Modern computer and video games rendering techniques and how they can be used besides games in other fields of computer graphics such as cinematic rendering.

Folker Schamel, Spinor GmbH

Talk presented at Eurographics 2006, Vienna, Austria

Rendering in Games

- Performance is very important.
  - Below 20 fps is not acceptable in any situation.
  - Static vertex / index buffers.
    - Skinning on the GPU.
    - Waving grass, plants, tree on the GPU
- Good quality by simple tricks.
  - Many detailed textures.
  - No wasting of polygons.
- A lot of faking.
  - For example shadowing.

Rendering in Games

- Static world and dynamic objects often use different rendering techniques.
  - For example regarding shadowing and animation.
- Special effects play an important role.
  - For example particles (explosions, clouds).
- Normal mapping
- Post rendering effects.
- APIs: Direct 3D and console APIs.
  - Open GL is basically not used anymore for games.

Precalculated Lighting and Shadowing

- Runtime performance is always the same for any number of light sources.
  - Artist can use any number of lights she wants.
- Typical techniques are:
  - Precalculated lighting in texture maps.
  - Precalculated radiosity as lightmap.
  - Precalculated specularity
    - For example stored in per-vertex or per-texel spherical harmonics

Partial Precalculated Lighting and Shadowing

- Position-dependent precalculated lighting for dynamic objects.
  - For example storing spherical harmonics coefficients for different positions in the world and interpolate between them for rendering a character.
- Projected textures on dynamic objects.
  - For example lighting of church window onto characters.

Per-Object Shadowing Textures

- Shadowing texture per-object,
  which is projected onto the environment.
- Possible rendering techniques:
  - Applied in main rendering pass.
  - Separate additive lighting pass.
  - Rectangle rendered in screen space.

Per-Object Shadowing Textures

- Simple dynamic soft shadows.
  - More efficient filtering than depth shadow maps: Work is logarithmic instead of linear in softness.
  - Allows faked distance dependent softness.
- Requires render-to-texture per object each frame.
  - Does not scale well with many objects.

Shadow Volumes

- Simple shadow volumes:
  - Dynamic objects cast shadow onto environment
  - No lighting computations.
    Shadowed area is just darkened.
- Full-scene shadow volumes:
  - Each object can cast shadows onto itself and all other objects.
  - Per pixel lighting possible.

Shadow Volumes

- Shadow volume extrusion with DX9:
  - Exact extrusion not possible on GPU in vertex program.
  - Theoretically possible on GPU by render-to-vertex-buffer, but not used in practice.
  - Approximated extrusion possible on GPU in vertex program. But has more artifacts.
- Shadow volume extrusion with DX10:
  - Will support exact extrusion in geometry shader.

Shadow Mapping

- Full-scene shadowing
- Quite simple to implement.
- Quite efficient.
- Used in many games.
- Main problem: Depth map pixel artifacts.

HDR

- Combined with post rendering effects
  - For example blooming
- Float textures:
  - Problem: Blending and transparencies.
  - Alternative to float textures:
    Non-linear mapping of color space.

Larger Levels

- Visibility techniques.
  - Precalculated, e.g. PVS.
  - Manually defined, e.g. portals.
  - Realtime on CPU.
  - GPU-based occlusion culling.
- Streaming

based award winning Dreamfall on console and PC
(captured from PC; resolution of screenshot reduced)

Natural Future Trends

- Simple, but effective:
  - More polygons.
  - More complex pixel programs.
  - Higher resolution.
  - More and better antialiasing.

Advanced Streaming

- No stuttering
  - Handle all streaming in a separate background thread.
  - Requires a thorough multithreading architecture
- Swapping out game states
  - Allows huge mutable game worlds

Multithreaded Rendering

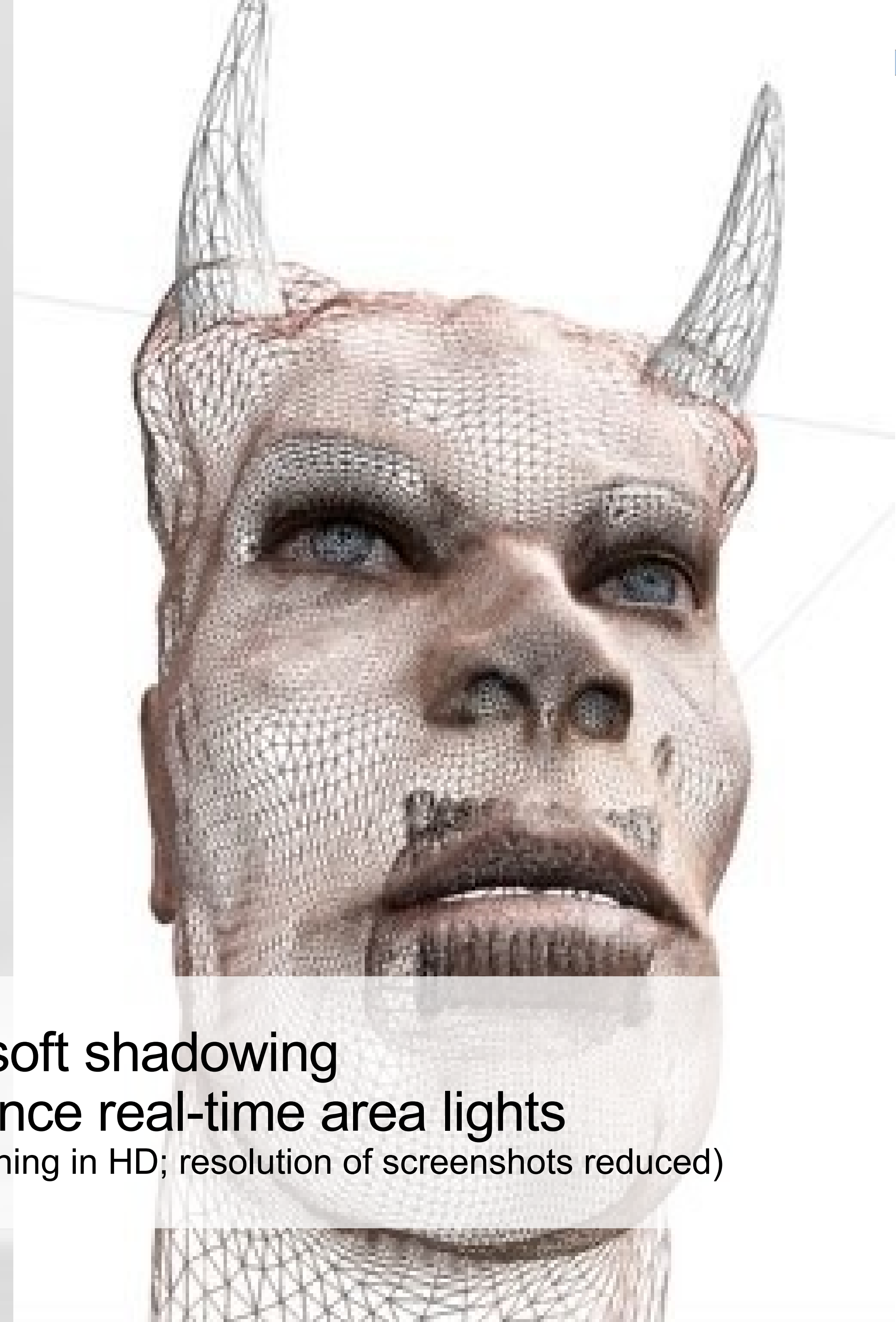
- Main thread does no work for rendering at all.
  - For example, rendering thread(s) do scene traversal and animation evaluation.
- Expensive work can be handed over to pooled threads.
  - For example, animation state evaluation for individual characters is a good candidate.

Future Trend: Better Rendering Quality

- Better lighting:
  - Better real-time lighting and shadowing.
  - More dynamic light sources.

Shark 3D™

Realistic hard and soft shadowing
from high-performance real-time area lights
(captured from Xbox 360 running in HD; resolution of screenshots reduced)



Shark 3D™

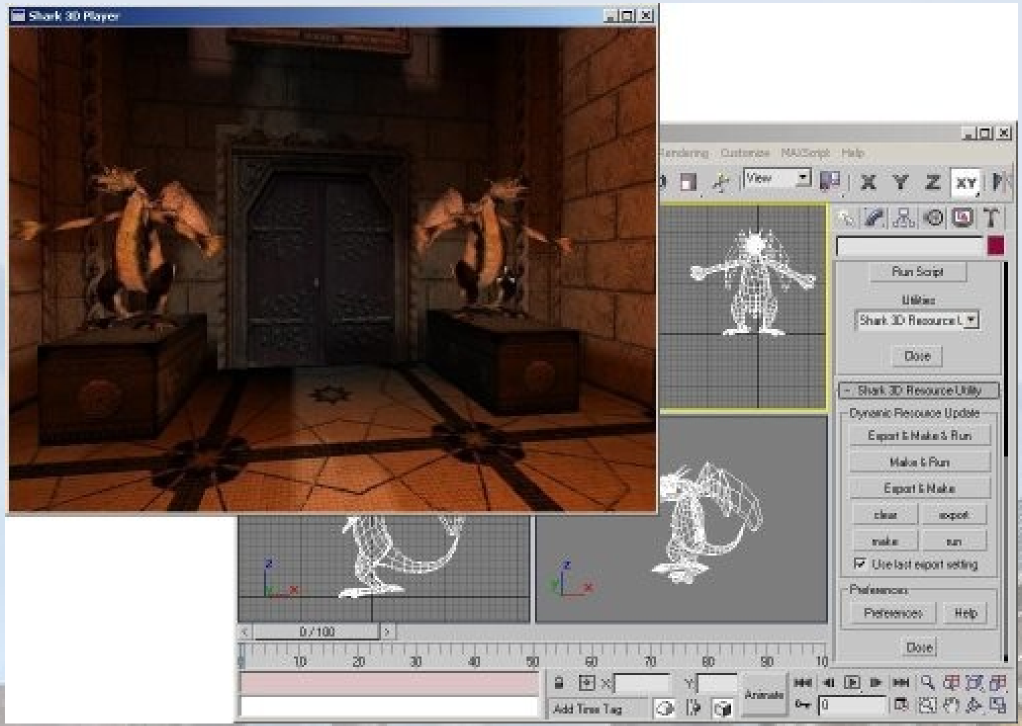
Realistic hard and soft shadowing
from high-performance real-time area lights
(captured from Xbox 360 running in HD; resolution of screenshots reduced)

Shark 3D Live Editing

Update changes inside editing tools(for example 3ds Max, Maya, Photoshop, proprietary tools) live into the running engine on all platforms (including consoles).

Changes you can update live include:

- Textures
- Sound files
- Shaders
- Vertex and pixel programs
- Lights
- Object positions
- Object geometry
- Mapping coordinates
- Animations
- Game logic objects (incl. error handling)
- Perch scripts (incl. error handling)
- ...

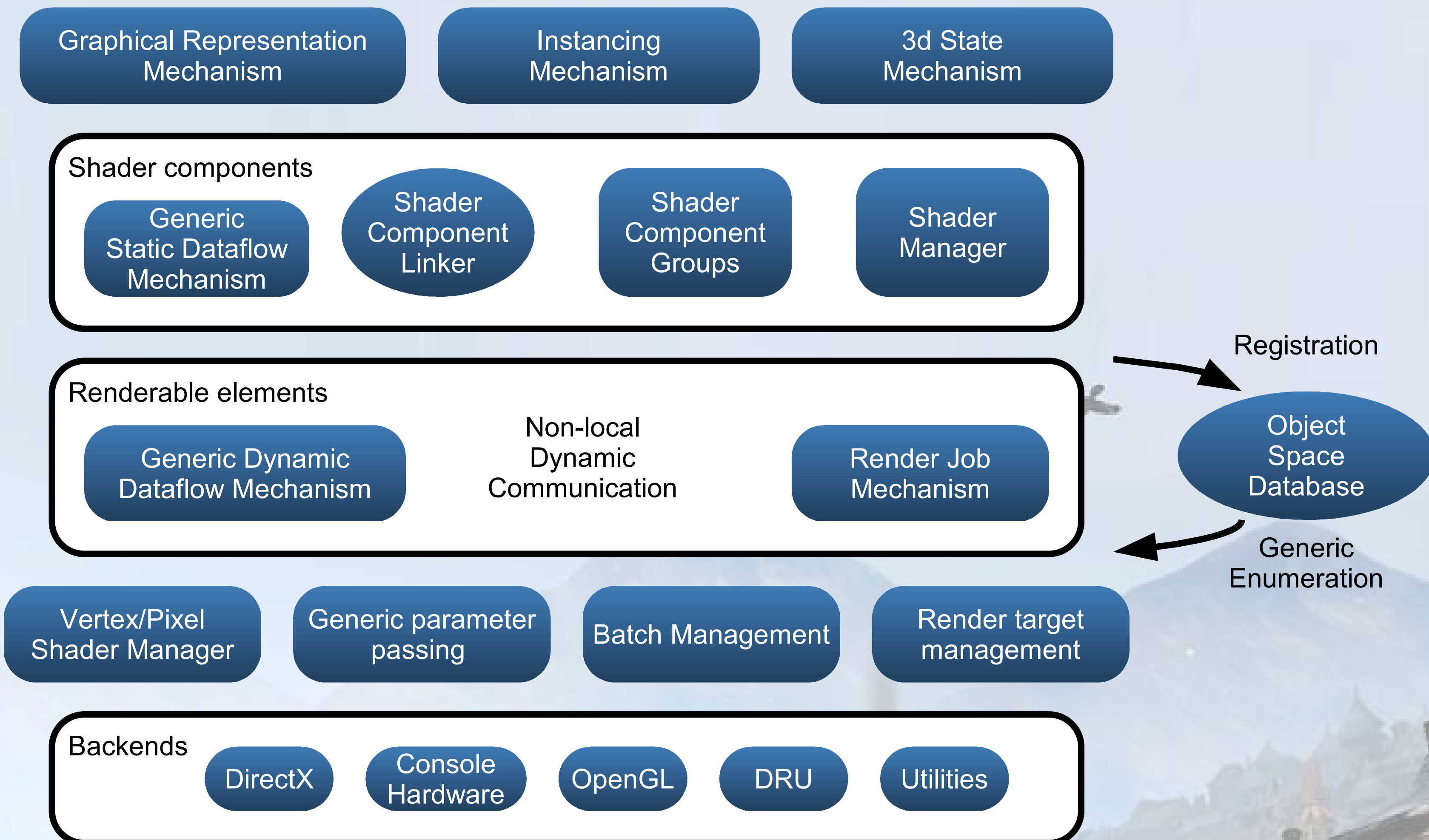


Open Architecture by Generic Scene Management

A generic scene management is an important foundation for high-end rendering features and for modularity in the renderer.

Static trees (BSP, static octree, ...)	- Do not work well for dynamic objects
Scene graphs	- Do not work well for dynamic objects - Do not work well with advanced non-local rendering techniques - Unnatural relation between logical object hierarchy and spacial relations
Generic object space	- Works excellently for many dynamic objects - Works excellently for advanced non-local rendering features - Shaders have generic access to object space - Decouples logical object hierarchy and spacial relations

Shark 3D as
Generic Open Renderer Platform



Shark 3D Modularity Sample: Main Rendering Code is Generic

Shark 3D's main rendering code:

```
s3d_CEngGfxTaskArray TaskArray;
s3d_CEngGfxCycle *Cycle = CollectNewCycle(
        Run, Cam, m_Trigger, TaskArray);

s3d_CEngUtilGfxElemJobBegin::AddGfxBegin(
        m_MsgHandler, m_Info.GetChars(), Cycle,
        m_DestProp, m_ClearParam, TaskArray, BeginMain);

s3d_CEngUtilGfxUtil::ExecTaskArray(TaskArray, 0);
```

The main rendering code is completely independent from particular advanced rendering features.

Shark 3D Modularity Sample: Rendering Features in Modules

Generic interface for implementing rendering modules in Shark 3D:

```
class s3d_CEngGfxElem: public s3d_CUtilRecogEyeBase
{
public:
    s3d_CEngGfxElem();

    virtual void GfxElemExec(
            s3d_CUtilRecogBase *GfxElemCtx,
            s3d_CUtilAtom *Trigger,
            s3d_CDrvVarBlk_cr ParamVarBlk,
            s3d_CEngGfxTaskArray &TaskArray);
};
```

Even advanced, non-local rendering techniques can be implemented in separate modules.

Examples: Different lighting techniques; multiple passes; rendering order; simple shadow volumes & shadow maps; advanced soft shadowing techniques; dynamic mirroring (planar, environment map etc.); post rendering effects; various render-to-texture techniques; effects requiring complex scene enumeration; PVS; ...

Shark 3D
Modular Shader System

Sample of Shark 3D standard shader components:

std

drvlightenum, meshenter, animactu
modeswitch, lightparam, animgen
modelmesh, lightenum, bundle
variants, lightenter, collexec
translmat, group, coloralpha
totex, fog, constfloat
regionenter, duptexchan, constmat
redirect, drvlightcoll, constvec
rectmesh, projmat, directtexchan
multi, addvec, mulmat
paintmesh

special

plain, projtotex
screenparam, filter
particenter, func
multilight, opticsdirect
envmap, opticstex
billboardmesh

shvol

enum, lenparam
perform, occluder
combineparam

user

user 1
user 2
user 3
user 4
user 4
user 5


logo!
10
0
Shark 3D™
ProSiebenSat.1 Produktion
KiKA
ARD.de
ZDF
is used by three of the four biggest German broadcasters
(captured from TV; resolution of screenshots reduced)
Projektor
Projektil

Game Renderer Features usually not used for non-Gaming

- Large world management usually not needed
  - No need for example for PVS, portals
- Console platform support

Game Renderer Features usually used also for non-Gaming

- Most rendering features
  - Including lighting and shadowing
- Performance optimizations
  - Runtime optimizations and tool pipeline optimizations

Additional Features required for non-Gaming

- Distributed rendering.
  - For example, Shark 3D was used for Cave rendering
- Linux platform support.

Thanks for your attention!