CMSC427
Scene graphs

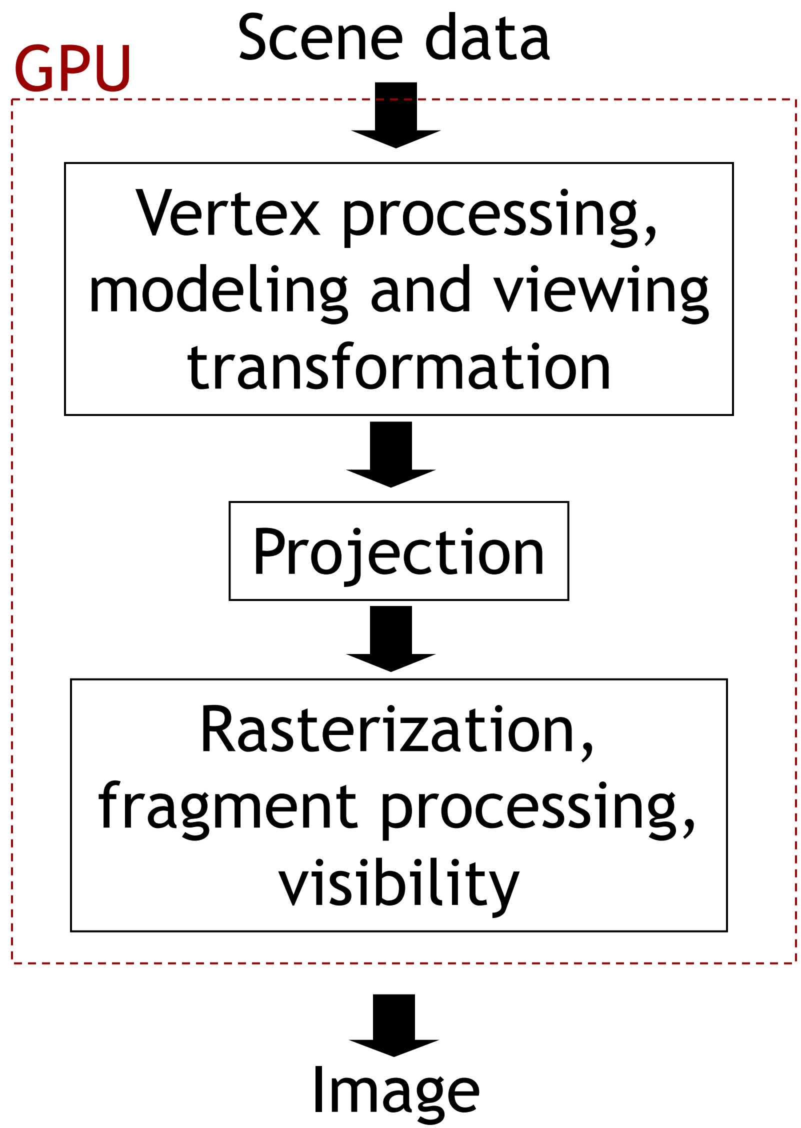
Credit: slides from Dr. Zwicker

Today

Scene graphs & hierarchies

- Introduction
- Scene graph data structures
- Rendering scene graphs
- Level-of-detail
- Culling

So far: rendering pipeline



System architecture

Interactive applications

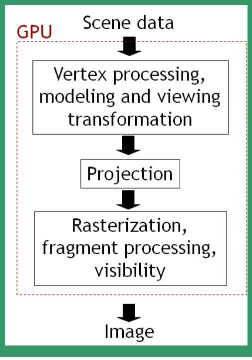
- Games, virtual reality, visualization

Rendering engine, scene graph API

- Implement functionality commonly required in applications
- Back-ends for different low-level APIs

Low-level graphics API

- Interface to graphics hardware



System architecture

Interactive applications

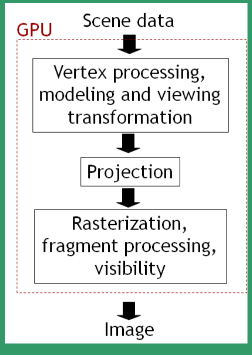
- Thousands

Rendering engine, scene graph API

- No broadly accepted standards
- Java3D, Ogre3D, OpenSceneGraph, jrtr, …

Low-level graphics API

- Highly standardized
- OpenGL (jogl), Direct3D



Scene graph APIs

Common functionality

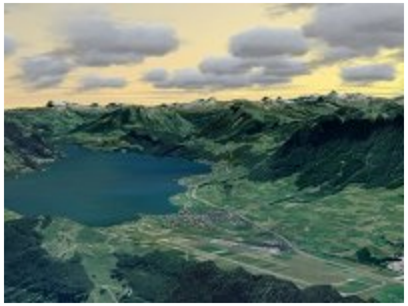
- Resource management
  - Content I/O (geometry, textures, materials, animation sequences)
  - Memory management
- High level scene representation
  - Scene graph
- Rendering
  - Efficiency
  - Advanced shading (materials, shadows, etc.)

Game engines

- Networking, physics, AI, etc.
  http://en.wikipedia.org/wiki/Game_engine

Scene graph APIs

- APIs focus on different clients/applications
- Java3D (https://java3d.dev.java.net/)
  - Simple, easy to use, web-based applications
- OpenSceneGraph (www.openscenegraph.org)
  - Scientific visualization, virtual reality, GIS (geographic information systems)
- Ogre3D (http://www.ogre3d.org/)
  - Games, high-performance rendering
- jrtr
  - Under development…



Common functionality: Ogre3D

_ FontManager (Ogre) ProgressiveMesh (Ogre) _ConfigOption (Ogre) FontPtr (Ogre) ProgressiveMesh::PMFaceVertex (Ogre) _finddata_t FrameEvent (Ogre) ProgressiveMesh::PMTriangle (Ogre) A FrameListener (Ogre) ProgressiveMesh::PMVertex (Ogre) AlignedAllocator (Ogre) FrameTimeControllerValue (Ogre) ProgressiveMesh::PMWorkingData (Ogre) AlignedAllocator::rebind (Ogre) FreeImageCodec (Ogre) Q AlignedMemory (Ogre) Frustum (Ogre) Quaternion (Ogre) Angle (Ogre) G QueuedRenderableCollection (Ogre) AnimableObject (Ogre) GpuConstantDefinition (Ogre) QueuedRenderableCollection::DepthSortDescendingLess (Ogre) AnimableValue (Ogre) GpuLogicalBufferStruct (Ogre) QueuedRenderableCollection::PassGroupLess (Ogre) Animation (Ogre) GpuLogicalIndexUse (Ogre) QueuedRenderableCollection::RadixSortFunctorDistance (Ogre) AnimationControllerFunction (Ogre) GpuNamedConstants (Ogre) QueuedRenderableCollection::RadixSortFunctorPass (Ogre) AnimationState (Ogre) GpuProgram (Ogre) QueuedRenderableVisitor (Ogre) AnimationStateControllerValue (Ogre) GpuProgram::CmdMorph (Ogre) R AnimationStateSet (Ogre) GpuProgram::CmdPose (Ogre) Radian (Ogre) AnimationTrack (Ogre) GpuProgram::CmdSkeletal (Ogre) RadixSort (Ogre) Any (Ogre) GpuProgram::CmdSyntax (Ogre) RadixSort::SortEntry (Ogre) Any::holder (Ogre) GpuProgram::CmdType (Ogre) Ray (Ogre) Any::placeholder (Ogre) GpuProgram::CmdVTF (Ogre) RaySceneQuery (Ogre) AnyNumeric (Ogre) GpuProgramManager (Ogre) RaySceneQueryListener (Ogre) AnyNumeric::numholder (Ogre) GpuProgramParameters (Ogre) RaySceneQueryResultEntry (Ogre) AnyNumeric::numplaceholder (Ogre) GpuProgramParameters::AutoConstantDefinition (Ogre) Rectangle (Ogre) Archive (Ogre) GpuProgramParameters::AutoConstantEntry (Ogre) Rectangle2D (Ogre) ArchiveFactory (Ogre) GpuProgramPtr (Ogre) RegionSceneQuery (Ogre) ArchiveManager (Ogre) GpuProgramUsage (Ogre) Renderable (Ogre) AutoParamDataSource (Ogre) H RenderablePass (Ogre) AxisAlignedBox (Ogre) HardwareBuffer (Ogre) RenderingAPIException (Ogre) AxisAlignedBoxSceneQuery (Ogre) HardwareBufferLicensee (Ogre) RenderOperation (Ogre) B HardwareBufferManager (Ogre) RenderPriorityGroup (Ogre) Billboard (Ogre) HardwareBufferManager::VertexBufferLicense (Ogre) RenderQueue (Ogre) BillboardChain (Ogre) HardwareIndexBuffer (Ogre) RenderQueue::RenderableListener (Ogre) BillboardChain::ChainSegment (Ogre) HardwareIndexBufferSharedPtr (Ogre) RenderQueueGroup (Ogre) BillboardChain::Element (Ogre) HardwareOcclusionQuery (Ogre) RenderQueueInvocation (Ogre) BillboardChainFactory (Ogre) HardwarePixelBuffer (Ogre) RenderQueueInvocationSequence (Ogre) BillboardParticleRenderer (Ogre) HardwarePixelBufferSharedPtr (Ogre) RenderQueueListener (Ogre) BillboardParticleRenderer::CmdAccurateFacing (Ogre) HardwareVertexBuffer (Ogre) RenderSystem (Ogre) BillboardParticleRenderer::CmdBillboardOrigin (Ogre) HardwareVertexBufferSharedPtr (Ogre) RenderSystem::Listener (Ogre) BillboardParticleRenderer::CmdBillboardRotationType (Ogre) HighLevelGpuProgram (Ogre) RenderSystemCapabilities (Ogre) BillboardParticleRenderer::CmdBillboardType (Ogre) HighLevelGpuProgramFactory (Ogre) RenderTarget (Ogre) BillboardParticleRenderer::CmdCommonDirection (Ogre) HighLevelGpuProgramManager (Ogre) RenderTarget::FrameStats (Ogre) BillboardParticleRenderer::CmdCommonUpVector (Ogre) HighLevelGpuProgramPtr (Ogre) RenderTarget::Impl (Ogre) BillboardParticleRenderer::CmdPointRendering (Ogre) I RenderTargetEvent (Ogre) BillboardParticleRendererFactory (Ogre) IlluminationPass (Ogre) RenderTargetListener (Ogre) BillboardSet (Ogre) Image (Ogre) RenderTargetViewportEvent (Ogre) BillboardSet::SortByDirectionFunctor (Ogre) ImageCodec (Ogre) RenderTexture (Ogre) BillboardSet::SortByDistanceFunctor (Ogre) ImageCodec::ImageData (Ogre) RenderWindow (Ogre) BillboardSetFactory (Ogre) IndexData (Ogre) Resource (Ogre) Bitwise (Ogre) InstancedGeometry (Ogre) Resource::Listener (Ogre) Bone (Ogre) InstancedGeometry::BatchInstance (Ogre) ResourceBackgroundQueue (Ogre) BorderPanelOverlayElement (Ogre) InstancedGeometry::GeometryBucket (Ogre) ResourceBackgroundQueue::Listener (Ogre) BorderPanelOverlayElement::CellUV (Ogre) InstancedGeometry::InstancedObject (Ogre) ResourceBackgroundQueue::QueuedNotification (Ogre) BorderPanelOverlayElement::CmdBorderBottomLeftUV (Ogre) InstancedGeometry::LODBucket (Ogre) ResourceBackgroundQueue::Request (Ogre) BorderPanelOverlayElement::CmdBorderBottomRightUV (Ogre) InstancedGeometry::MaterialBucket (Ogre) ResourceGroupListener (Ogre) BorderPanelOverlayElement::CmdBorderBottomUV (Ogre) InstancedGeometry::OptimisedSubMeshGeometry (Ogre) ResourceGroupManager (Ogre) BorderPanelOverlayElement::CmdBorderLeftUV (Ogre) InstancedGeometry::QueuedGeometry (Ogre) ResourceGroupManager::ResourceDeclaration (Ogre) BorderPanelOverlayElement::CmdBorderMaterial (Ogre) InstancedGeometry::QueuedSubMesh (Ogre) ResourceGroupManager::ResourceGroup (Ogre) BorderPanelOverlayElement::CmdBorderRightUV (Ogre) InstancedGeometry::SubMeshLodGeometryLink (Ogre) ResourceGroupManager::ResourceLocation (Ogre) BorderPanelOverlayElement::CmdBorderSize (Ogre) InternalErrorException (Ogre) ResourceManager (Ogre) BorderPanelOverlayElement::CmdBorderTopLeftUV (Ogre) IntersectionSceneQuery (Ogre) RibbonTrail (Ogre) BorderPanelOverlayElement::CmdBorderTopRightUV (Ogre) IntersectionSceneQueryListener (Ogre) RibbonTrailFactory (Ogre) BorderPanelOverlayElement::CmdBorderTopUV (Ogre) IntersectionSceneQueryResult (Ogre) Root (Ogre) BorderPanelOverlayElementFactory (Ogre) InvalidParametersException (Ogre) RotationalSpline (Ogre) BorderRenderable (Ogre) InvalidStateException (Ogre) RuntimeAssertionException (Ogre) Box (Ogre) IOException (Ogre) S C ItemIdentityException (Ogre) ScaleControllerFunction (Ogre) Camera (Ogre) K SceneManager (Ogre) CmdAngle (Ogre::EmitterCommands) KeyFrame (Ogre) SceneManager::LightInfo (Ogre) CmdCaption (Ogre::OverlayElementCommands) L SceneManager::lightLess (Ogre) CmdColour (Ogre::EmitterCommands) LayerBlendModeEx (Ogre) SceneManager::lightsForShadowTextureLess (Ogre) CmdColourRangeEnd (Ogre::EmitterCommands) Light (Ogre) SceneManager::materialLess (Ogre) CmdColourRangeStart (Ogre::EmitterCommands) LightFactory (Ogre) SceneManager::MovableObjectCollection (Ogre) CmdDirection (Ogre::EmitterCommands) LinearResampler (Ogre) SceneManager::SceneMgrQueuedRenderableVisitor (Ogre) CmdDuration (Ogre::EmitterCommands) LinearResampler_Byte (Ogre) SceneManager::ShadowCasterSceneQueryListener (Ogre) CmdEmissionRate (Ogre::EmitterCommands) LinearResampler_Float32 (Ogre) SceneManager::SkyBoxGenParameters (Ogre) CmdEmittedEmitter (Ogre::EmitterCommands) LinkedSkeletonAnimationSource (Ogre) SceneManager::SkyDomeGenParameters (Ogre) CmdHeight (Ogre::OverlayElementCommands) LiSPSMShadowCameraSetup (Ogre) SceneManager::SkyPlaneGenParameters (Ogre) CmdHorizontalAlign (Ogre::OverlayElementCommands) Log (Ogre) SceneManagerEnumerator (Ogre) CmdLeft (Ogre::OverlayElementCommands) LogListener (Ogre) SceneManagerFactory (Ogre) CmdMaterial (Ogre::OverlayElementCommands) LogManager (Ogre) SceneManagerMetaData (Ogre) CmdMaxDuration (Ogre::EmitterCommands) M SceneNode (Ogre) CmdMaxRepeatDelay (Ogre::EmitterCommands) ManualObject (Ogre) SceneQuery (Ogre) CmdMaxTTL (Ogre::EmitterCommands) ManualObject::ManualObjectSection (Ogre) SceneQuery::WorldFragment (Ogre) CmdMaxVelocity (Ogre::EmitterCommands) ManualObject::ManualObjectSectionShadowRenderable (Ogre) SceneQueryListener (Ogre) CmdMetricsMode (Ogre::OverlayElementCommands) ManualObject::TempVertex (Ogre) SceneQueryResult (Ogre) CmdMinDuration (Ogre::EmitterCommands) ManualObjectFactory (Ogre) ScriptLoader (Ogre) CmdMinRepeatDelay (Ogre::EmitterCommands) ManualResourceLoader (Ogre) Serializer (Ogre) CmdMinTTL (Ogre::EmitterCommands) MapIterator (Ogre) ShadowCameraSetup (Ogre) CmdMinVelocity (Ogre::EmitterCommands) Material (Ogre) ShadowCaster (Ogre) CmdName (Ogre::EmitterCommands) MaterialManager (Ogre) ShadowListener (Ogre) CmdPosition (Ogre::EmitterCommands) MaterialPtr (Ogre) ShadowRenderable (Ogre) CmdRepeatDelay (Ogre::EmitterCommands) MaterialScriptCompiler (Ogre) ShadowTextureConfig (Ogre) CmdTop (Ogre::OverlayElementCommands) MaterialScriptCompiler::MaterialScriptContext (Ogre) ShadowTextureManager (Ogre) CmdTTL (Ogre::EmitterCommands) MaterialScriptCompiler::MaterialScriptProgramDefinition (Ogre) ShadowVolumeExtrudeProgram (Ogre) CmdVelocity (Ogre::EmitterCommands) MaterialScriptContext (Ogre) SharedPtr (Ogre) CmdVerticalAlign (Ogre::OverlayElementCommands) MaterialScriptProgramDefinition (Ogre) SimpleRenderable (Ogre) CmdVisible (Ogre::OverlayElementCommands) MaterialSerializer (Ogre) SimpleSpline (Ogre) CmdWidth (Ogre::OverlayElementCommands) Math (Ogre) Singleton (Ogre) Codec (Ogre) Matrix3 (Ogre) Skeleton (Ogre) Codec::CodecData (Ogre) Matrix4 (Ogre) SkeletonInstance (Ogre) ColourValue (Ogre) MemoryDataStream (Ogre) SkeletonManager (Ogre) Compiler2Pass (Ogre) MemoryManager (Ogre) SkeletonPtr (Ogre) Compiler2Pass::LexemeTokenDef (Ogre) Mesh (Ogre) SkeletonSerializer (Ogre) Compiler2Pass::TokenInst (Ogre) MeshLodUsage (Ogre) Sphere (Ogre) Compiler2Pass::TokenRule (Ogre) MeshManager (Ogre) SphereSceneQuery (Ogre) Compiler2Pass::TokenState (Ogre) MeshManager::MeshBuildParams (Ogre) StaticFaceGroup (Ogre) CompositionPass (Ogre) MeshPtr (Ogre) StaticGeometry (Ogre) CompositionTargetPass (Ogre) MeshSerializer (Ogre) StaticGeometry::GeometryBucket (Ogre) CompositionTechnique (Ogre) MeshSerializerImpl (Ogre) StaticGeometry::LODBucket (Ogre) CompositionTechnique::TextureDefinition (Ogre) MeshSerializerImpl_v1_1 (Ogre) StaticGeometry::MaterialBucket (Ogre) Compositor (Ogre) MeshSerializerImpl_v1_2 (Ogre) StaticGeometry::OptimisedSubMeshGeometry (Ogre) CompositorChain (Ogre) MeshSerializerImpl_v1_3 (Ogre) StaticGeometry::QueuedGeometry (Ogre) CompositorChain::RQListener (Ogre) MovableObject (Ogre) StaticGeometry::QueuedSubMesh (Ogre) CompositorInstance (Ogre) MovableObject::Listener (Ogre) StaticGeometry::Region (Ogre) CompositorInstance::Listener (Ogre) MovableObjectFactory (Ogre) StaticGeometry::Region::RegionShadowRenderable (Ogre) CompositorInstance::RenderSystemOperation (Ogre) MovablePlane (Ogre) StaticGeometry::SubMeshLodGeometryLink (Ogre) CompositorInstance::TargetOperation (Ogre) MultiRenderTarget (Ogre) StringConverter (Ogre) CompositorManager (Ogre) N StringInterface (Ogre) CompositorPtr (Ogre) NearestResampler (Ogre) StringUtil (Ogre) CompositorScriptCompiler (Ogre) Node (Ogre) SubEntity (Ogre) CompositorScriptCompiler::CompositorScriptContext (Ogre) Node::Listener (Ogre) SubMesh (Ogre) CompositorSerializer (Ogre) NodeAnimationTrack (Ogre) T ConfigDialog (Ogre) NodeAnimationTrack::Splines (Ogre) TagPoint (Ogre) ConfigFile (Ogre) NumericAnimationTrack (Ogre) Technique (Ogre) ConstMapIterator (Ogre) NumericKeyFrame (Ogre) TempBlendedBufferInfo (Ogre) ConstVectorIterator (Ogre) NumericSolver (Ogre) TexCoordModifierControllerValue (Ogre) Controller (Ogre) O TextAreaOverlayElement (Ogre) ControllerFunction (Ogre) OptimisedUtil (Ogre) TextAreaOverlayElement::CmdAlignment (Ogre) ControllerManager (Ogre) Overlay (Ogre) TextAreaOverlayElement::CmdCaption (Ogre) ControllerValue (Ogre) OverlayContainer (Ogre) TextAreaOverlayElement::CmdCharHeight (Ogre) ConvexBody (Ogre) OverlayElement (Ogre) TextAreaOverlayElement::CmdColour (Ogre) D OverlayElementFactory (Ogre) TextAreaOverlayElement::CmdColourBottom (Ogre) DataStream (Ogre) OverlayManager (Ogre) TextAreaOverlayElement::CmdColourTop (Ogre) DDSCodec (Ogre) P TextAreaOverlayElement::CmdFontName (Ogre) DefaultAxisAlignedBoxSceneQuery (Ogre) PanelOverlayElement (Ogre) TextAreaOverlayElement::CmdSpaceWidth (Ogre) DefaultHardwareBufferManager (Ogre) PanelOverlayElement::CmdTiling (Ogre) TextAreaOverlayElementFactory (Ogre) DefaultHardwareIndexBuffer (Ogre) PanelOverlayElement::CmdTransparent (Ogre) Texture (Ogre) DefaultHardwareVertexBuffer (Ogre) PanelOverlayElement::CmdUVCoords (Ogre) TextureFrameControllerValue (Ogre) DefaultIntersectionSceneQuery (Ogre) PanelOverlayElementFactory (Ogre) TextureManager (Ogre) DefaultPlaneBoundedVolumeListSceneQuery (Ogre) ParamCommand (Ogre) TexturePtr (Ogre) DefaultRaySceneQuery (Ogre) ParamDictionary (Ogre) TextureUnitState (Ogre) DefaultSceneManager (Ogre) ParameterDef (Ogre) TextureUnitState::TextureEffect (Ogre) DefaultSceneManagerFactory (Ogre) Particle (Ogre) TextureUnitState::UVWAddressingMode (Ogre) DefaultShadowCameraSetup (Ogre) ParticleAffector (Ogre) TimeIndex (Ogre) DefaultSphereSceneQuery (Ogre) ParticleAffectorFactory (Ogre) Timer (Ogre) Degree (Ogre) ParticleEmitter (Ogre) TransformKeyFrame (Ogre) DynLib (Ogre) ParticleEmitterFactory (Ogre) TRect (Ogre) DynLibManager (Ogre) ParticleIterator (Ogre) U E ParticleSystem (Ogre) UnifiedHighLevelGpuProgram (Ogre) EdgeData (Ogre) ParticleSystem::CmdCull (Ogre) UnifiedHighLevelGpuProgram::CmdDelegate (Ogre) EdgeData::Edge (Ogre) ParticleSystem::CmdEmittedEmitterQuota (Ogre) UnifiedHighLevelGpuProgramFactory (Ogre) EdgeData::EdgeGroup (Ogre) ParticleSystem::CmdHeight (Ogre) UnimplementedException (Ogre) EdgeData::Triangle (Ogre) ParticleSystem::CmdIterationInterval (Ogre) UserDefinedObject (Ogre) EdgeListBuilder (Ogre) ParticleSystem::CmdLocalSpace (Ogre) UTFString (Ogre) EdgeListBuilder::CommonVertex (Ogre) ParticleSystem::CmdMaterial (Ogre) UTFString::_base_iterator (Ogre) EdgeListBuilder::Geometry (Ogre) ParticleSystem::CmdNonvisibleTimeout (Ogre) UTFString::_const_fwd_iterator (Ogre) EdgeListBuilder::geometryLess (Ogre) ParticleSystem::CmdQuota (Ogre) UTFString::_const_rev_iterator (Ogre) EdgeListBuilder::vectorLess (Ogre) ParticleSystem::CmdRenderer (Ogre) UTFString::_fwd_iterator (Ogre) Entity (Ogre) ParticleSystem::CmdSorted (Ogre) UTFString::_rev_iterator (Ogre) Entity::EntityShadowRenderable (Ogre) ParticleSystem::CmdWidth (Ogre) UTFString::invalid_data (Ogre) EntityFactory (Ogre) ParticleSystem::SortByDirectionFunctor (Ogre) V ErrorDialog (Ogre) ParticleSystem::SortByDistanceFunctor (Ogre) Vector2 (Ogre) Exception (Ogre) ParticleSystemFactory (Ogre) Vector3 (Ogre) ExceptionCodeType (Ogre) ParticleSystemManager (Ogre) Vector4 (Ogre) ExceptionFactory (Ogre) ParticleSystemRenderer (Ogre) VectorIterator (Ogre) ExternalTextureSource (Ogre) ParticleSystemRendererFactory (Ogre) VertexAnimationTrack (Ogre) ExternalTextureSource::CmdFPS (Ogre) ParticleVisualData (Ogre) VertexBoneAssignment_s (Ogre) ExternalTextureSource::CmdInputFileName (Ogre) Pass (Ogre) VertexBufferBinding (Ogre) ExternalTextureSource::CmdPlayMode (Ogre) Pass::HashFunc (Ogre) VertexCacheProfiler (Ogre) ExternalTextureSource::CmdTecPassState (Ogre) PassthroughControllerFunction (Ogre) VertexData (Ogre) ExternalTextureSourceManager (Ogre) PatchMesh (Ogre) VertexData::HardwareAnimationData (Ogre) F PatchMeshPtr (Ogre) VertexDeclaration (Ogre) FactoryObj (Ogre) PatchSurface (Ogre) VertexElement (Ogre) FileHandleDataStream (Ogre) PixelBox (Ogre) VertexMorphKeyFrame (Ogre) FileInfo (Ogre) PixelUtil (Ogre) VertexPoseKeyFrame (Ogre) FileNotFoundException (Ogre) Plane (Ogre) VertexPoseKeyFrame::PoseRef (Ogre) FileStreamDataStream (Ogre) PlaneBoundedVolume (Ogre) ViewPoint (Ogre) FileSystemArchive (Ogre) PlaneBoundedVolumeListSceneQuery (Ogre) Viewport (Ogre) FileSystemArchiveFactory (Ogre) PlaneOptimalShadowCameraSetup (Ogre) VisibleObjectsBoundsInfo (Ogre) FloatGpuParameterControllerValue (Ogre) PlatformInformation (Ogre) W FocusedShadowCameraSetup (Ogre) Plugin (Ogre) WaveformControllerFunction (Ogre) FocusedShadowCameraSetup::PointListBody (Ogre) Polygon (Ogre) WindowEventListener (Ogre) Font (Ogre) Pose (Ogre) WindowEventUtilities (Ogre) Font::CmdCodePoints (Ogre) PrefabFactory (Ogre) WireBoundingBox (Ogre) Font::CmdResolution (Ogre) Profile (Ogre) Z Font::CmdSize (Ogre) Profiler (Ogre) ZipArchive (Ogre) Font::CmdSource (Ogre) Profiler::ProfileFrame (Ogre) ZipArchiveFactory (Ogre) Font::CmdType (Ogre) Profiler::ProfileHistory (Ogre) ZipDataStream (Ogre) Font::GlyphInfo (Ogre) Profiler::ProfileInstance (Ogre)

Today

Scene graphs & hierarchies

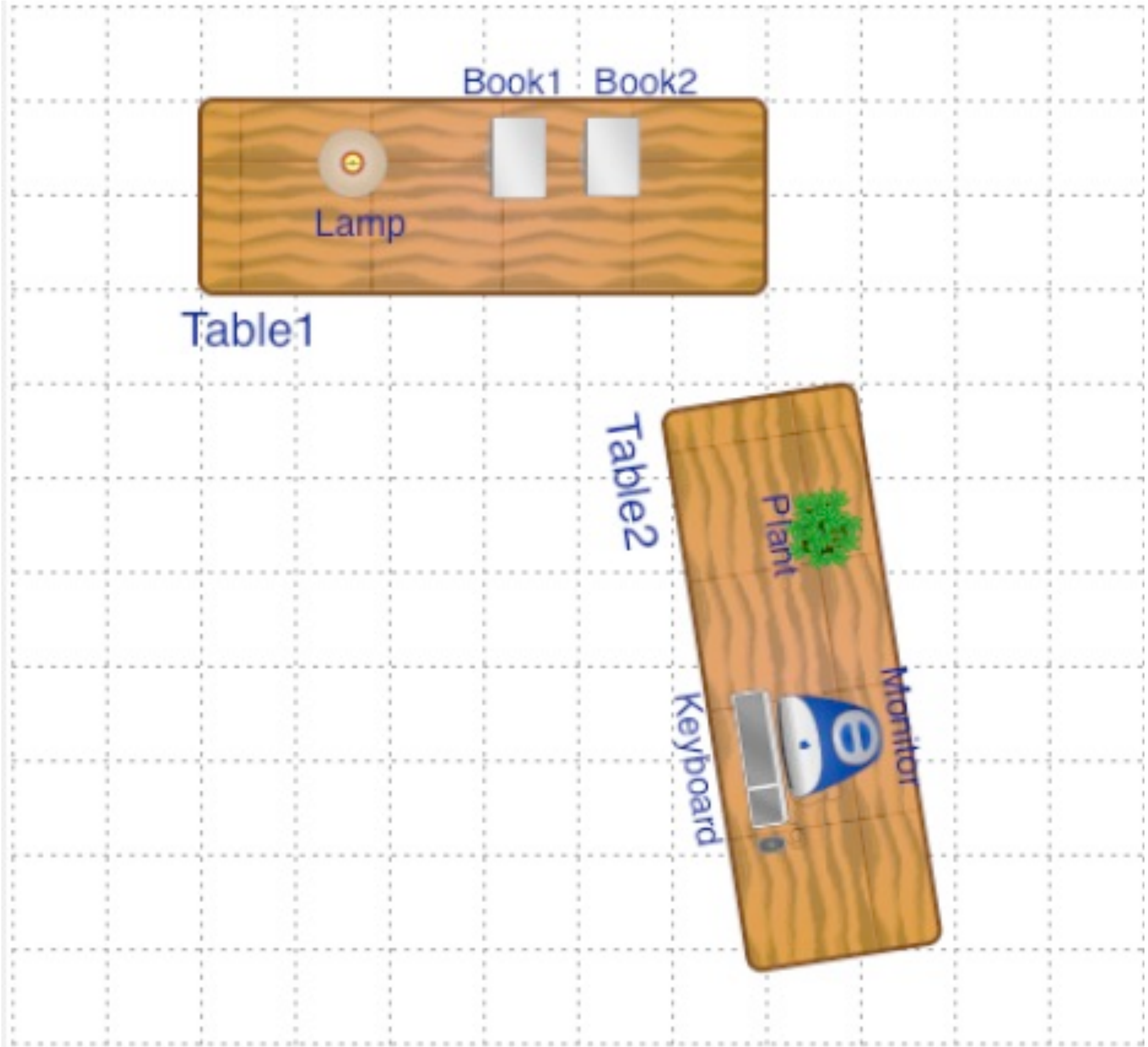
- Introduction
- Scene graph data structures
- Rendering scene graphs
- Level-of-detail
- Culling

Scene graphs

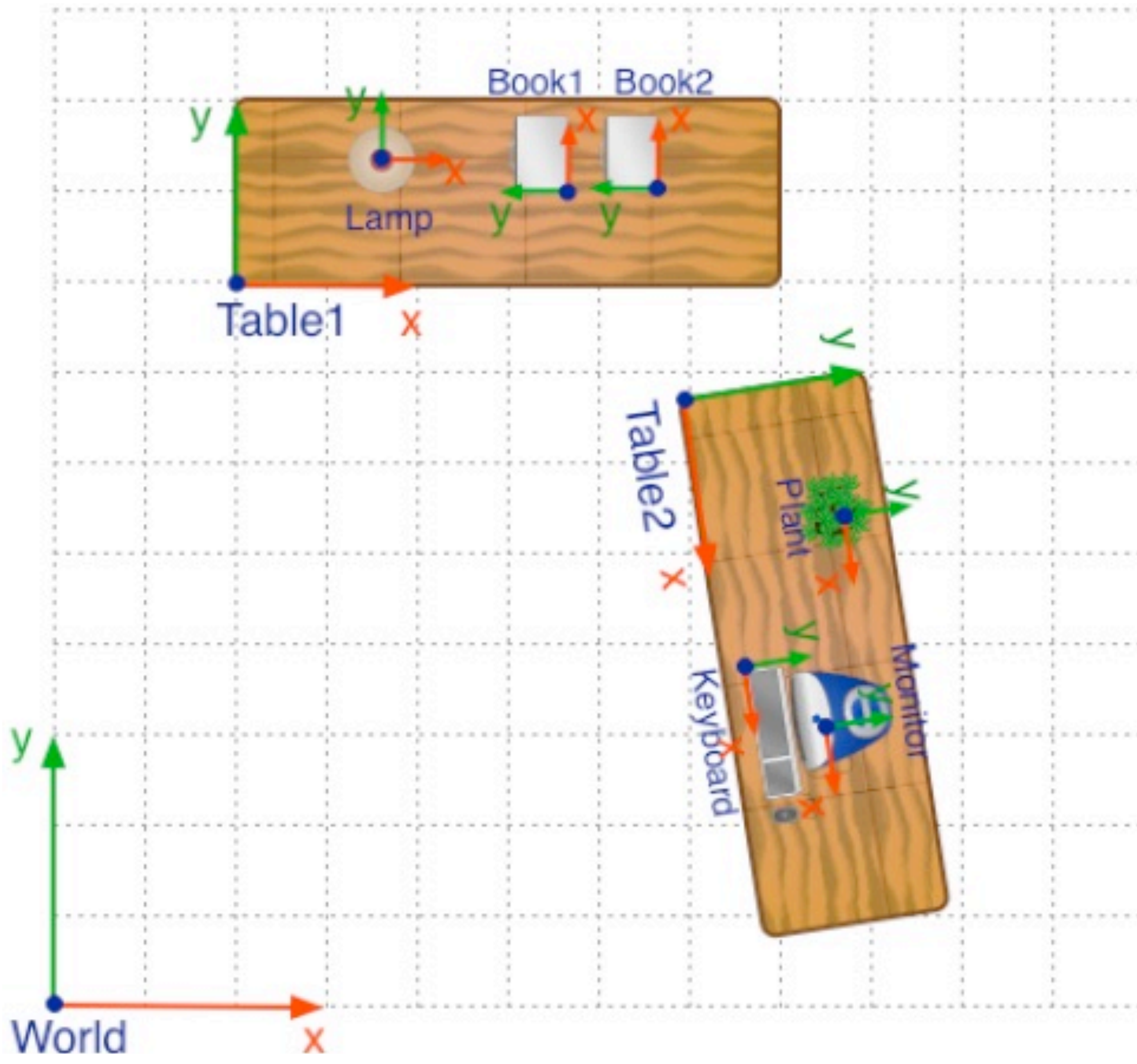
- Data structure for intuitive construction of 3D scenes
- So far we've seen object list –
  list of objects
- Ideas for improvement?
  - Disadvantages of list structure?
  - What functionality could an improved data structure support?

Sample scene

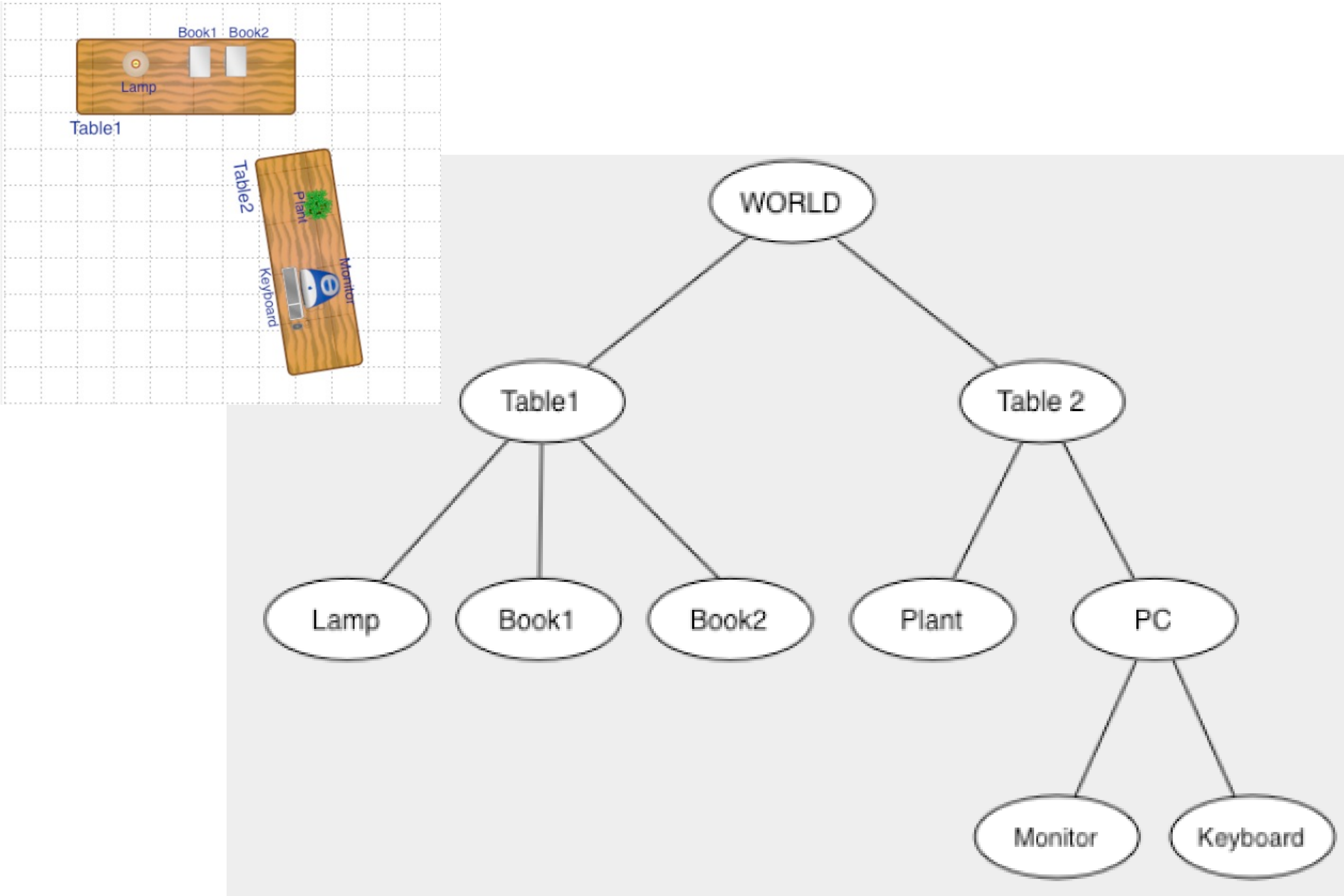
Top view



Top view with coordinates



Hierarchical organization



Data structure

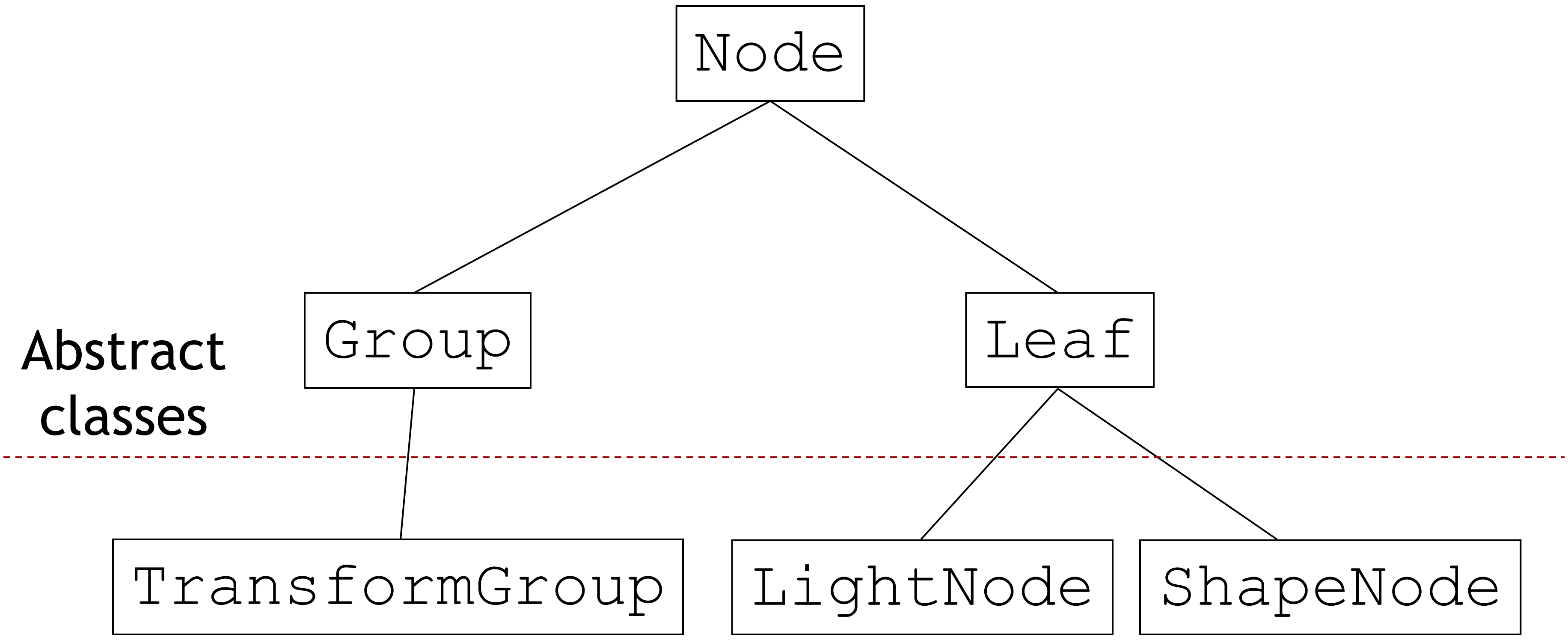
- Requirements
  - Collections of individual models/objects
  - Organized in groups
  - Related via hierarchical transformations (transformations apply to groups of objects)
- Use a tree or graph structure
- Each node has associated local coordinates
  - Its own coordinate system
- Different types of graph nodes
  - Geometry (shapes, objects)
  - Transformations
  - Lights
  - …

Class hierarchy

- Use a class hierarchy for graph nodes
  - All derived from same base class
  - Classes provide different functionality based on their intended use
- Many designs possible
  - Concepts are the same, details may differ
- Design driven by intended application
  - Games: optimize for speed
  - Large-scale visualization: optimize for memory requirements
  - Modeling system: optimize for editing flexibility

Class hierarchy

- Inspired by Java3D



Class hierarchy

`Node`

- Access to local-to-world coordinate transform

`Group`

- List of children
- Get, add, remove child

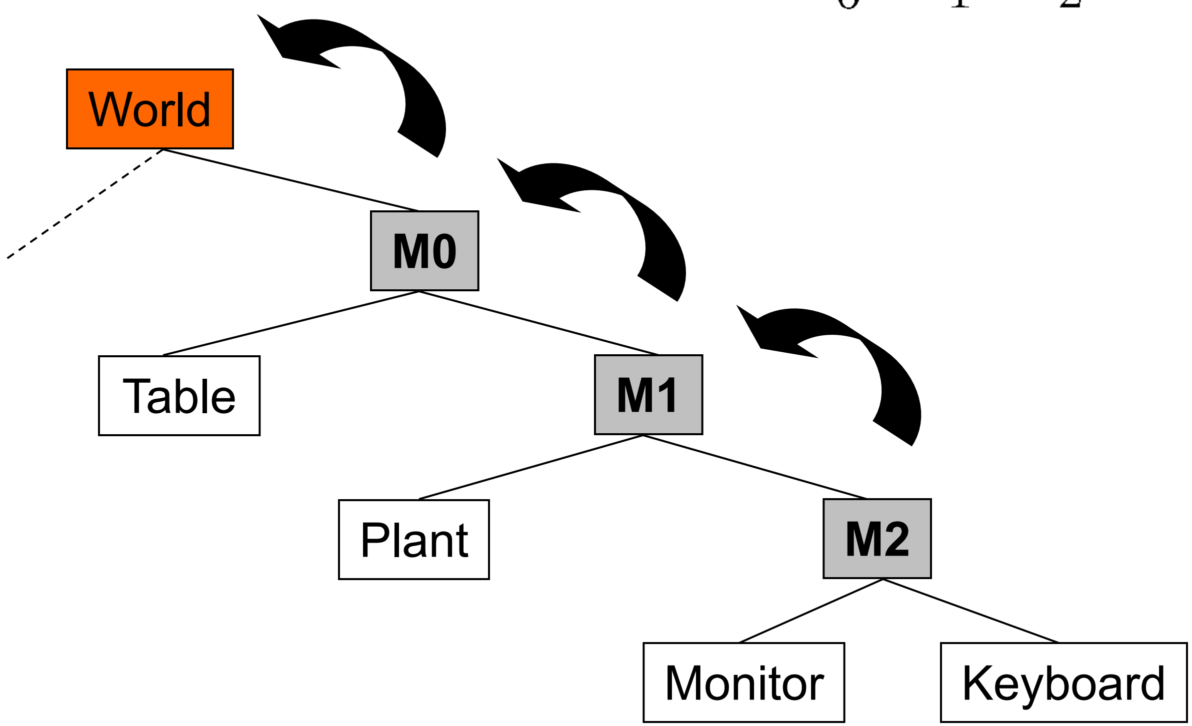
`Leaf`

- Node with no children

Class hierarchy

`TransformGroup`

- Stores additional transformation $\mathbf{M}$
- $\mathbf{M}$ applies to complete subtree below node
- Keyboard-to-world transform $\mathbf{M}_0\mathbf{M}_1\mathbf{M}_2$



Class hierarchy

Subclasses of `Leaf`

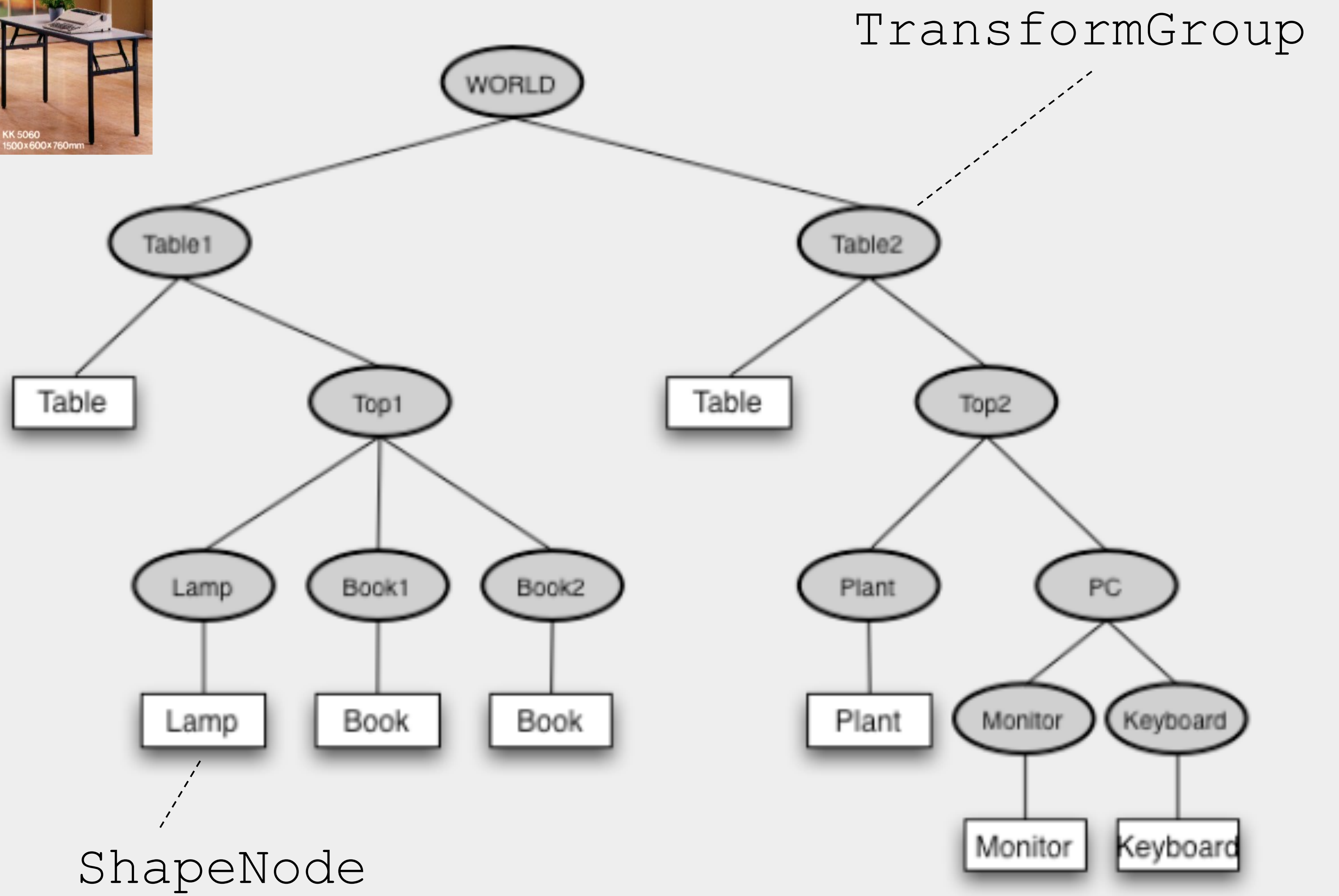
`Light`

- Stores (or references) a light source (position, direction, strength, etc.)

`ShapeNode`

- References a geometric object and associated properties, e.g., material

Scene graph for sample scene



Building sample scene

```
WORLD = new Group();
table1Trafo = new TransformGroup(…);  WORLD.addChild(table1Trafo);
table1 = makeTable(); table1Trafo.addChild(table1);
top1Trafo = new TransformGroup(…); table1Trafo.addChild(top1Trafo);

lampTrafo = new TransformGroup(…); top1Trafo.addChild(lampTrafo);
lamp = makeLamp(); lampTrafo.addChild(lamp);

book1Trafo = new TransformGroup(…); top1Trafo.addChild(book1Trafo);
book1 = makeBook(); book1Trafo.addChild(book1);

...
```

- More convenient to construct scenes than using linear list of objects
- Easier to manipulate
  - Transformations automatically apply to whole subtree

Modifying the scene

- Change tree structure
  - Add, delete, rearrange nodes
- Change node parameters
  - Transformation matrices, create animations
  - Shape of geometry data
  - Materials
- Define specific subclasses
  - Animation, triggered by timer events…

Modifying the scene

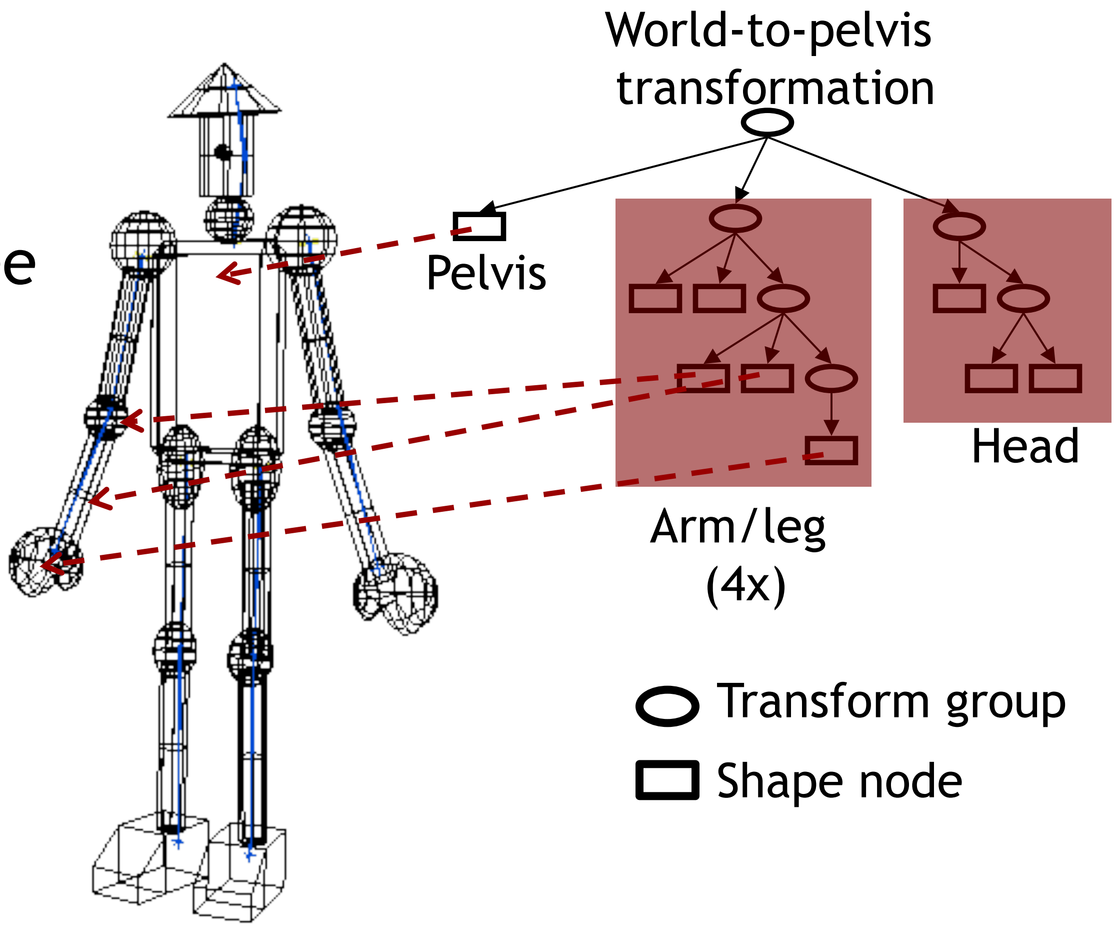
- Change a transform in the tree

```
table1Trafo.setRotationZ(23);
```

- Table rotates, everything on the table moves with it
- Allows easy animation
  - Build scene once at start of program
  - Update parameters to draw each frame
- Allows interactive model manipulation tools
  - Add objects relative to parent objects
  - E.g., book on table

Articulated character

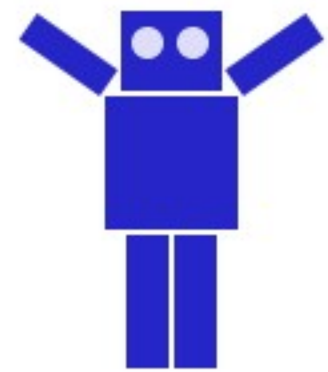
- Separate rigid parts
- Joint angles define transformation matrices
- Hierarchy
  - Rooted at pelvis
  - Neck, head subtree
  - Arms subtree
  - Legs subtree



Example in Processing

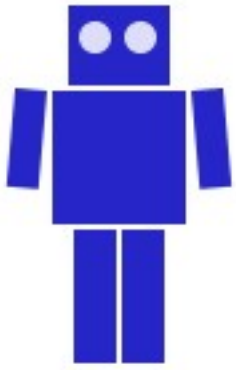
- Hierarchy of parts
- Matrix stack
  - pushMatrix()
  - popMatrix()

```
drawRobot()

pushMatrix()
   rotate()
   drawLeftArm()
popMatrix()

pushMatrix()
   rotate()
   drawRightArm()
popMatrix()
```



Example in Processing

```
void draw() {
  background(255);
  pushMatrix();
  translate(mouseX,mouseY);
  drawRobot();
  popMatrix();
}
```



```
void drawRobot() {
  noStroke();
  fill(38, 38, 200);
  rect(20, 0, 38, 30); // head
  rect(14, 32, 50, 50); // body
  drawLeftArm();
  drawRightArm();
  rect(22, 84, 16, 50); // left leg
  rect(40, 84, 16, 50); // right leg
  fill(222, 222, 249);
  ellipse(30, 12, 12, 12); // left eye
  ellipse(47, 12, 12, 12);  // right eye}.
```

```
void drawRightArm() {
  pushMatrix();
  translate(66, 32);
  rotate(radians(-armAngle));
  rect(0, 0, 12, 37); // right arm
  popMatrix();
}
```

Parameteric models

- Scene graph can implement parametric model
  - Encapsulate functionality of a model in separate class
- Parameters for
  - Position in world
  - Relationship between parts (e.g., joint angles)
  - Shape of individual parts (e.g., length of limbs)
- Degrees of freedom (DOFs)
  - Total number of float parameters in the model

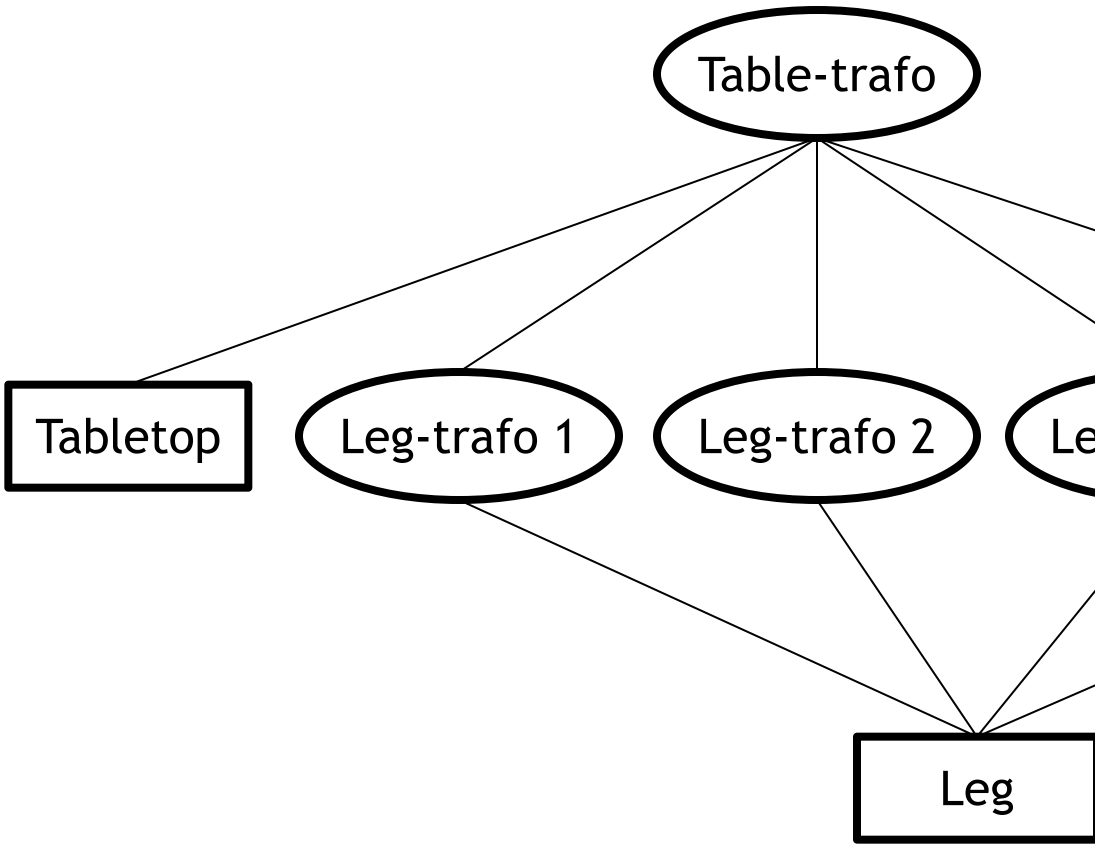
More node types

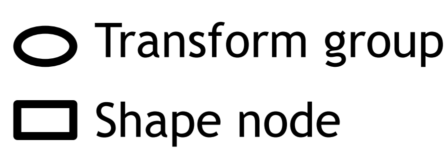
- Shape nodes
  - Cube, sphere, curved surface, etc…
- Nodes that control graph structure
  - Switch/Select: parameters choose whether or which children to enable, etc…
- Nodes that define other properties
  - Camera
  - Advantage: easy to specify location/orientation relative to some other scene part

Multiple instantiation

- A scene may have many copies of a model
- A model might use several copies of a part
- Multiple instantiation
  - One copy of node or subtree in memory
  - Reference (pointer) inserted as child of many parents
  - Object appears in scene multiple times, with different coordinates
- Not the same as instantiation in C++/Java terminology
- Scene is a directed acyclic graph (DAG), not a tree

Multiple instantiation





Multiple instantiation

Advantages

- Saves memory
- May save rendering time, depending on caching/optimization

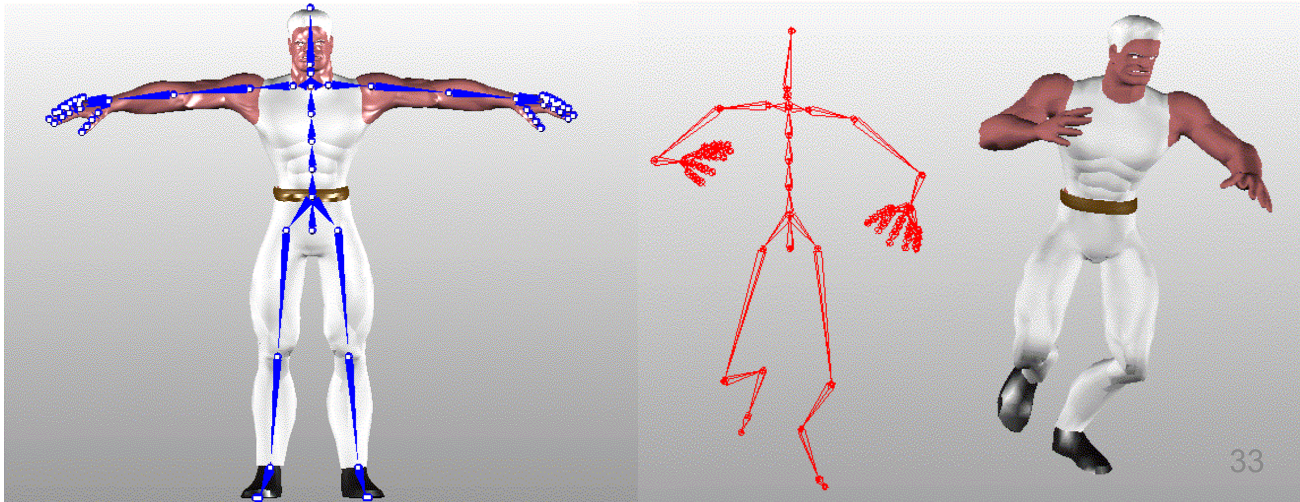
Possible disadvantage

- Change parameter once, affects all instances
  - Can be good or bad, depending on what you want
- Solution: let children inherit properties from parent
  - Different instances have different properties

Fancier operations

Given articulated character, i.e., skeleton, compute skin

- Shape nodes that compute surface across multiple joint nodes
- Nodes that change shape of geometry
- Extremely popular in games



Today

Scene graphs & hierarchies

- Introduction
- Scene graph data structures
- Rendering scene graphs
- Level-of-detail
- Culling

Basic rendering

- Traverse the tree recursively

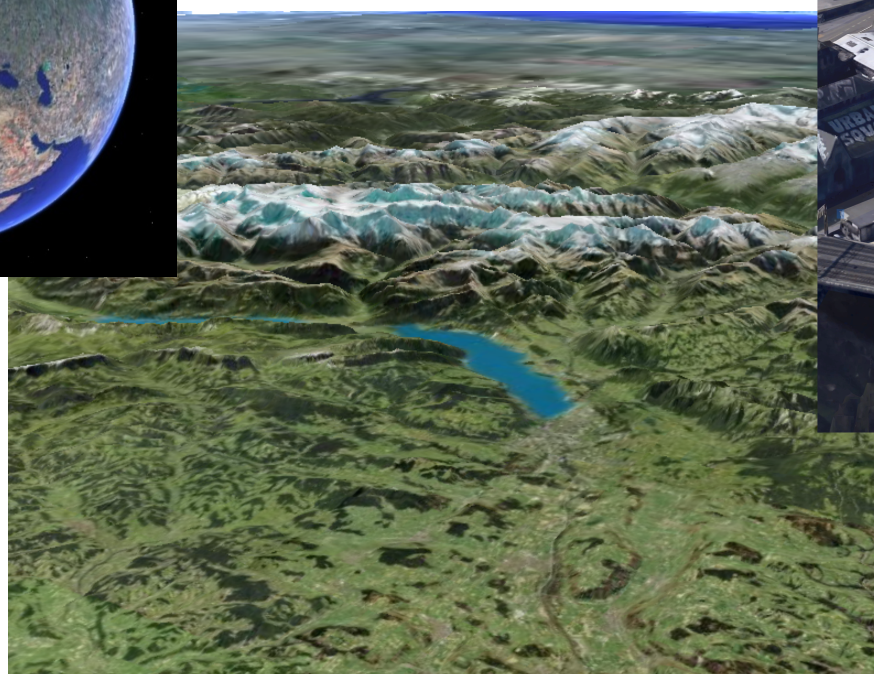
```
TransformGroup::draw(Matrix4 C) {
  C_new = C*M;   // matrix M is a class member
  for all children
    draw(C_new);
}

ShapeNode::draw(Matrix4 C) {
  setModelView(C);
  setMaterial(myMaterial);
  render(myObject);
}
```

Initiate rendering with root node, here called `world`

```
world->draw(IDENTITY);
```

Rendering

- How about rendering huge scenes?
- For example, Google Earth



http://earth.google.com

Problems

- Too much data to store in main memory
- Too slow to render all geometry in each frame

Performance optimization

- Culling
  - Quickly discard invisible parts of the scene
- Level-of-detail (LOD) techniques
  - Use lower quality for distant (small) objects
  - Lower quality of geometry
  - Lower quality of shading

Performance optimization

- Scene graph compilation
  - Efficient use of low-level API
  - Minimize state changes (switching shaders, etc.) in rendering pipeline
  - Render objects with similar properties (geometry, shaders, materials) in batches (groups)
- Data management
  - Manage memory hierarchy (network, disk, CPU RAM, GPU RAM)
  - Load data into desired level of memory hierarchy on-demand

Today

Scene graphs & hierarchies

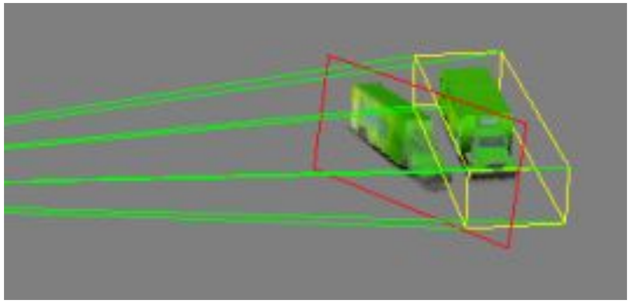
- Introduction
- Scene graph data structures
- Rendering scene graphs
- Level-of-detail
- Culling

Level-of-detail techniques

- "Adapt rendering technique to level of detail visible in the objects that are rendered"
  http://en.wikipedia.org/wiki/Level_of_detail
  - Level of detail depends on distance of objects from camera
- Simplest approach: don't draw small objects
- Use threshold
  - E.g., size in pixels
- Problem: popping artifacts

Impostors

- Replace objects by impostors
  - Textured planes representing the objects
  - Faster to render than original geometry
- Impostor generation
  - Exploits frame-to-frame coherence
  - Dynamic: generate periodically, reuse for a few frames

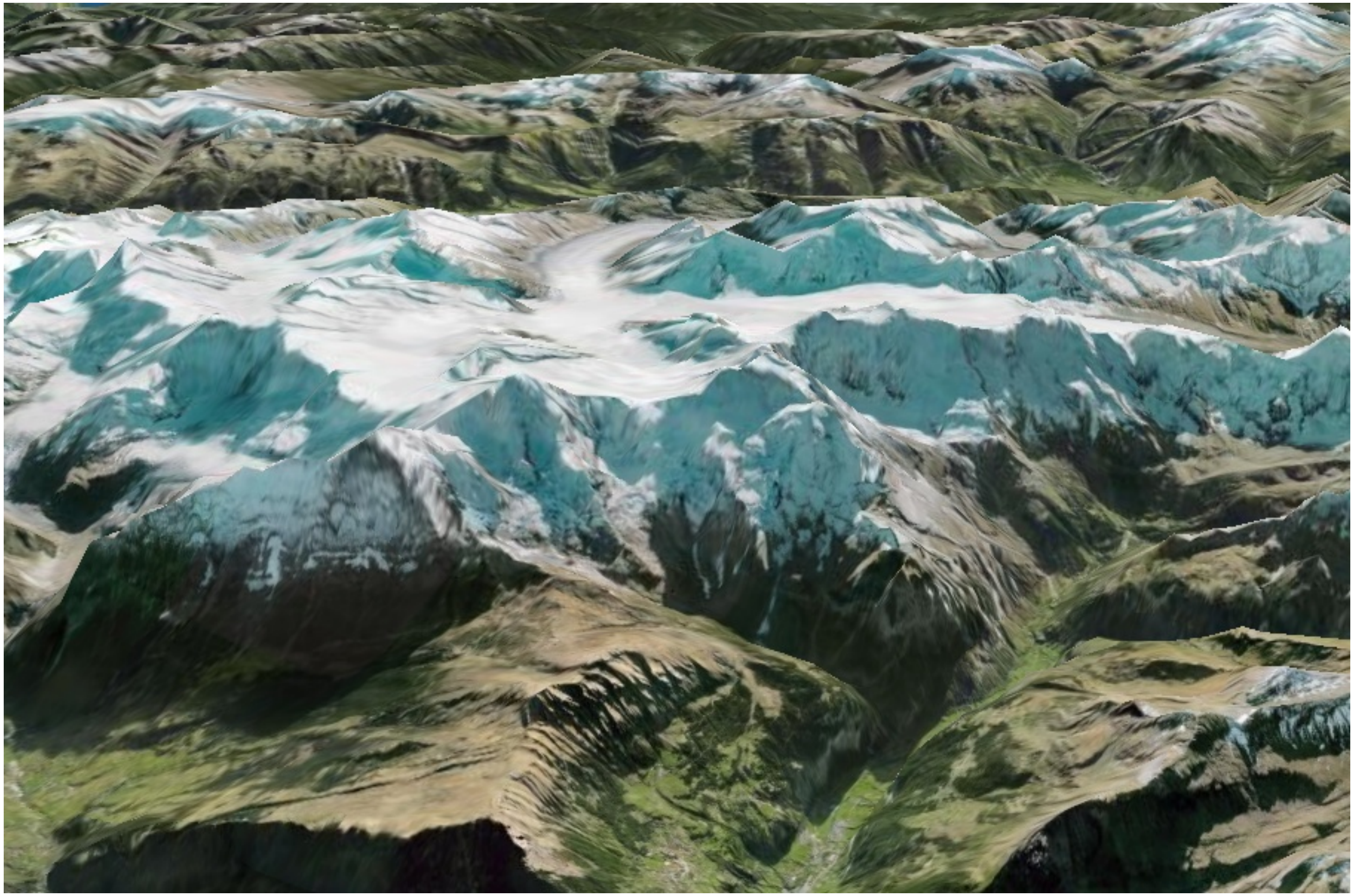


Dynamic impostor generation

Original vs. impostor

Geometric LOD

- Use fewer triangles for objects that are further away from the viewer



Shading LOD

- Use simpler shader for objects that are further away

With normal maps/
bump mapping

Without normal maps/
bump mapping

Today

Scene graphs & hierarchies

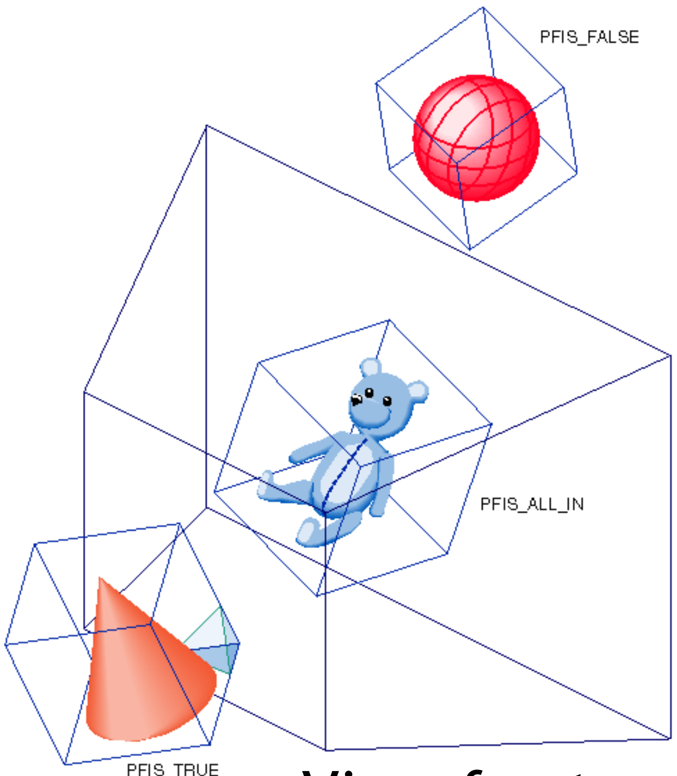
- Introduction
- Scene graph data structures
- Rendering scene graphs
- Level-of-detail
- Culling

Culling

- "Don't attempt to draw objects that are not visible"
- Essential for interactive performance with large scenes
- View frustum culling
  - Discard objects outside view frustum
- Occlusion culling
  - Discard objects that are within view frustum, but hidden behind other objects

View frustum culling

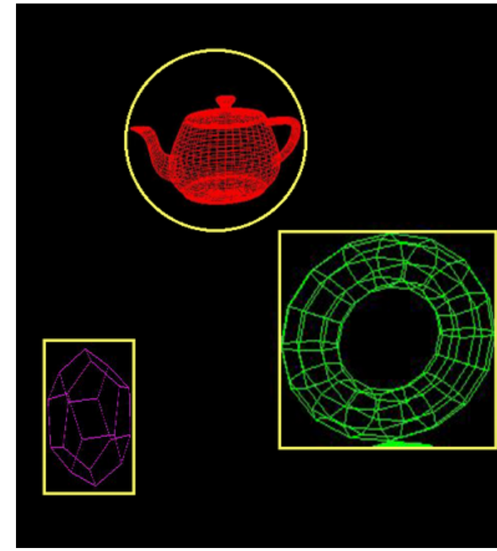
- Frustum defined by 6 planes
- Each plane divides space into "outside", "inside"
- Check each object against each plane
  - Outside, inside, intersecting
- If "outside" at least one plane
  - Outside the frustum
- If "inside" all planes
  - Inside the frustum
- Else partly inside and partly out
- Challenge: compute intersections efficiently

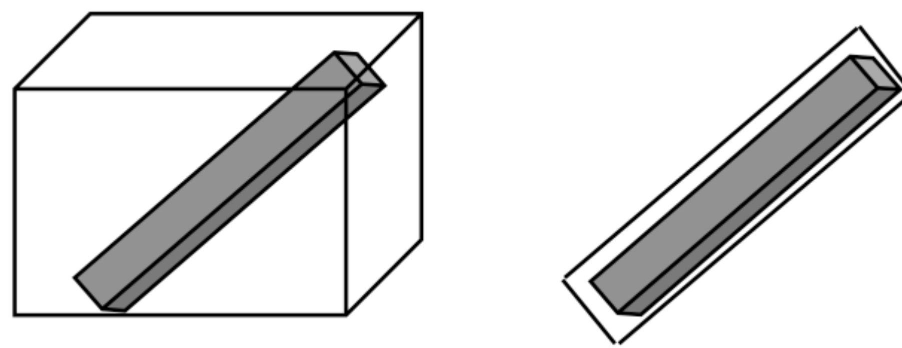


View frustum

Bounding volumes

- Simple shape that completely encloses an object
  http://en.wikipedia.org/wiki/Bounding_volume
- Generally a box or sphere
  - Spheres easiest to work with, but hard to get tight fits





Bounding volumes

Culling with bounding volumes

- Intersect bounding volume with view frustum, instead of full geometry
  - Intersection computation much simpler
- Culling is conservative
  - Sometimes, bounding volume intersects view frustum, but actual geometry does not
  - No artifacts, but some performance loss

Culling with bounding spheres

Precomputation

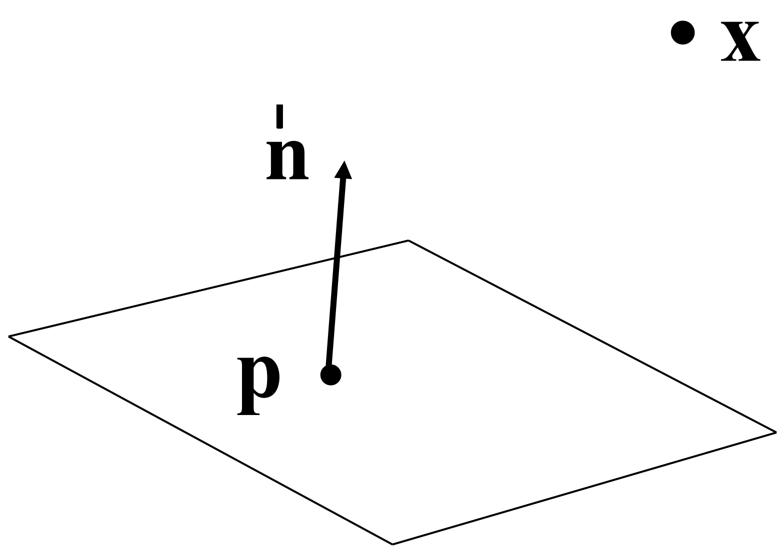
- Simple computation of bounding spheres
  - Compute object „center“, e.g., average of all vertices
  - Sphere radius is largest distance from center to any vertex
- Tightest possible bounding sphere is harder to find, http://en.wikipedia.org/wiki/Bounding_sphere

Rendering

- Intersection of sphere with view frustum

Distance to plane

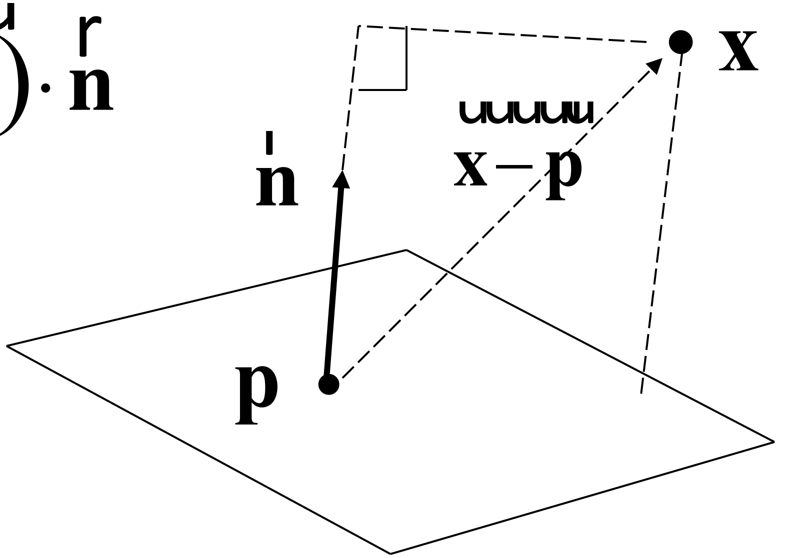
- A plane is described by a point $\mathbf{p}$ on the plane and a unit normal $\vec{\mathbf{n}}$
- Find the (perpendicular) distance from point $\mathbf{x}$ to the plane



Distance to plane

- The distance is the length of the projection of $\overrightarrow{\mathbf{x}-\mathbf{p}}$ onto $\vec{\mathbf{n}}$

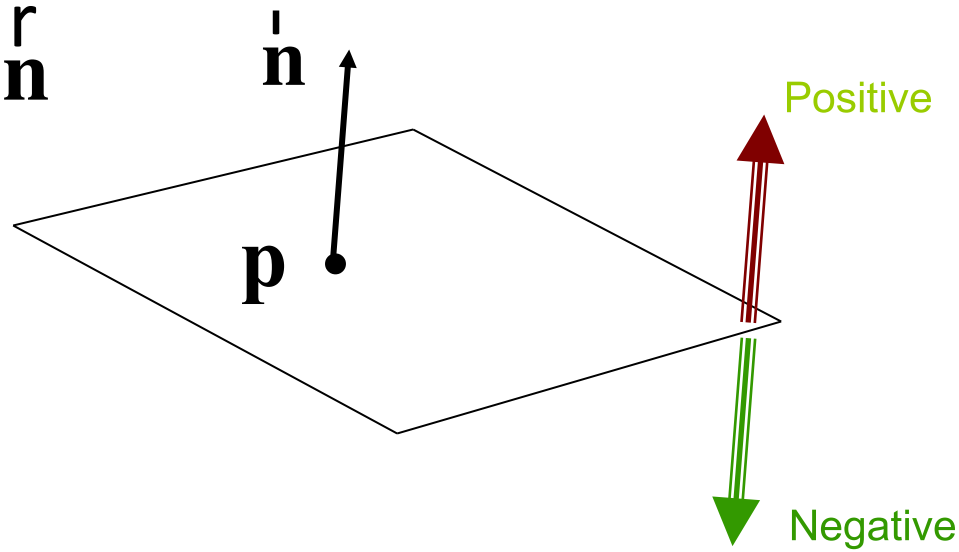
$$dist = \left(\overrightarrow{\mathbf{x}-\mathbf{p}}\right)\cdot\vec{\mathbf{n}}$$



Distance to plane

- The distance has a sign
  - positive on the side of the plane the normal points to
  - negative on the opposite side
  - 0 exactly on the plane
- Divides all of space into two infinite half-spaces

$$dist(\mathbf{x}) = \left(\vec{\mathbf{x}} - \vec{\mathbf{p}}\right) \cdot \vec{\mathbf{n}}$$



Distance to plane

- Simplification

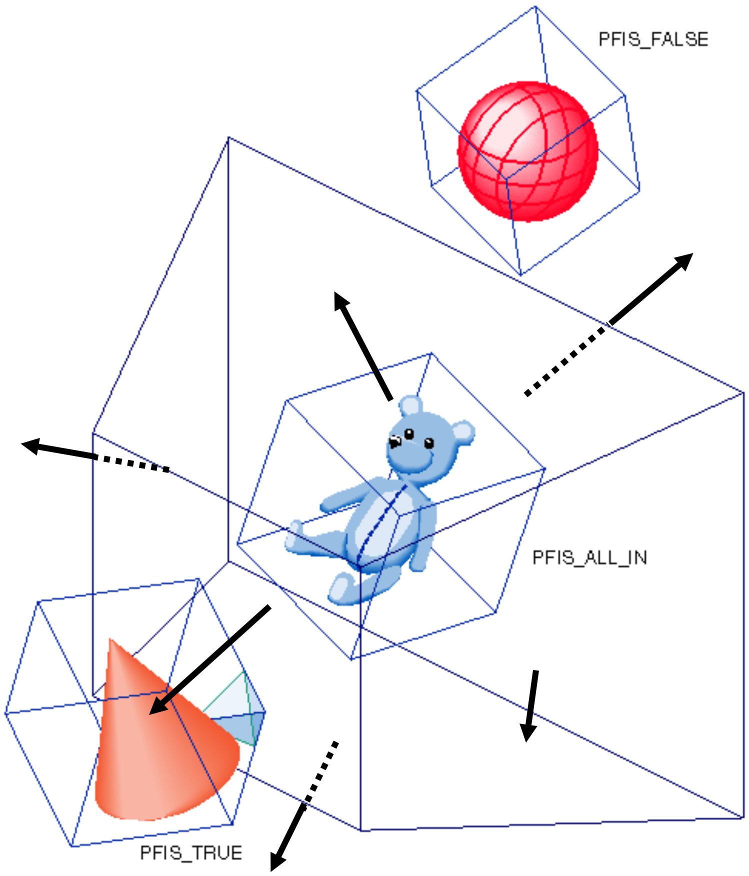
$$\begin{aligned} dist(\mathbf{x}) &= (\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} \\ &= \mathbf{x} \cdot \mathbf{n} - \mathbf{p} \cdot \mathbf{n} \end{aligned}$$

$$dist(\mathbf{x}) = \mathbf{x} \cdot \mathbf{n} - d, \quad d = \mathbf{pn}$$

- d is independent of $\mathbf{p}$
- d is distance from the origin to the plane
- We can represent a plane with just d and $\vec{\mathbf{n}}$

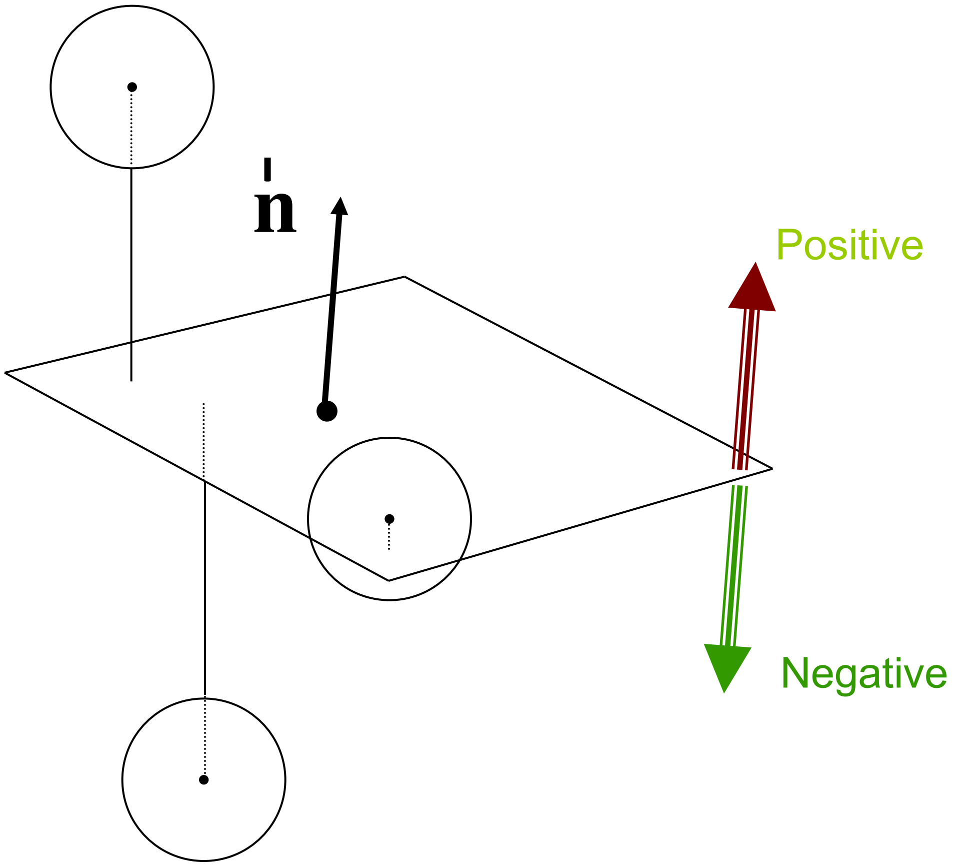
Frustum with signed planes

- Normal of each plane points outside
  - "outside" means positive distance
  - "inside" means negative distance



Test sphere and plane

- For sphere with radius r and origin $\mathbf{x}$, test the distance to the origin, and see if it's beyond the radius
- Three cases
  - $dist(\mathbf{x})>r$
    - completely above
  - $dist(\mathbf{x})<-r$
    - completely below
  - $-r<dist(\mathbf{x})<r$
    - intersects

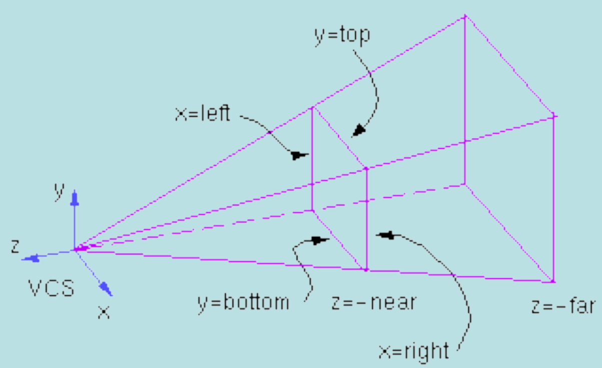


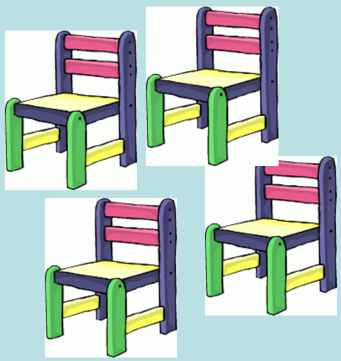
Summary

- Precompute the normal $\mathbf{n}$ and value d for each of the six planes
- Given a sphere with center $\mathbf{x}$ and radius r
- For each plane:
  - if $dist(\mathbf{x}) > r$: sphere is outside! (no need to continue loop)
  - add 1 to count if $dist(\mathbf{x}) < -r$
- If we made it through the loop, check the count:
  - if the count is 6, the sphere is completely inside
  - otherwise the sphere intersects the frustum
  - *(can use a flag instead of a count)*

Culling groups of objects

- If whole group of objects outside view frustum, want to be able to cull the whole group quickly
- But if the group is partly in and partly out, need to be able to cull individual objects
- Should we use scene graph hierarchy for culling?

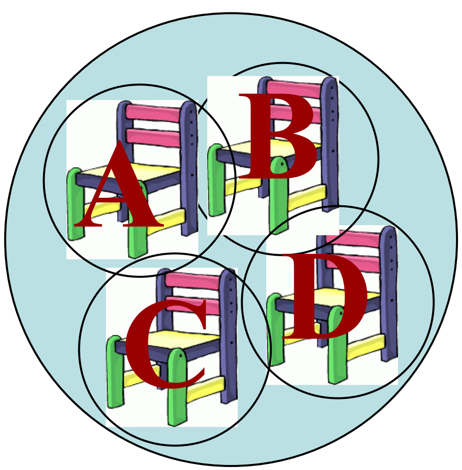


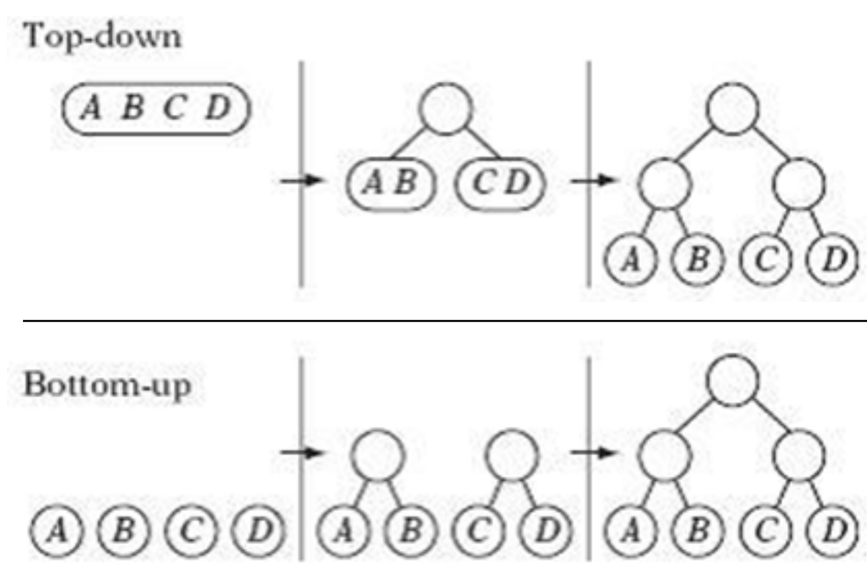


Bounding volume hierarchies (BVH)

http://en.wikipedia.org/wiki/Bounding_volume_hierarchy

- Construct hierarchy of objects in a tree
- Bounding volume of each parent node encloses the bounding volumes of all its children
- Top-down or bottom-up construction



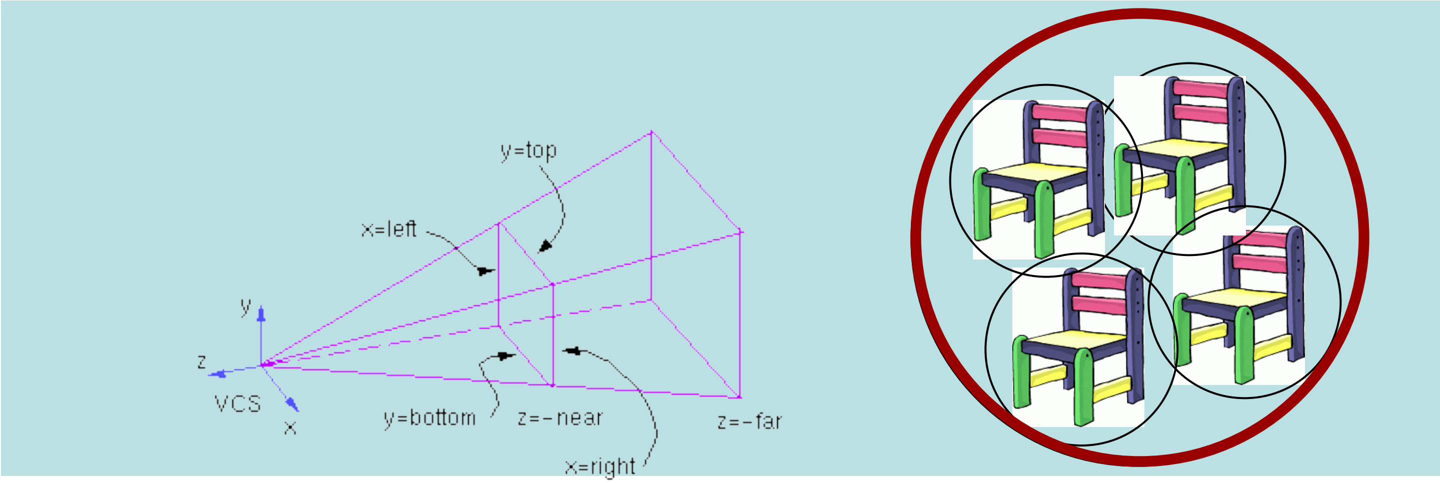


Recursively split groups

Recursively merge groups

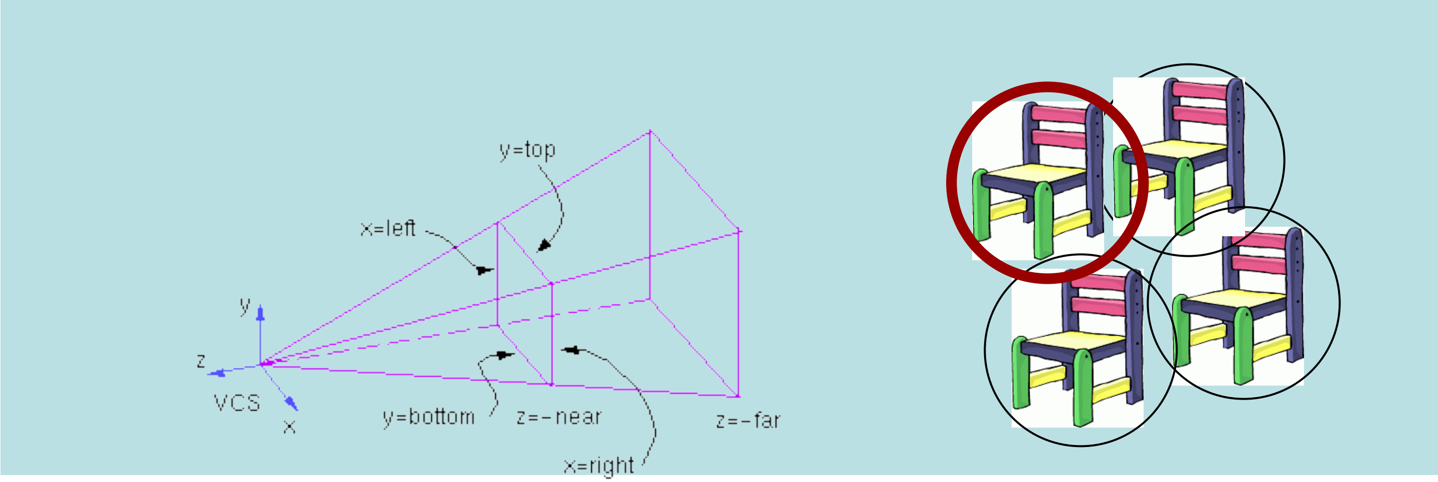
Hierarchical view frustum culling

- Start by testing the outermost bounding volume
  - If it's entirely out, don't draw the group at all
  - If it's entirely in, draw the whole group



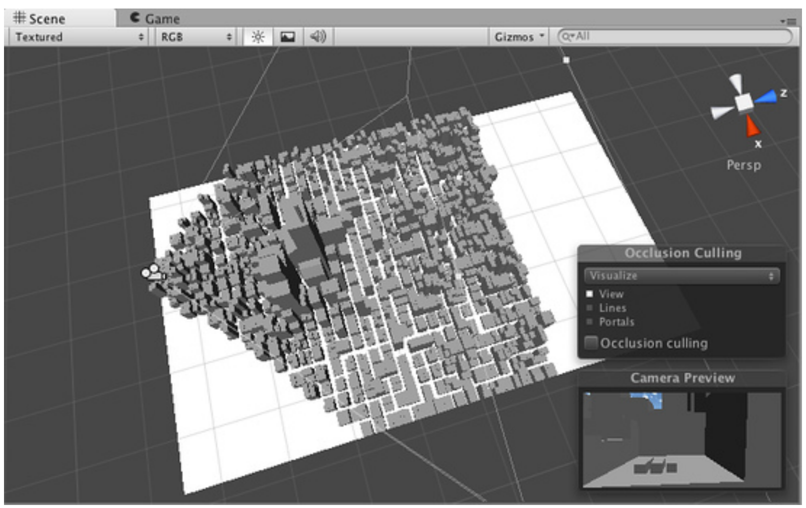
Hierarchical view frustum culling

- If bounding volume is partly inside and partly outside
  - Test each child's bounding volume individually
  - If the child is in, draw it; if it's out cull it; if it's partly in and partly out, recurse.
  - If recursion reaches a leaf node, draw it normally

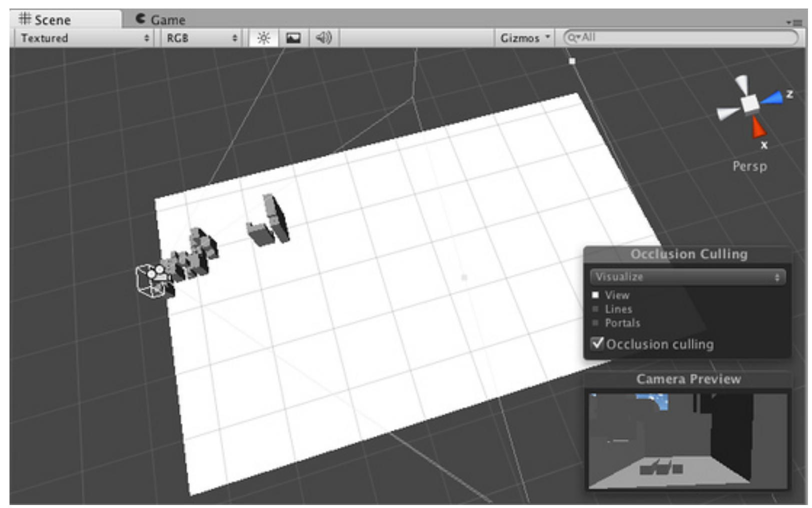


Occlusion culling

- „Don‘t attempt to draw objects that are occluded behind others“
- Related to the hidden surface determination problem
  http://en.wikipedia.org/wiki/Hidden_surface_determination



View frustum culling



Occlusion culling

http://docs.unity3d.com/Documentation/Manual/OcclusionCulling.html

Occlusion culling

- Cell-based occlusion culling using potentially visible sets
  http://en.wikipedia.org/wiki/Potentially_visible_set
  http://www.gamedev.net/reference/articles/article1212.asp

1. Preprocessing
   - Divide scene into cells (3D cubes)
   - Each cell stores list of objects that intersect it
   - Determine potentially visible set (PVS) for each cell
   - PVS: For each cell, set of other cells potentially visible from any viewpoint within first cell
2. Rendering
   - When camera is in one cell, do not need to render any cells (i.e., any objects associated with that cell) not in PVS

PVS computation

- Conservative
  - Never exclude a potentially visible cell from the PVS, but some cells in PVS may actually not be visible from certain viewpoints
  - Problem: performance degradation
- Aggressive
  - No invisible cells are in PVS, but some visible cells may be ommitted
  - Problem: visibility errors
- Exact
  - No visibility errors and no redundancy
  - Challenge: tractable algorithms to compute

http://en.wikipedia.org/wiki/Potentially_visible_set

Occlusion culling

- Example video using “Unity” game engine
  http://unity3d.com/
  http://www.youtube.com/watch?v=S5l3unhW4e0
- Specialized algorithms for different types of geometry
  - Indoor scenes (scenes with portals)
    http://en.wikipedia.org/wiki/Portal_rendering
  - Terrain (height field, 2.5D)
  - Urban scenes

Dynamic scenes

- Challenge: objects move or change shape
- Hierarchical culling with bounding volume hierarchies (BVH)
  - Need to update BVH
- Occlusion culling with potentially visible sets (PVS)
  - Need to update PVS
- Require efficient algorithms!