A Trip Down The (2011) Rasterization Pipeline

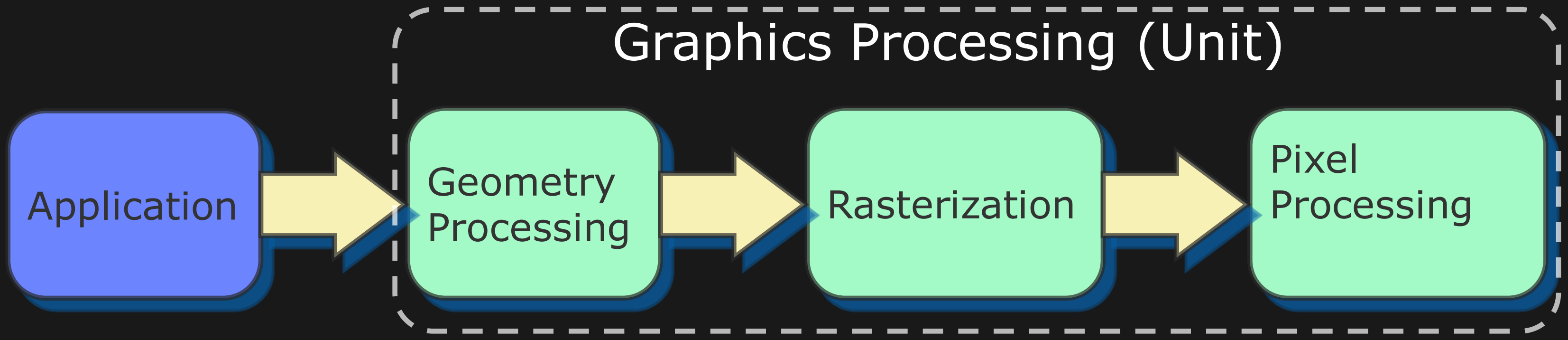
Aaron Lefohn - Intel / University of Washington
Mike Houston – AMD / Stanford

This talk

- Overview of the real-time rendering pipeline available in ~2011 corresponding to graphics APIs:
  - DirectX 11
  - OpenGL 4.x

- Discuss
  - What changes from DX9 to DX11
  - Key uses of these new features

General Rasterization Pipeline

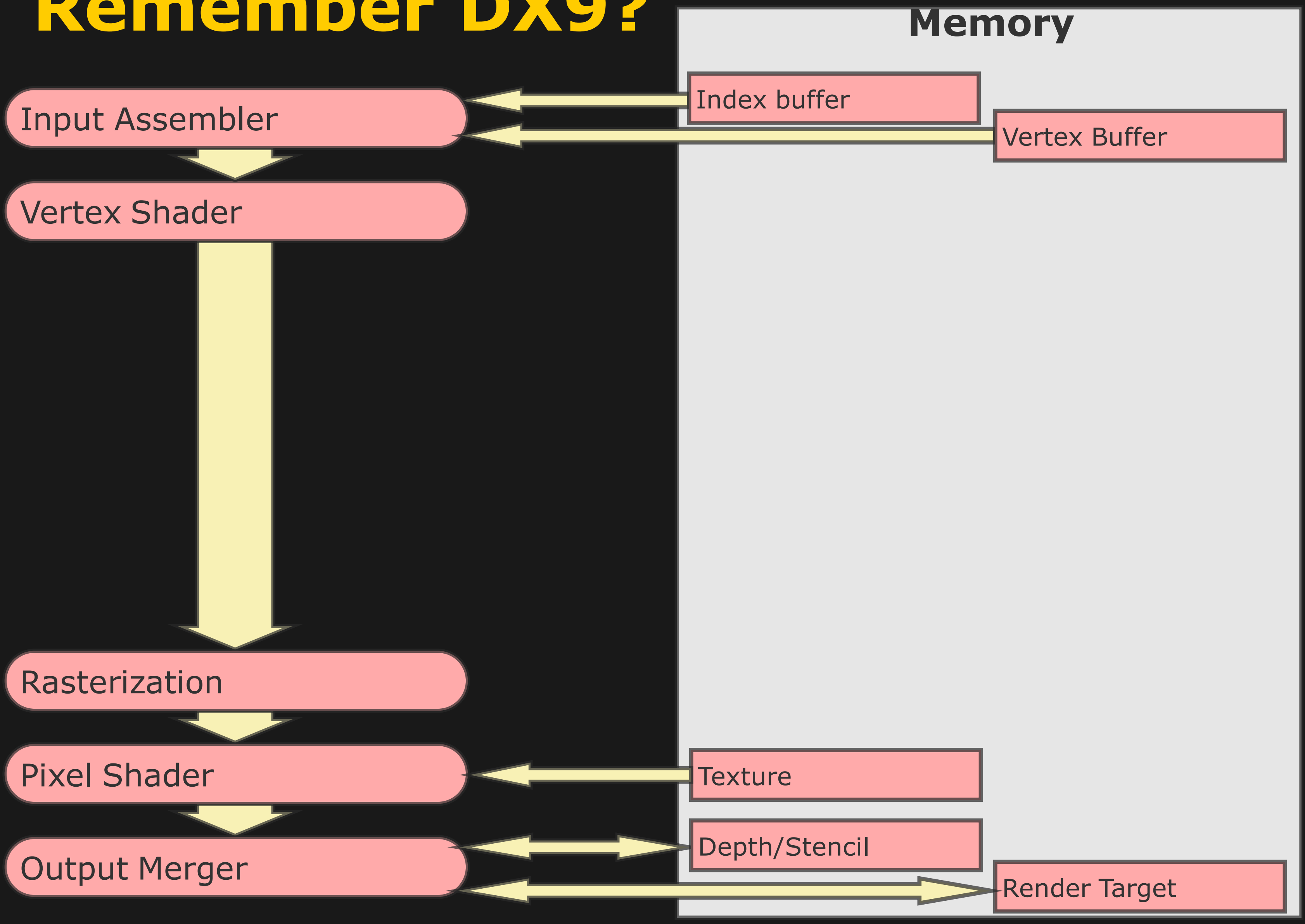


- Geometry processing:
  - Transforms geometry, generates more geometry, computes per-vertex attributes
- Rasterization:
  - Sets up a primitive (e.g., triangle), and finds all samples inside the primitive
- Pixel processing
  - Interpolates vertex attributes, and computes pixel color

Slide by Tomas Akenine-Möller

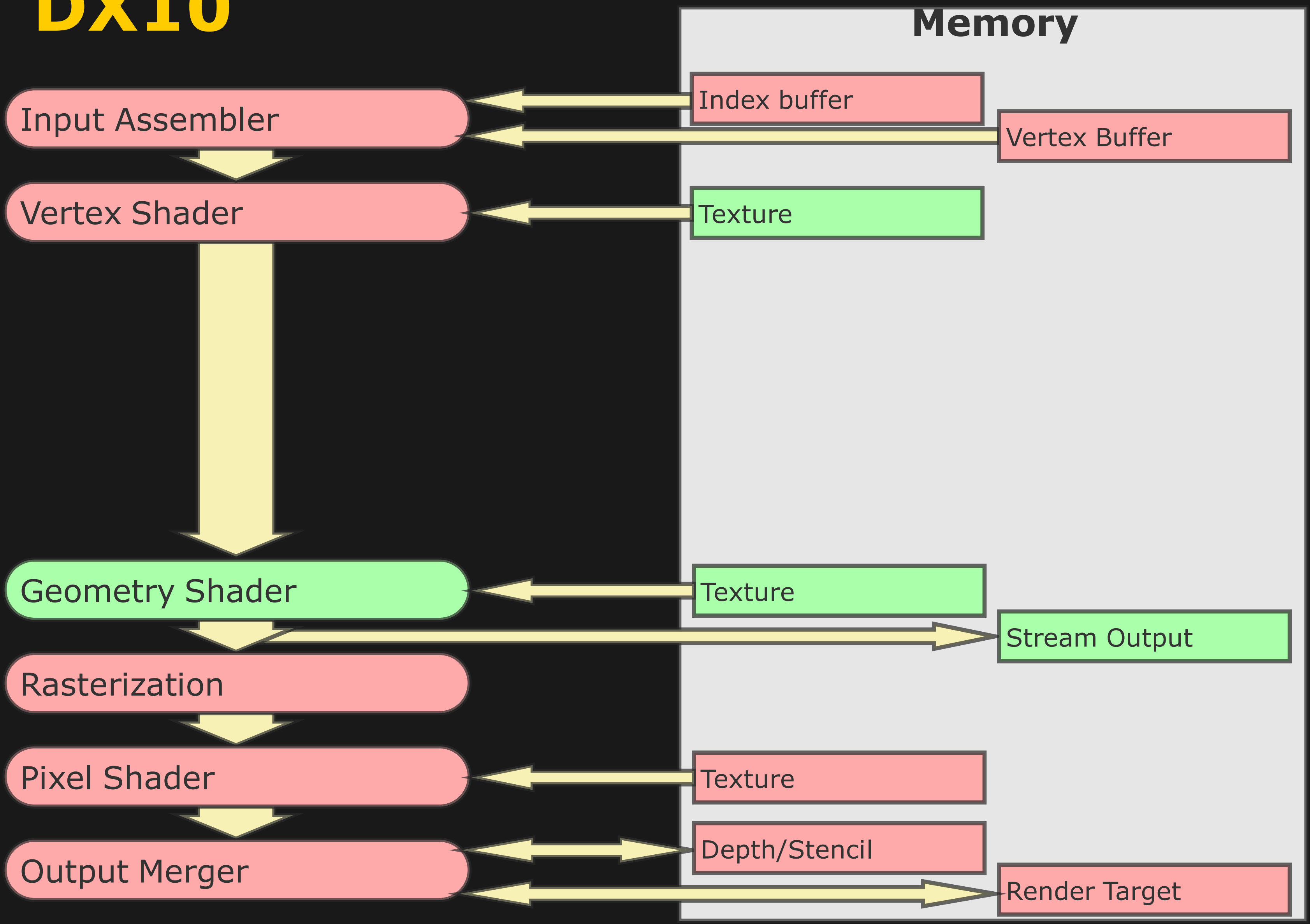
DX10

Remember DX9?



Slide by Tomas Akenine-Möller

DX10

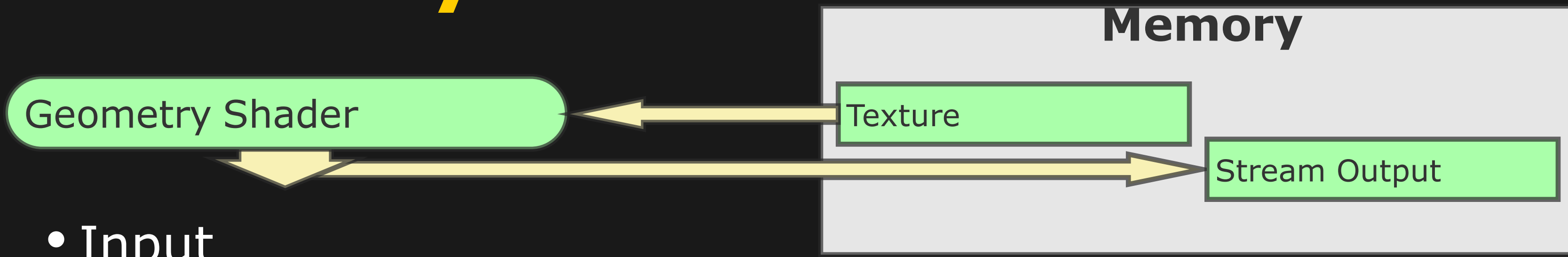


Slide by Tomas Akenine-Möller

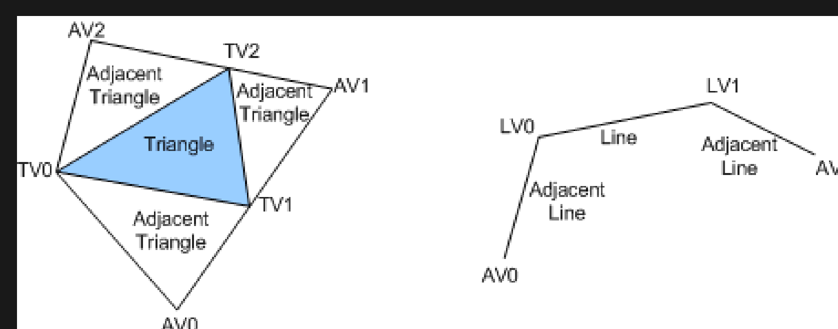
DX10 Increased Shading Capability

- In addition to adding a new pipeline stage, DX10 greatly increased programmability of vertex and fragment stages
  - All shader stages have same limitations (unified shading cores)
  - Pixel/fragment shaders “grow up”
    - Instruction limits, register limits, flow control, etc.
    - Well-specified floating point precision requirements
    - Write up to 8, fp32x4 outputs per fragment

Geometry Shader



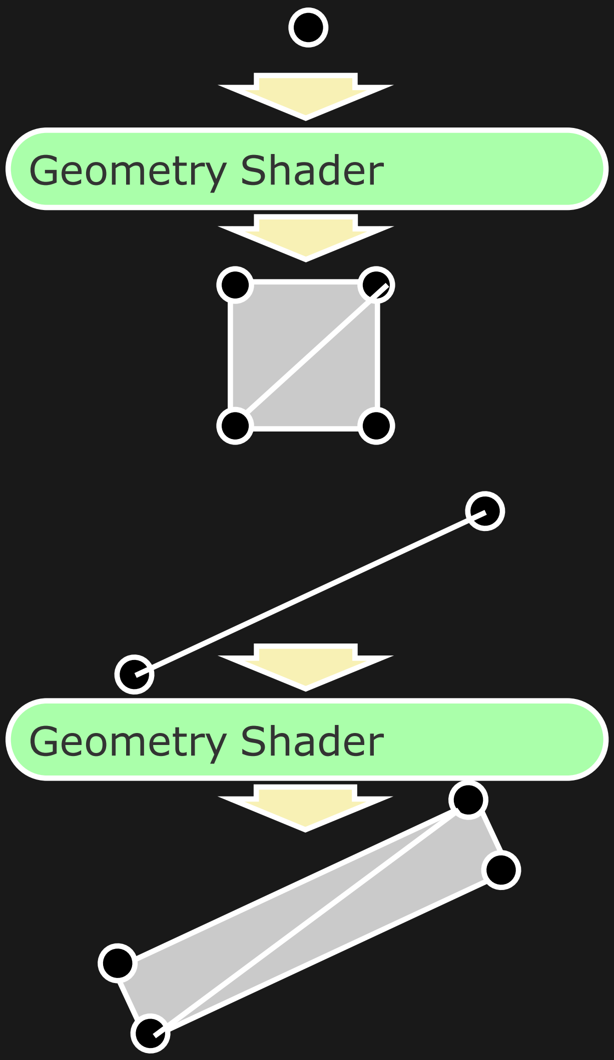
- Input
  - Vertices of a *full* primitive (1 for point, 2 for line, 3 for triangle)
  - Vertices of edge-*adjacent* primitives
  - Primitive ID - allows to fetch or compute per-face data
- Output
  - Point-, line- or triangle strip
  - Sent to rasterizer or vertex buffer in memory
  - Variable output size
    - makes it hard to parallelize



Slide by Tomas Akenine-Möller

Geometry Shader

- Usage examples
  - **Point Sprite Tessellation**: The shader takes in a single vertex and generates four vertexes (two output triangles) that represent the four corners of a quad
  - **Wide Line Tessellation**: The shader receives two line vertexes and generates four vertexes for a quad that represents a widened line.
  - Other uses: generation of shadow volumes and cube maps



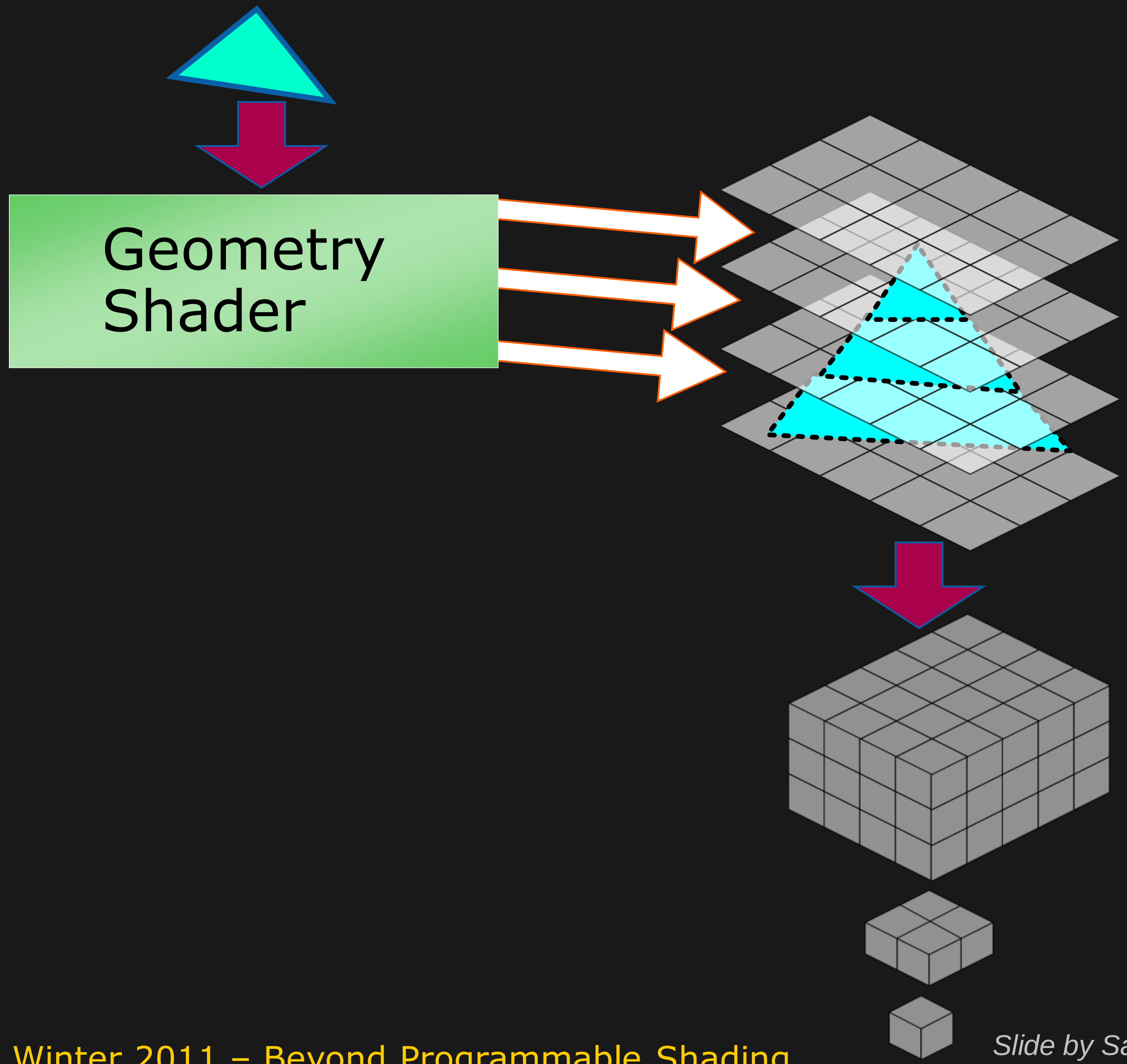
Slide by Tomas Akenine-Möller

Point Sprites and Wide Lines

- Smoke represented by point sprites
  - Points converted to quads in geometry shader
- Hair represented by "wide lines"
  - Lines converted to quads in geometry shader

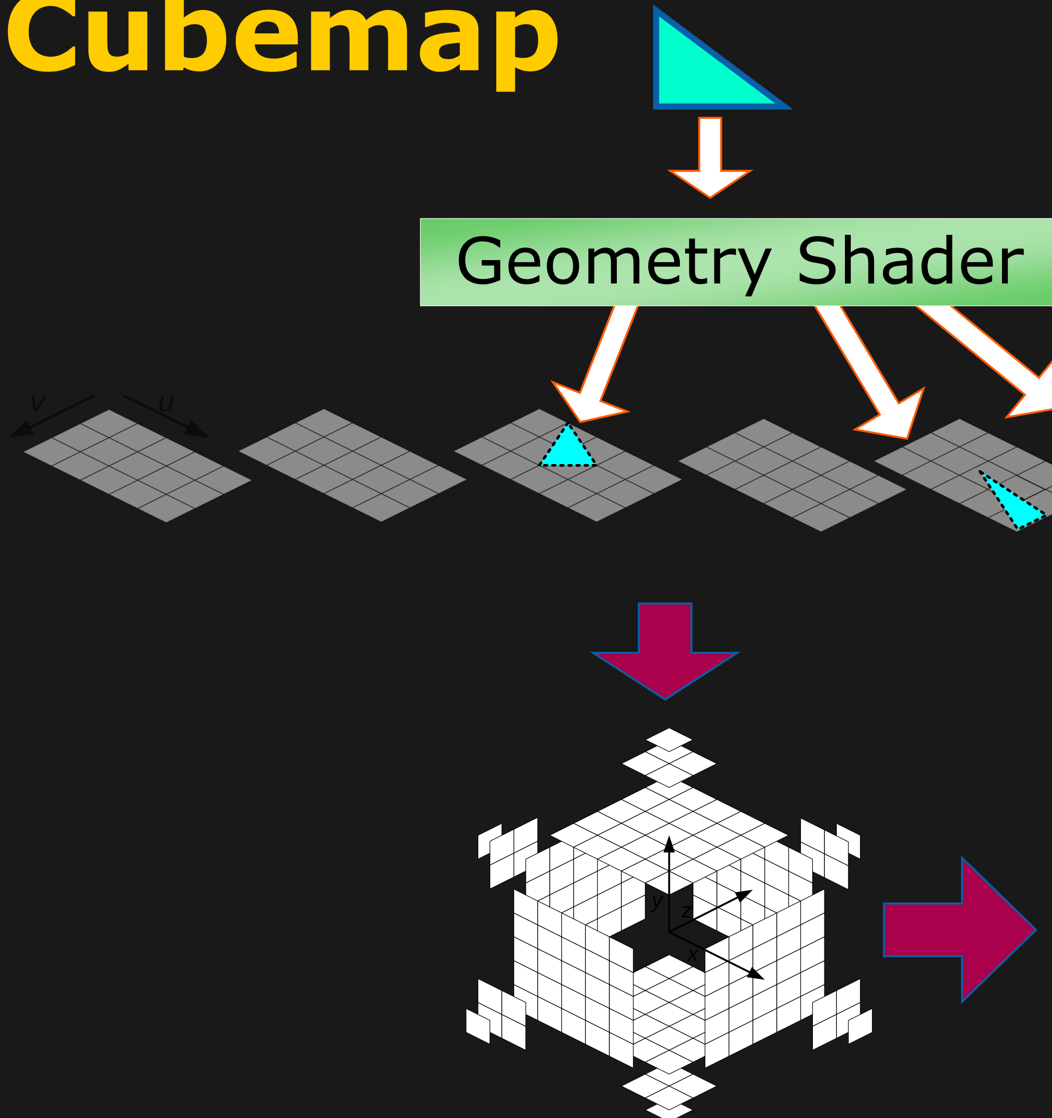
Images from "Adaptive Volumetric Shadow Maps," EGSR 2010, Salvi et al.

Render-To-Volume



Slide by Sam Glassenberg

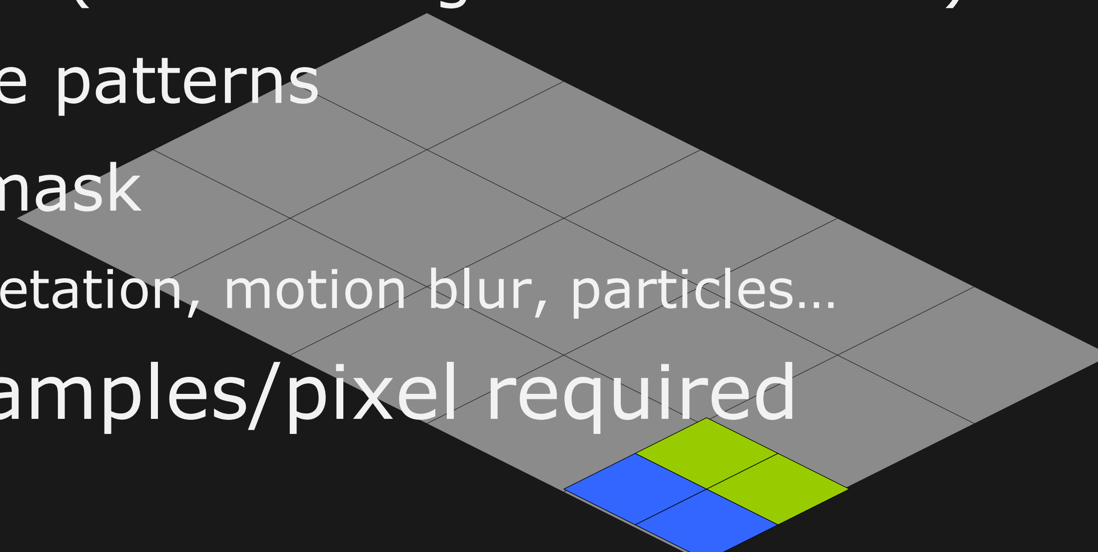
Single Pass Render-To-Cubemap



Direct3D 10.1 Features
Full anti-aliasing control

- Application control over:
  - Multi-sample AA (smooth edges)

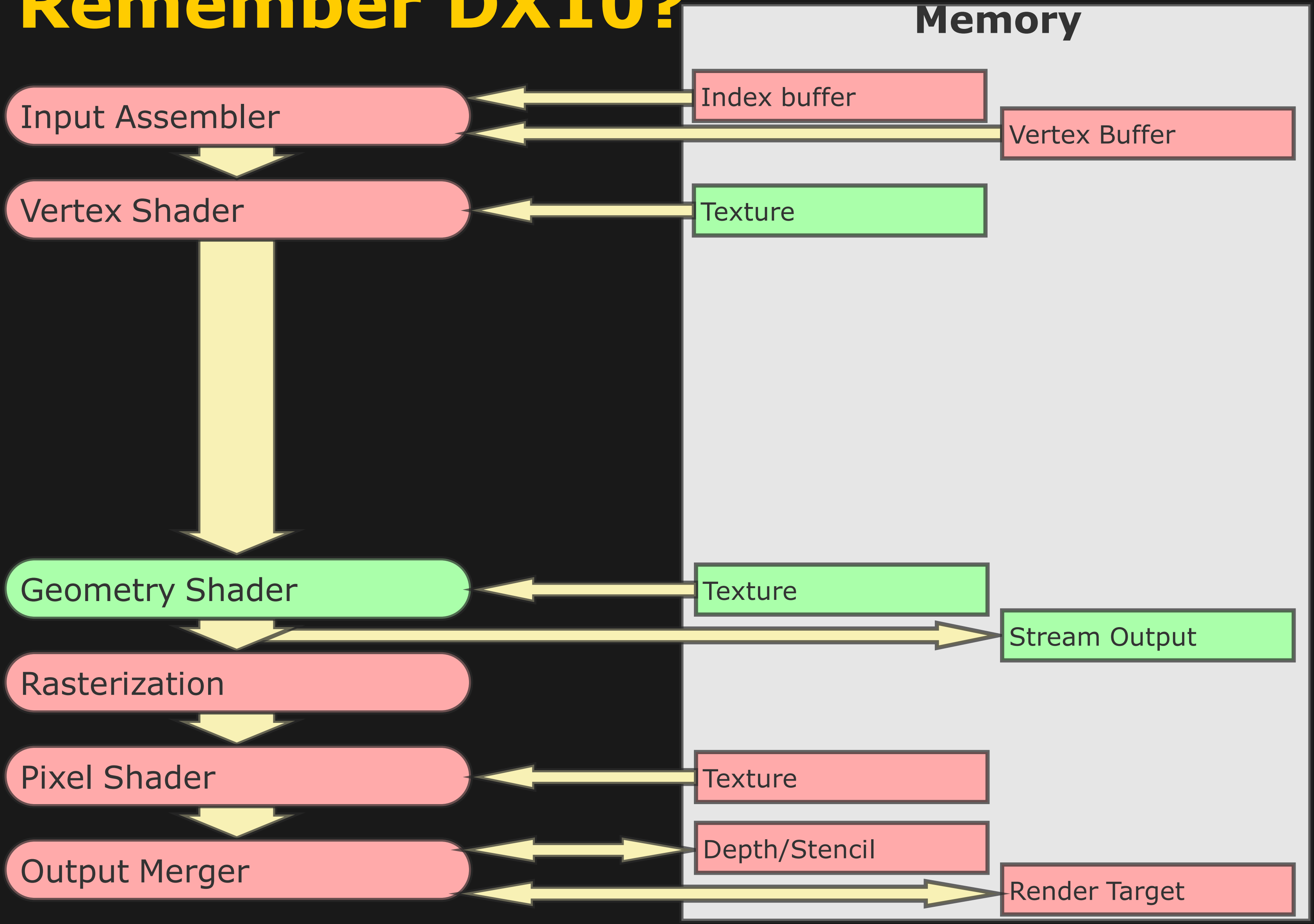
    or
  - Super-sample AA (smooth edges and interior)
  - Selecting sample patterns
  - Pixel coverage mask
    - High-quality vegetation, motion blur, particles…
- Minimum of 4 samples/pixel required



Slide by Sam Glassenberg

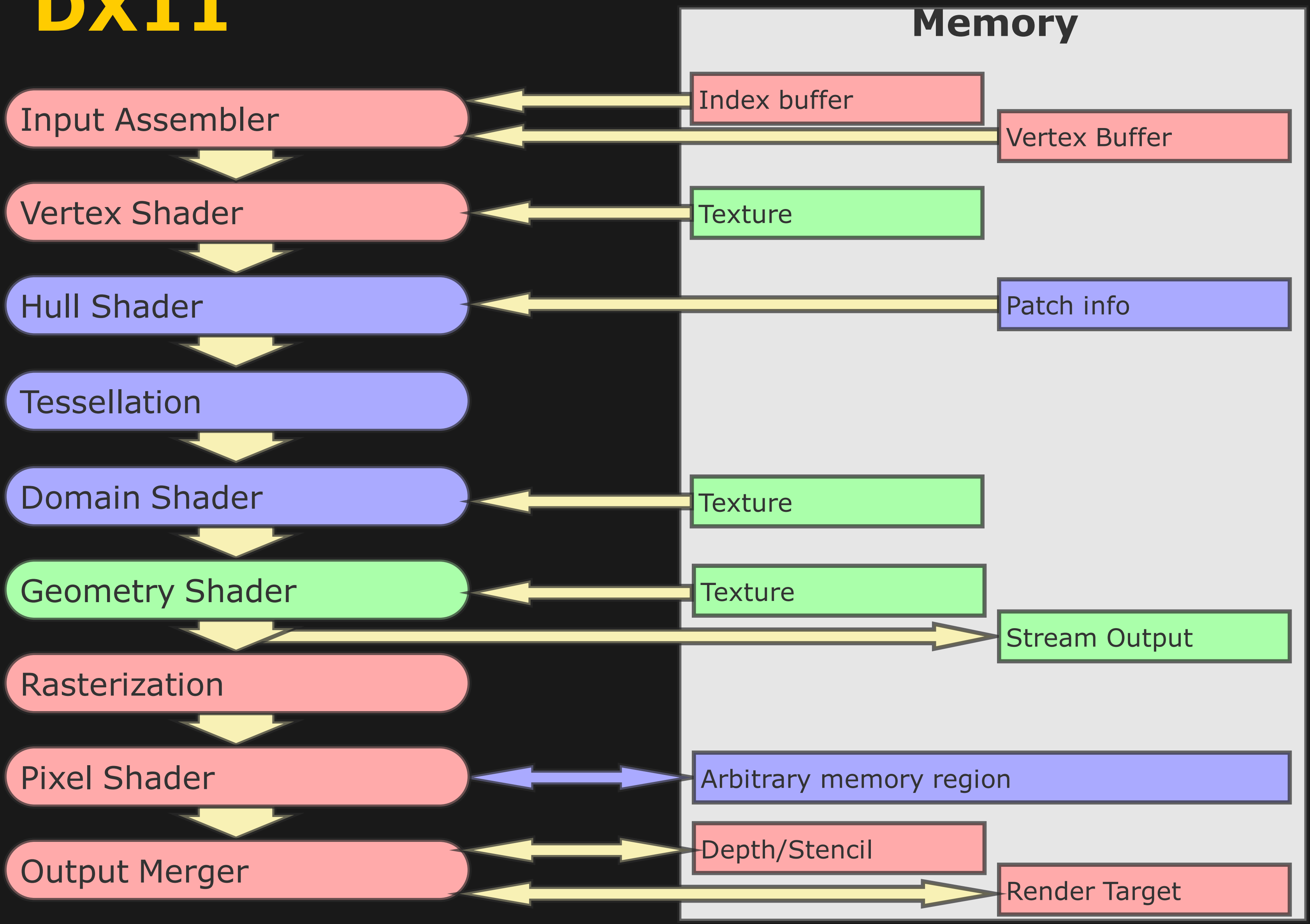
DX11

Remember DX10?



Slide by Tomas Akenine-Möller

DX11

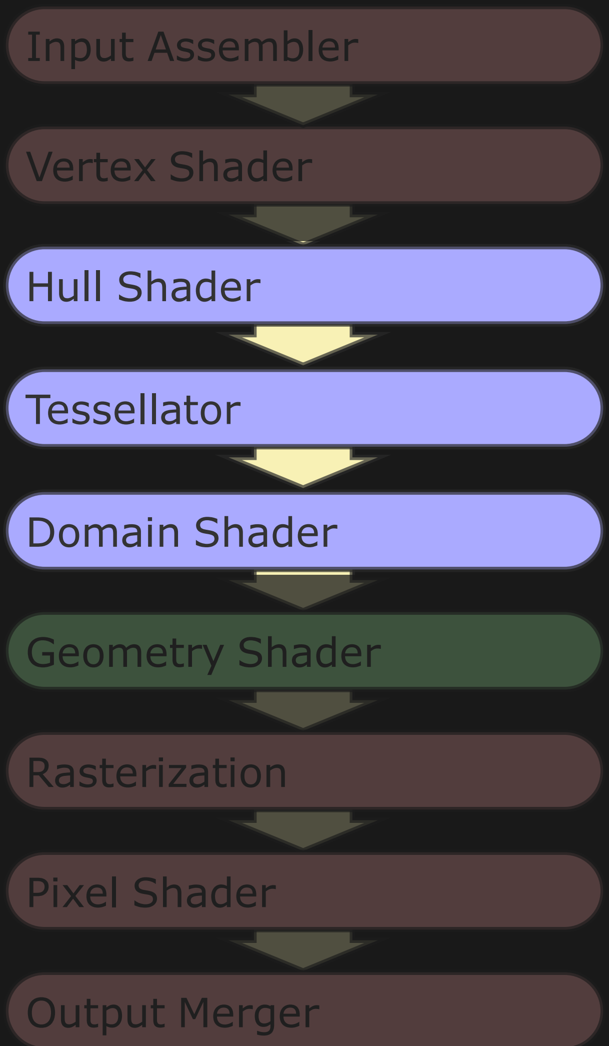


Slide by Tomas Akenine-Möller

DX11 Tessellation example: Displaced subdivision surfaces

Slide by Tomas Akenine-Möller

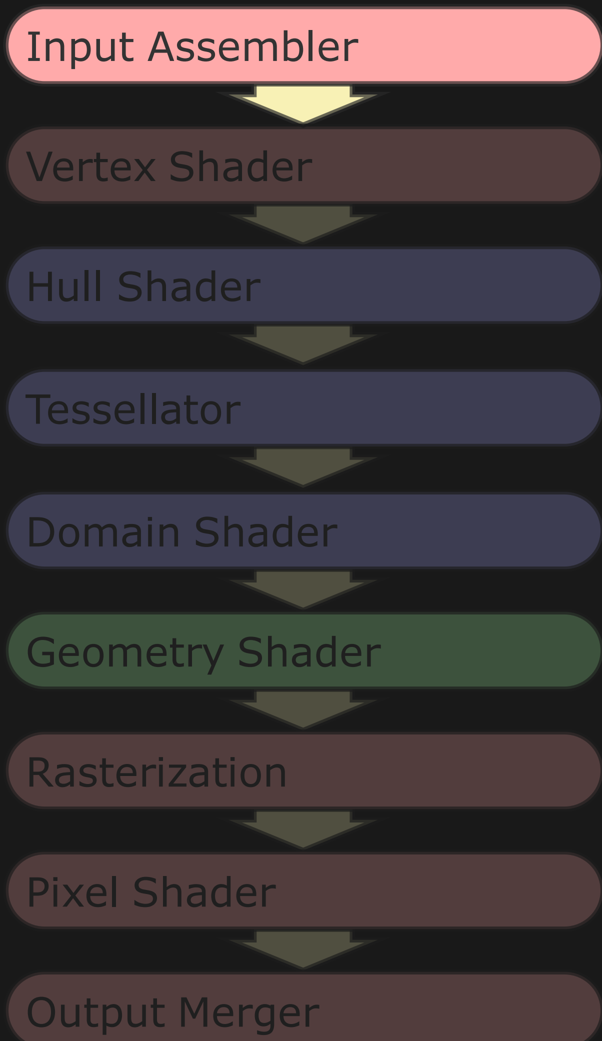
DX11 - new stages for tessellation



- New ***Programmable*** stages
  - Hull Shader
  - Domain Shader
- Fixed Function Stage
  - Tessellator

Slide by Tomas Akenine-Möller

Tessellation - Input Assembler



- New Patch primitive type
- Outputs vertices to vertex shader and patch control points to hull shader

Slide by Tomas Akenine-Möller

Tessellation - Vertex Shader

- One invocation per *control point*
- For example, skin/animate the control points

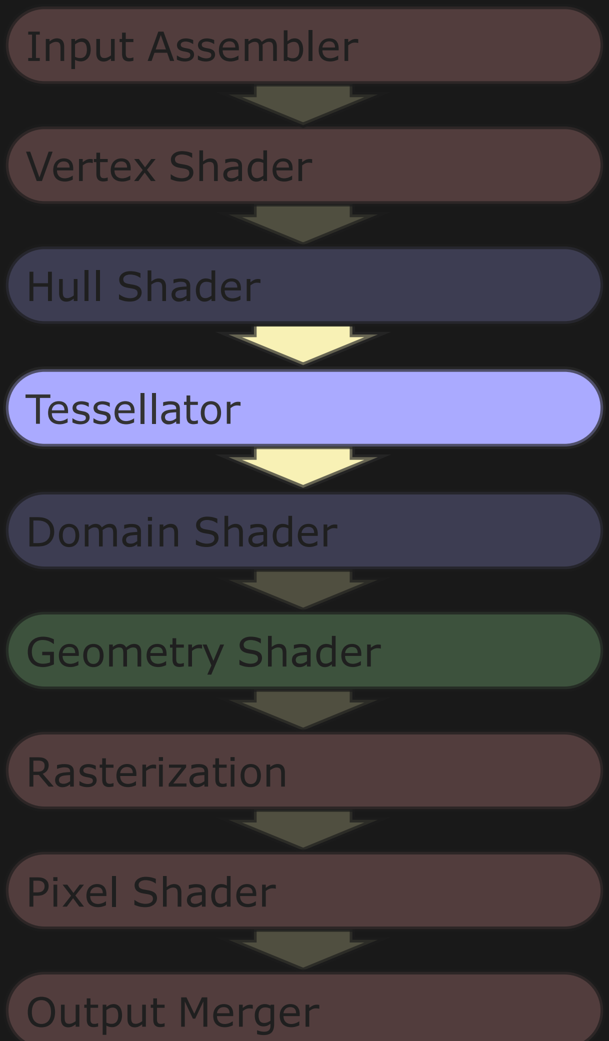
Slide by Tomas Akenine-Möller

Hull Shader

- HS works on an entire patch
- HS can access all the input and output control points
- Typical functions
  - Assigns edge LODs
  - Change the basis

Slide by Tomas Akenine-Möller

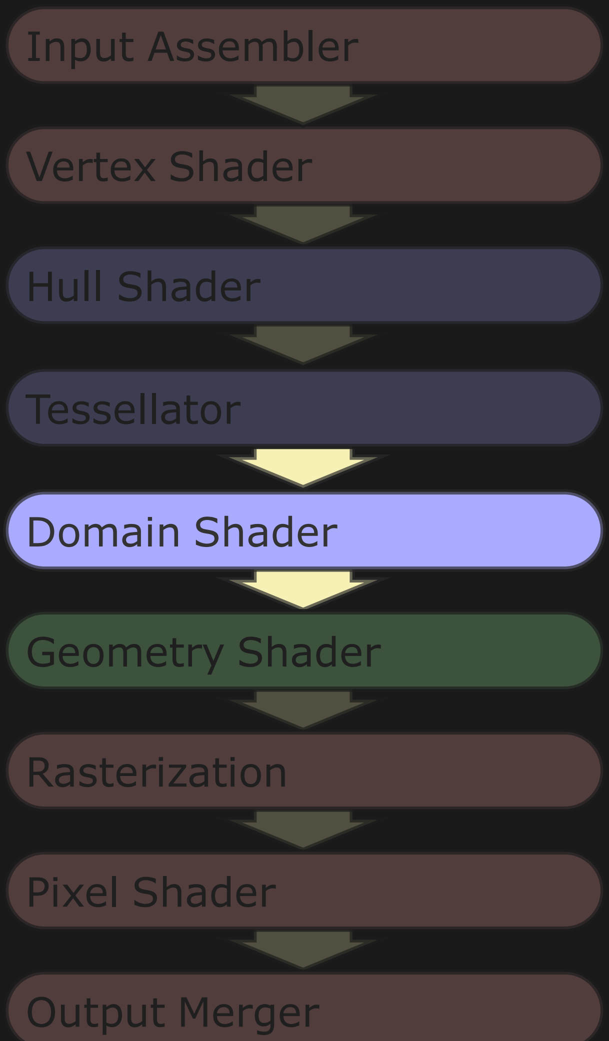
Tessellator



- TS inputs are edge LODs and additional knob values
  - Inner tessellation + tess mode
- TS generates (*u*,*v*) coordinates and connectivity information
- The (*u*,*v*) coordinates are in the domain [0,1]
  - Calculated using fixed point math to ensure water tight edges

Slide by Tomas Akenine-Möller

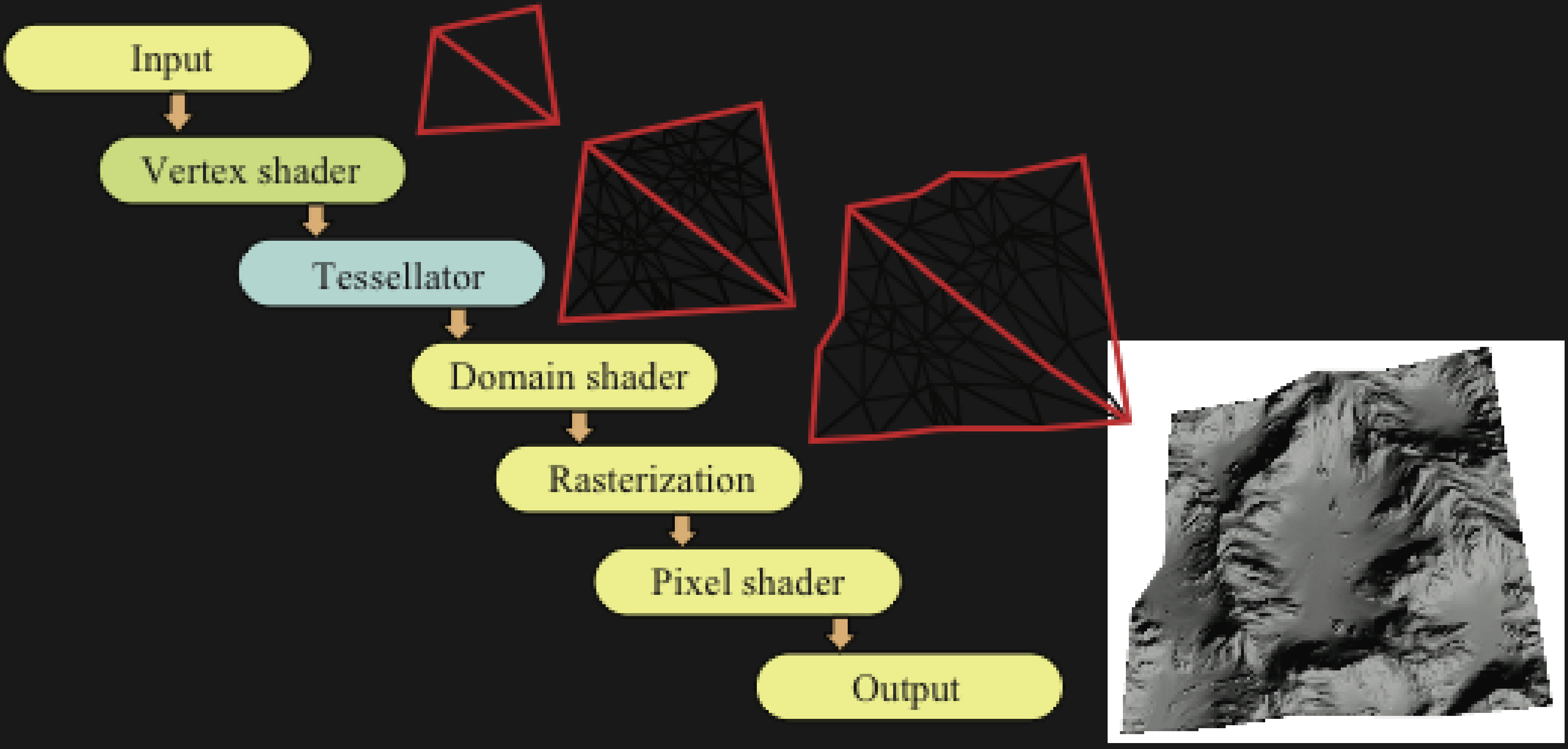
Domain Shader



- One domain shader invocation per (u,v) pair
- DS gets (u,v) from tessellator and control points from HS
- Computes a real 3D point from a domain location (u,v)
  - For example, displace the point using displacement map
  - Project the point
- Calculate auxiliary per vertex data
  - Texture coordinates
  - Tangent space vectors

Slide by Tomas Akenine-Möller

An example through the tessellation pipeline

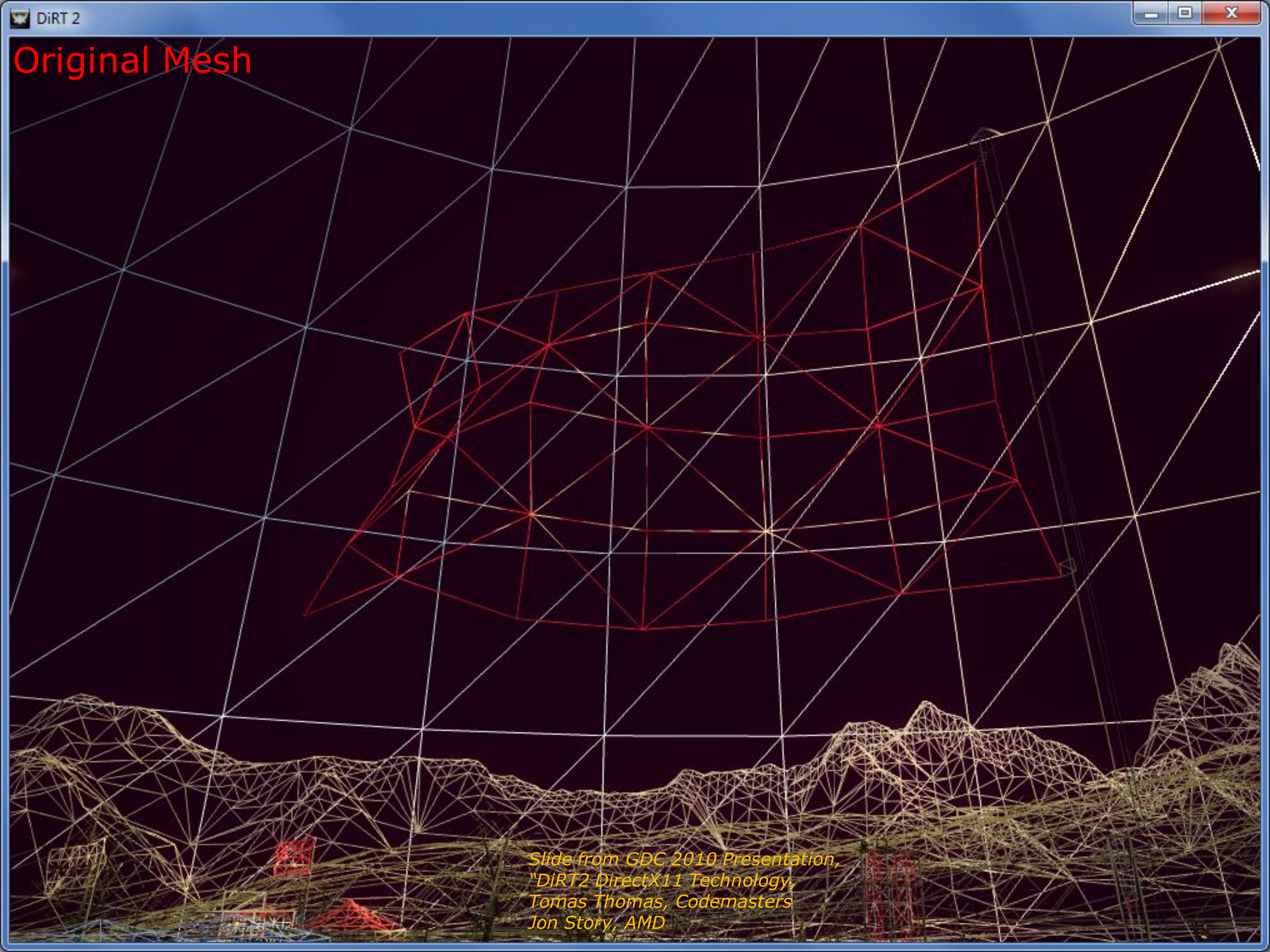


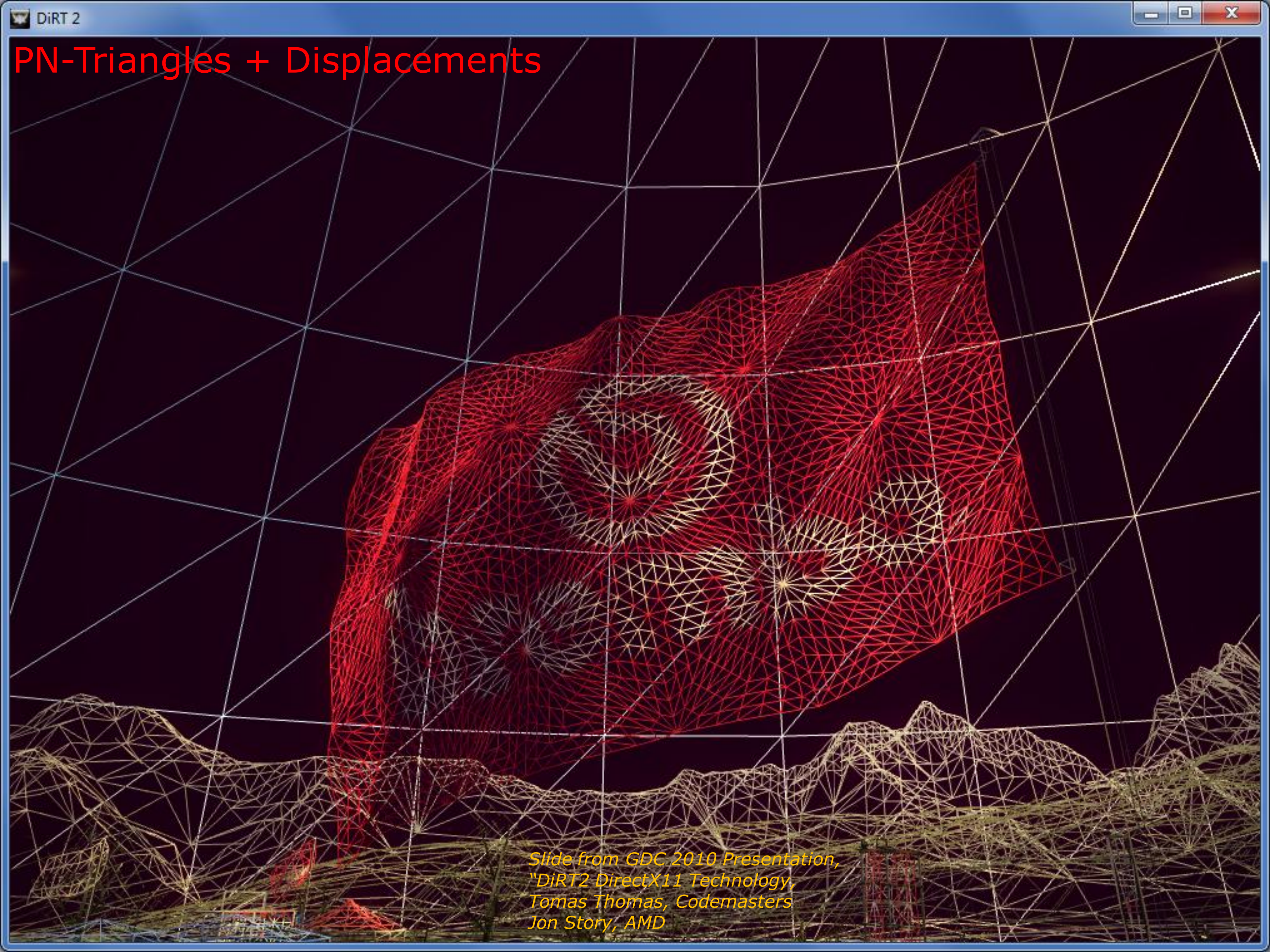
Slide by Tomas Akenine-Möller

DX11 Tessellation Examples: Codemasters DiRT2

Slides from “DiRT2 DirectX 11 Technology“, Thomas and Story, GDC 2010


DiRT 2
Original Mesh
brembo
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD


DiRT 2
Original Mesh
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD


DiRT 2
PN-Triangles + Displacements
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD


DiRT 2
PN-Triangles + Displacements
brembo
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD


DiRT 2
Original Mesh
brembo
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

Original Mesh

Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

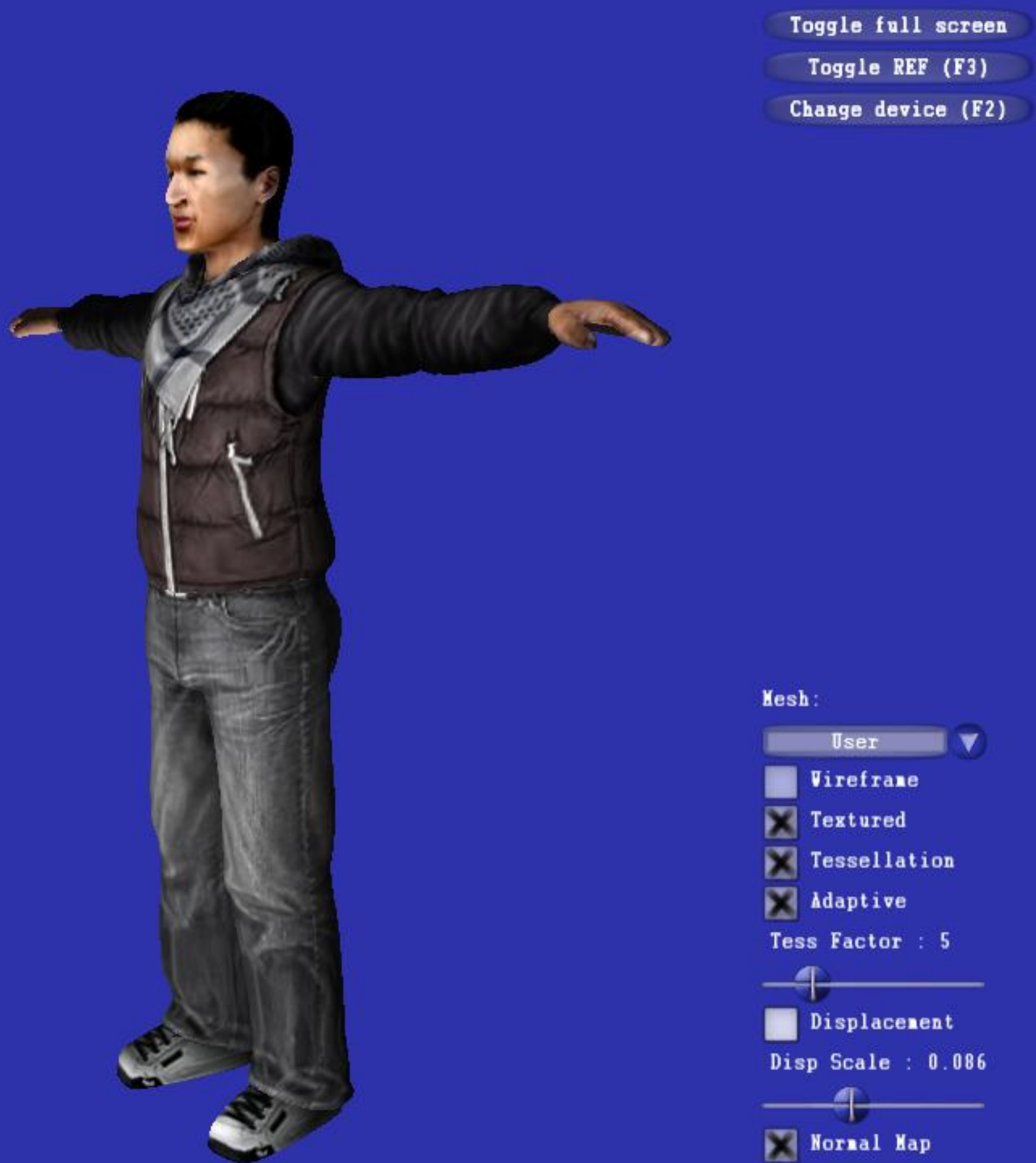
Original Mesh

Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

PN-Triangles

Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

PN-Triangles

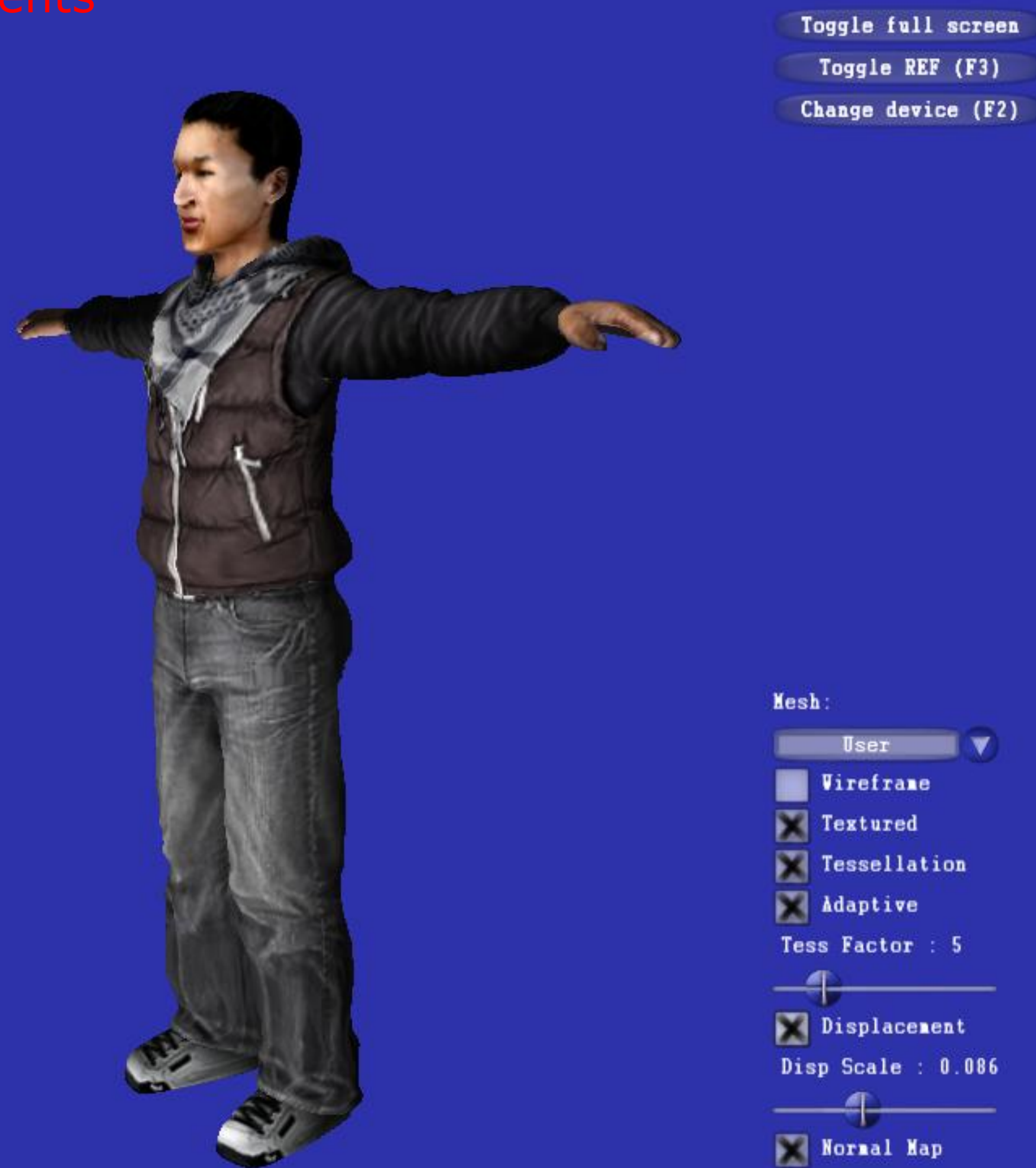


Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

PN-Triangles + Displacements

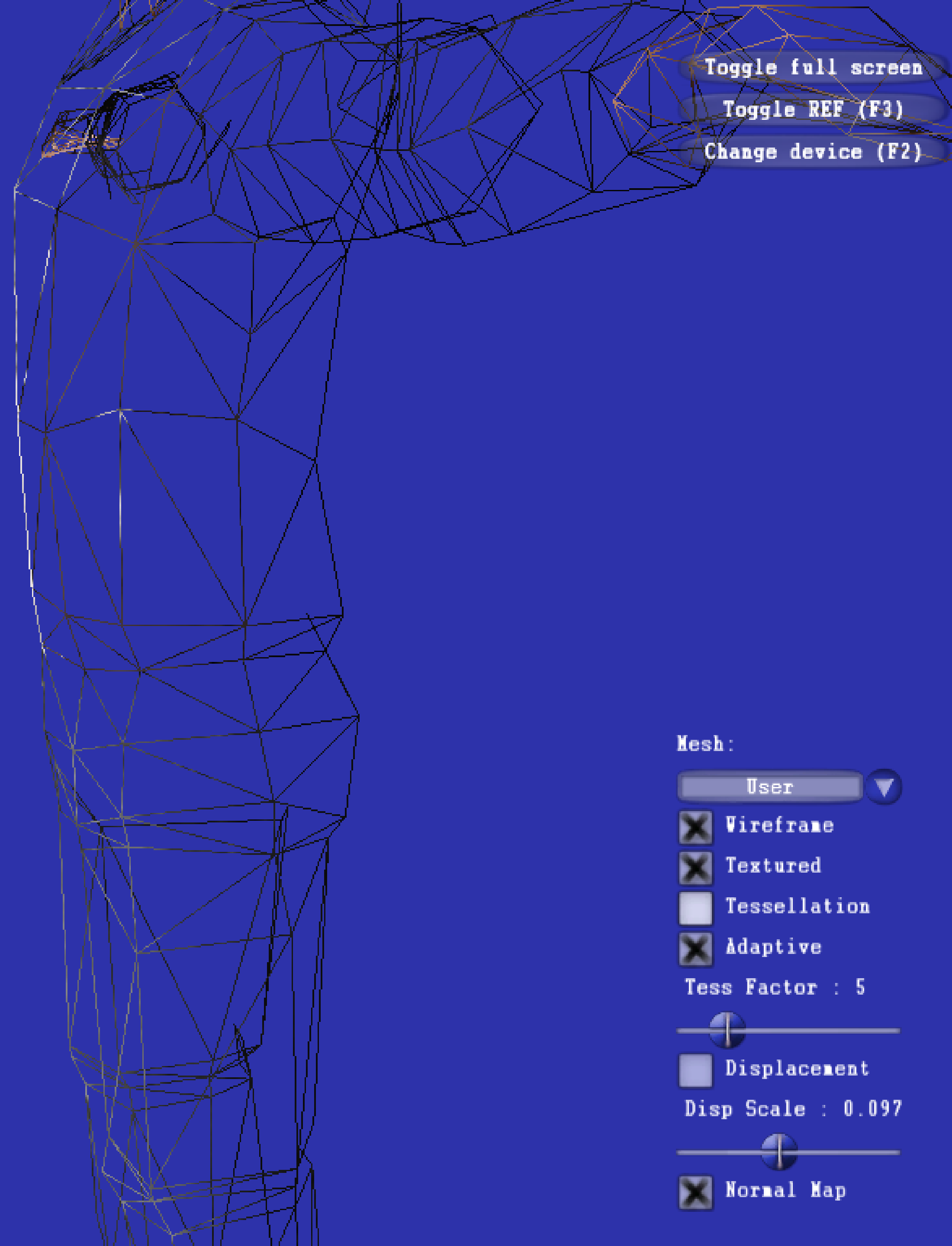
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

PN-Triangles + Displacements



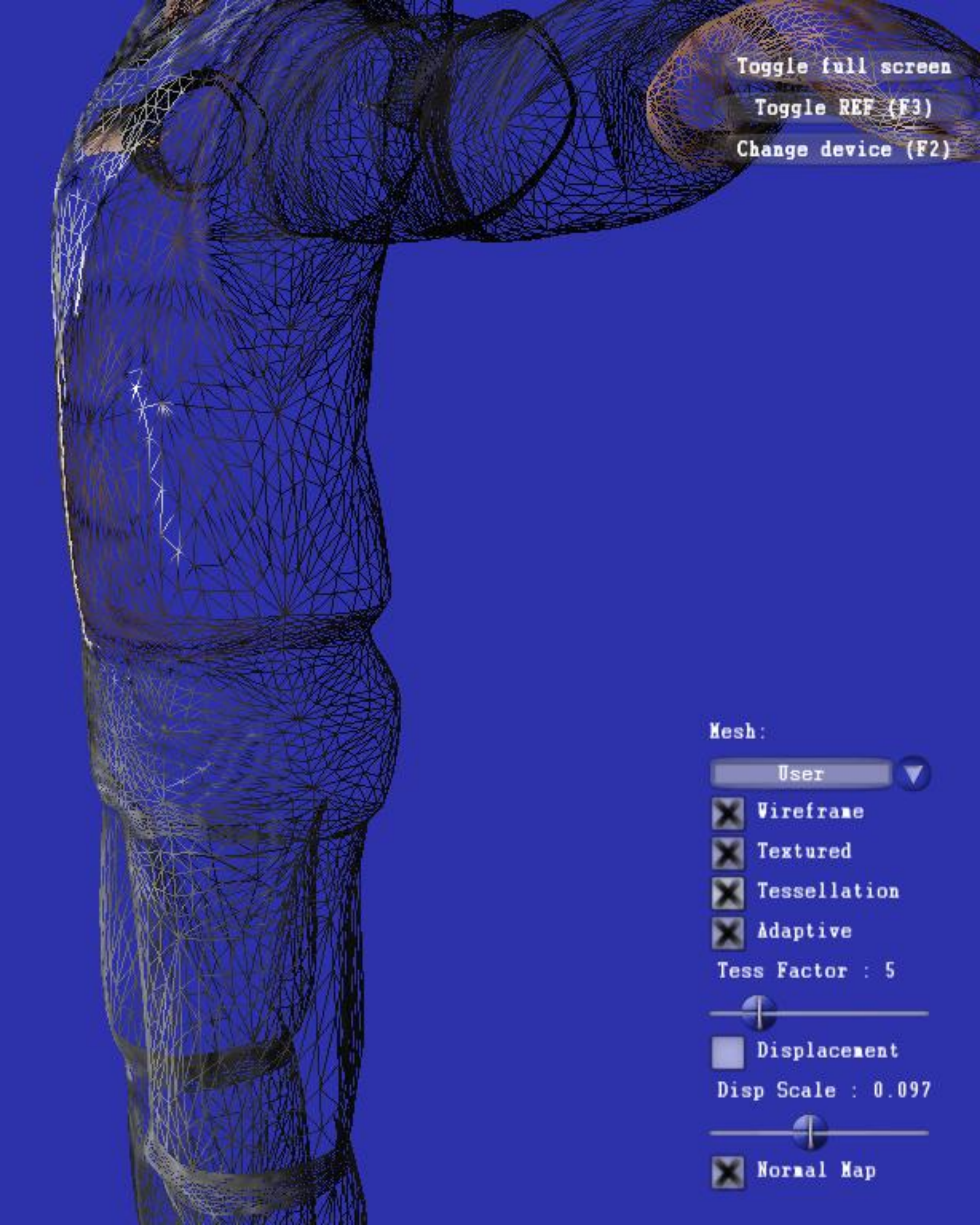
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

Original Mesh



Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

PN-Triangles



Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

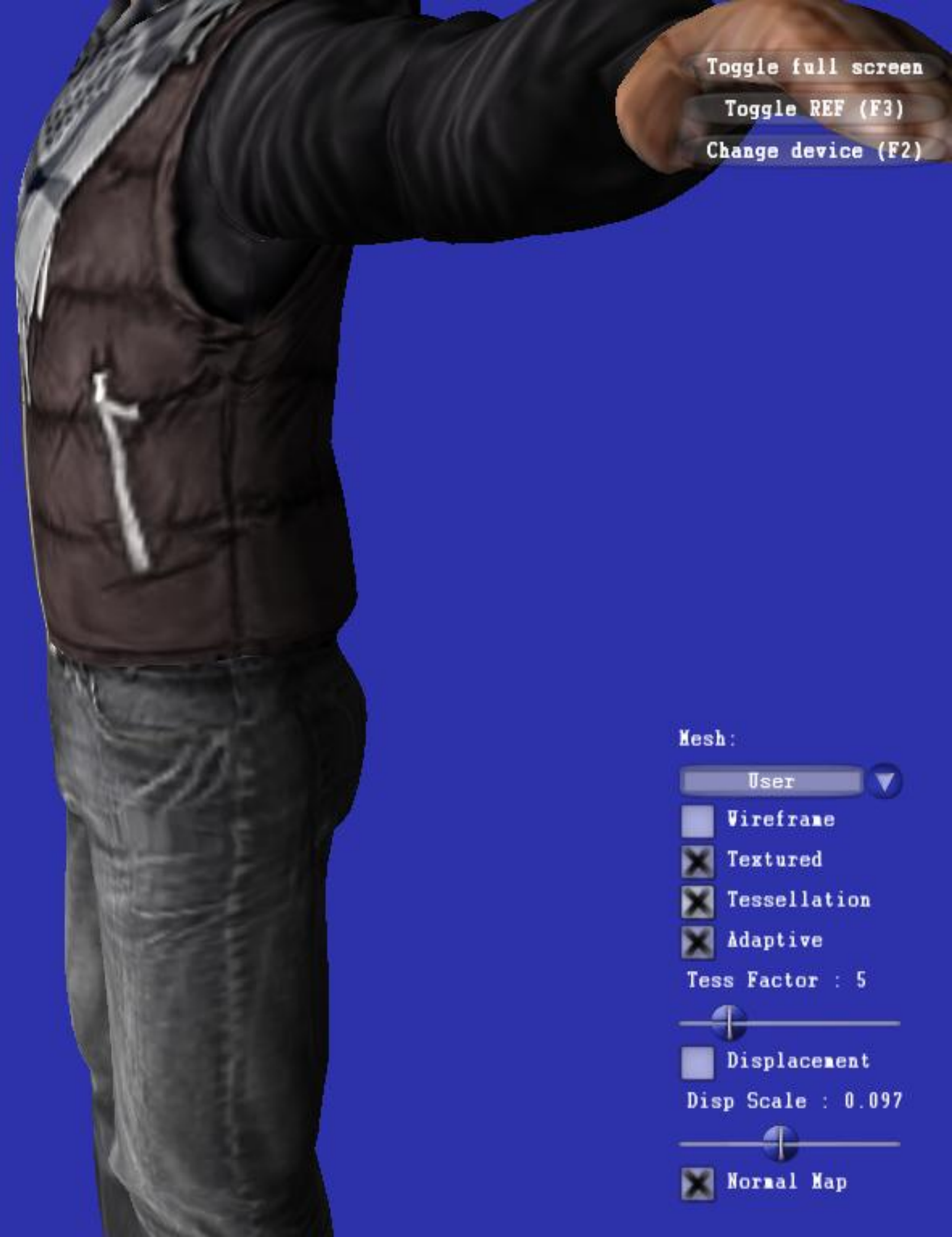
PN-Triangles + Displacements

Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

Original Mesh

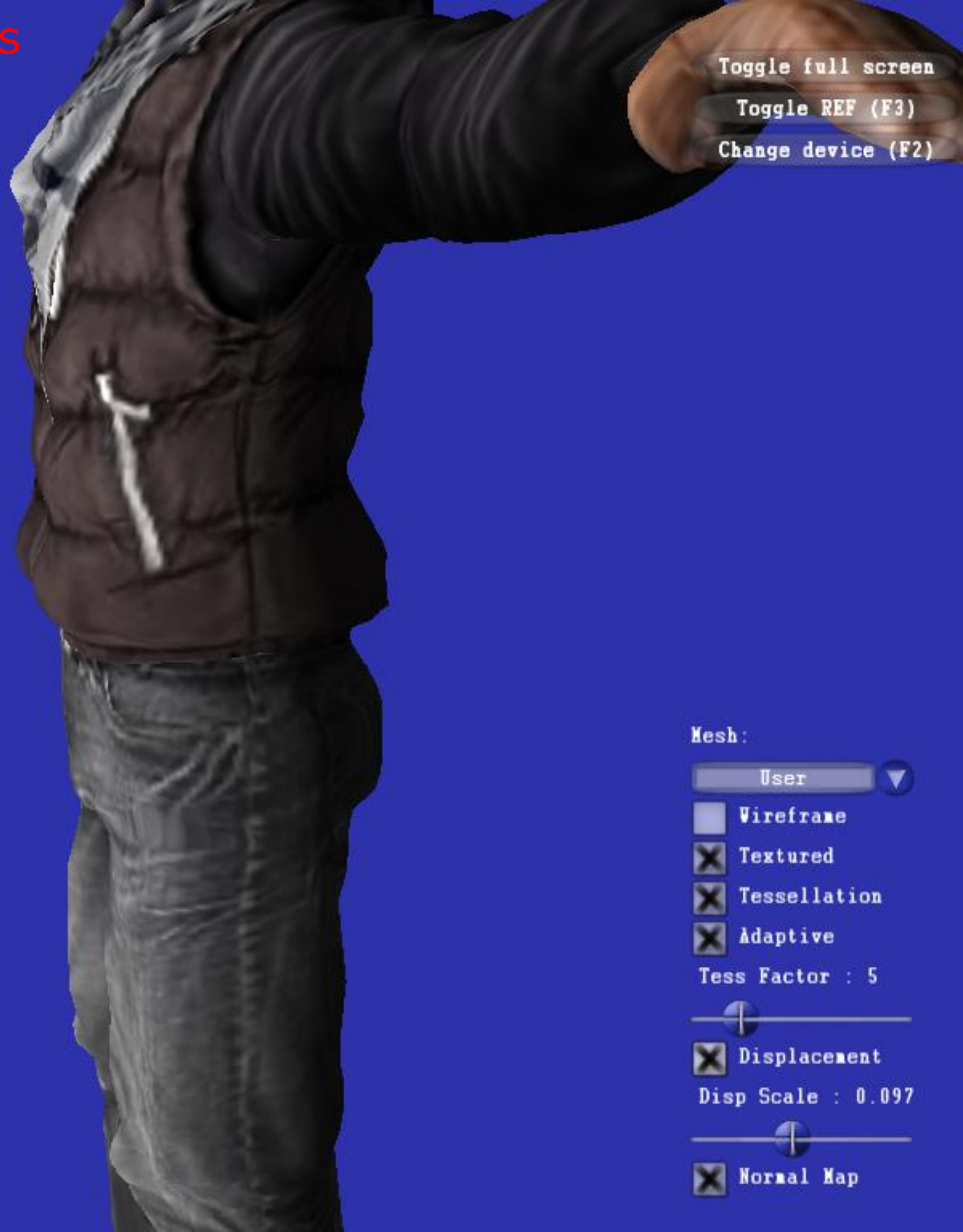
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

PN-Triangles

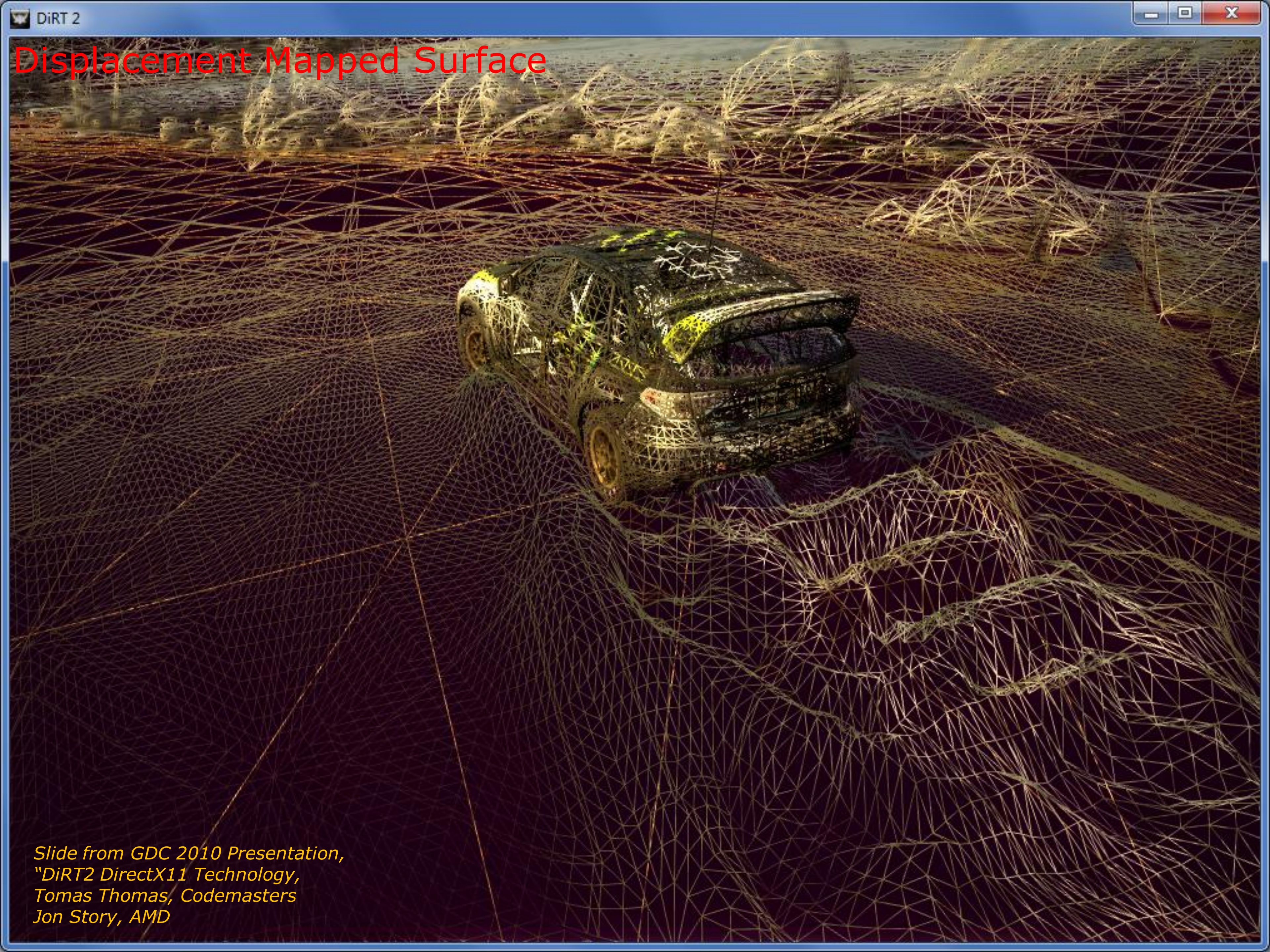


Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD

PN-Triangles + Displacements



Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD


DiRT 2
Displacement Mapped Surface
Slide from GDC 2010 Presentation,
"DiRT2 DirectX11 Technology,
Tomas Thomas, Codemasters
Jon Story, AMD


Displacement Mapping: OFF
REPLAY
ATI RADEON PREMIUM GRAPHICS
x4

Displacement Mapping: ON

DX11 Pixel Shader

DirectX11 Pixel Shader Changes

- Read and write anywhere in memory
  - "Unordered access views" (UAVs)
  - Pixel shaders can now write somewhere other than pixel position (!)
- Atomic read-modify-write operations
  - add, subtract, min, max, compare-exchange, …, or, xor

DX11 Pixel Shader Implications

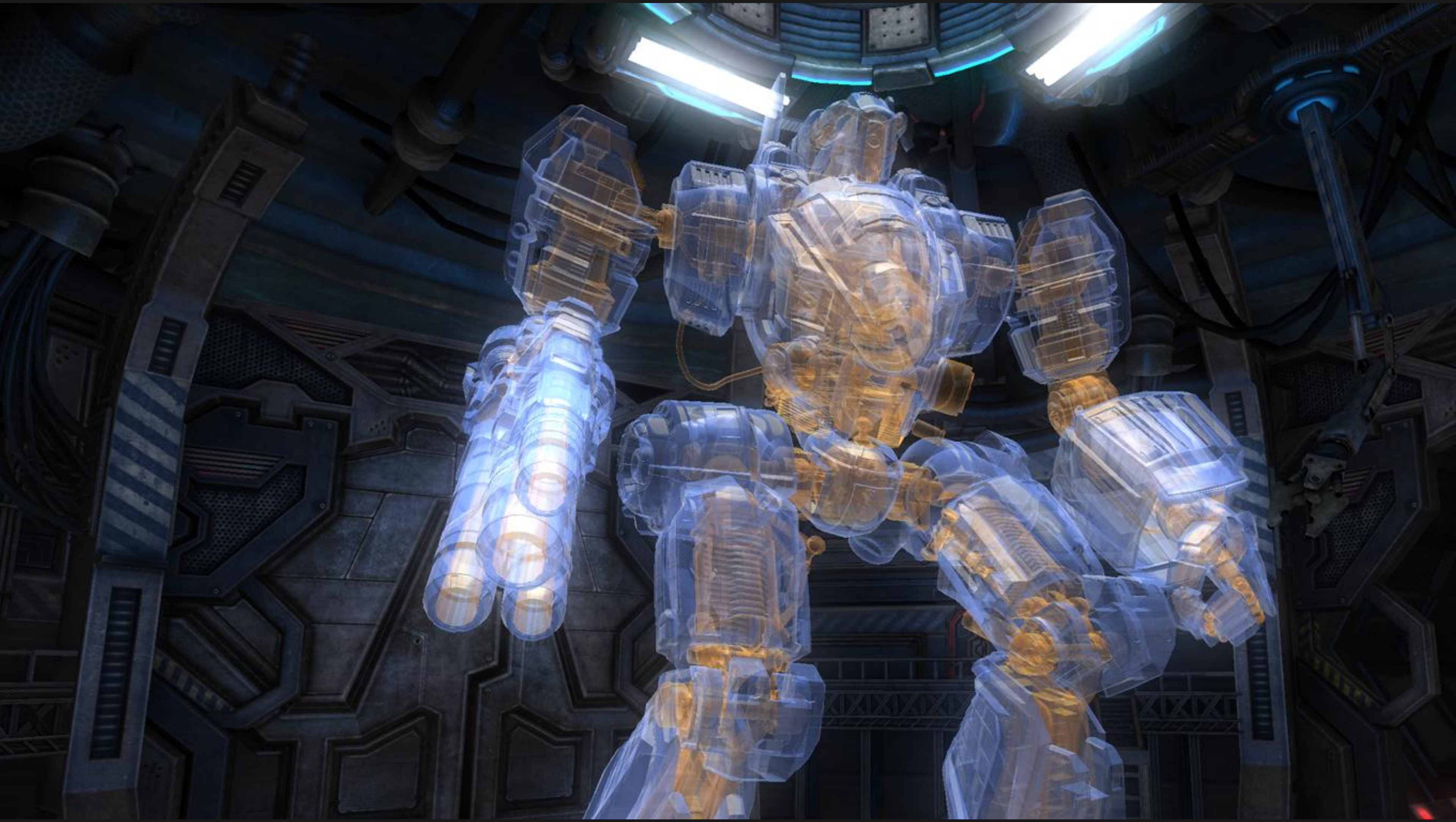
- ***Pixel shaders can build data structures other than images***
  - Scatter/gather and atomic operations
  - “Render to user-defined data structure”

- Limitations
  - Atomic RMW operations only on 32-bit integers (no atomic struct RMW)
  - No critical sections / mutexes

Real-time A-Buffer Generation

Pixel Shader 5 Example

Order Independent Transparency I



"Real-Time Concurrent Linked List Construction on the GPU,"
Yang et al., EGSR 2010

The Problem with Transparency

With Sorting

"Real-Time Concurrent Linked List Construction on the GPU,"
Yang et al., EGSR 2010

Skeleton hidden

No Sorting

Arm appears in front of body

Sorting is hard!

Solution: Sort Fragments Per Pixel

- Create A-Buffer using DX11 rendering pipeline
  - "The A -buffer, an antialiased hidden surface method," Carpenter, SIGGRAPH 1984
- Render pass 1
  - Store linked list of fragments per-pixel in 2 separate buffers
    - UAV 1: Storage for all [RGBA,Z,next] values from all fragments
    - UAV 2: "Image-sized" storage for a head pointer per pixel position
- Render pass 2
  - Full-screen pass that sorts and blends fragments to create pixel color

DX11 A-Buffer

- Pros
  - Correct solution to order-independent transparency
  - Runs at interactive rates on current hardware
- Cons
  - A-buffers use unbounded amount of memory
  - Performance variable depending on order fragments stored in memory

DX11 ComputeShader / DirectCompute

- "Out-of-pipeline" new programming model for "general" computation, designed to interact tightly with the graphics API (language is HLSL)
- Similar in semantics and capability to OpenCL and CUDA
- We will discuss DirectCompute later in the course

Summary

- Jump from DX9 to DX11 (OGL2 to OGL4)
  - Adds 4 new pipeline stages (hull, tessellator, domain, geometry)
  - Makes shader stages far more general compute engines
  - Enables users to render to user-defined data structures (and defer visibility determination)
  - Greatly relaxes memory model of shaders (especially scatter/atomics)
  - Adds DirectCompute, which begins to blur lines between pipeline and user-defined parallel programs
  - Is still new wrt adoption in the game world (but a number of DX11 titles shipping)
- Get to know DX11/GL4 well---you can do *a lot* within the confines of the rendering pipeline

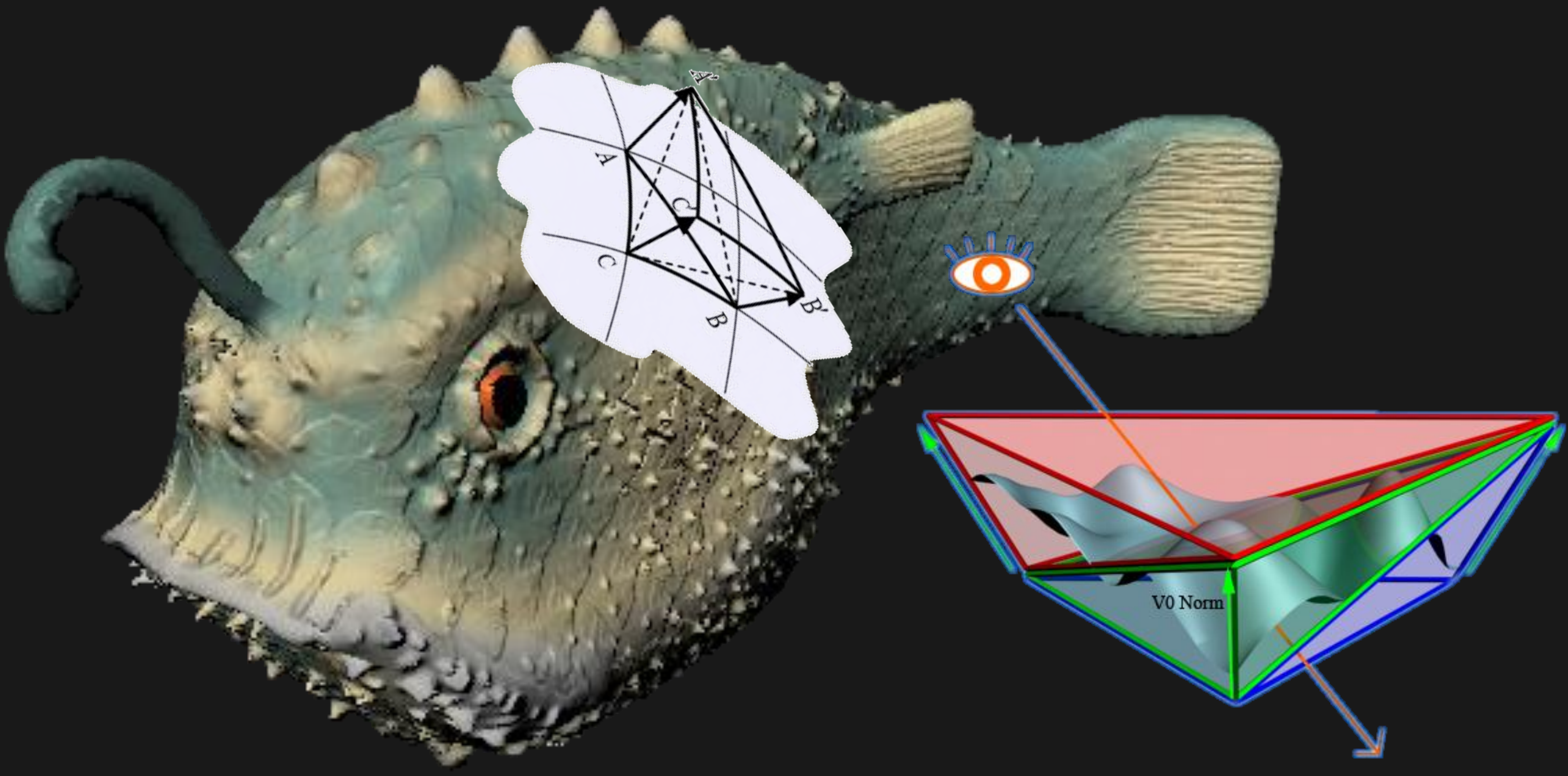
Backup

Geometry Shader Example
Shadow volume generation

Slide by Sam Glassenberg

Geometry Shader Example
Generalized displacement maps

- Displacement Mapping (Direct3D 10)



Slide by Sam Glassenberg

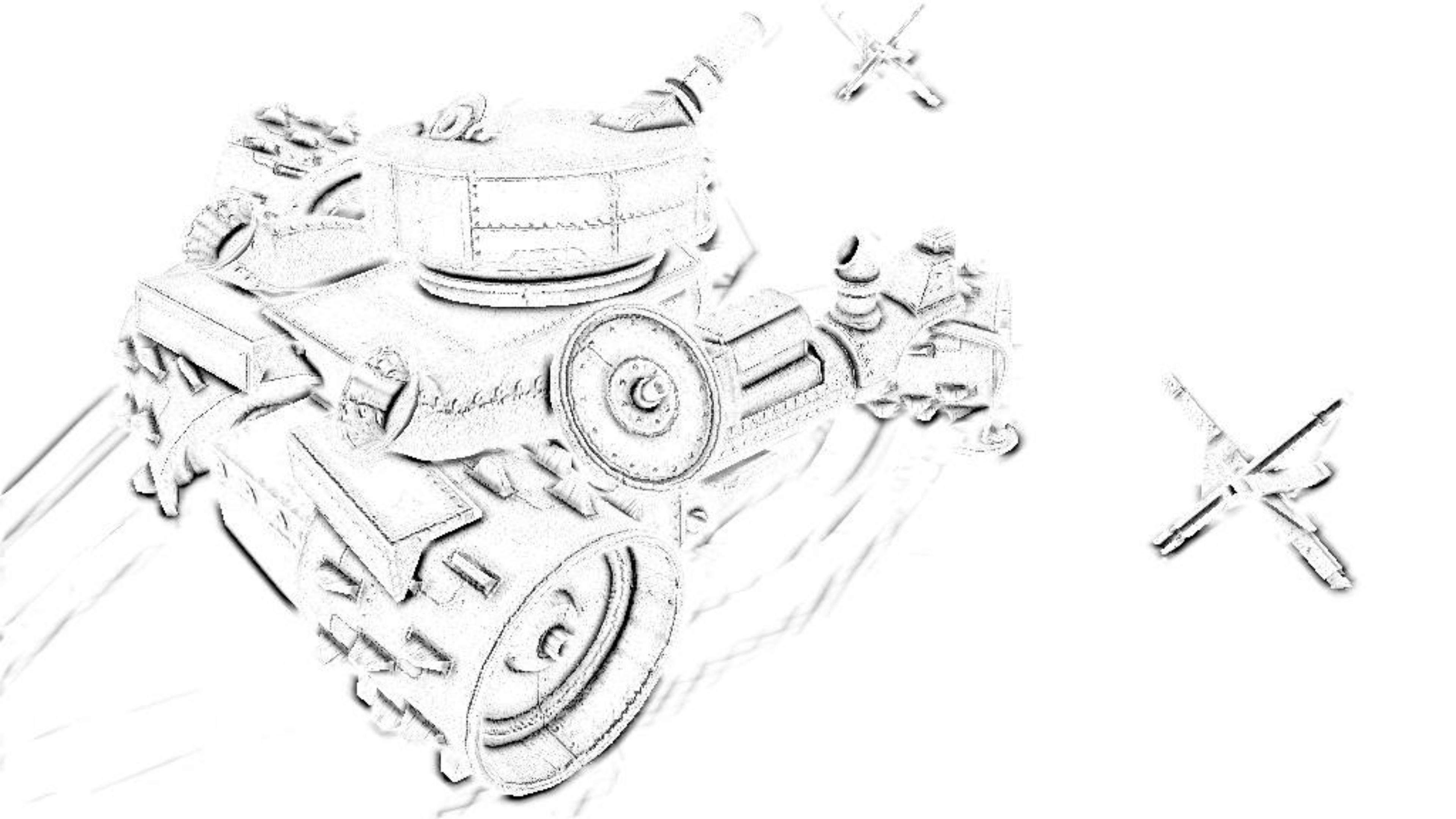
New DX10 and DX10.1 features heavily used for shadow techniques in production games

Examples of some from *"High Quality Direct3D 10.0 & 10.1 Accelerated Techniques", Story and Gruen, GDC 2009*


HDAO On (depth only)
HDAO Off (depth only)
- 40 Gathers
- No filtering needed

HDAO On (depth & normals)

- 80 **Gathers** (could use alot less)
- No filtering needed



Tom Clancy's HAWX
Publisher: Ubisoft
Developer: Ubisoft Romania

Stormrise
Publisher: SEGA
Developer: The Creative Assembly Australia

BattleForge
Publisher: EA
Developer: EA Phenomic


From DICE’s Frostbite Engine:
Uniform Shadow Filtering

From DICE's Frostbite Engine:
Unique Weight Shadow Filtering


From DICE’s Frostbite Engine:
Standard 2x2 Shadow Filtering


From DICE's Frostbite Engine:
5x5 Unique Weight Filtering

Tom Clancy`s HAWX
Publisher: Ubisoft
Developer: Ubisoft Romania
Blurred VSM

Tom Clancy`s HAWX
Publisher: Ubisoft
Developer: Ubisoft Romania
Unique Weight Shadows