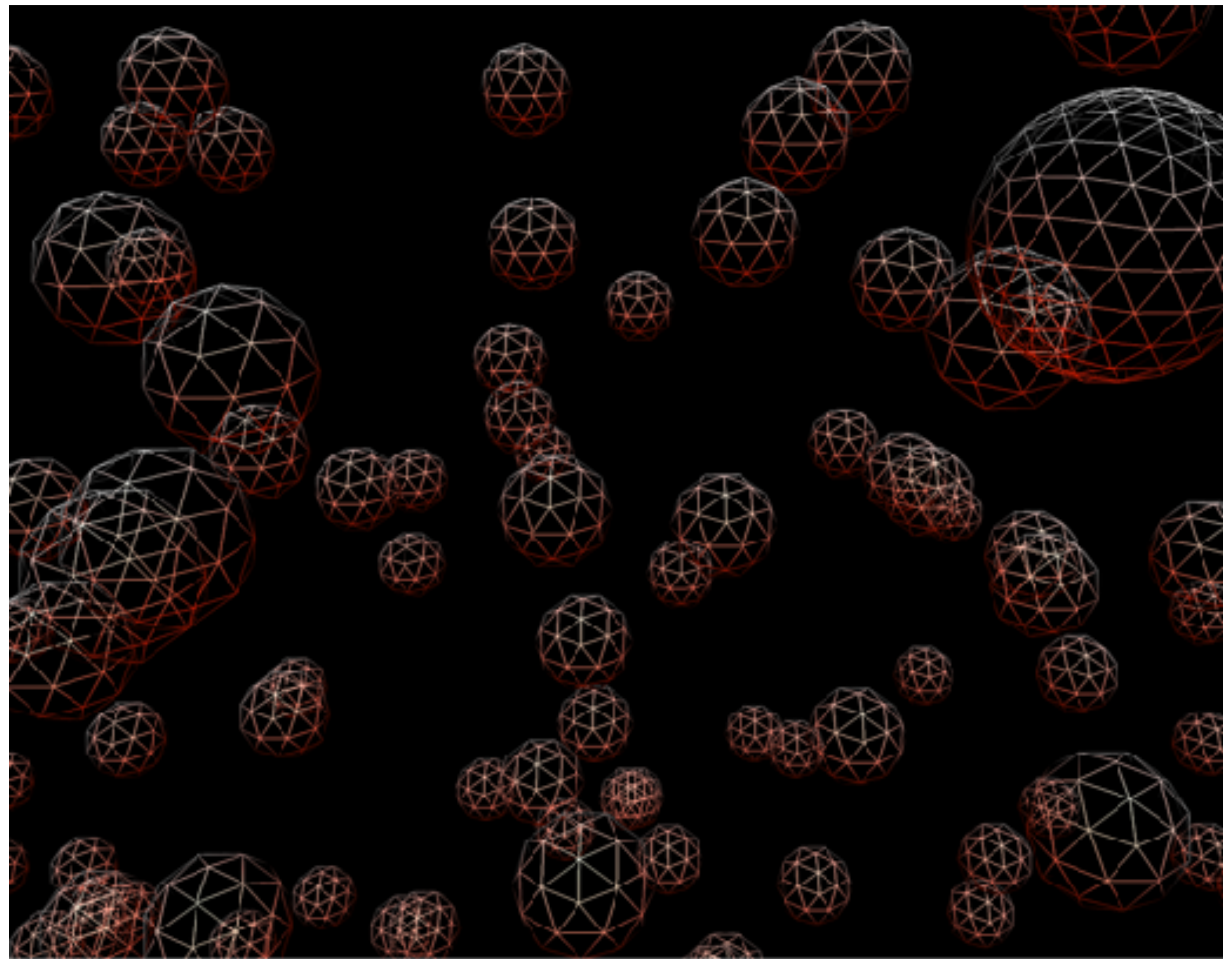
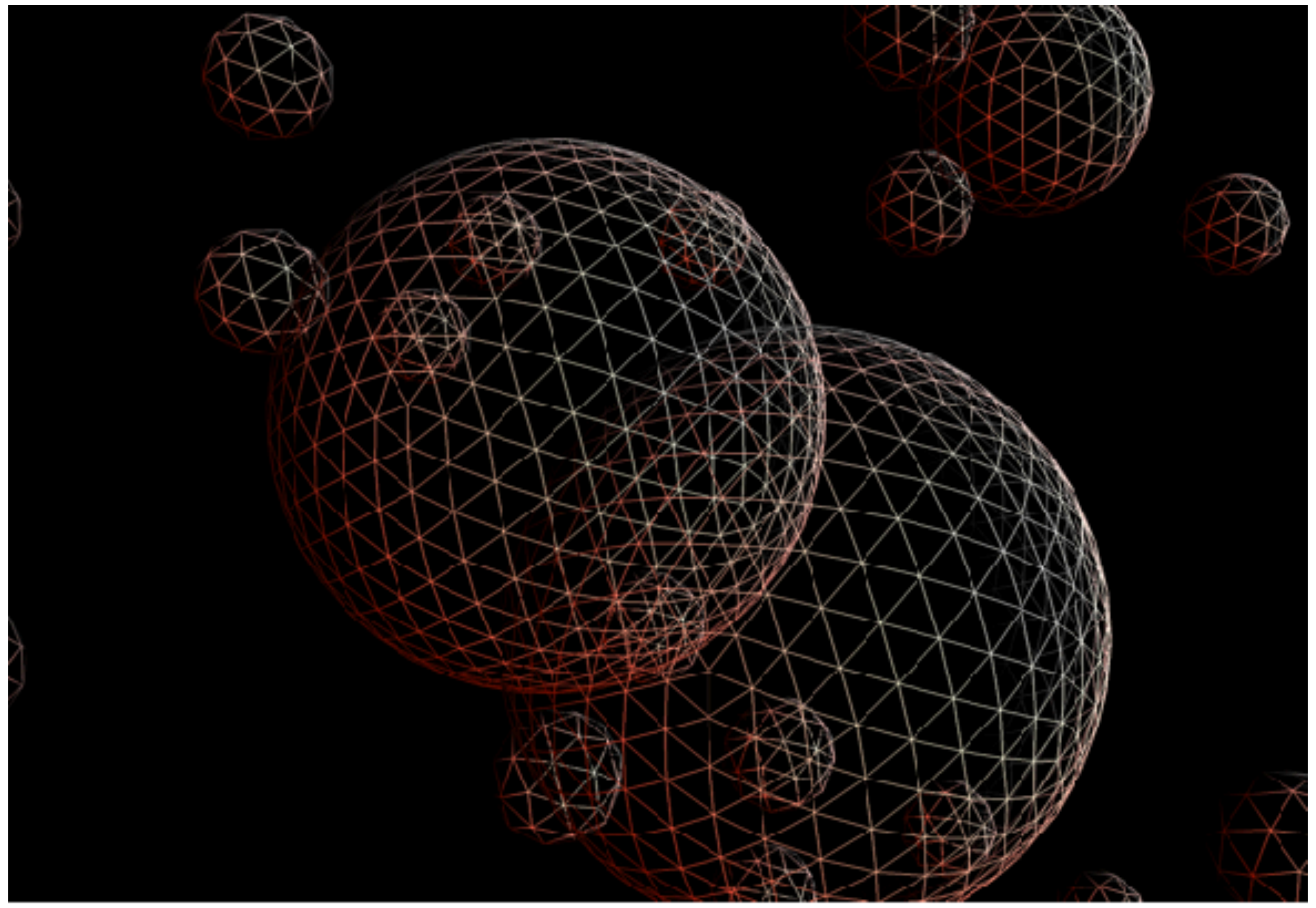
COSC342: Computer Graphics

2017





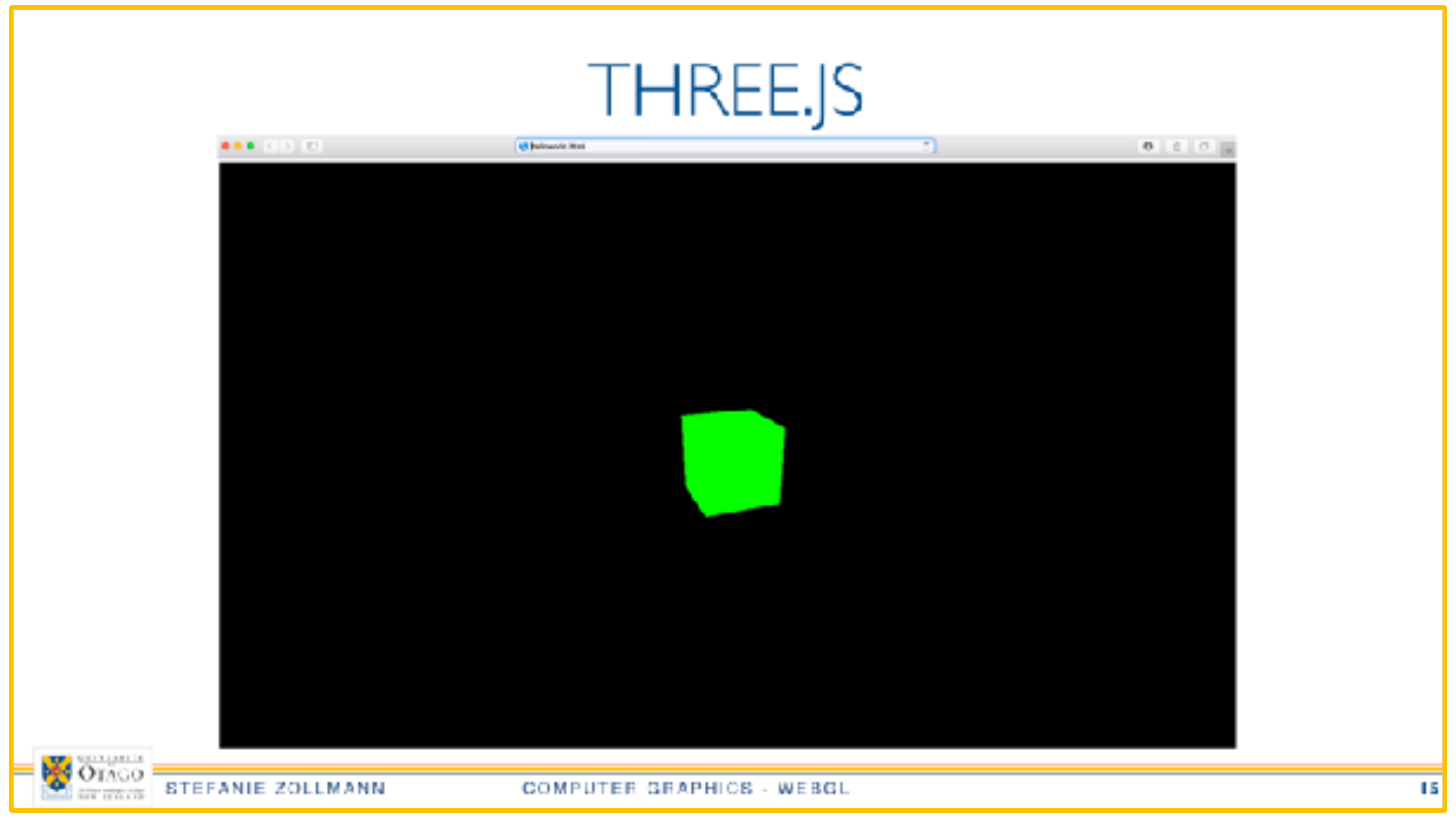


Lecture 21

SCENE GRAPHS

Stefanie Zollmann

LAST TIME

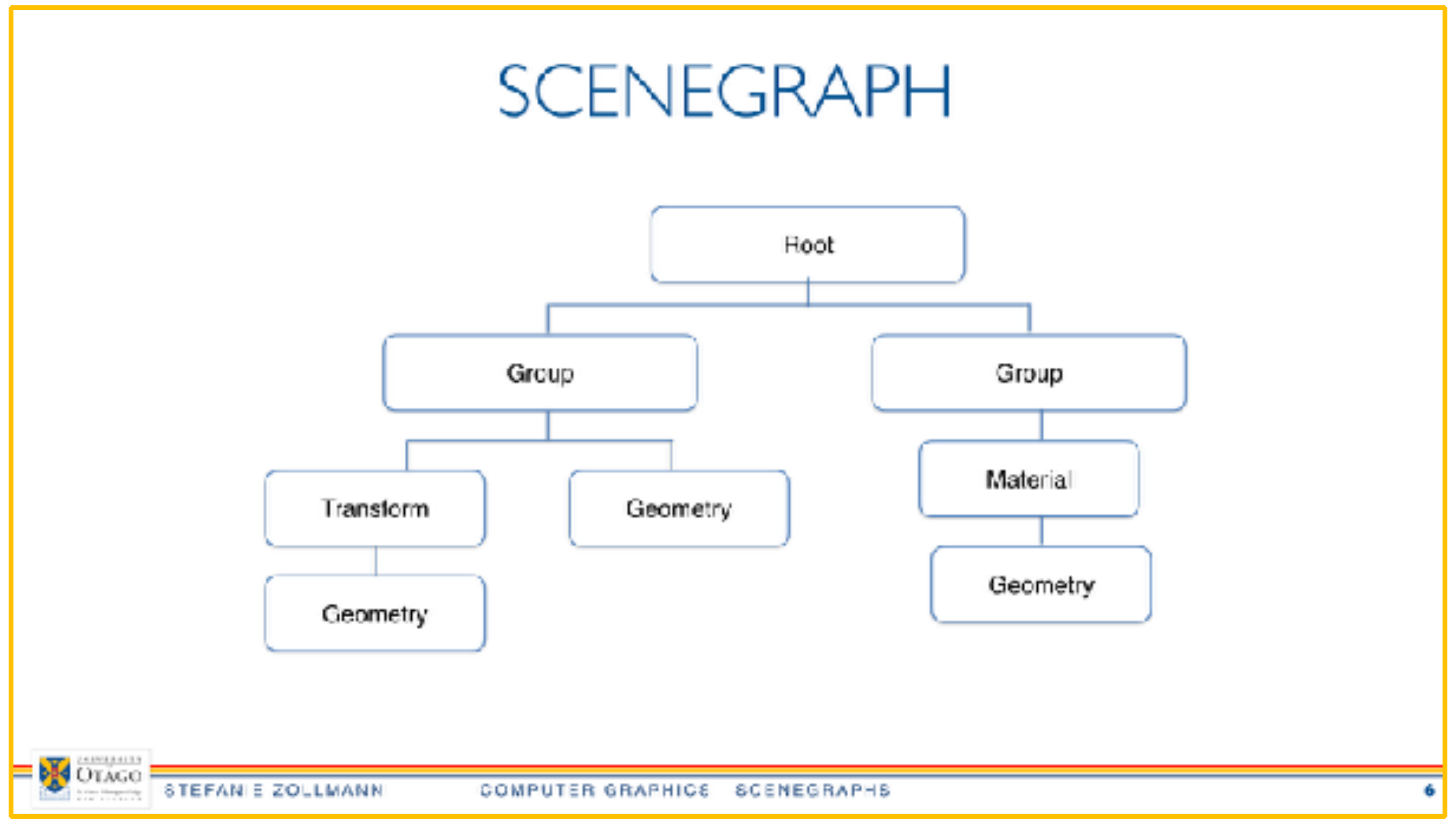


WEBGL

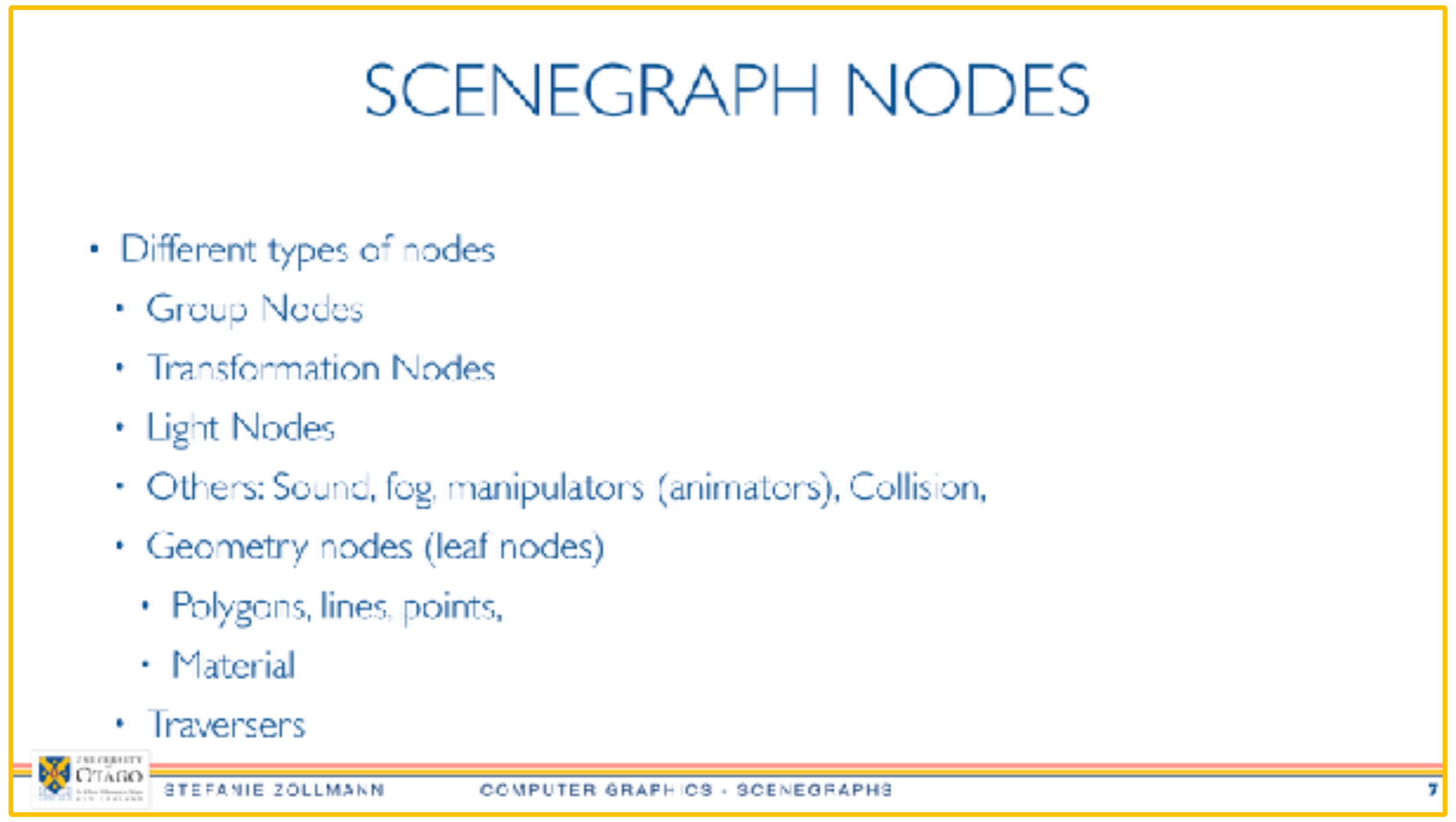
THREE.JS

INTERACTIVE DEMOS

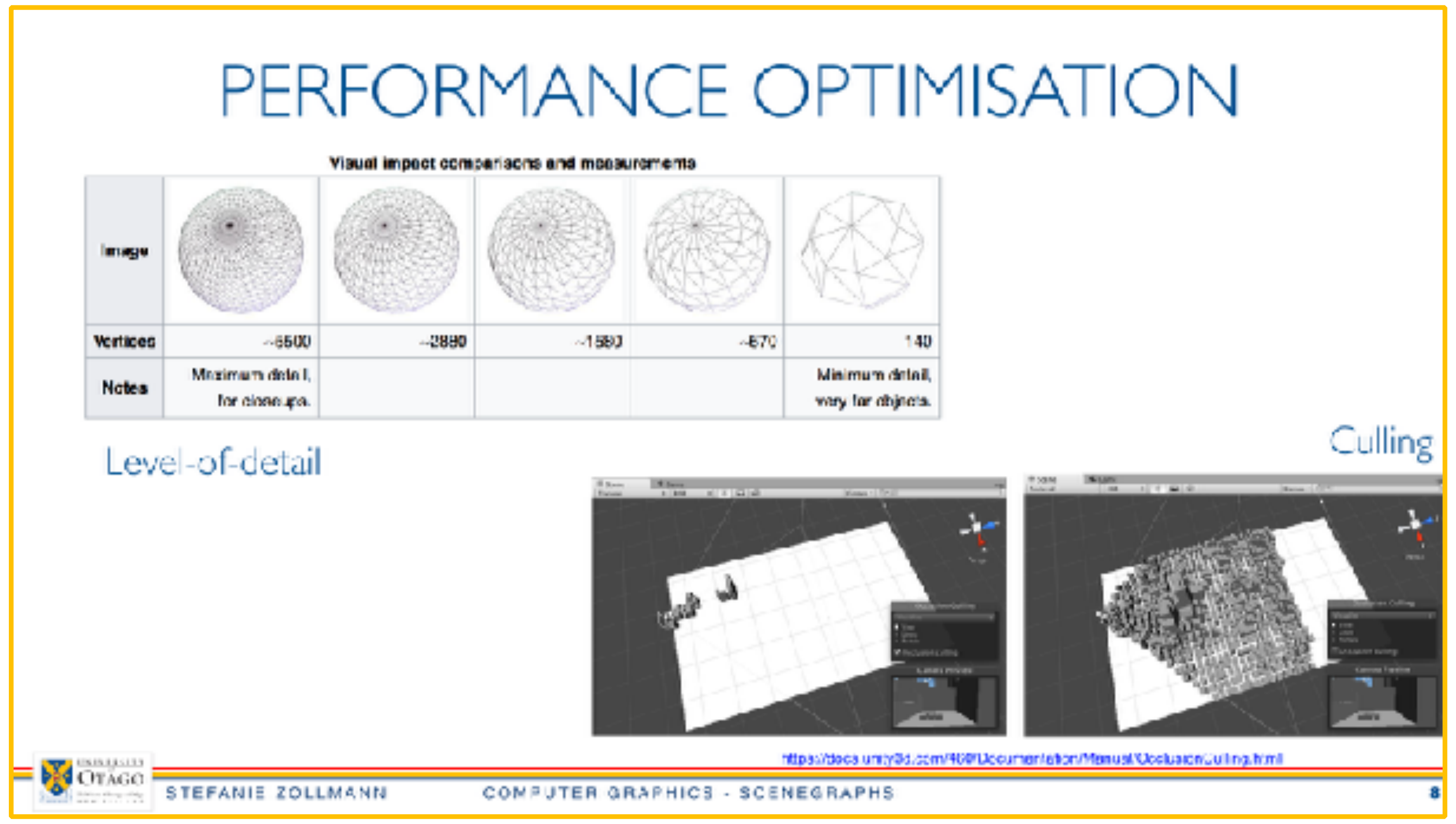
TODAY



SCENEGRAPH



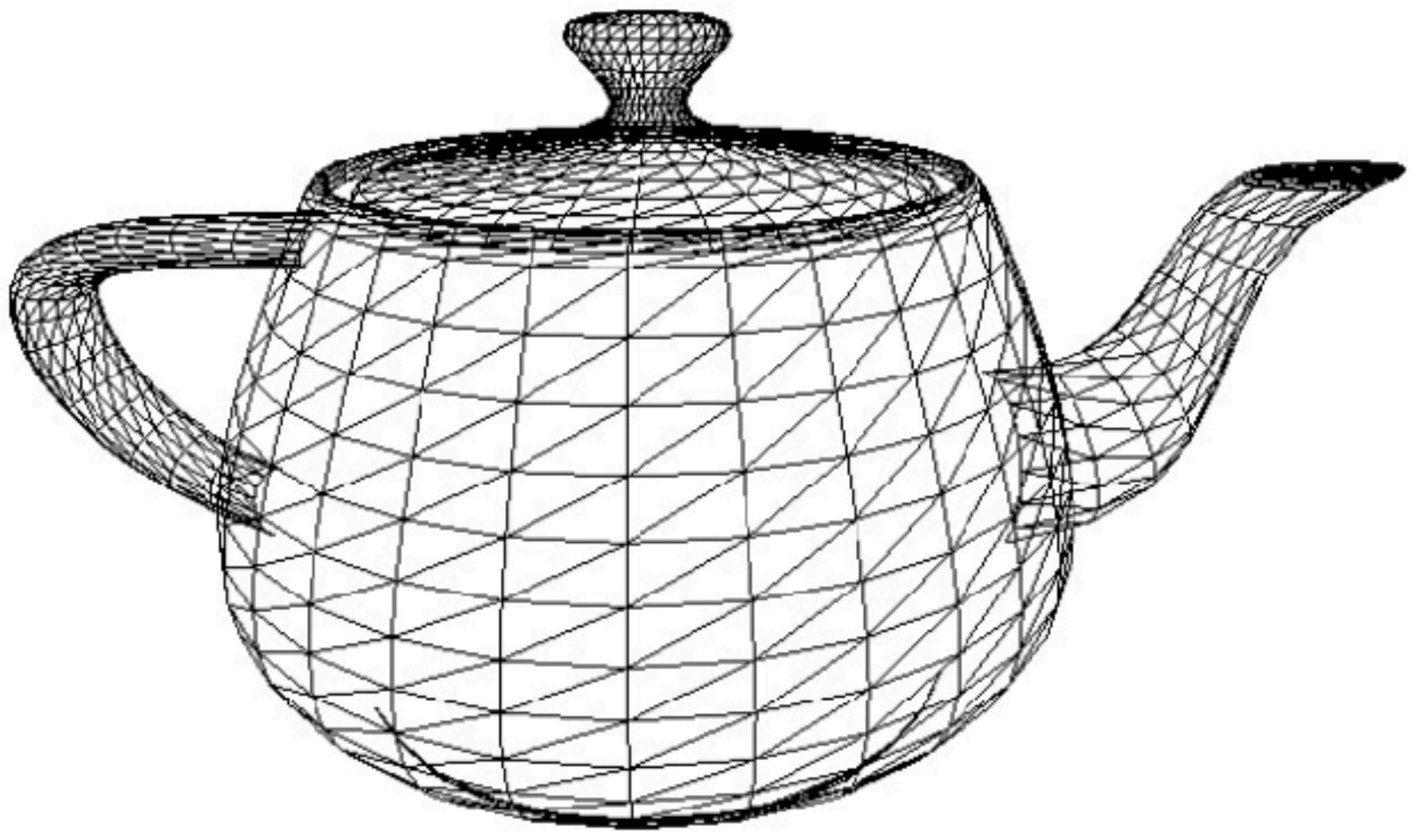
NODES



OPTIMISATIONS

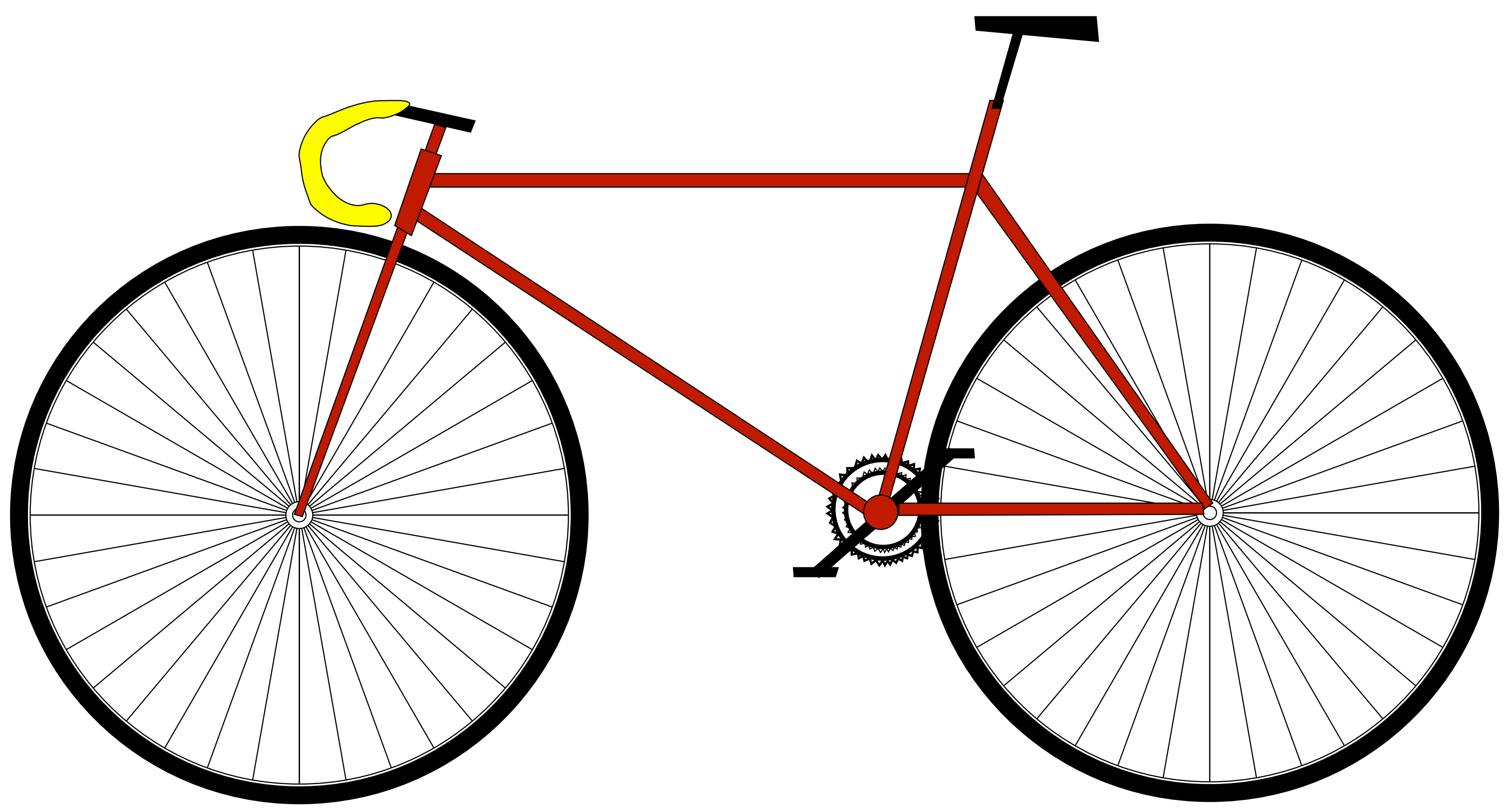
WHY SCENE GRAPHS?

- OpenGL, WebGL
  - Designed as a state machine
  - Only aware of primitives (e.g. triangles)
  - No knowledge about complete scene



PROBLEM

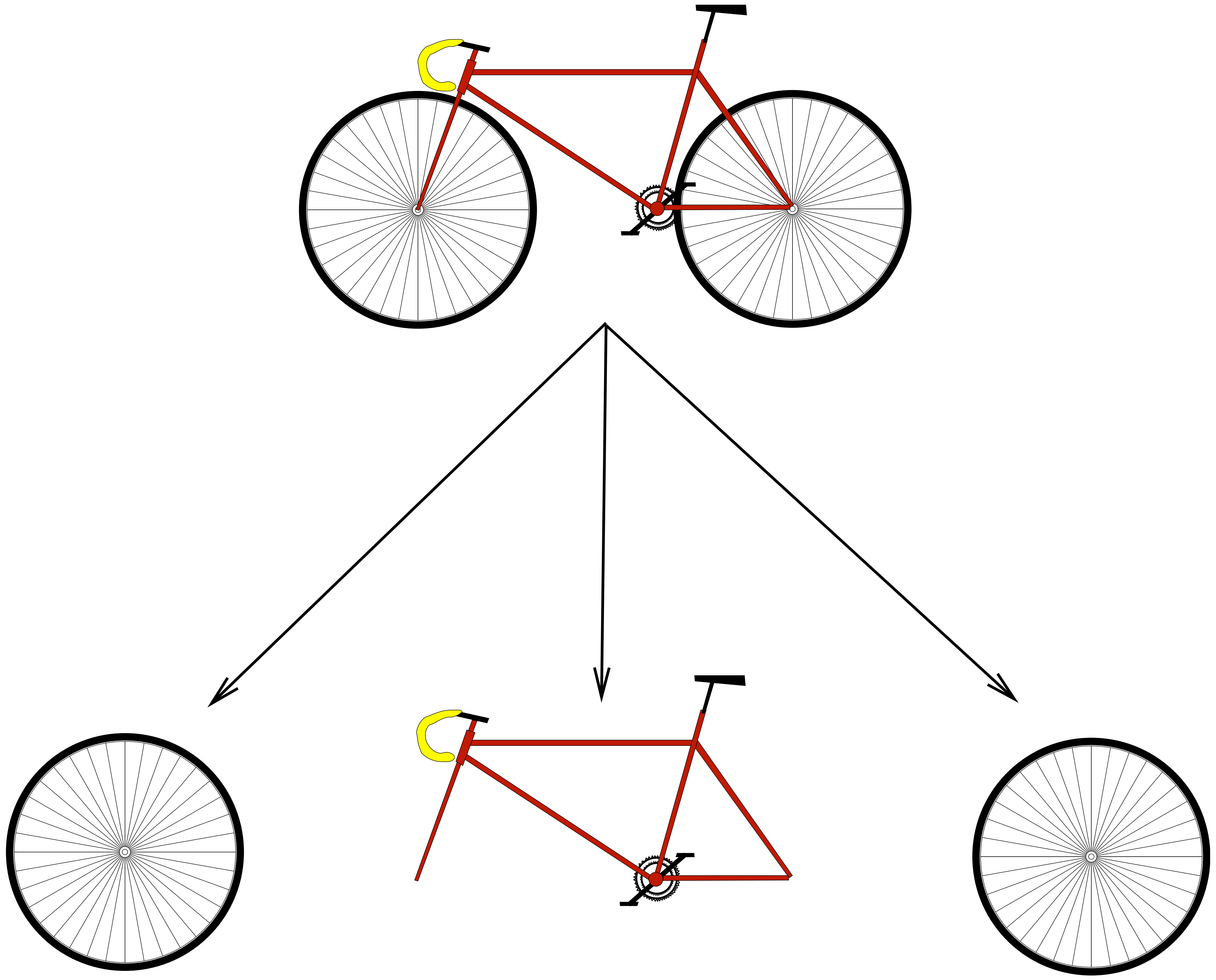
```
void drawBike(){

    drawFrame();
    drawFrontWheel();
    drawRearWheel();

}
```



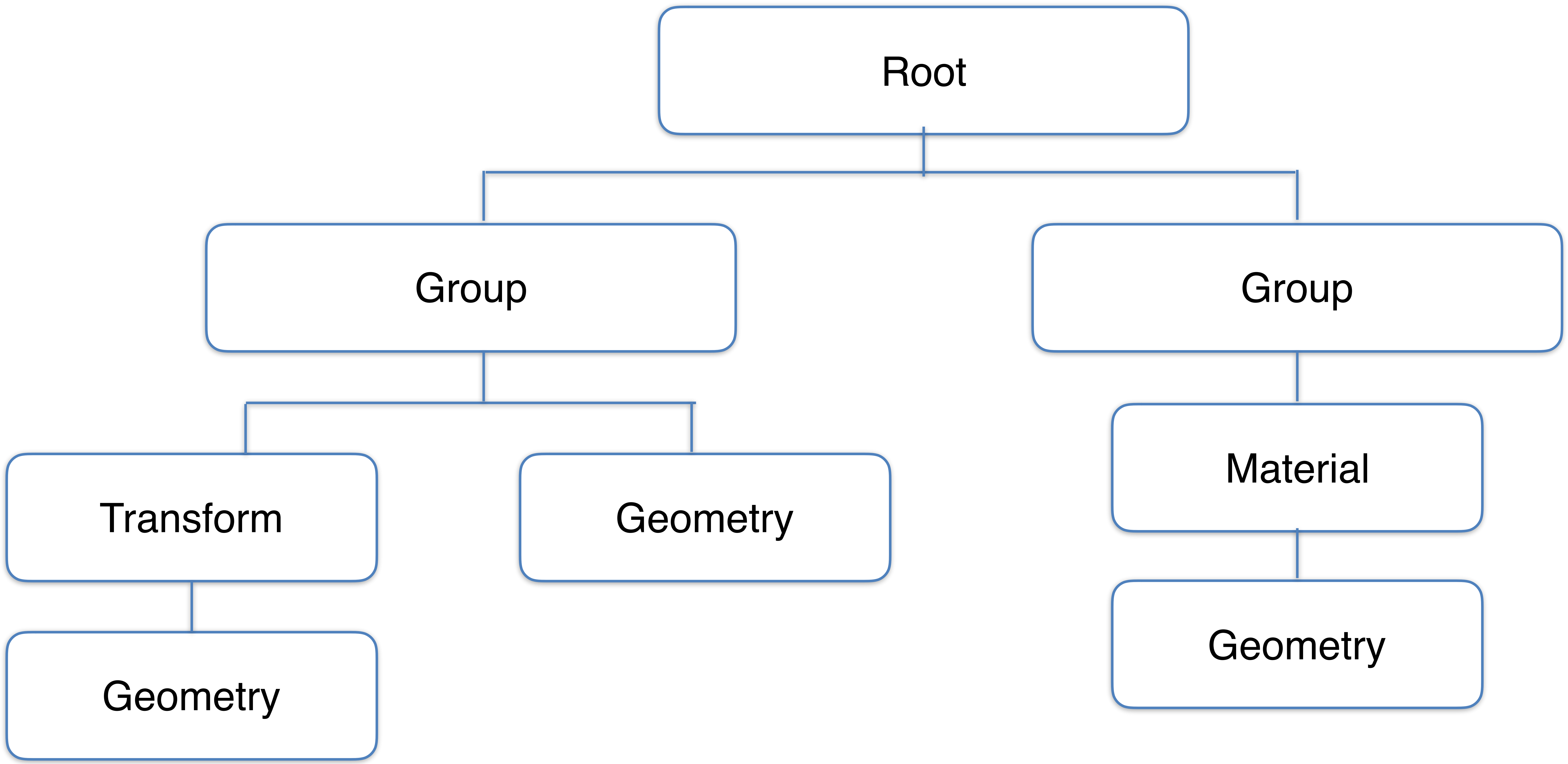
- Spatial relationship not stored
- How to move parts at the same time?
- Identical wheels are drawn twice

WHY SCENEGRAPHS?

- Scene graphs
  - Store objects in a scene in a graph
  - Group related objects that share some properties together
  - Can optimise the way data is passed to the graphics hardware

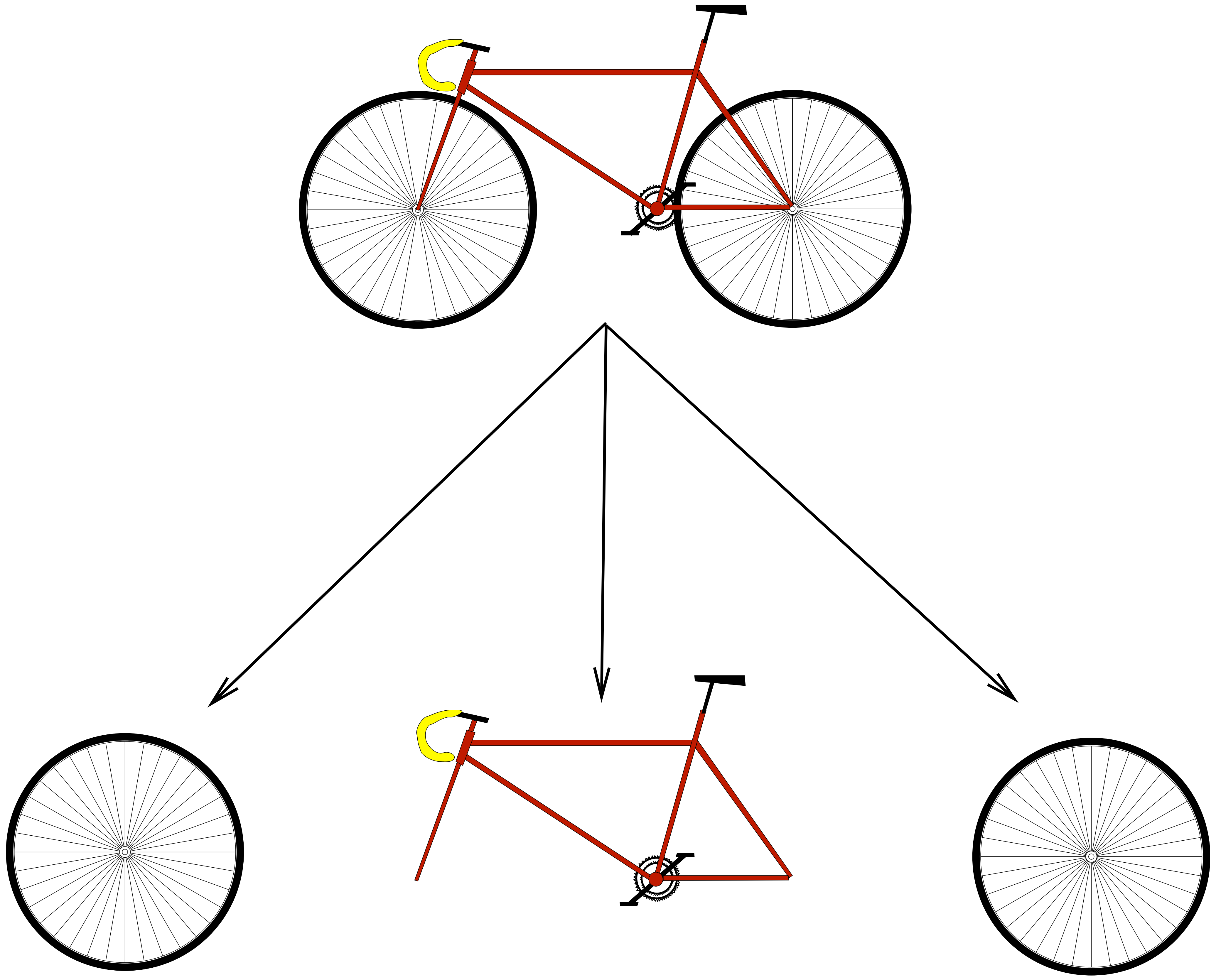


SCENE GRAPH


Root
Group
Group
Transform
Geometry
Material
Geometry
Geometry

SCENEGRAPH

- Data structure
- Represented by a tree
  - Directed acyclic graph
  - Root on top
  - Leaves representing geometries
  - Objects are placed relative to each other
- Represents logical and spatial relationship between objects in a scene



ADVANTAGES

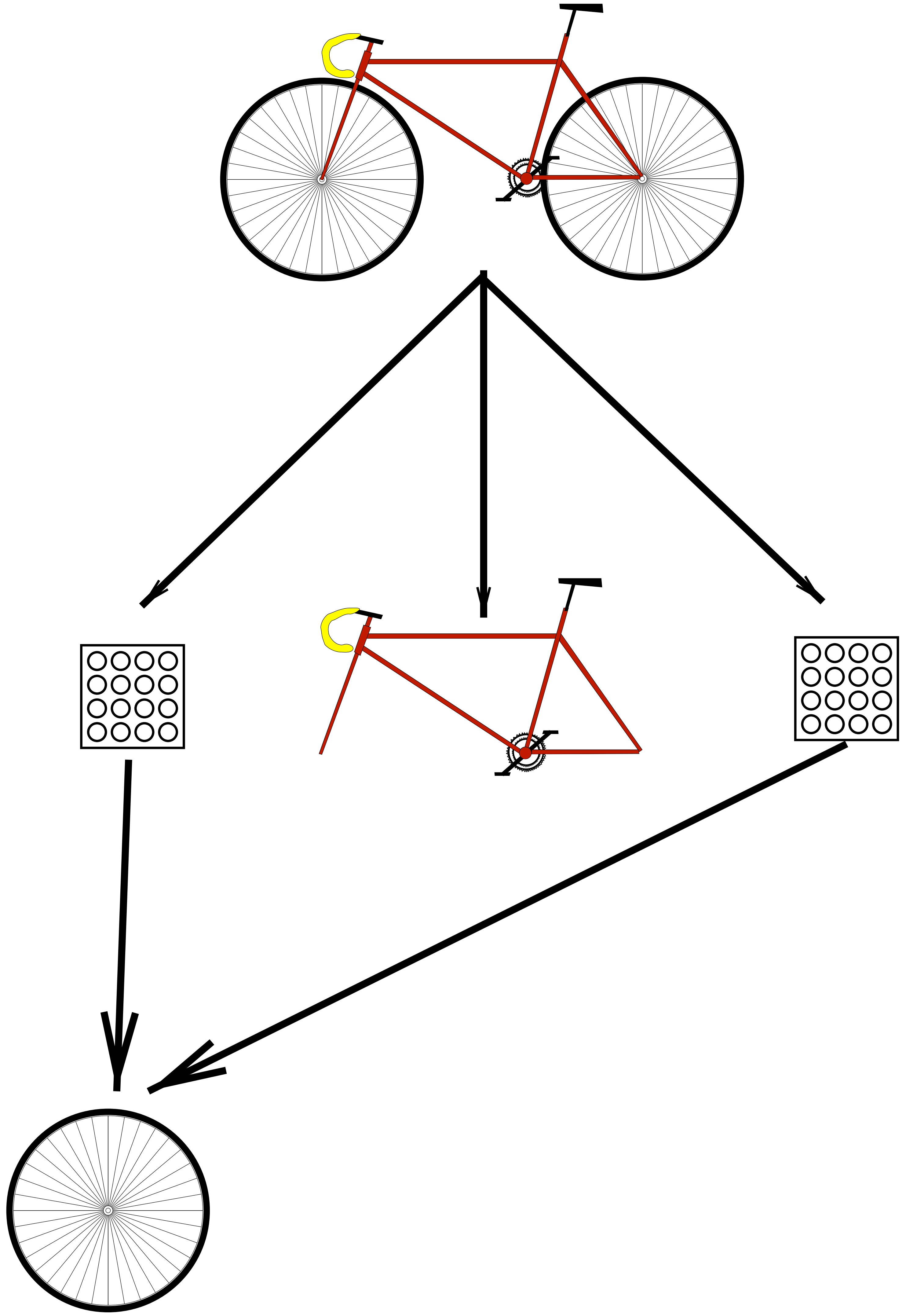
- Reuse geometry
- Reuse materials
- Optimisation based on view settings
- Optimisation based on occlusions
- Optimisation based on distance

SCENEGRAPH

- Advantages:
  - Performance
  - Productivity
  - Portability
  - Scalability

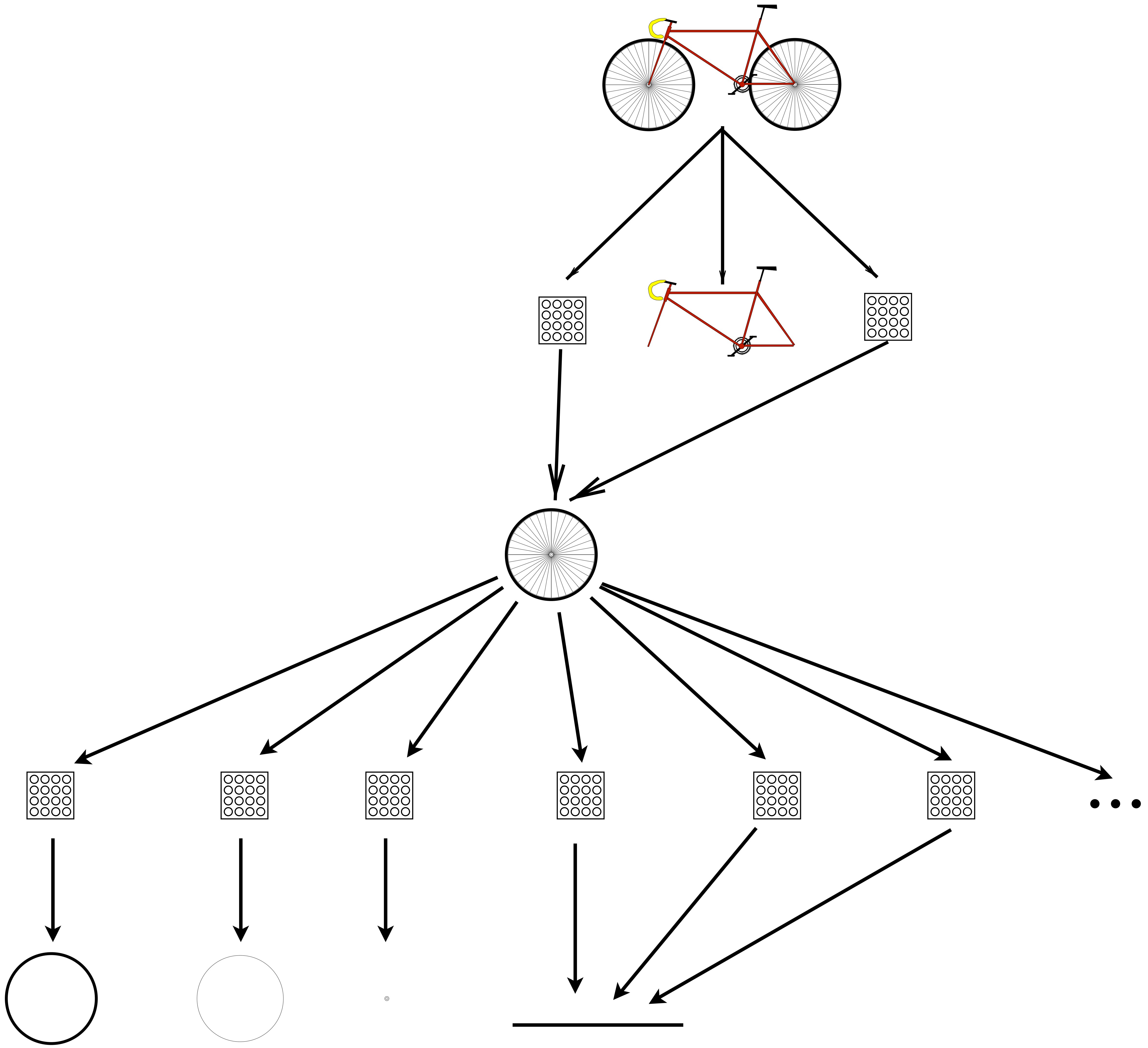
SCENEGRAPH EXAMPLE

- Duplicated objects in our scene should share the same nodes in the scene-graph
- Note that we can use nodes to hold matrices, as well as geometry
- The transformations in a matrix node will apply to all children of that node



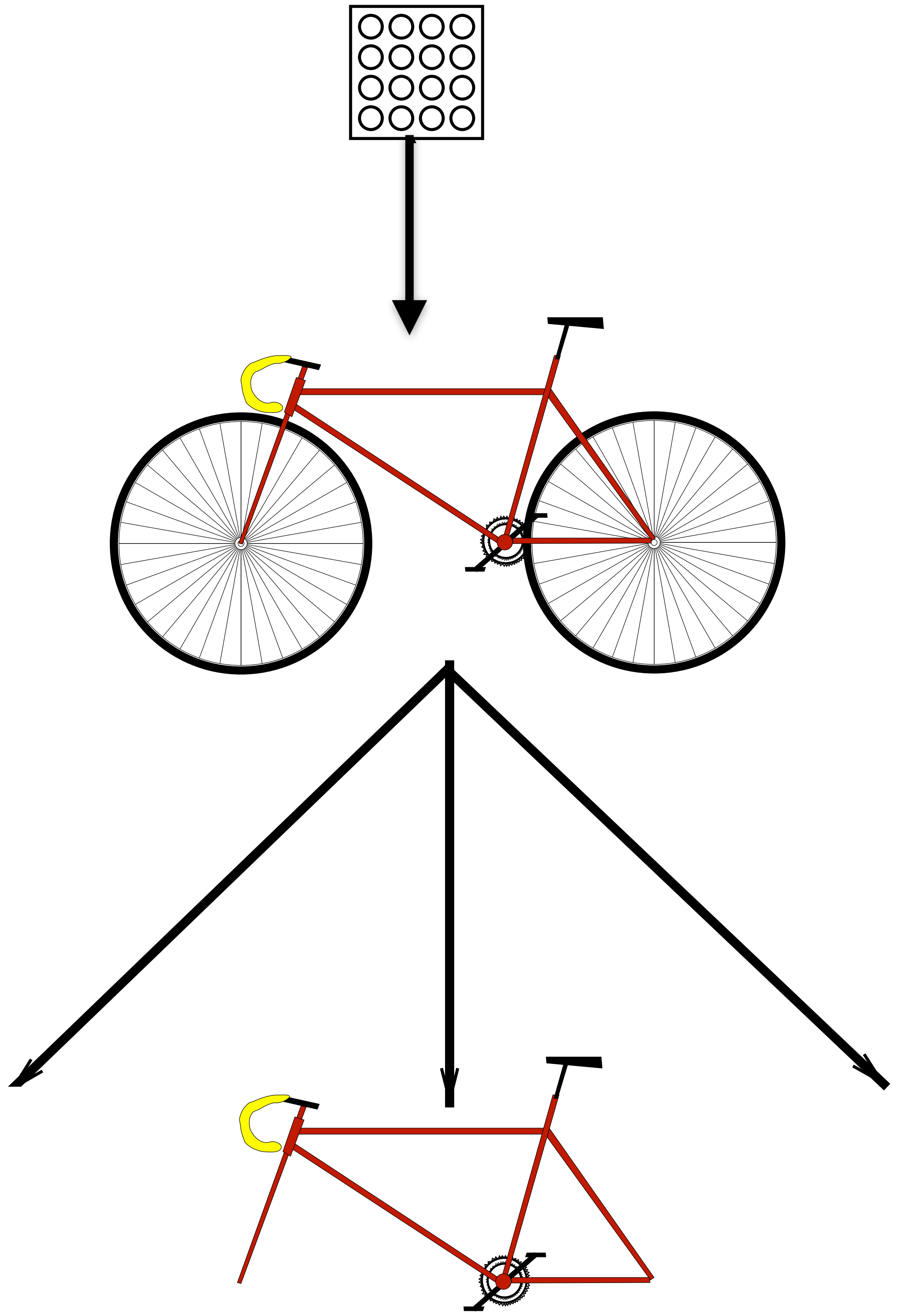
SCENEGRAPH EXAMPLE

- Further geometric decomposition:
  - Wheel is itself made up of subcomponents



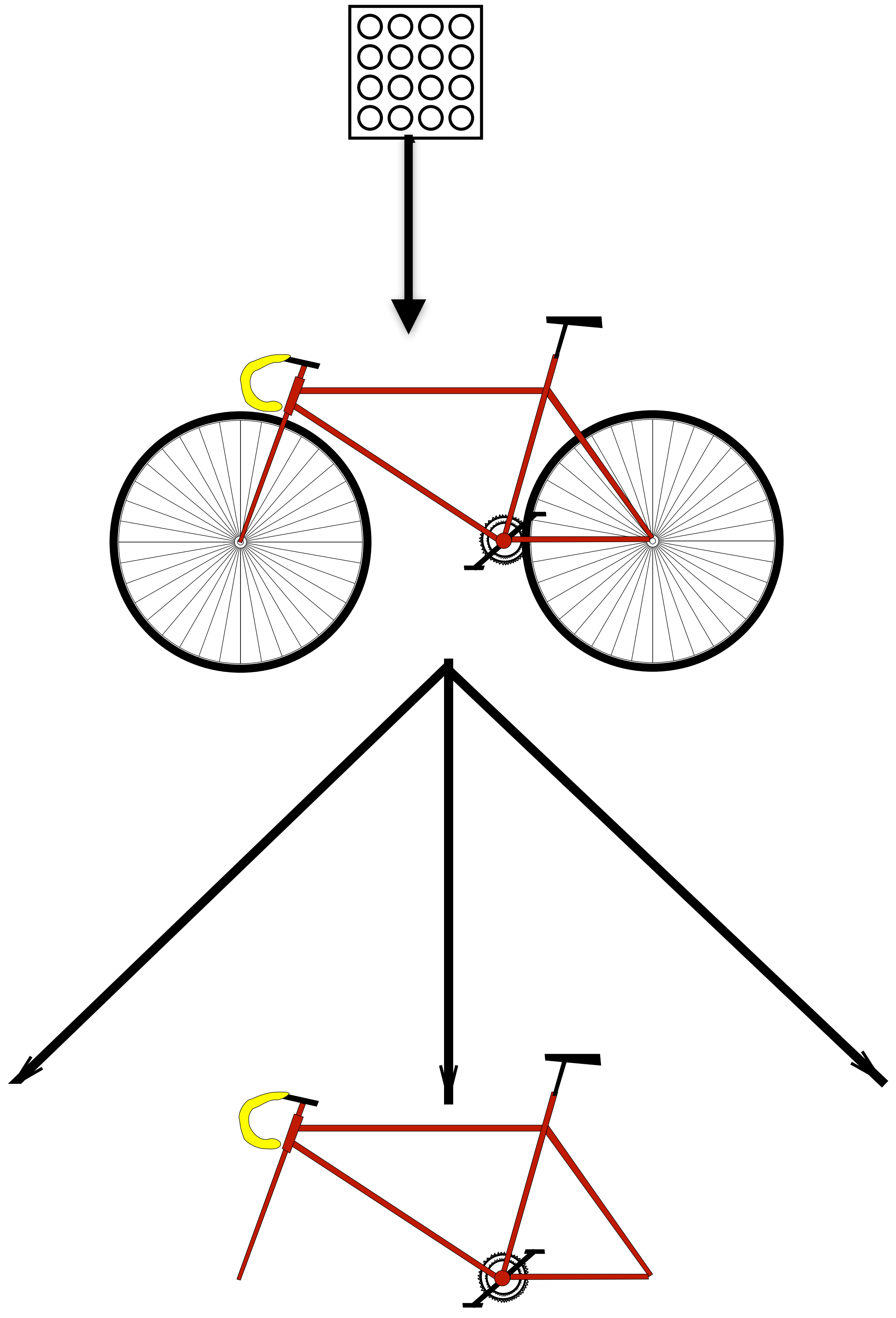
TRANSFORMATIONS

- Object has a local transformation relative to its parent
- If bike root is transformed the frame and all child nodes would be transformed as well
- The root of the graph corresponds to "world coordinates"
- Each geometry is defined in local coordinates



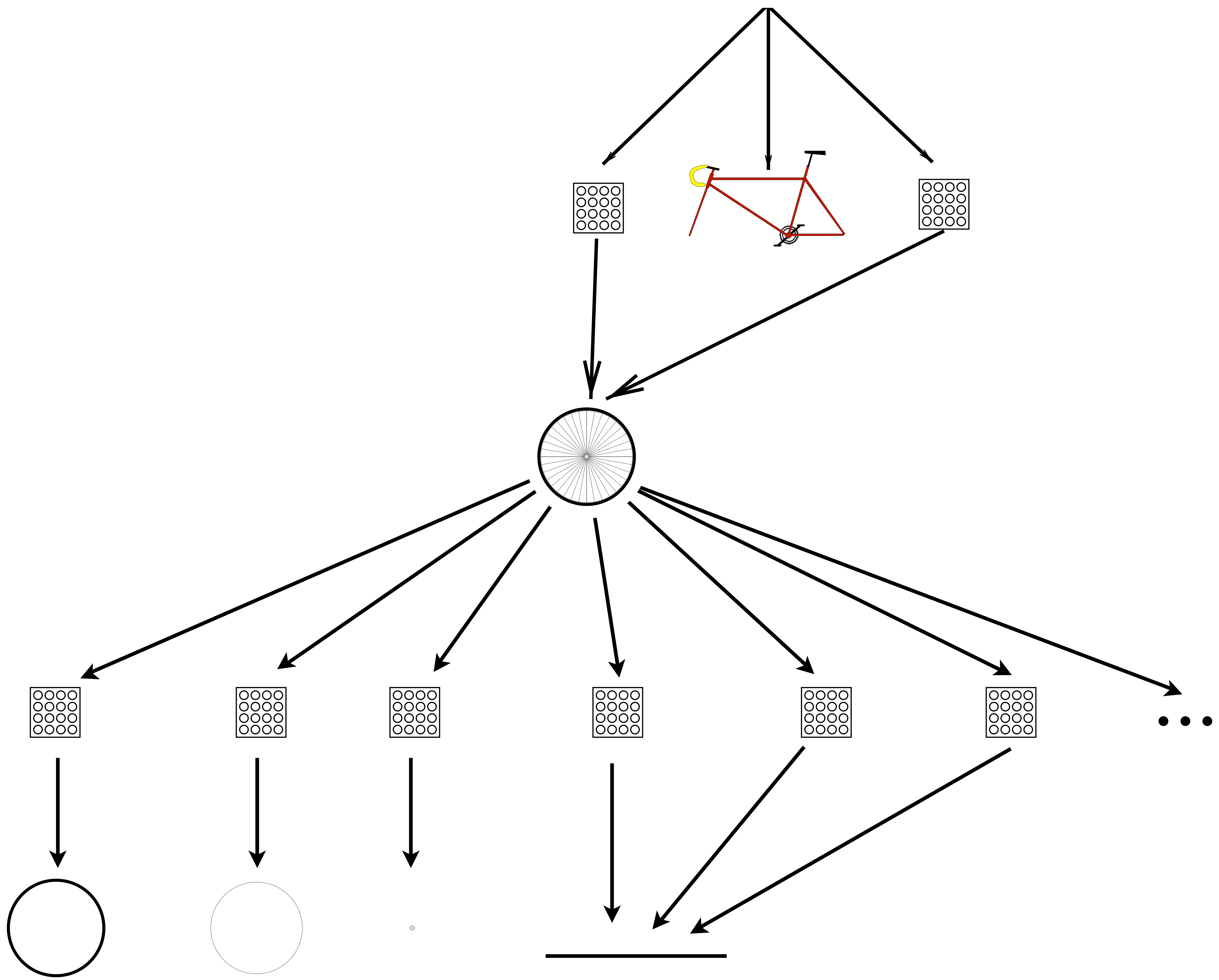
WORLD COORDINATE SYSTEM

- Everything is positioned relative to the world coordinates
- Model matrix: Convert points in an object's local coordinate system into world coordinate system
- View matrix: Convert world coordinates to viewing coordinates

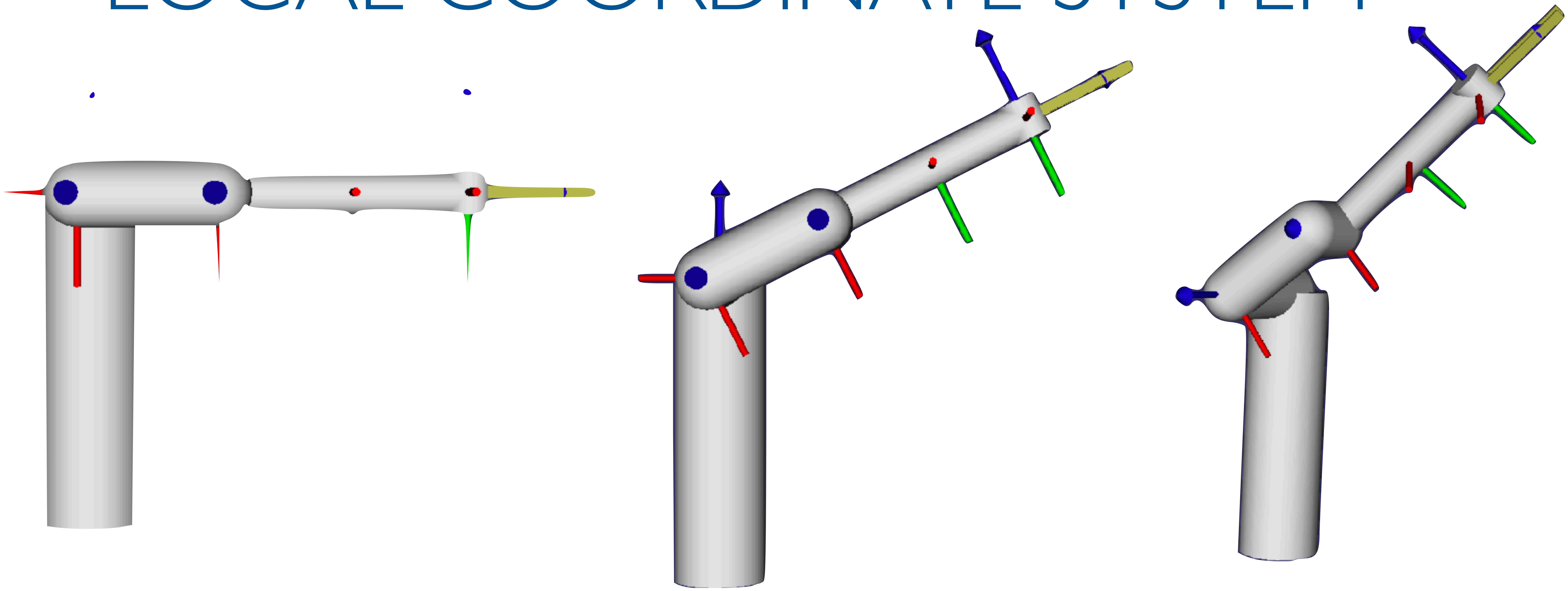


LOCAL COORDINATE SYSTEM

- An object's local transformation describes the mapping between the object's local coordinate system to the parent's local coordinate system
- E.g. the left spoke's transformation describes the mapping between the wheel and the spoke



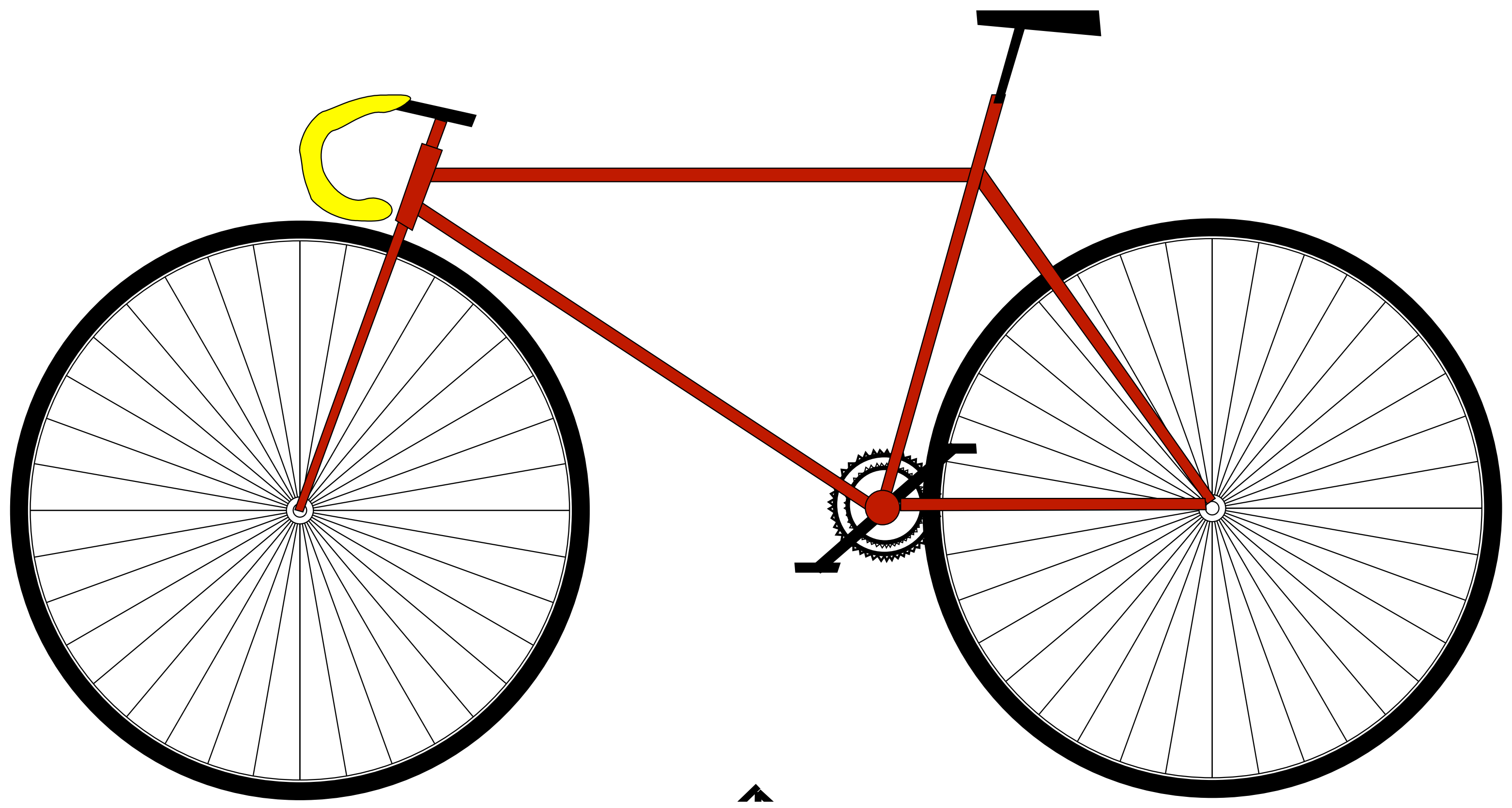
LOCAL COORDINATE SYSTEM



- Each part of the robot is modelled in its own local coordinate system
- If the transformation of a part is changed it affects all its child nodes

COMPARED TO OUR INITIAL PROBLEM

```
void drawBike(){

    drawFrame();
    drawFrontWheel();
    drawRearWheel();

}
```



- Spatial relationship not stored
- How to move parts at the same time?
- Identical wheels are drawn twice

SCENE GRAPH IMPLEMENTATIONS

- OpenInventor
- OpenSceneGraph
- OpenSG (http://www.opensg.org)
- COIN3D (API compatible with Open Inventor 2.1)

OPENSCENEGRAPH

http://trac.openscenegraph.org/projects/osg/wiki/Screenshots/WSPSweden

OPENSCENEGRAPH

The Foundation of the Hellenic World: http://trac.openscenegraph.org/projects/osg/wiki/Screenshots/FHW

OPENSCENEGRAPH

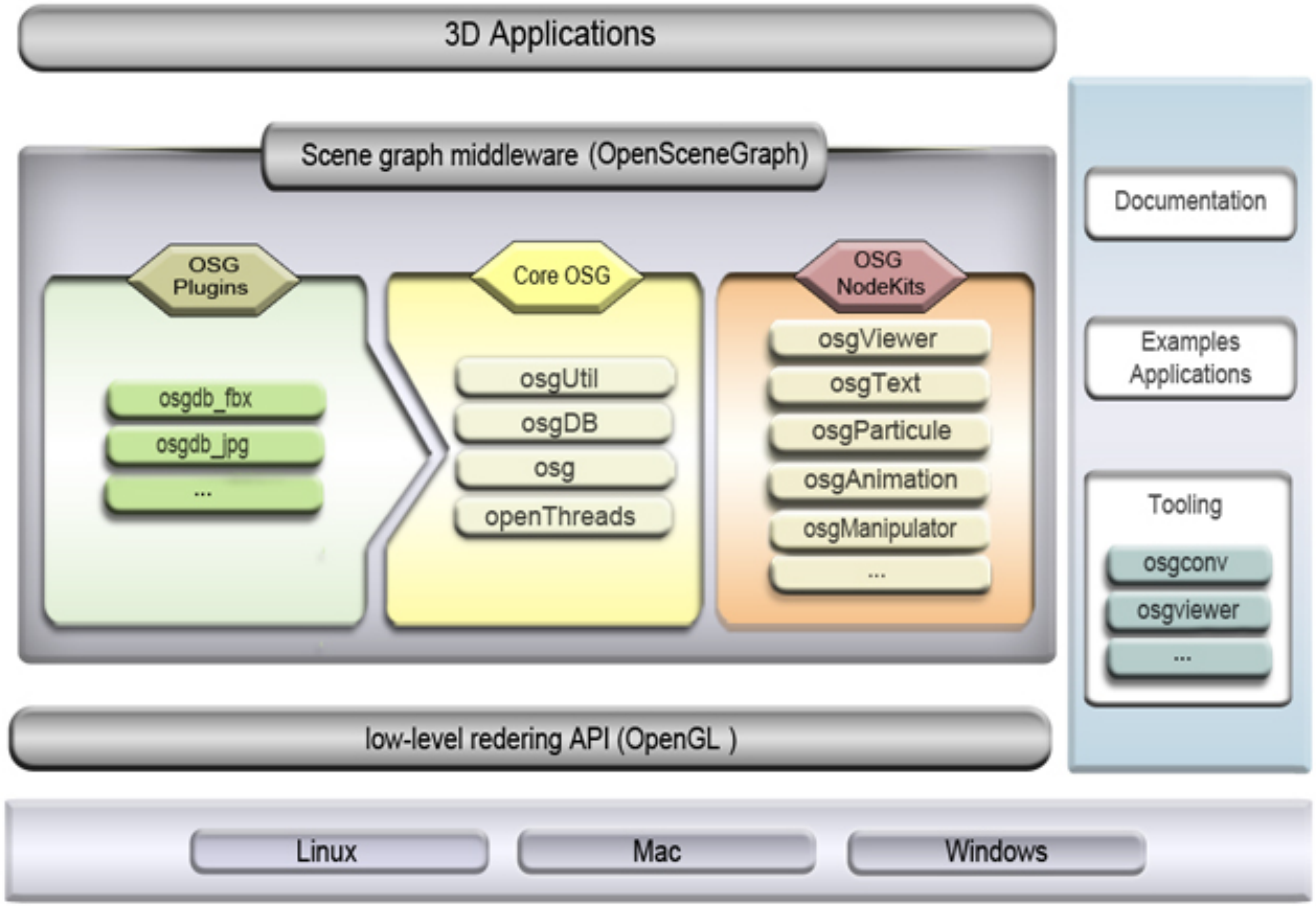
https://live.osgeo.org/_images/osgearth3.jpg

ADDITIONAL LIBRARIES

OSG NODEKITS

- osgText - anti-aliased text
- osgFX - special effects framework
- osgShadow - shadow framework
- osgManipulator - 3d interactive controls
- osgTerrain - terrain rendering
- osgAnimation - character and rigid body animation
- osgVolume - volume rendering
- Etc.

OPENSCENEGRAPH



By ALJI Mohamed - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=18209753

SCENEGRAPH NODES

- Child of Node parent class
- Types
  - Group Nodes
  - Transformation Nodes
  - Switch nodes
  - LOD nodes
  - Geode, geometry leaf node

SCENEGRAPH NODES

- Drawable: abstract base class for drawable graphics
  - Geometry: vertices, normals, faces, texture coords
  - Text: drawing text
  - DrawPixels: drawing images using glDrawPixels
- StateSet: encapsulates OpenGL states and attributes
- Texture: encapsulates OpenGL texture functionality

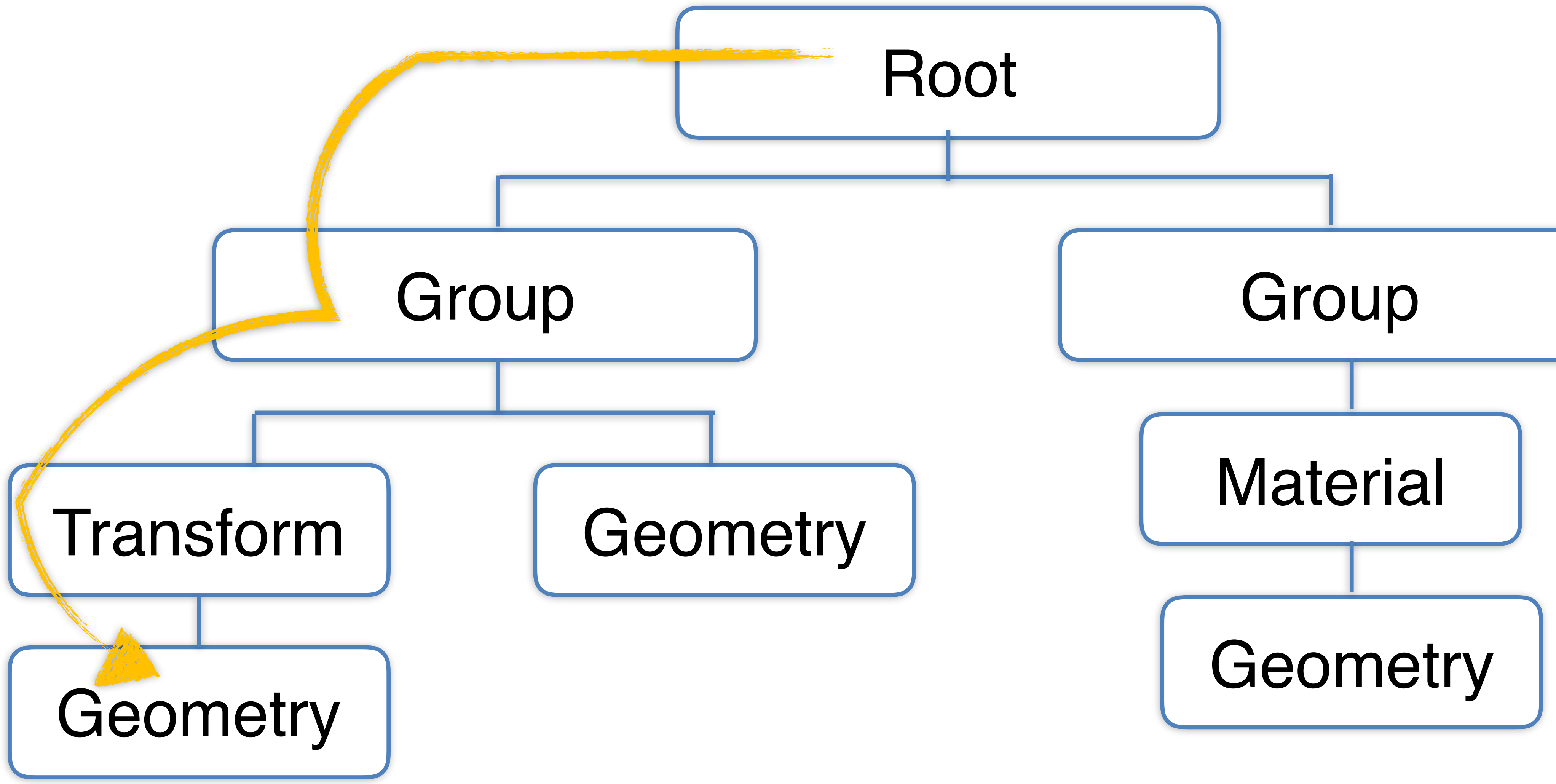
SHAPES

- Box
- Sphere
- Cone
- Cylinder
- Capsule
- Special shapes (e.g. InfinitePlane)

TRAVERSAL

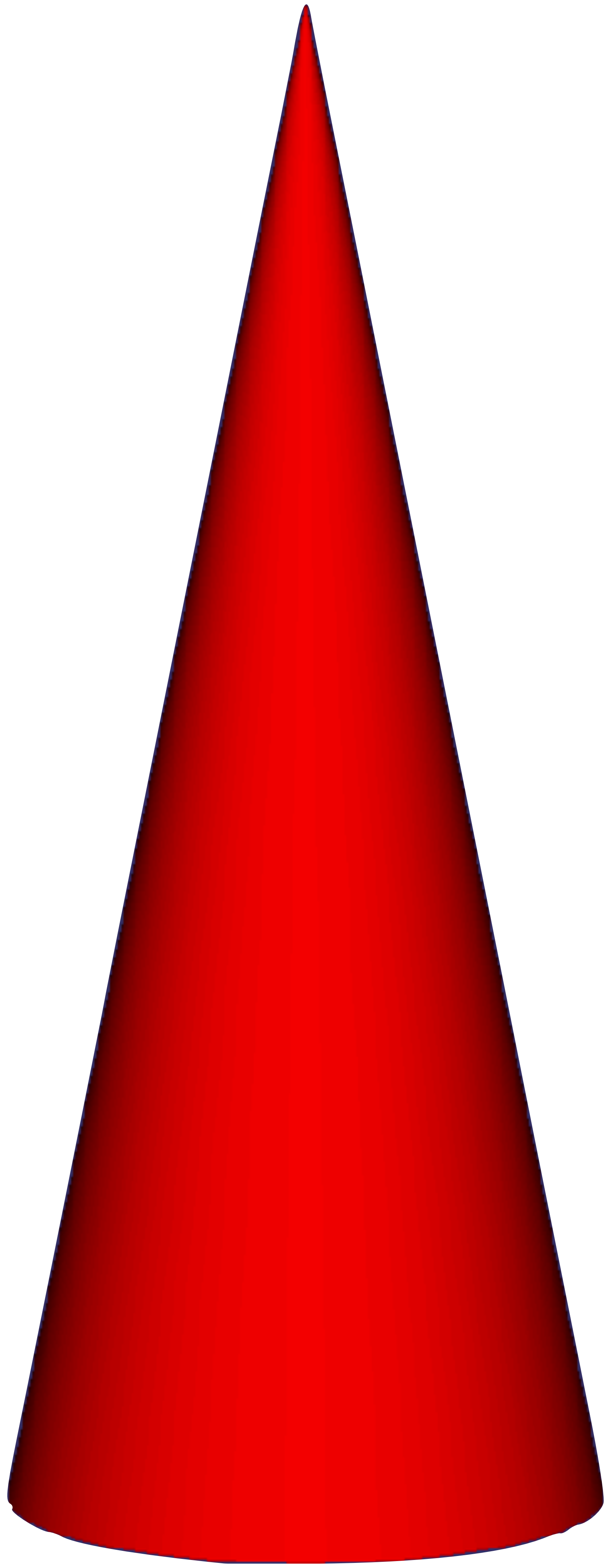
Rendering consists of traversing the graph:

- Geometry nodes have to be rendered (primitives, meshes),
- Transform node modifies transformation matrices,
- State node change a render state (colour, material, shader)



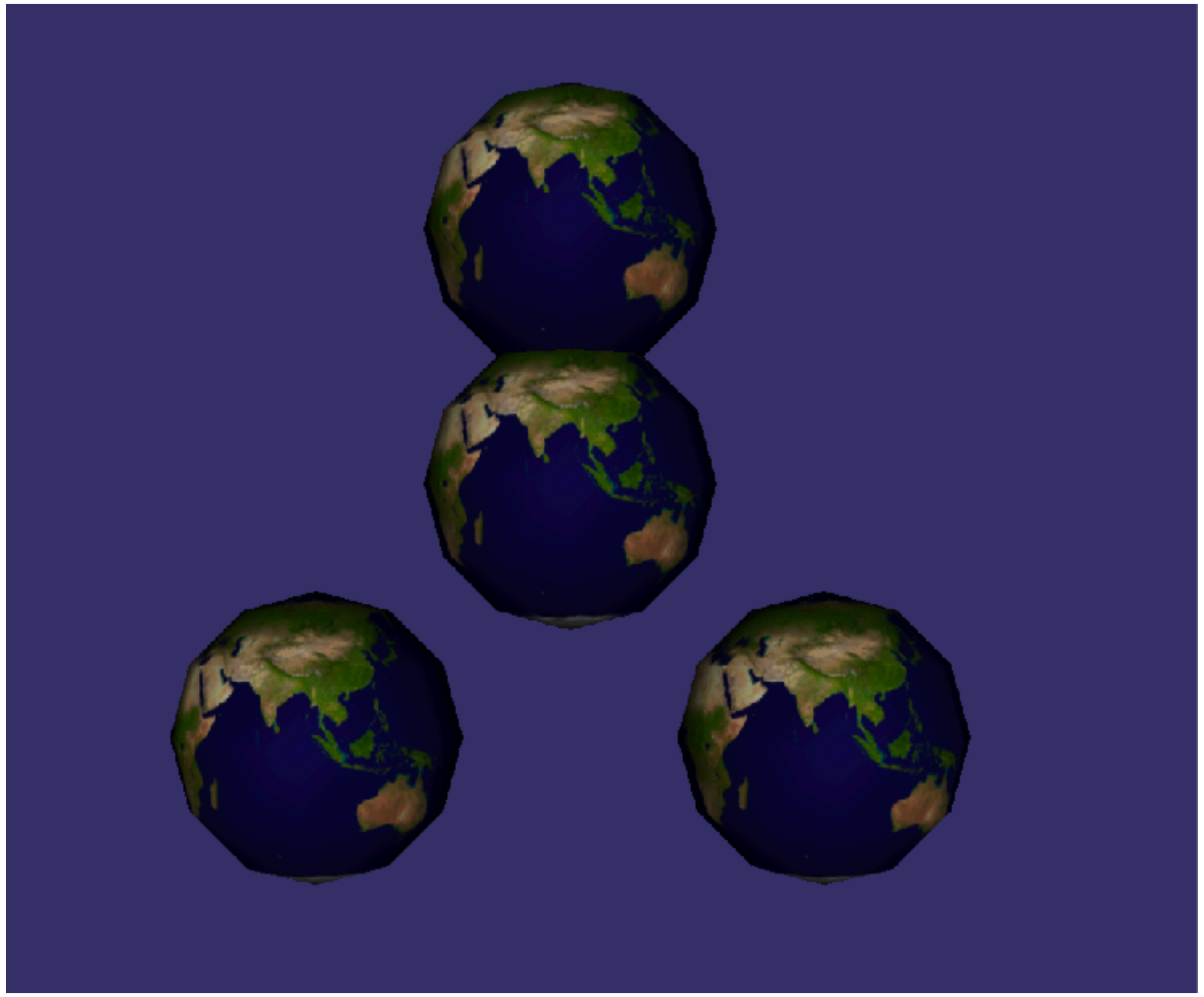
GEODES

```
osg:: Vec3 vcen(1.0, 1.0, 1.0);

 osg::Cone* cone = new osg::Cone(vcen, 2.0, 10.0);
 // Create a drawable object based on the cone

 osg:: ShapeDrawable *drawable = new osg::ShapeDrawable(cone);

 drawable->setColor(osg::Vec4(1.0f, 0.0f, 0.0f, 1.0f));
 // create a new geode

 osg:: Geode* geode = new osg::Geode();
 geode->addDrawable(drawable);

  osg::Group * node = new osg::Group();
 node->addChild(geode);
```



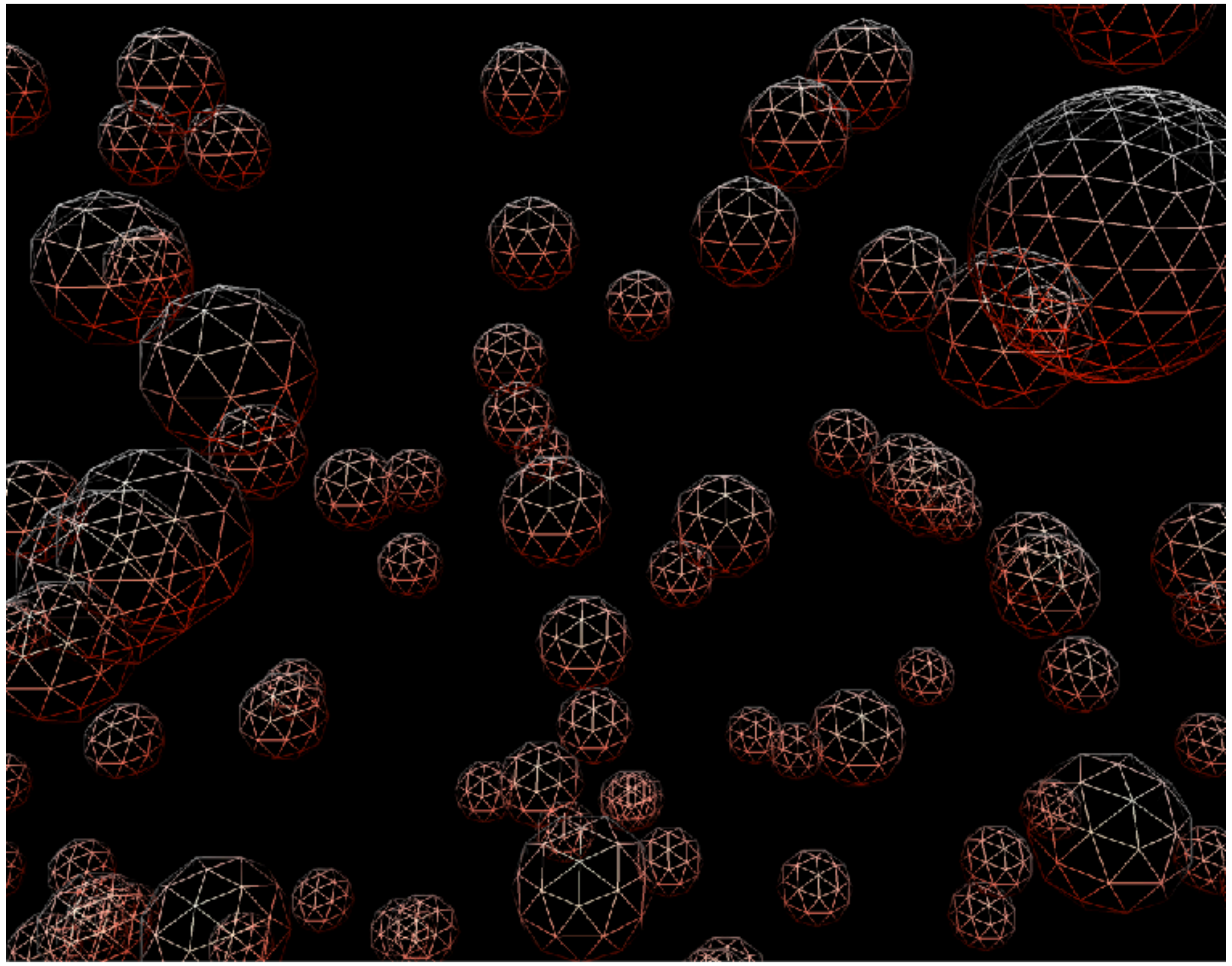
GROUP NODES

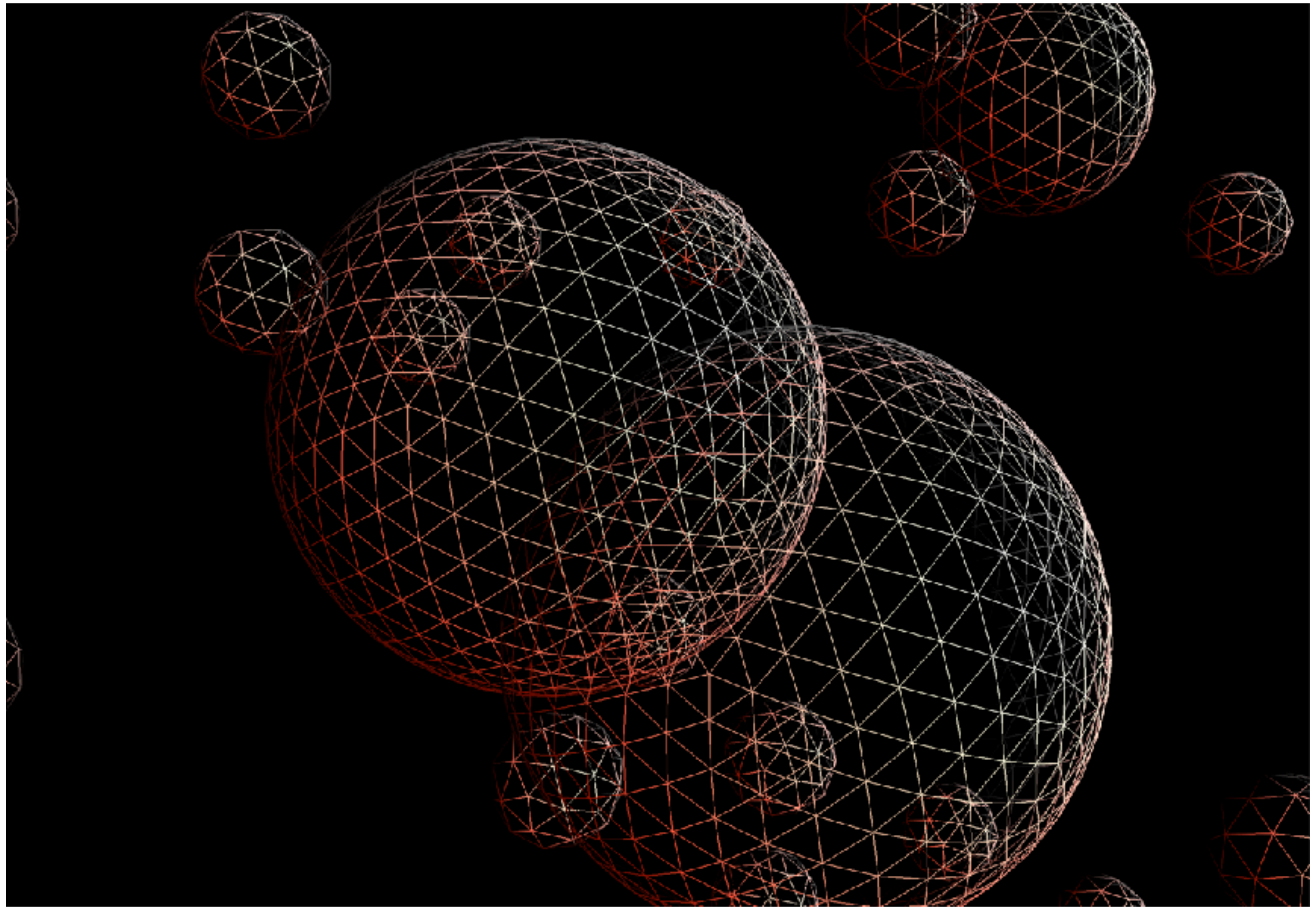
```
float r = node->getBound().radius(); // diameter
osg::Group * root = new osg::Group();
osg::Group * group = new osg::Group();

// Create four matrices for translated instances of the cow
osg::MatrixTransform * transform0  = new osg::MatrixTransform( );
transform0->setMatrix( osg::Matrix::translate( 0,0,r ) );

osg::MatrixTransform * transform1  = new osg::MatrixTransform( );
transform1->setMatrix( osg::Matrix::translate( 0,0,0 ) );

osg::MatrixTransform * transform2  = new osg::MatrixTransform( );
transform2->setMatrix( osg::Matrix::translate( -r,0,-r ) );

osg::MatrixTransform * transform3  = new osg::MatrixTransform( );
transform3->setMatrix( osg::Matrix::translate(  r,0,-r ) );

root->addChild( transform0 );
root->addChild( group );
group->addChild( transform1 );
group->addChild( transform2 );
group->addChild( transform3 );

transform0->addChild( node );
transform1->addChild( node );
transform2->addChild( node );
transform3->addChild( node );
```



PERFORMANCE OPTIMISATION

LEVEL OF DETAIL





- Rendering Optimisation
- Change level of detail depending on distance to the camera

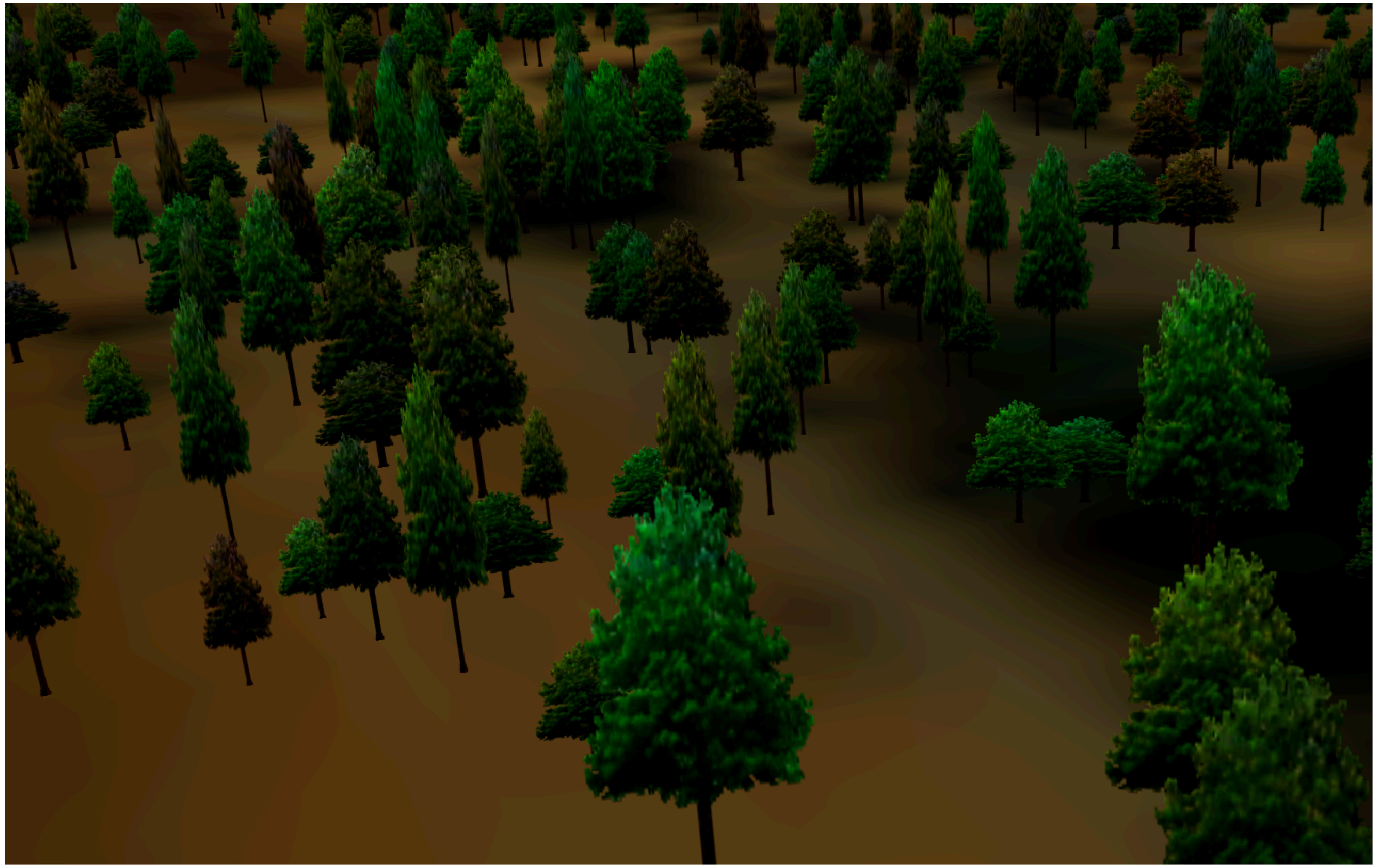
LOD NODE

- Node allows switching between children depending on distance from eye point
- Objects further away from the eye point are rendered at a lower level of detail
- Each LOD child has a corresponding valid range consisting of a minimum and maximum distance. Given a distance to the viewer (d), LOD displays a child if min <= d < max.
- Usage like group node: lod.addChild(detailedNode);
- Set the visible range from the viewer to the object:
  - load.setRange(childNumber, near, far);

LOD NODES

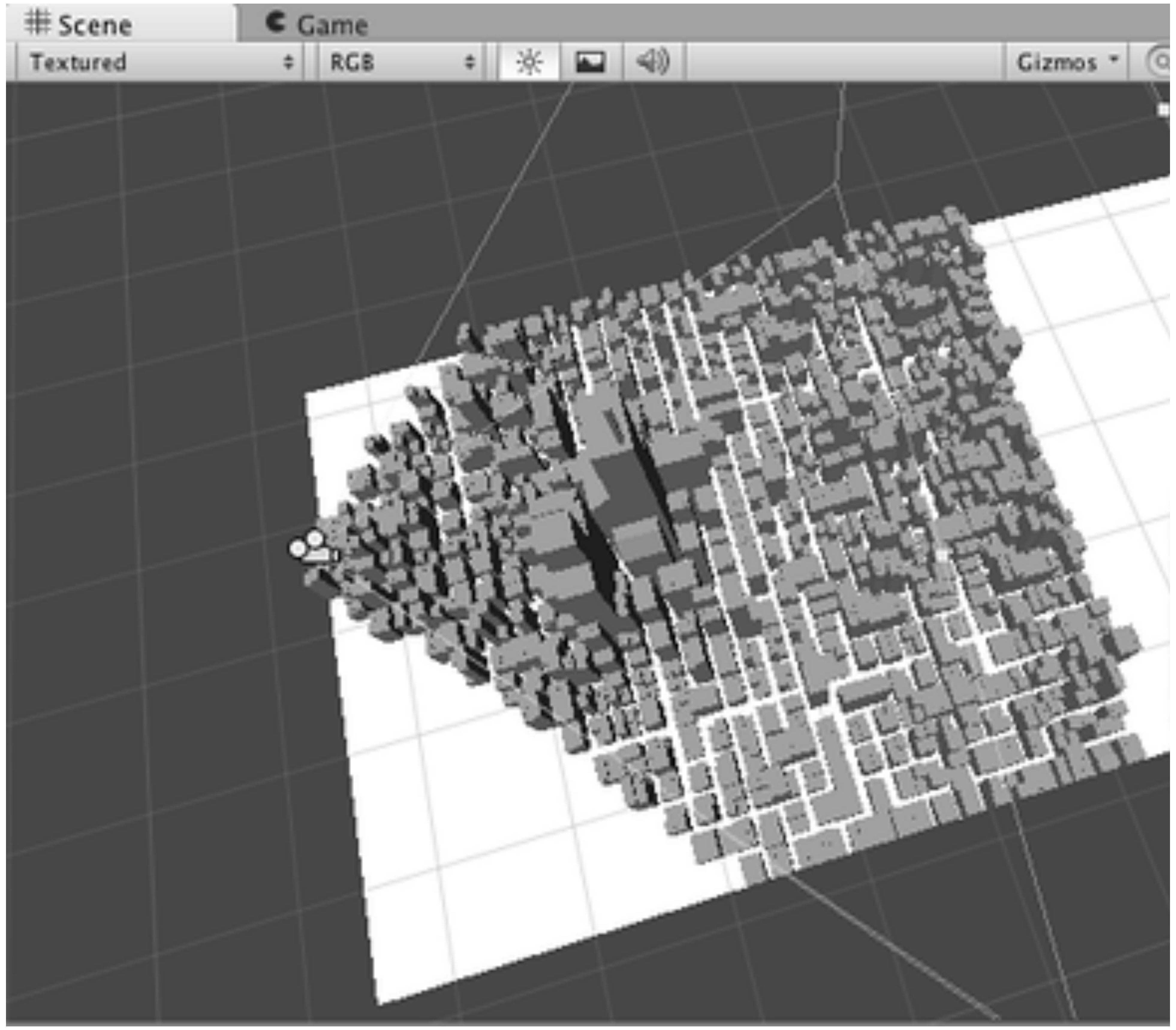
```
LOD lod = new LOD();

lod.addChild(highDetail);

lod.setRange(0, 0, 10);
lod.addChild(medDetail);

lod.setRange(1, 10, 100);
lod.addChild(lowDetail);

lod.setRange(2, 100, 1000);
lod.addChild(noDetails);

lod.setRange(2,1000,10000);
```

BILLBOARDS



CULLING

- Scene graph structure supports efficient culling
  - Bounding volume is calculated at each node in the scene graph
  - Includes geometry of node and child nodes
  - When traversing the scene graph, bounding volumes are evaluated against view frustum
  - If bounding volume does not intersect with view frustum -> culled
  - Culled nodes and their children are not sent to the graphic card



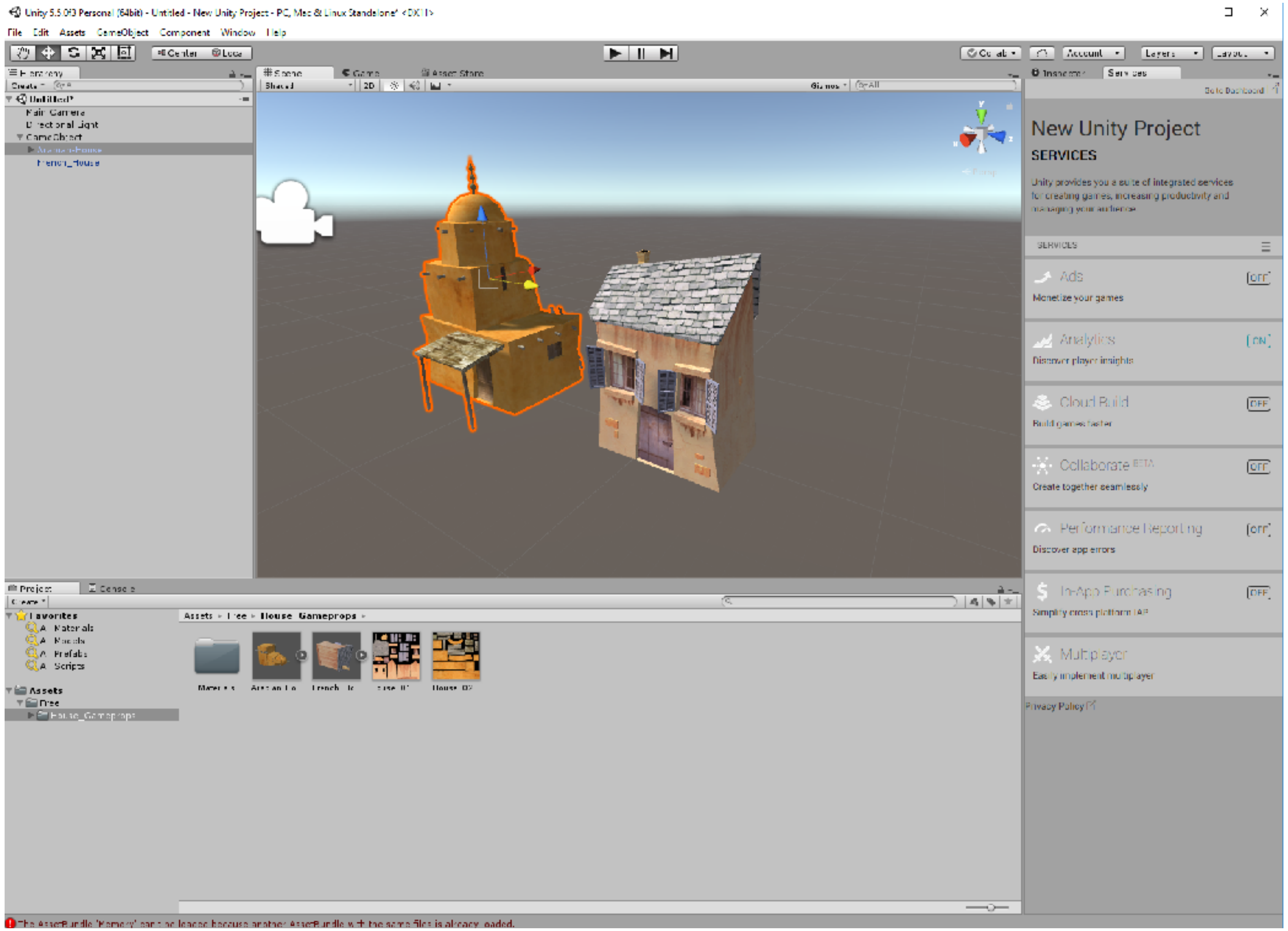
https://docs.unity3d.com/460/Documentation/Manual/OcclusionCulling.html

GAME ENGINES

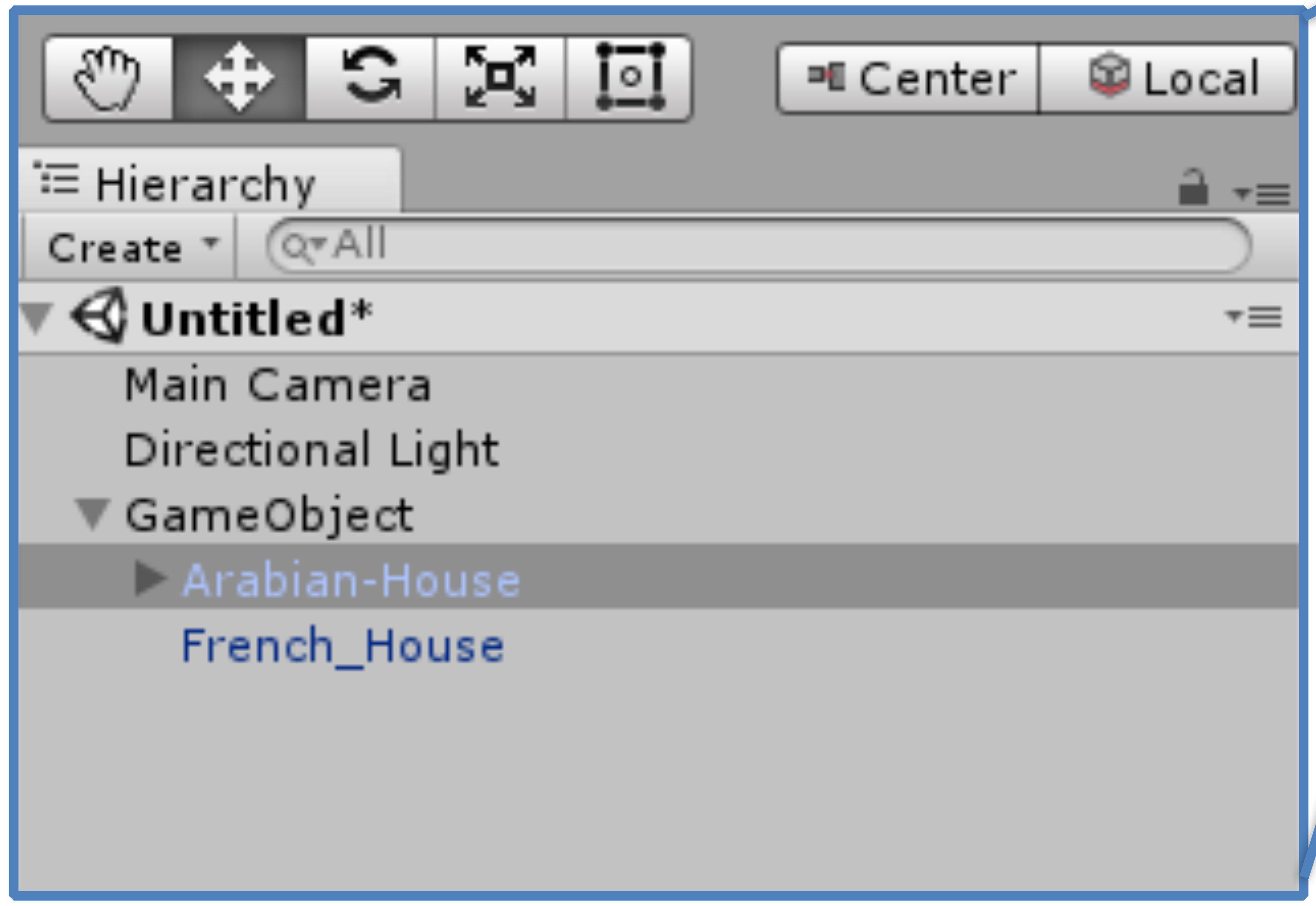
- Provide:
  - Rendering
  - Scene graph
  - Animation
  - Physics Engine
- Visual UI for scene editing

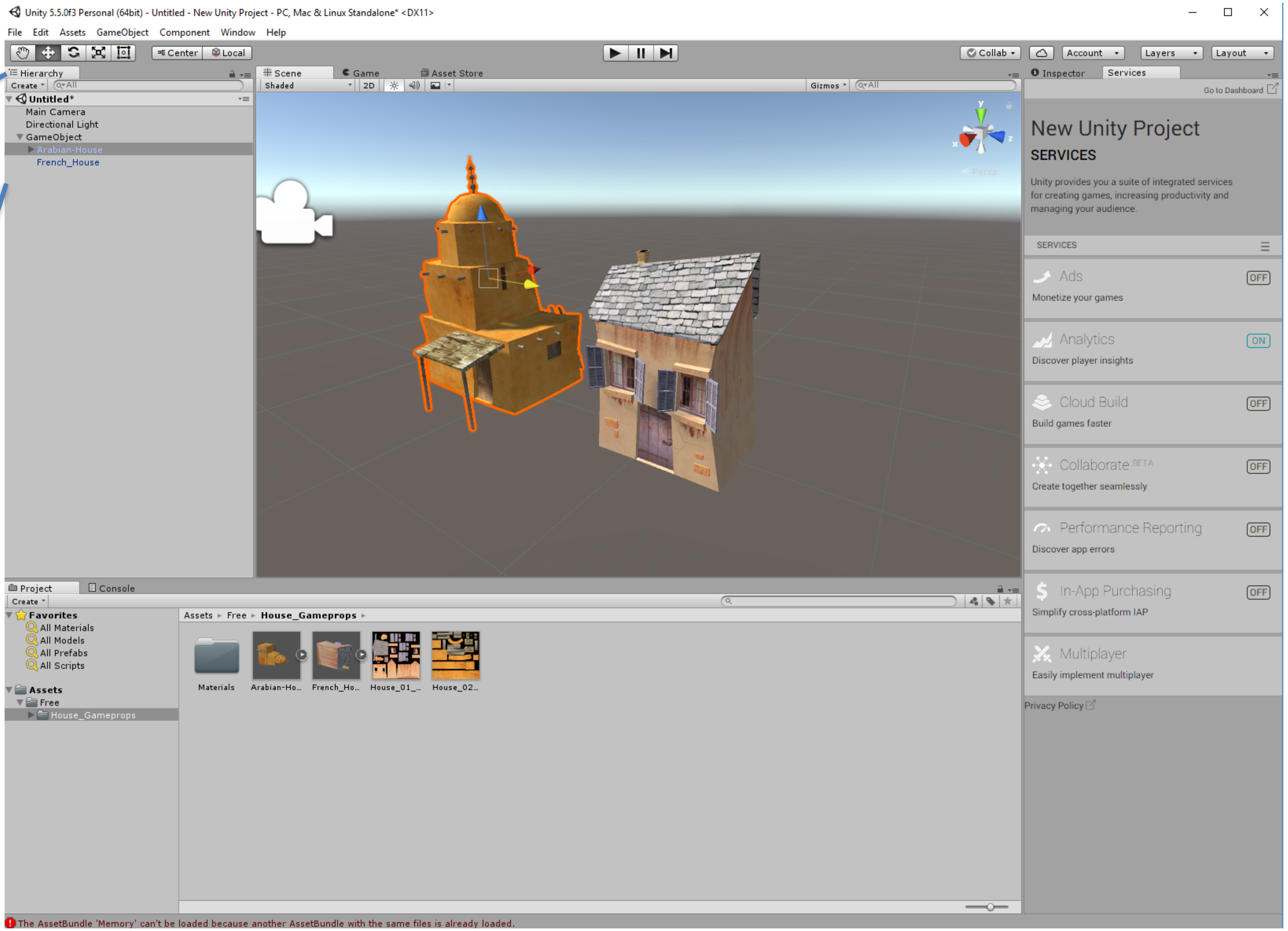
RENDER/GAME ENGINES

- Unity
- Unreal
- CryEngine
- Etc.

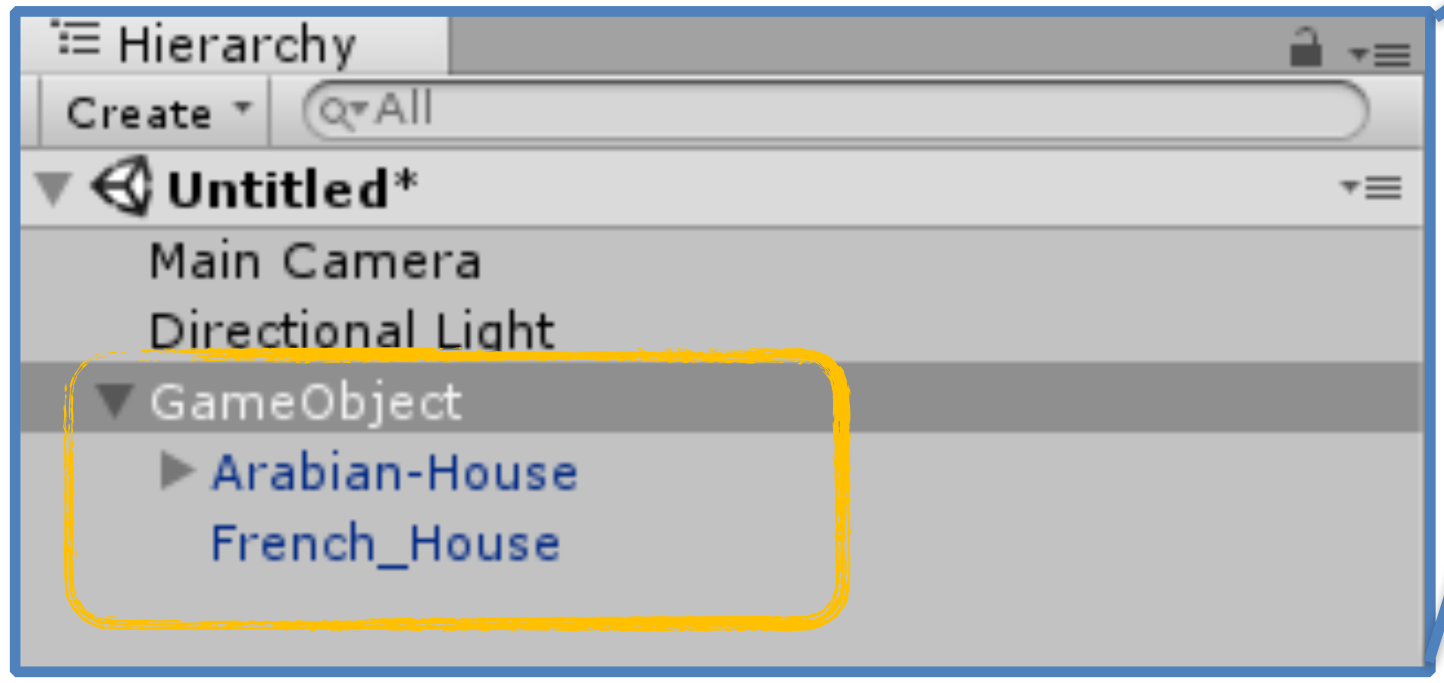


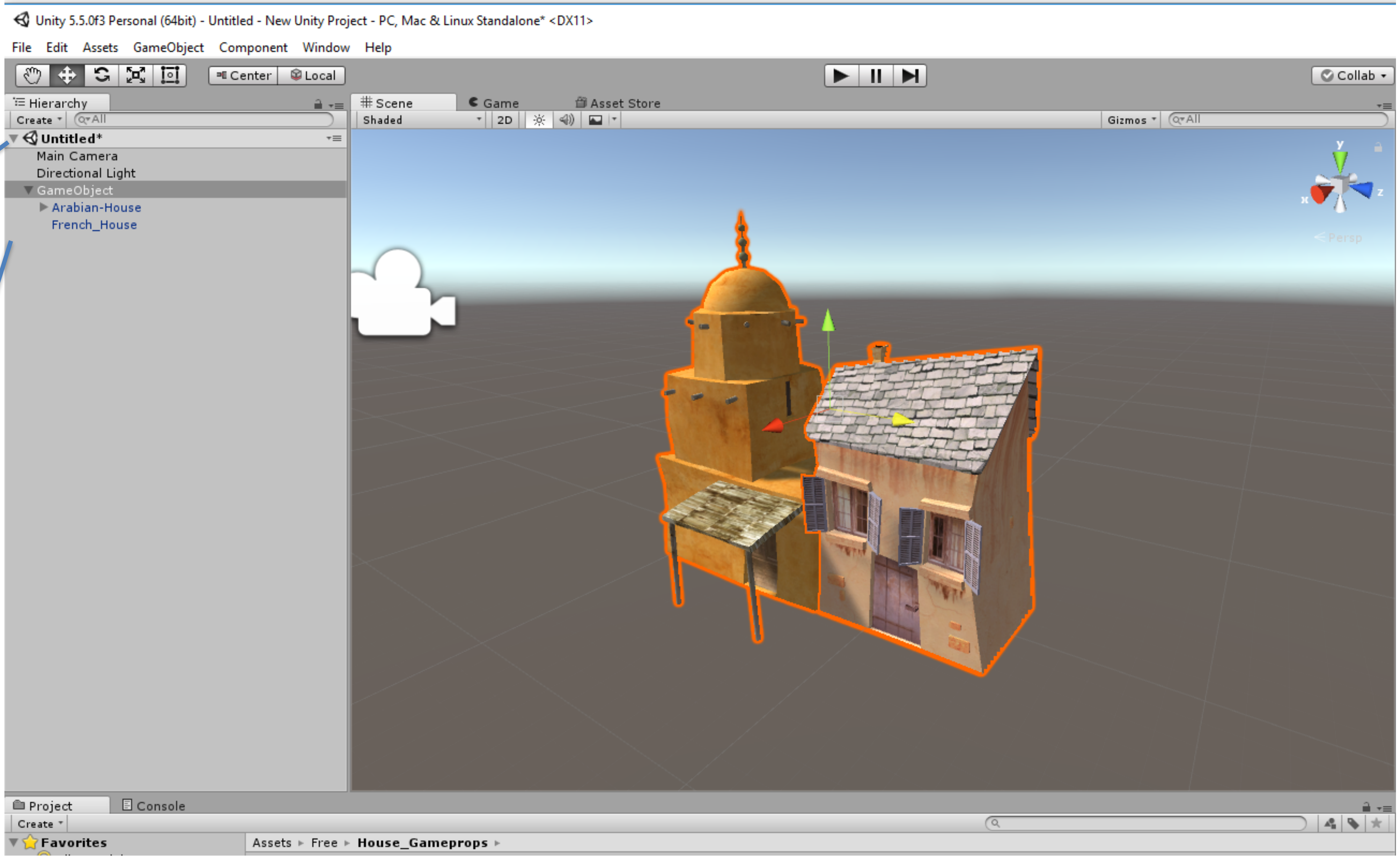
UNITY


Center
Local
Hierarchy
Create
Untitled*
Main Camera
Directional Light
GameObject
Arabian-House
French_House

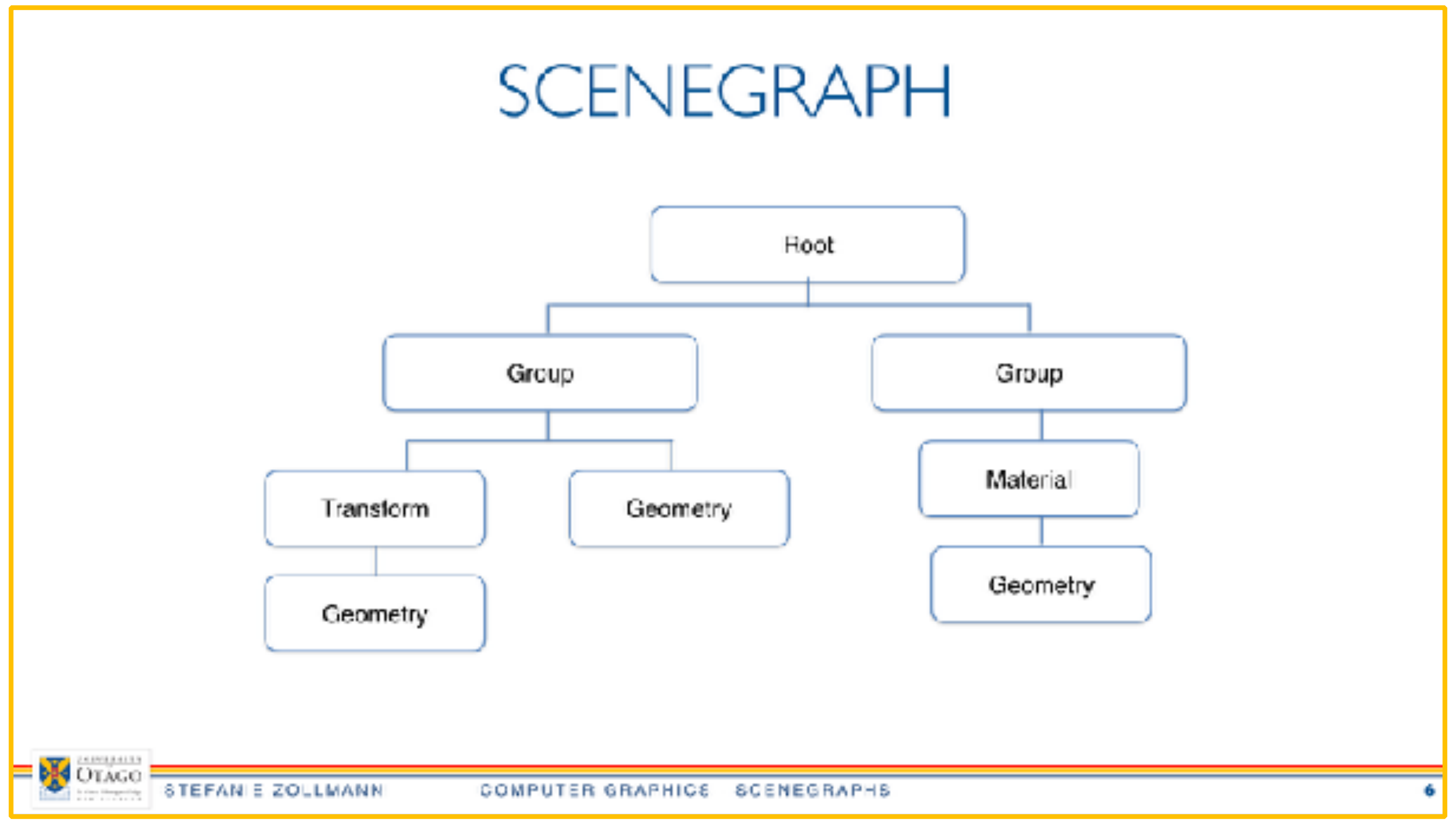

Unity 5.5.0f3 Personal (64bit) - Untitled - New Unity Project - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Scene Game Asset Store
Inspector Services
New Unity Project
SERVICES
Unity provides you a suite of integrated services for creating games, increasing productivity and managing your audience.
Ads
Monetize your games
Analytics
Discover player insights
Cloud Build
Build games faster
Collaborate
Create together seamlessly
Performance Reporting
Discover app errors
In-App Purchasing
Simplify cross-platform IAP
Multiplayer
Easily implement multiplayer
Privacy Policy
Project Console
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Free
House_Gameprops
Materials
The AssetBundle 'Memory' can't be loaded because another AssetBundle with the same files is already loaded.

UNITY

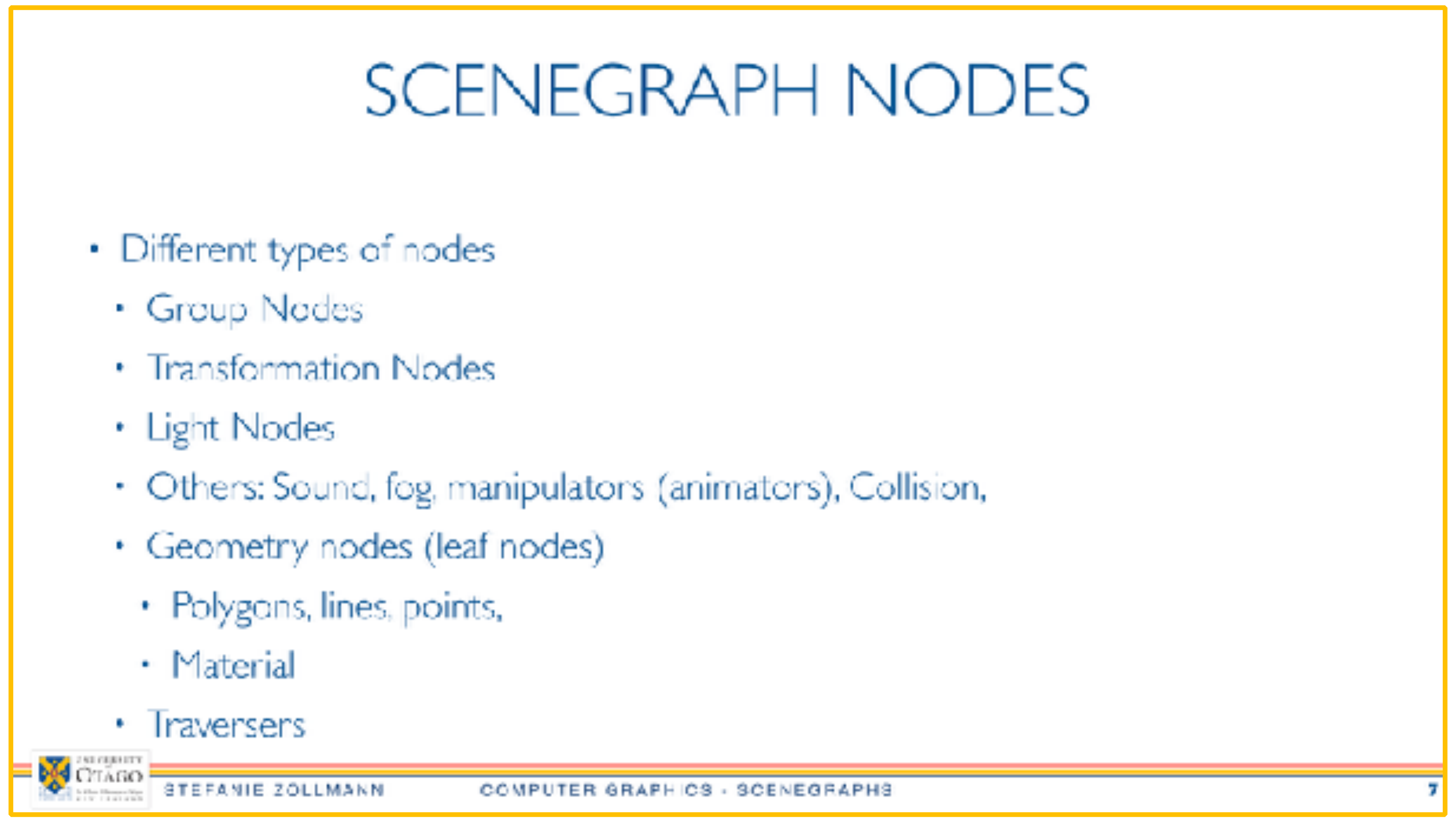

Hierarchy
Create
All
Untitled*
Main Camera
Directional Light
GameObject
Arabian-House
French_House


Unity 5.5.0f3 Personal (64bit) - Untitled - New Unity Project - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center
Local
Collab
Hierarchy
Create
Untitled*
Main Camera
Directional Light
GameObject
Arabian-House
French_House
Scene
Game
Asset Store
Shaded
2D
Gizmos
Project
Console
Favorites
Assets ▸ Free ▸ House_Gameprops ▸

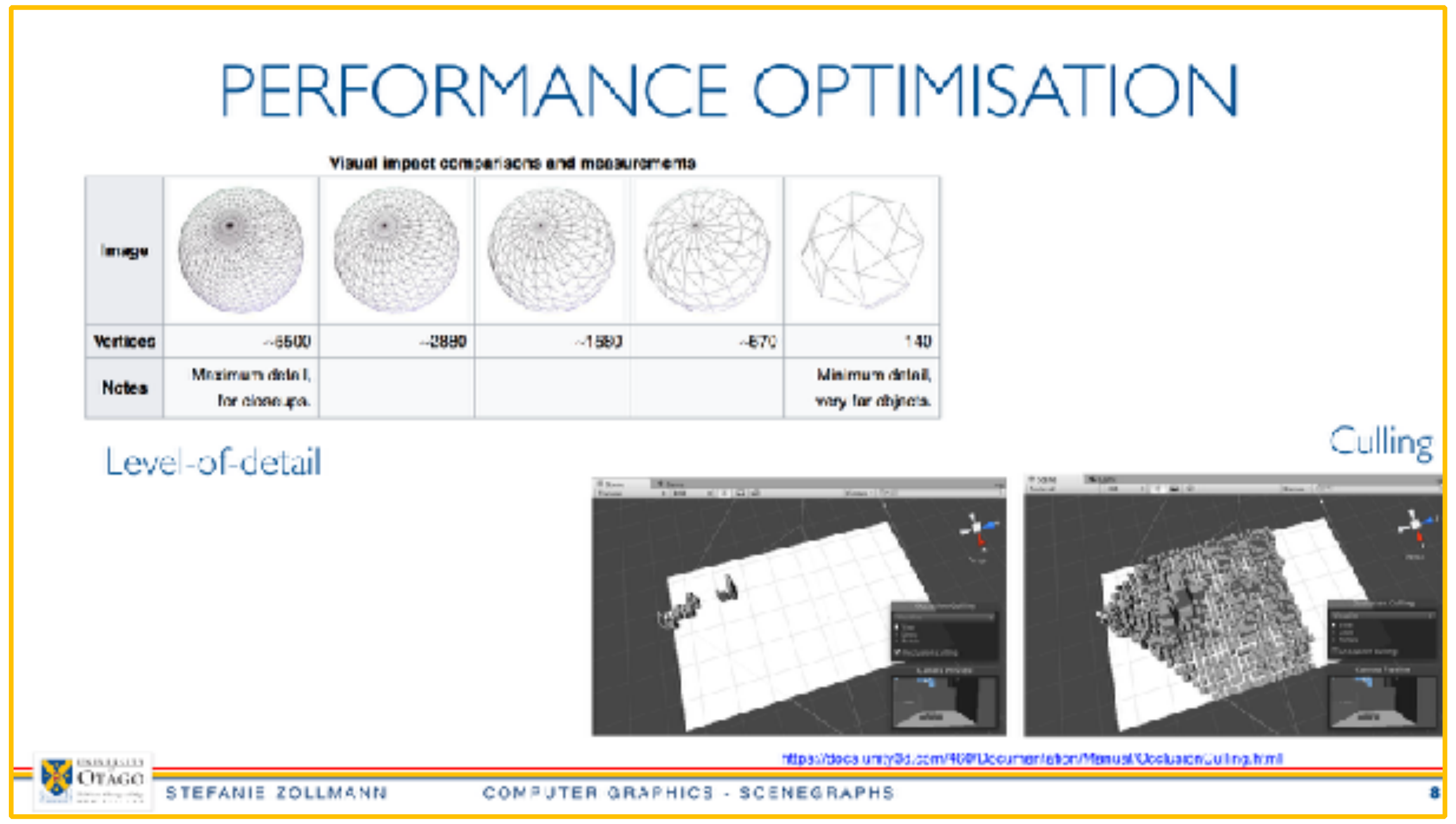
SUMMARY



SCENEGRAPH



NODES



OPTIMISATIONS

Thank You!

For more material visit

http://www.cs.otago.ac.nz/cosc342/