Enhancing X3D for Advanced MR Appliances

Yvonne Jung, Tobias Franke, Patrick Dähne, Johannes Behr

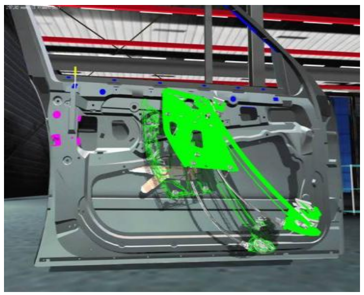
Fraunhofer-IGD, Darmstadt, Germany

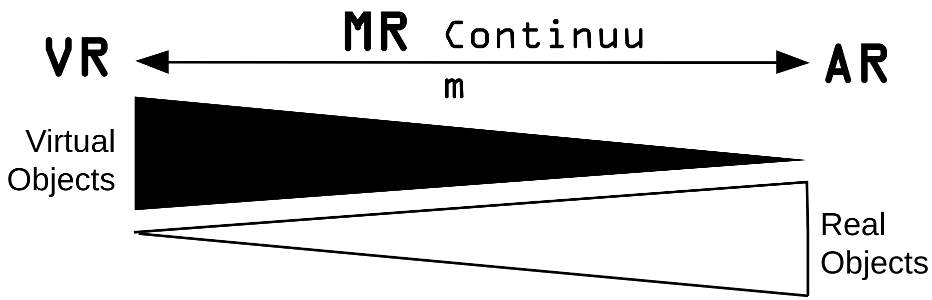
http://www.igd.fhg.de/igd-a4/

Overview

- Motivation
- Data Stream Sensors / Camera model
- Lighting in Mixed Reality
- Shadows in X3D
- Multi-pass techniques
- Differential Rendering
- Results
- Conclusion

Motivation





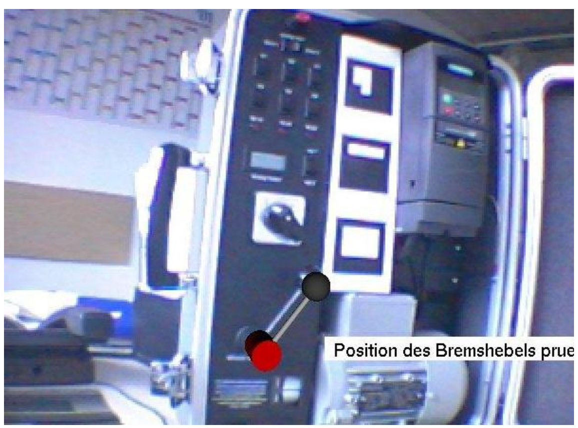
- Virtual Reality (VR) – Only virtual objects/ data
  - X3D well established standard as application description language
- Augmented Reality (AR) – Virtual and real objects
  - X3D (sometimes) used as loader for geometric models
- Mixed Reality (MR) – Continuum between VR and AR
  - X3D currently lacks important features needed for MR →
  - Integration of sensor data streams + rendering extensions needed!

Motivation

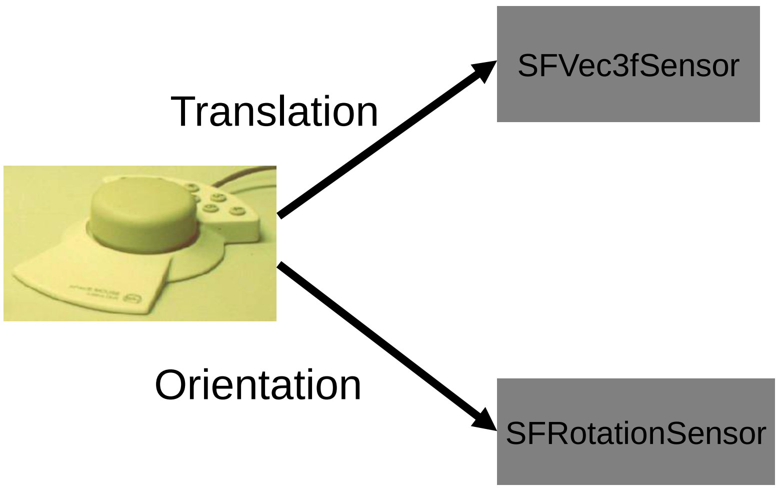
- Augmented Reality aims at *integrating* additional data/ virtual objects into real scenes (assembly simulations…)
  - Video of real scene needs to be put behind virtual objects
  - Exact pose of user or camera needs to be determined (→ CV)
- Next step: Fitting virtual objects *seamlessly* into real scenes
  - Realistic and photometrically consistent lighting is needed
  - Changes in lighting caused by virtual objects need to be updated
- Features not provided by *X3D*, but it already provides
  - Scene- and behaviour-graph
  - Interaction and navigation
  - Animation and Simulation

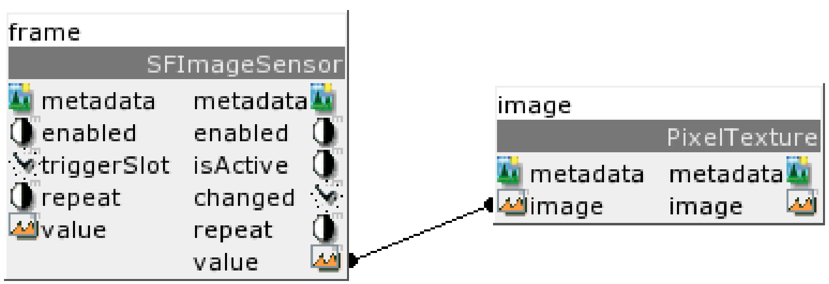
Motivation

- Problem: We want to augment images/ videos not only with some additional information but also with nice looking virtual objects.
- Challenge 1: Integration of various sensor data streams in X3D
- Challenge 2: Usage of advanced rendering methods like shadows



Integration of Data Stream Sensors

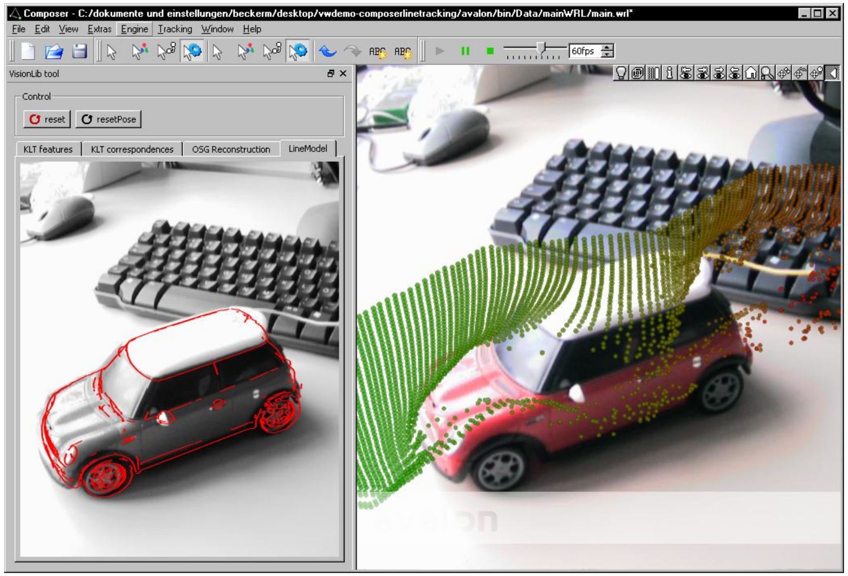




- Device Independent
- One physical device can be mapped to multiple sensors
- One sensor for every base type (Float, Rotation, Image, Matrix…)
- Label used to map the sensor to the data stream
- Node interface for type “x”:

```
xSensor : X3DDirectSensorNode {
  x          [in,out] value
  SFBool   []       out   FALSE
  SFString []       label ""
}
```

Flexible Camera



- New camera node for easier integration with visualization and tracking systems

```
Viewfrustum : X3DViewpointNode {
  …
  SFMatrix4f [in,out] modelview  ident.
  SFMatrix4f [in,out] projection   ident.
}
```

- Likewise new transform node

```
MatrixTransform : X3DGroupingNode {
  …
  SFBool [in,out]   render      TRUE
  SFMatrix4f [in,out] matrix identity
}
```

Relighting problems in Mixed Reality

- Geometry reconstruction
  - Including camera pose estimation (→ Computer Vision)
  - Needed for occlusion handling and shadows (+ tracking)

- Lighting reconstruction and simulation
  - Recover number, type and position of primary light sources
  - Consistent lighting of virtual objects with real world illumination

- Material reconstruction
  - Interreflections/ color bleeding; shadow color
  - Enables changes in material and geometry

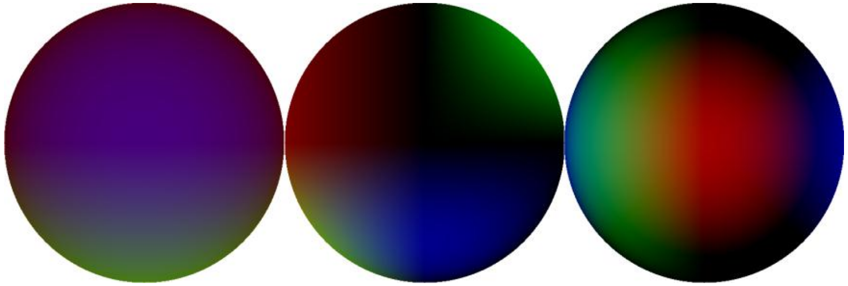
Image based lighting (IBL)

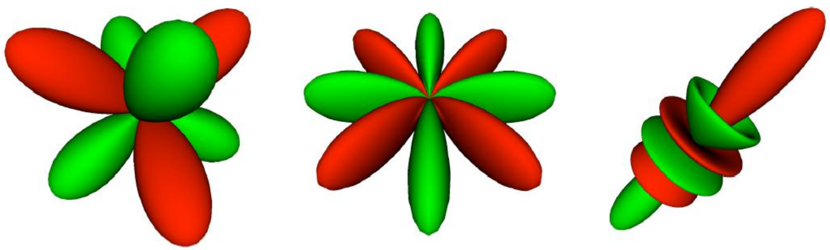
- Capture real world lighting
  - 180° fish-eye lens
  - Light probe image
- High dynamic range (HDR) for capturing scene radiance
  - P. Debevec, J. Malik: Recovering High Dynamic Range Radiance Maps from Photographs. 1997.
  - P. Debevec: Rendering Synth. Objects Into Real Scenes: Bridging Traditional a. Image-Based Graphics With Global Illumination a. HDR Photography. 1998.
- Environment Mapping for transfer
  - Irradiance mapping as “extension” for others than purely reflective materials
  - → pre-integration of simple BRDF’s

Spherical Harmonics (1)

$$y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta), & m > 0 \\ \sqrt{2}K_l^m \sin(-m\phi)P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0 \end{cases}$$

$$K_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}$$



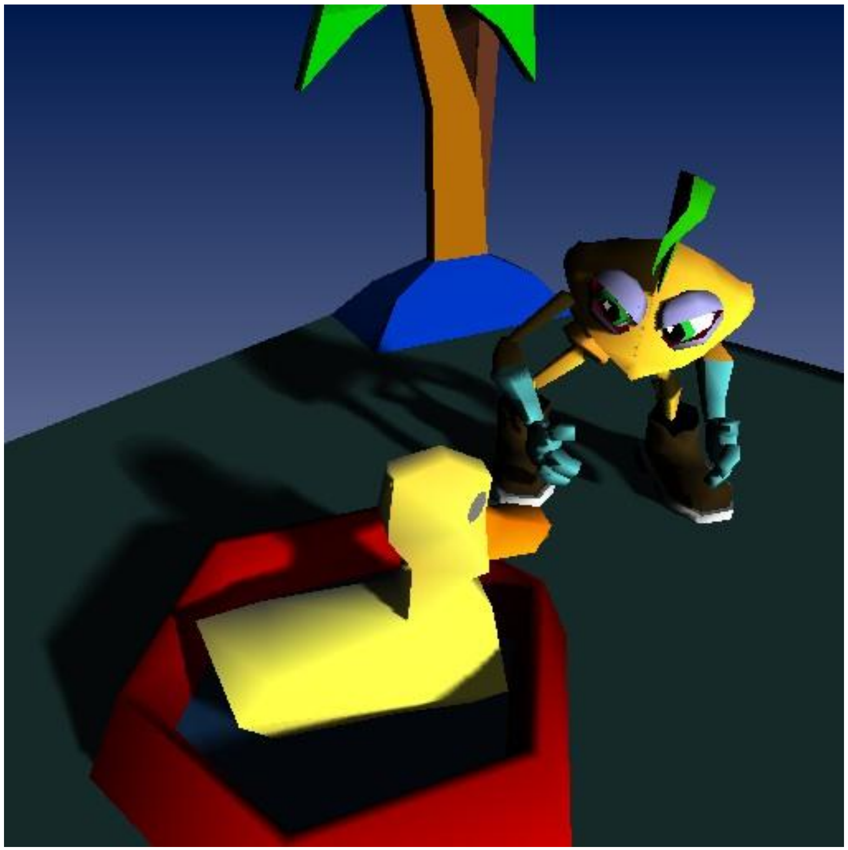


Spherical Harmonics (2)

- Low-frequency representation of spheremaps are efficiently calculated with SH analysis
- Function values of Legendre polynomial (base funcs) pre-computed a. stored in texture
- Reconstruction in shader which dots the constant SH values with corresponding coefficients
- SphericalHarmonicsGenerator
  - is a special texture node
  - generates irradiance map from input texture ("irradianceMap")
  - irradiance map is used to simulate reflected light for certain material
  - "numBand" defines type of map, "numSamples" scales integration
  - "ambient_changed" determines ambient brightness and can be used to adjust shadow intensity

```
SphericalHarmonicsGenerator :
   X3DTextureNode {
 SFNode  []          textureProperties NULL
 SFImage [in,out] irradianceMap NULL
 SFInt32 [in,out]  numBands    3
 SFInt32 [in,out]  numSamples  1000
 MFFloat [out]     coefficients_changed
 SFFloat [out]      ambient_changed
}
```

Shadows

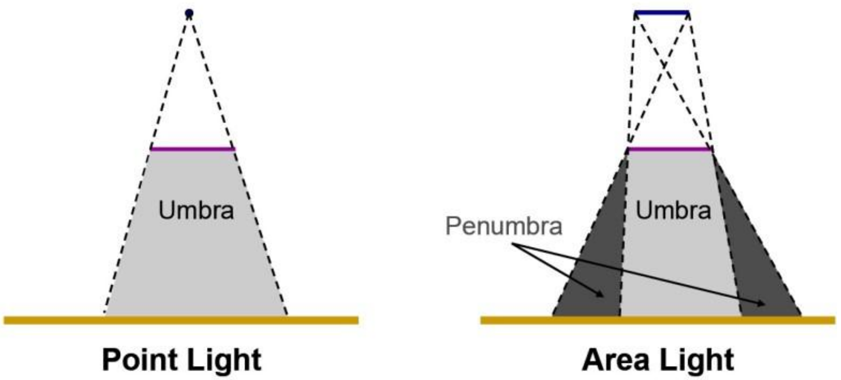


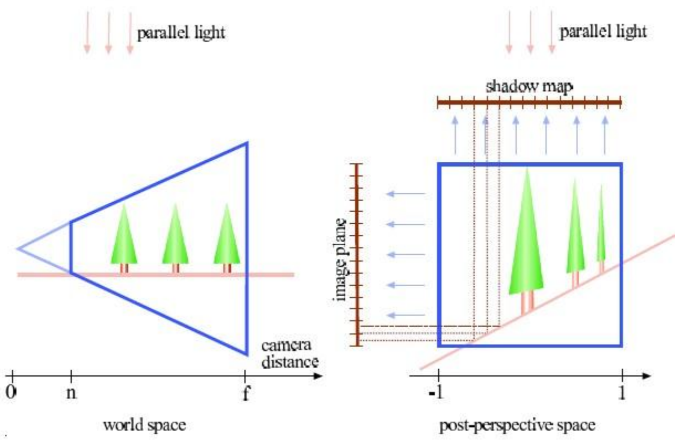
- Requirements:
  - Robust and intuitive usage
  - Applicable for every type of scene
  - No special treatment for shaders
- Solution:
  - No special shadow nodes, but extension of existing light nodes, for regulating light and shadow
  - Generic parameter/ abstraction level for supporting different types of implementations
  - Example (values in [0;1]):

```
SpotLight {
    shadowIntensity 0.7
    mode "auto"
    direction 0 -1 0
    location -2 14 2
}
```

Implementation

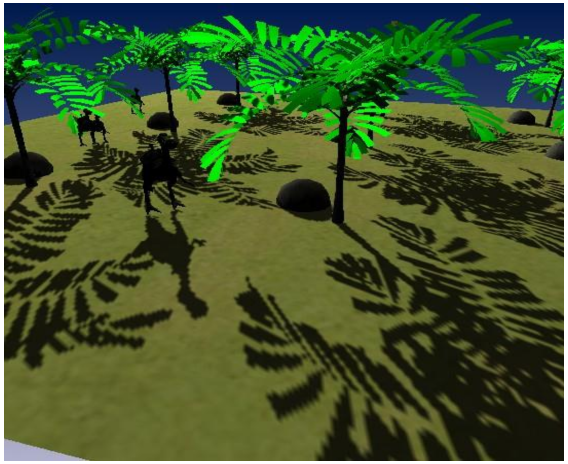
- Real-time shadow algorithms
  - Light maps (only static scenes)
  - Plane projections (only planes)
  - Shadow volumes (good quality)
  - Shadow mapping (scales best)
- Shadow mapping
  - Create depth map from light view and project onto scene; in shadow if depth is greater than map value
  - PCF (percentage-closer filtering): simulates shadows from area lights
  - LISP (light space perspective sha.): maps calculated in post-perspective camera space (frustum → cube); virtual camera calc. in light space



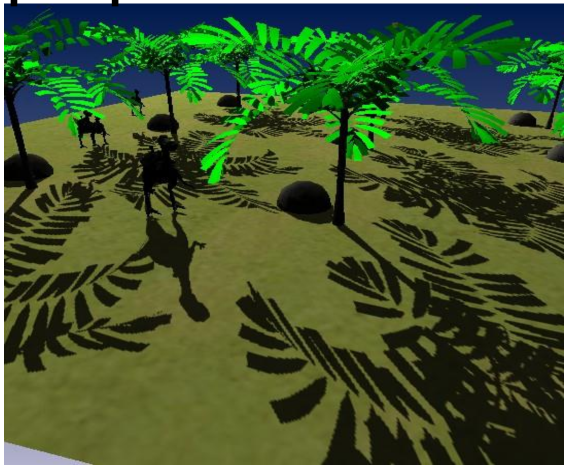


Shadow Modes

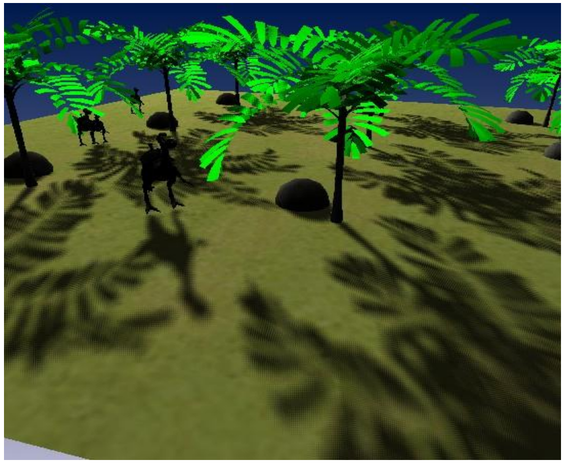
uniformHardShadow



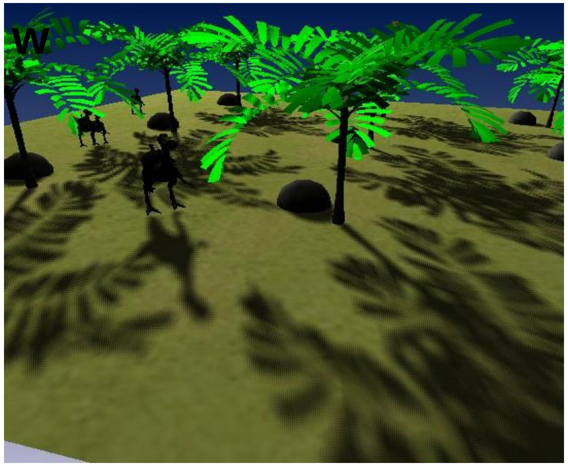
perspectiveHardShadow



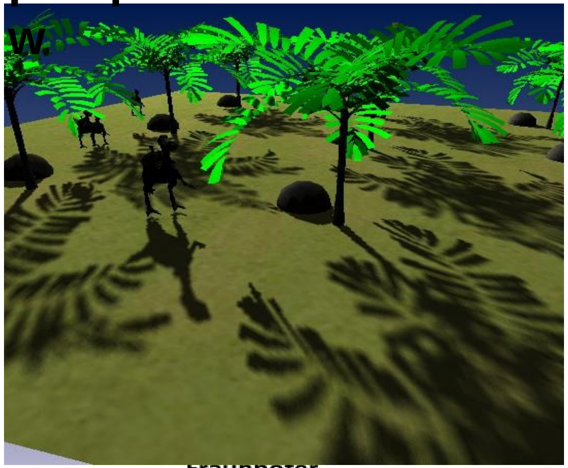
fastUniformSoftShadow



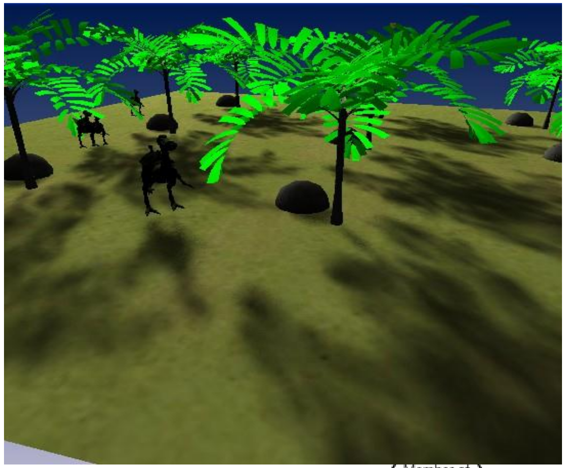
niceUniformSoftShadow



perspectiveSoftShadow

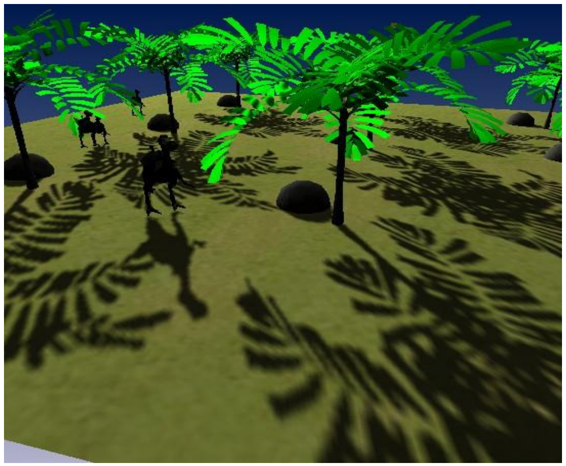


uniformSoftShadow

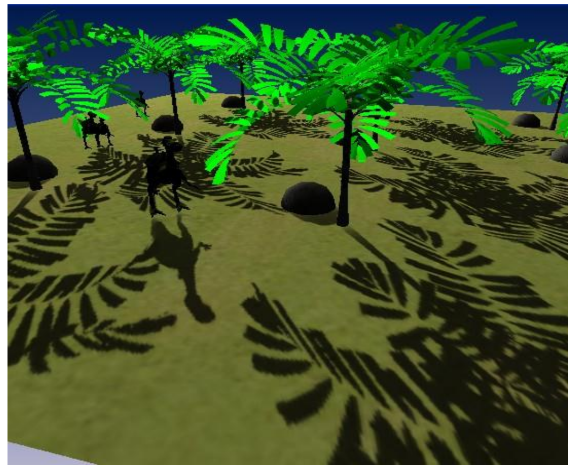


Shadow Smoothness

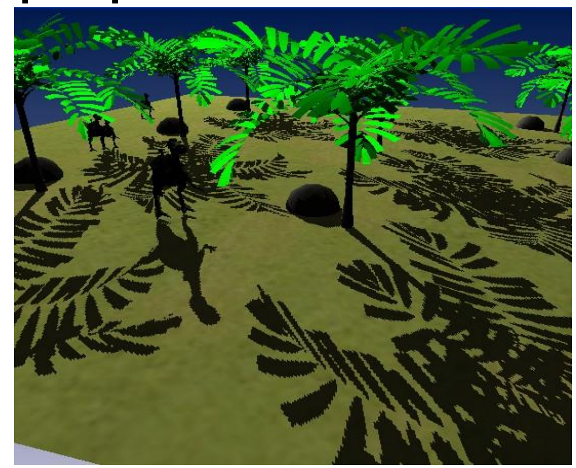
niceUniformSoftShadow - 0.0



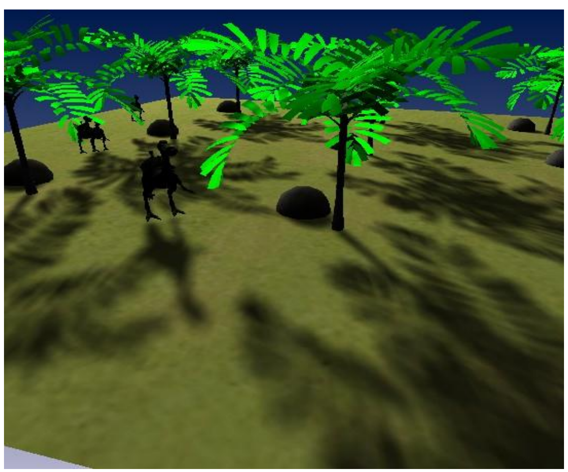
uniformSoftShadow - 0.0



perspectiveSoftShadow - 0.0



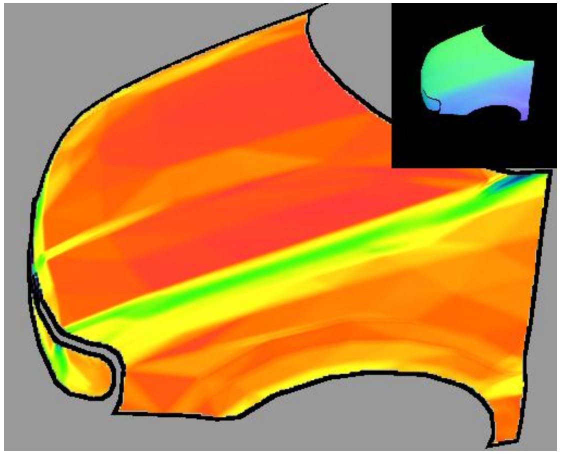
niceUniformSoftShadow - 1.0

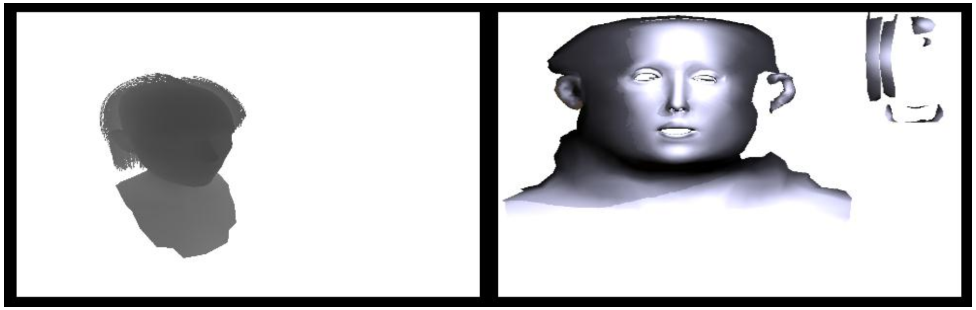


uniformSoftShadow - 1.0

perspectiveSoftShadow - 1.0

Multi pass rendering (1)





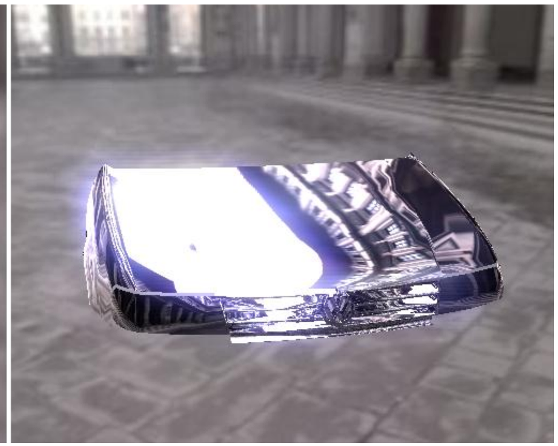
- Term "multi pass" is twofold, it means both the ability to
  - dynamically render a partial scene graph to an offscreen texture
  - render in an ordered sequence with different drawing operations
- RenderedTexture can be seen as FBO/ PBuffer abstraction
  - first proposed in “http://www.xj3d.org/extensions/render_texture.html”
  - floating point textures can be forced → higher precision + HDR rendering

Multi pass rendering (2)

- Image space rendering ops (e.g. rendering to texture space or NPR rendering)
  - Accessing e.g. neighboring information in shader programs
- Field "depthMap“ allows generation of depth maps; only useful in combination with appropriate transforms
  - projection (modelview projection matrix of camera space)
  - viewing (model matrix of parent)
- TextureGrabLayer node
  - Useful for special effects
  - Contains grabbed framebuffer (depending on its ordering)
  - Field “texture” can be re-USE-d

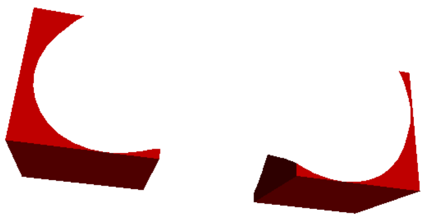
```
RenderedTexture : X3DEnvironmentTextureNode{
  SFNode      []        textureProperties NULL
  SFString    [in,out] update "NONE"
  MFNode      []        excludeNodes []
  SFNode      [in,out] viewpoint NULL
  SFNode      [in,out] background NULL
  SFNode      [in,out] fog NULL
  SFNode      [in,out] scene NULL
  MFInt32     [in,out] dimensions [128 128 4]
  SFBool      [in,out]  depthMap FALSE
  SFMatrix4f [out]    projection identity
  SFMatrix4f [out]    viewing     identity
}

TextureGrabLayer : X3DLayerNode {
  SFBool [in,out] isPickable   FALSE
  SFNode [in,out] viewport   NULL
  SFNode [in,out] texture      NULL
}
```

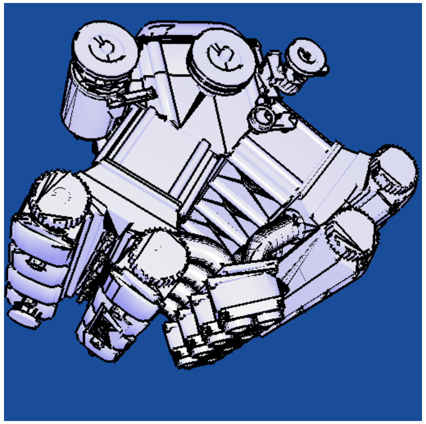
Layered rendering



- X3D 3.2: Layering/ layout for interaction and screen-space-text
  - Only sub-trees, rendering order and 2d-positions are defined now (→ HUD)
  - Still need for screen space compositing effects (e.g. blur, glow, scetch → IBR)
- Post processing step in image space to create special visual effects
  - Render window-sized and view-aligned quads with some Appearance
  - Additionally provide some way to control the composition methods

Render state control





- Access to color masking and arbitrary masking (i.e. stencil) in combination with defined rendering order for compositing
- Special materials for front/ back faces beyond *TwoSidedMaterial*
- Possibility to disable depth writing/ using different depth funcs
- Compositing of objects or layers via blending, discarding etc.

X3D node extensions (1)

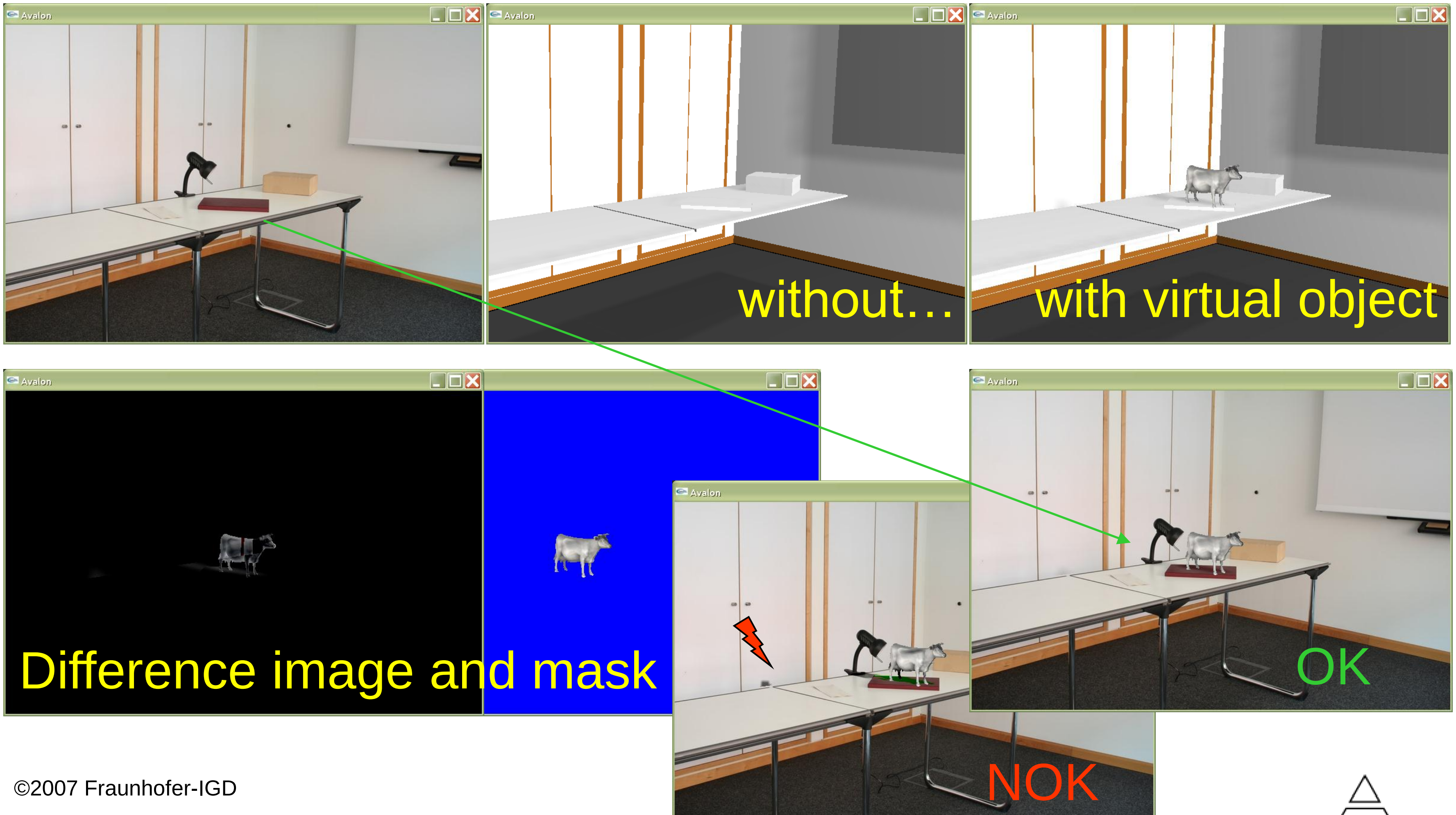

```
Appearance : X3DAppearanceNode {
  SFInt32  []       sortKey  0
  SFNode  [in,out] fillProperties  NULL
  SFNode  [in,out] lineProperties  NULL
  SFNode  [in,out] material  NULL
  MFNode  [in,out] shaders  []
  SFNode  [in,out] texture  NULL
  SFNode  [in,out] textureTransform NULL
  SFNode  [in,out] blendMode  NULL
  SFNode  [in,out] stencilMode  NULL
  SFNode  [in,out] colorMaskMode  NULL
  SFNode  [in,out] depthMode  NULL
  SFNode  [in,out] faceMode  NULL
}
AppearanceGroup : X3DGroupingNode {
  SFBool [in,out]        render        TRUE
  MFNode [in,out]        children      []
  SFNode [in,out]        appearance NULL
}
```

- Appearance reveals how Shape node looks like → extend shape component with some new nodes for setting different render states and Appearance with suiting fields
  - Maps to GPU, no PROTOs possible!
- Need to control the color-/ stencil-/ depth-buffer writing and merging → Requirement: control over rendering order
  - Introduce "sortKey" field (default is 0)
  - More robust and intuitive than e.g. a special ordering group for rendering
- Nodes for fine grained render state control
  - If corresponding fields in “Appearance” not set, browser uses standard settings
- New AppearanceGroup node is useful if a whole group of Shape nodes should share the same material properties
  - Field “render” (shared by all grouping nodes) simplifies setting of visibility

X3D node extensions (2)


```
StencilMode : X3DAppearanceChildNode {
  SFString [in,out] stencilFunc "none"
  SFInt32  [in,out] stencilValue 0
  SFInt32  [in,out] stencilMask  0
  SFString [in,out] stencilOpFail  "keep"
  SFString [in,out] stencilOpZFail "keep"
  SFString [in,out] stencilOpZPass "keep"
  SFInt32  [in,out] bitMask  -1
}

ColorMaskMode : X3DAppearanceChildNode {
  SFBool [in,out] maskR TRUE
  SFBool [in,out] maskG TRUE
  SFBool [in,out] maskB TRUE
  SFBool [in,out] maskA TRUE
}

BlendMode : X3DAppearanceChildNode {
  SFString [in,out] srcFactor  "one"
  SFString [in,out] destFactor "zero"
  SFColor  [in,out] color 1 1 1
  SFFloat  [in,out] colorTransparency 0
  SFString [in,out] alphaFunc "none"
  SFFloat  [in,out] alphaFuncValue 0
}

DepthMode : X3DAppearanceChildNode {
  SFBool   [in,out] enableDepthTest TRUE
  SFString [in,out] depthFunc "none"
  SFBool   [in,out] readOnly FALSE
  SFFloat  [in,out] zNearRange -1
  SFFloat  [in,out] zFarRange  -1
}
```

Steps in differential rendering



Differential Rendering with X3D

- Compositing method to augment img. with consistent illumination
- Given: reconstructed scene model
  - Geometry and Material
- Two passes/ lighting simulations
  - Original scene image/ video
  - Scene without virtual objects
  - Scene with add. virtual objects
- Reconstruction is inaccurate
  - Virtual objects must be rendered last, for creating a valid mask only at pixel positions finally containing the object
- Composition in after effects layer
  - Last step: render tex. layer with mask

- With *L* being the respective image radiance it holds that

$$L_{err} = L_{without} - L_{orig}$$

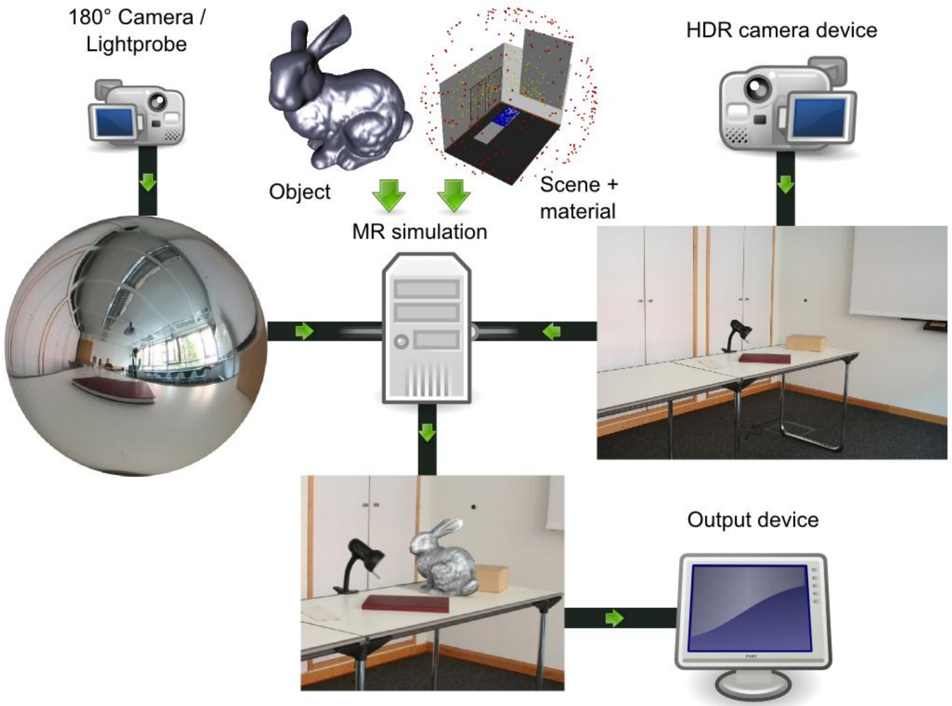
$$L_{final} = L_{with} - L_{err} \quad \Rightarrow$$

$$L_{final} = L_{orig} + (L_{with} - L_{without})$$

- GLSL fragment shader program

```
[…]
void main()
{
  vec4 virt=texture2D(with,gl_TexCoord[0].st);
  vec4 real=texture2D(without,gl_TexCoord[0].st);
  vec4 img=texture2D(orig,gl_TexCoord[1].st);
  gl_FragColor = img + virt - real;
}
```

System setup for MR simulation



- Preprocessing: material and geometry reconstr.
- Runtime: acquiring scene image, camera pose, illumination info
  - Routing to application
- Render scene twice, with shadows enabled
  - with/ without virtual obj.
  - Framebuffer + Texture
- Compositing by using 3 layers, with different render states enabled

Results (1)

Results (2)

- Left: “Archeoguide” (outdoor AR with video in background)
- Right: Combination of all presented techniques – dragon model is rendered in a 1500x1000 pixel context with 8xFSAA, a SH analysis of incoming lighting data with 9 coefficients, a mixture of 25% diffuse and 75% specular lighting, static AO and PCF shadows, at 10 fps on a standard Intel P4/ 2.4 GHZ PC with a NVidia 6600 GT

Conclusion

- Discussed how current X3D standard could be utilized for advanced augmented and mixed reality applications
- Data stream sensor nodes and new camera node for integrating devices like cameras or tracking systems
- Extension proposals for advanced rendering techniques
  - irradiance mapping (HDR)
  - robust, intuitive shadows
  - render state control
  - full multi pass support
  - layers for post processing
- Node extensions allow application developers to create complex photo-realistic mixed reality environments easily

Thanks for your attention!

Video/ Demo? Questions?

Player and technical information:
http://instant-reality.igd.fhg.de/