Visual Engineering of SSX

GDC 2002

Introduction

What is SSX?

PS2 Architecture

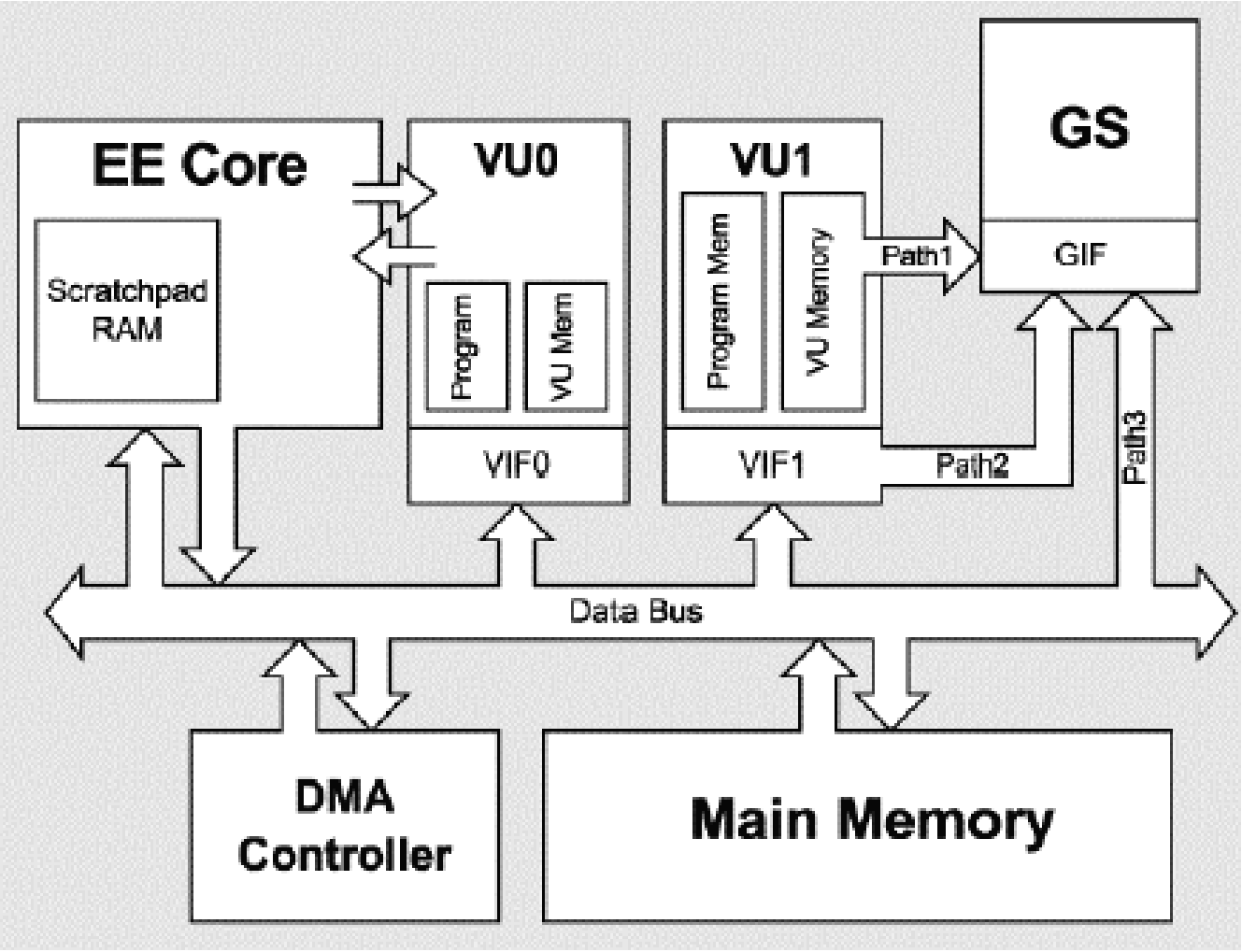
SSX rendering techniques

- Characters
- Facial Animation
- Terrain system
- World lighting

Pipelines

Next generation PS2 games

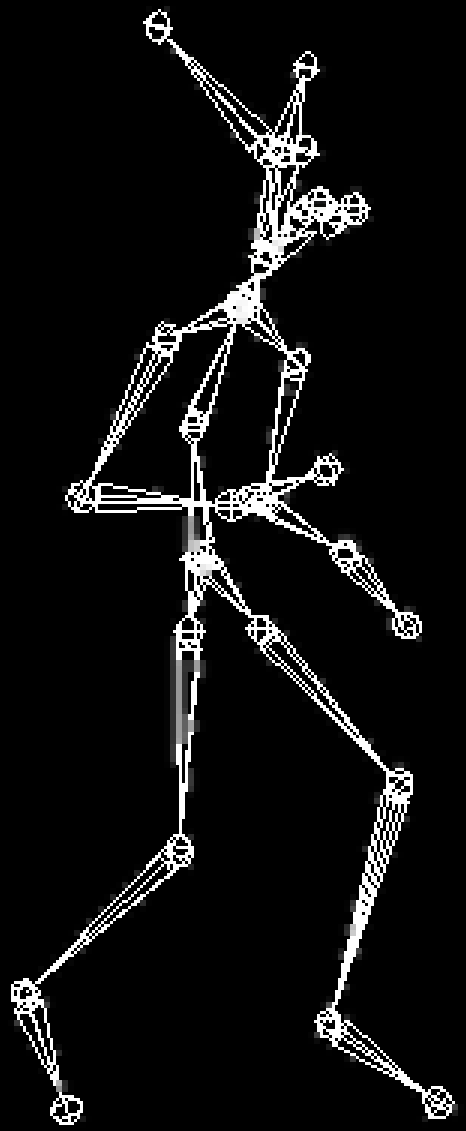
PS2 Architecture



Character Models

Skeleton based animation

- Bones represent animation
- Positional and rotational information per bone
- Vertices weighted to multiple bones
- Animation sharing between models
- Generally not as fast as key frames (static model)
- Small data footprint per animation (compared to key frames)
- Highly compressible



Skeleton Models

SSX & SSX Tricky

- 25 bones in skeleton
- 2500 (SSX) and 3000 (SSX Tricky) polygons per model
- 6 models on screen

Post Transform weighting (SSX)

- Vertices stored in local bone space

$$w_i = \text{weighting factor to bone } i$$

$$v_{Bi} = \text{vertex in bone space } i$$

$$M_{Bi} = \text{Matrix Bone } i \text{ (Bone space to world space)}$$

$$v = w_0 \times (v_{B0} \times M_{B0}) + w_1 \times (v_{B1} \times M_{B1}) + \ldots$$

Skeleton Models

Post transform implementation

- All bone matrices transferred to VU1
- Vertices transformed and weighted using VU1
- Post weighting Indexed triangle strips
- No weighting for normals
- VU1 double buffer layout
  - 2 input buffers (DMA in)
  - 2 output buffers (to GS)
  - Synchronized by micro program activation
- 3 directional plus one ambient light per vertex

Skeleton Models

Blended matrices (SSX Tricky)

- generate weighted matrices at run time
- No post transform weighting required
- quantized weighting to limit the number of matrices
- Good cross platform approach (GameCube)

$$M^{*}_{Bi} = \text{Matrix Bone } i \text{ (Character pose space to world space)}$$

$$v_p = \text{vertex in default character pose space}$$

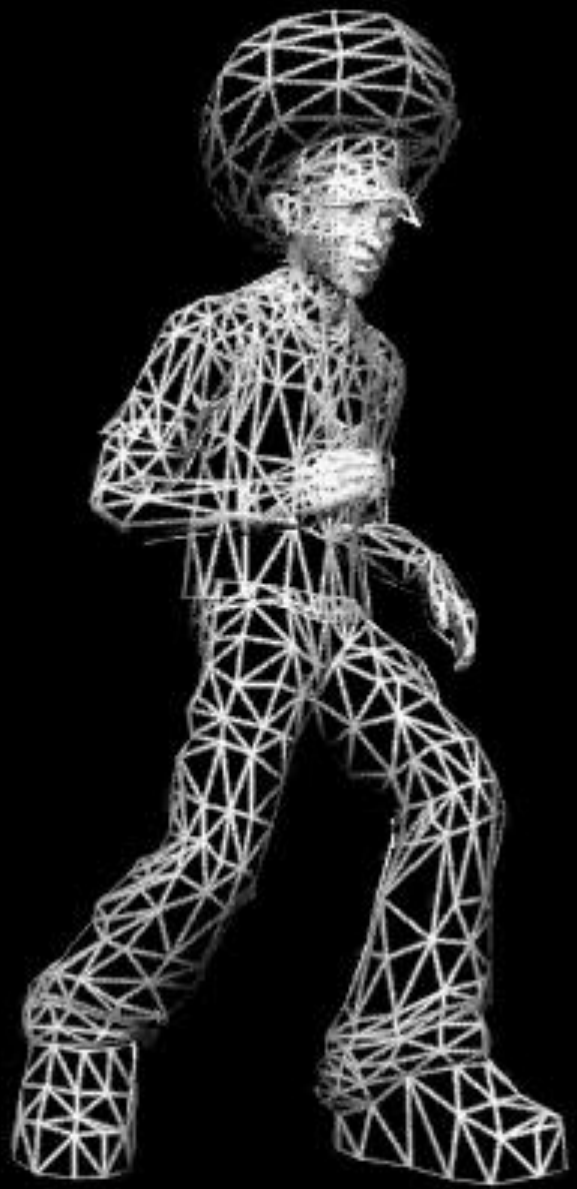
$$v = w_0 \times \left(v_p \times M^{*}_{B0}\right) + w_1 \times \left(v_p \times M^{*}_{B1}\right) + \ldots$$

$$M_w = \left(w_0 \times M^{*}_{B0}\right) + \left(w_1 \times M^{*}_{B1}\right) + \ldots$$

Skeleton Models

Blended matrices implementation

- CPU weighted matrices generated per frame
- 10%-25% quantized weighting to maximum 3 bones
- ~250 matrices for high res. ~150 for medium, ~75 low
- Matrix batches sent to VU1
- Vertices with matrix index transformed using VU1
- Normals with matrix index transformed using VU1
- Non indexed strips generated
- VU1 double buffer layout



Facial Animation

Different methods of Facial Animation

- Translation Null Bones (SSX)
  - Requires many bones
  - Static pose still expensive
  - Constrained system
- Morph Targets (SSX Tricky)
  - No bones
  - Unconstrained
  - Static pose is free
  - Double (triple) buffering required

Morph Targets

Character Rendering

Lighting

- Simple directional lighting
- 1 ambient + 3 directional lights
- Specular map (second pass)

Shadows

- Low resolution model is rendered into a texture from the lights perspective
- Shadow Texture is projected onto terrain
- Extra pass for shadowed terrain patches


SSX
SNOWBOARDS
Show Help

Terrain System

A curve net of Bi-Cubic Bézier patches

Pros

- Data compression
- Improved physics intersection
- Dynamic level of detail

Cons

- Tessellation overhead
- Seaming issues
- Loss of vertex level tweaking

Terrain System

Bi-Cubic Bézier patches

- A high order parametric representation
- Surface is a function of 3 bivariate functions

$$Q(u, v) = (X(u,v), Y(u,v), Z(u,v))$$

$$\text{Where : } 0.0 \le u \le 1.0 \text{ and } 0.0 \le v \le 1.0$$

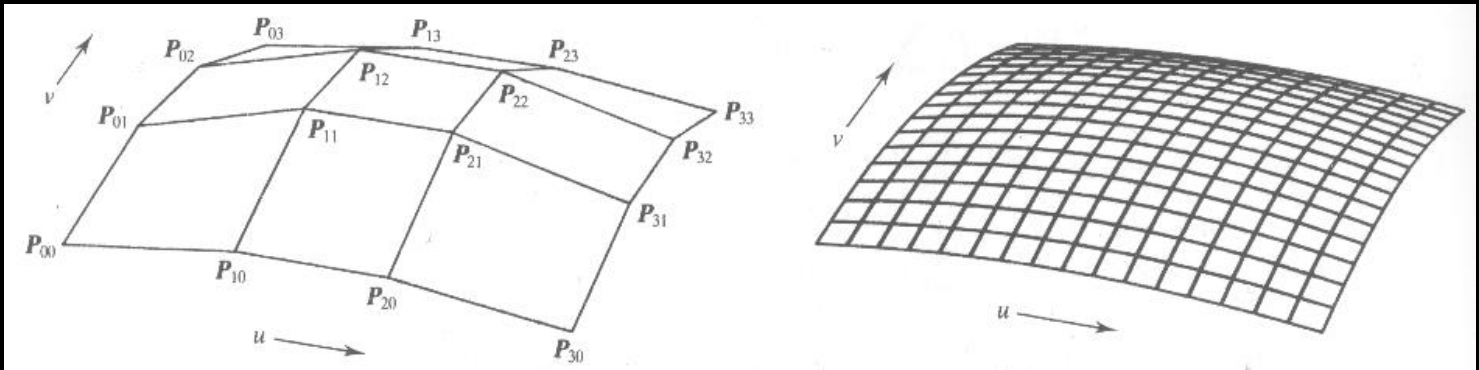
Control points & Basis functions

$$Q(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} p_{ij}\, b_i(u) b_j(v)$$

Terrain System

Matrix notation (VU friendly format)

$$Q(u,v) = [u^3, u^2, u, 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

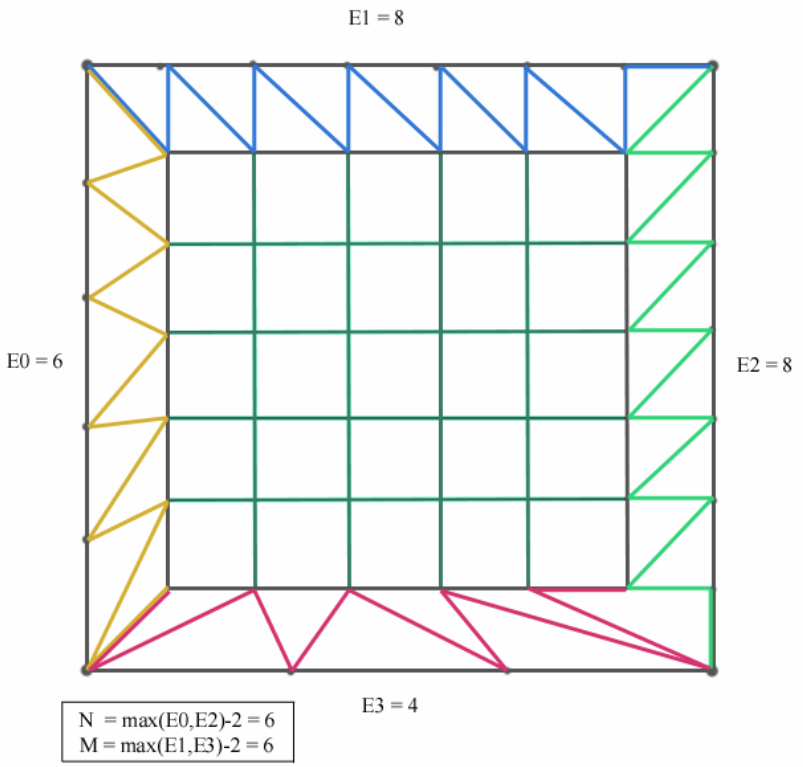


Level of Detail

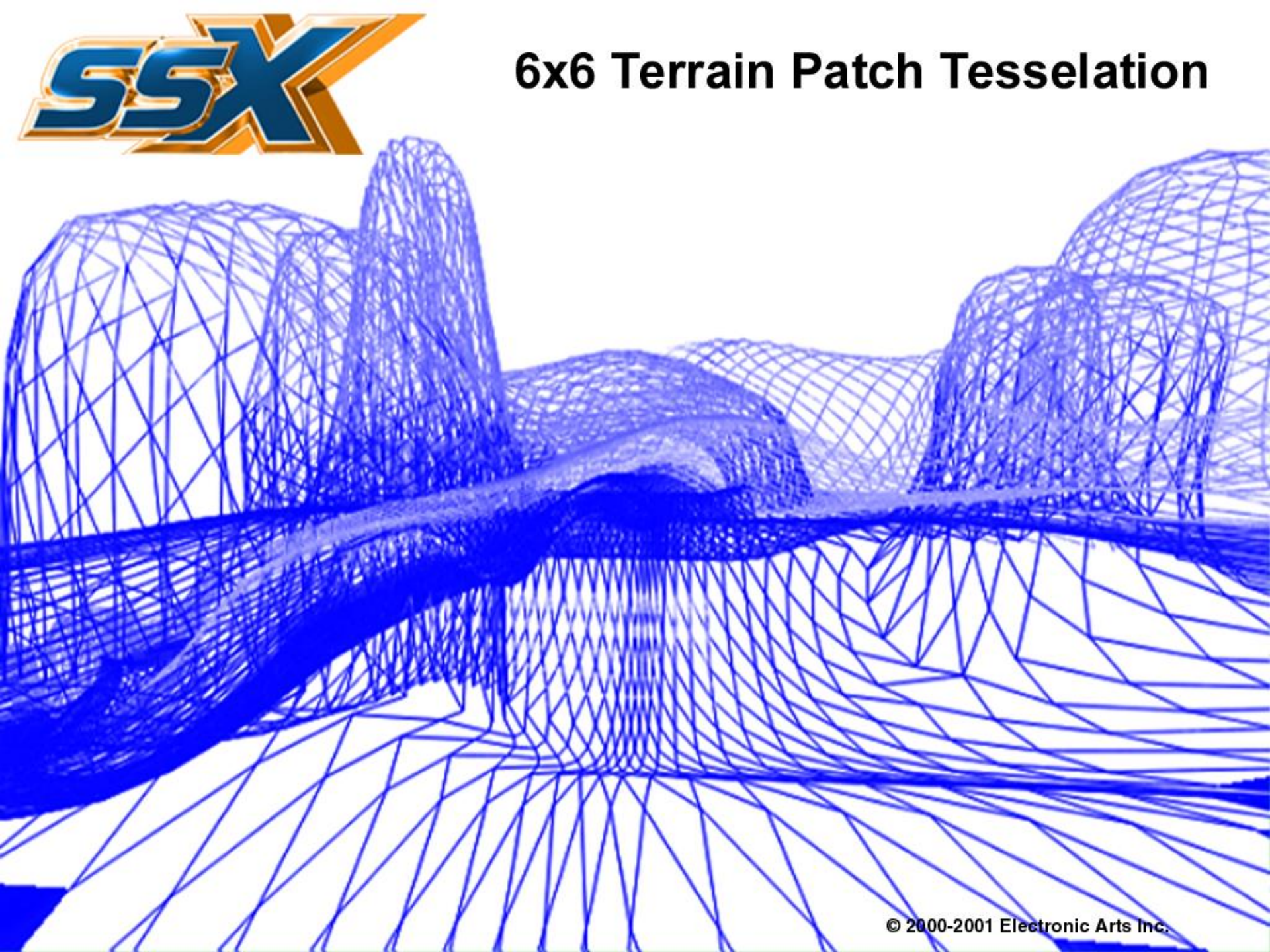
- Distance based heuristic applied to each edge
- Evaluated to 4,6, or 8 vertices per edge
- If all edges the same then use uniform tessellation
- Other wise Non Uniform “glue patch”
- Requires C_0 continuity only
- See GDC proceedings for white paper.

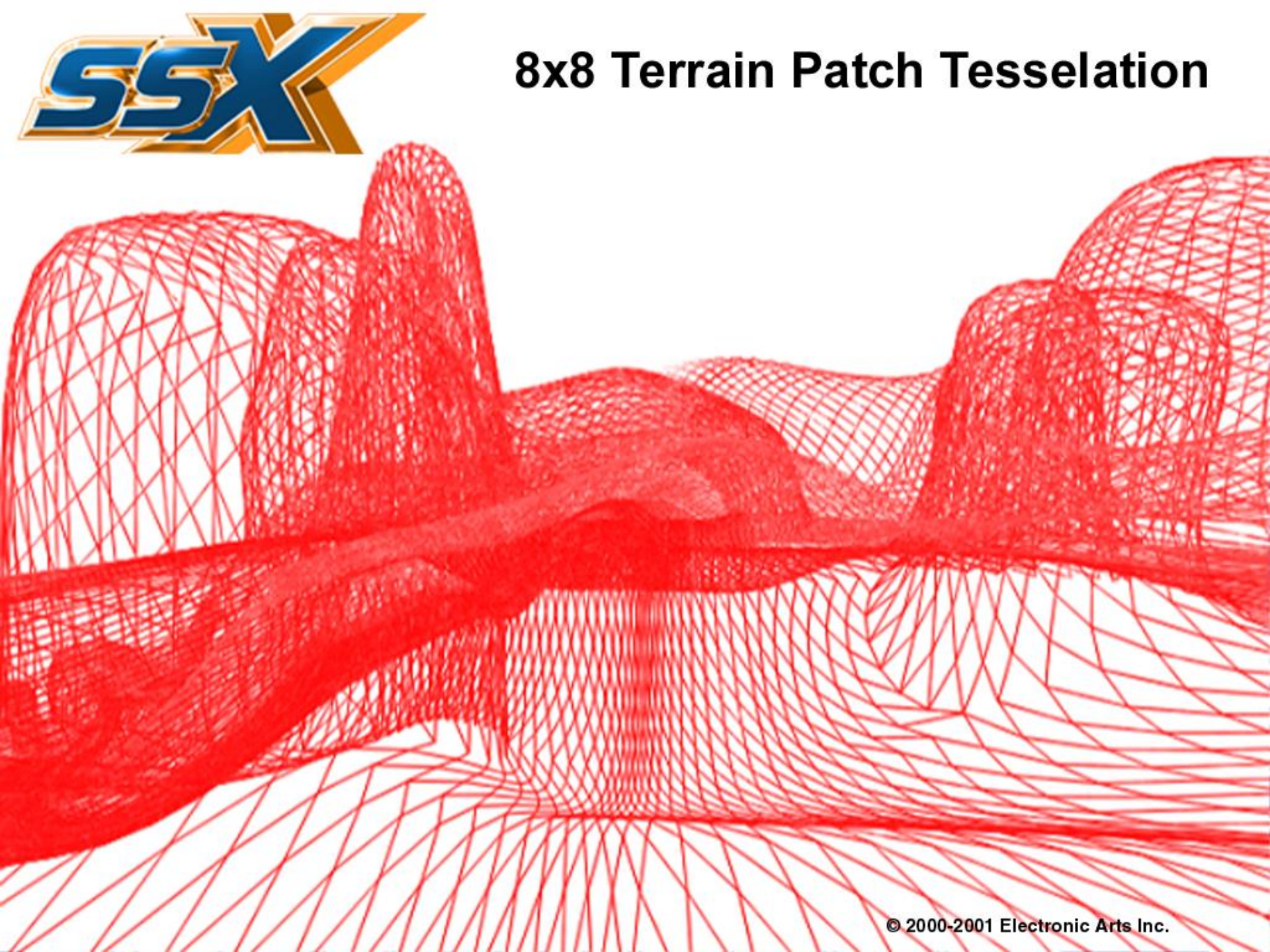
Non Uniform Mesh

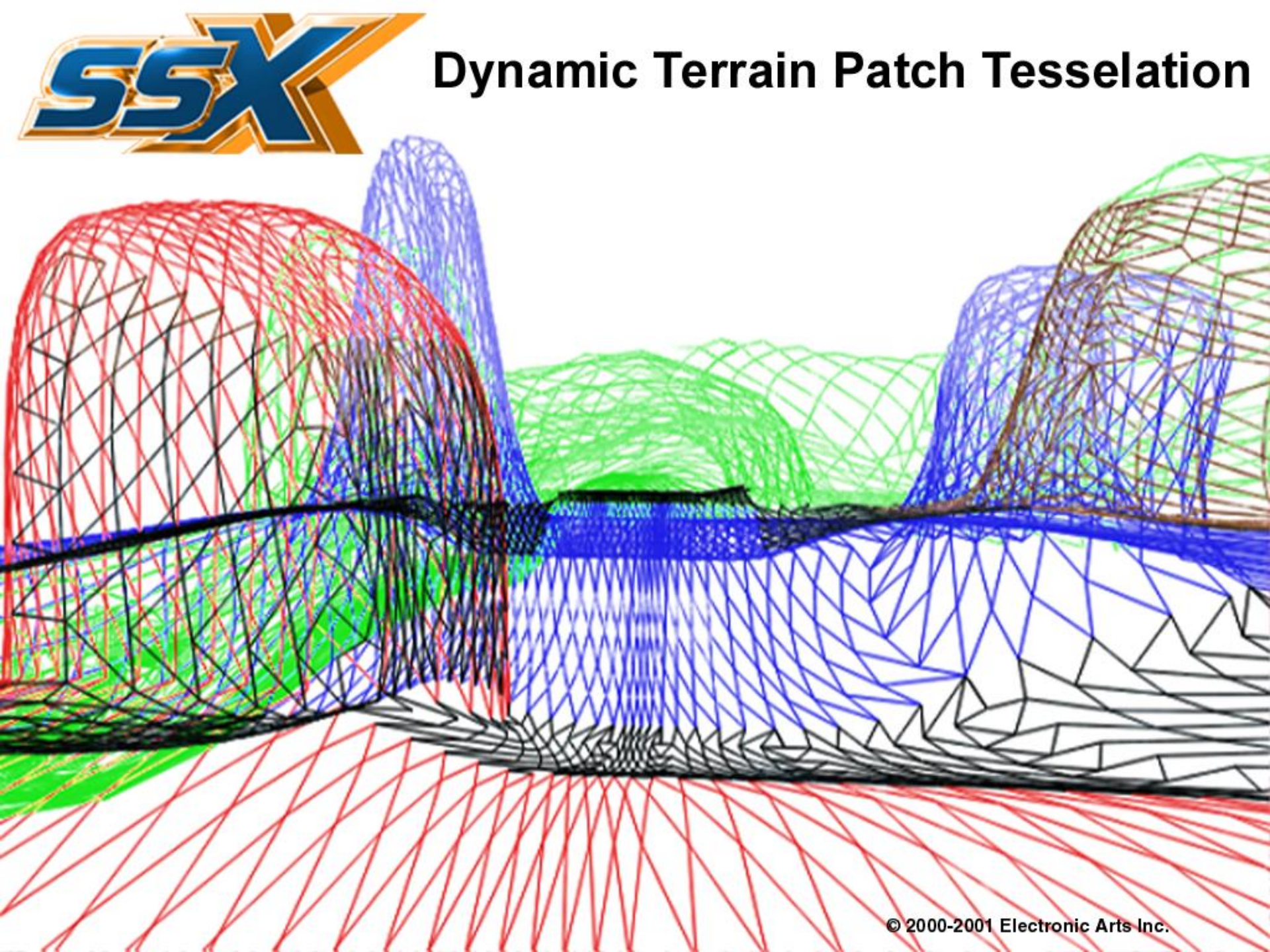
- 4 glue strips
- Uniform inner mesh




SSX
4x4 Terrain Patch Tesselation
© 2000-2001 Electronic Arts Inc.


SSX
6x6 Terrain Patch Tesselation
© 2000-2001 Electronic Arts Inc.


SSX
8x8 Terrain Patch Tesselation
© 2000-2001 Electronic Arts Inc.


SSX
Dynamic Terrain Patch Tesselation
© 2000-2001 Electronic Arts Inc.


SSX
Terrain with Base Texture
© 2000-2001 Electronic Arts Inc.

Terrain System

PS2 Implementation

- Tessellate using VU1
- VU1 double buffer layout
- Pre calculated blend tables
- Fast Uniform tessellation for 4x4,6x6 and 8x8
- Special case non-uniform patches
- Generate 2+ sets of UVs
  - Base texture
  - Light map
  - Projected textures

Terrain System

Performance

With LOD typical scene consists of

- 77% 4x4s, 14% 6x6s, 3% 8x8s, and 6% non-uniform
- Average polygons per patch is 29.68
- Profiled at 5.9 Mpolys/sec (2 pass)

Without LOD

- All 8x8s would require 98 polygons per patch
- Requires 330% more polygons
- Effectively need 19.48 Mpolys/sec (2 pass)

Ambient,directional, point and spot lights

- Negative lights for shadowing

Off line lighting (light maps)

- Each patch evaluated as a grid of points
- Full lighting equation for each sample point
- Lighting results stored in a 2D texture
- 8x8 or 16x16 texture
- Eliminates level of detail light popping

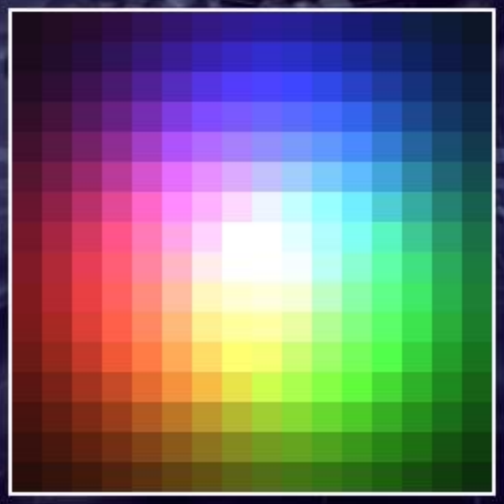
Lighting with textures

Base Texture 128x128

Point Sample

Bilinear Filtering

Light map 16x16



Light maps on the PS2

$C_D = \text{destination colour} \quad C_S = \text{source colour}$

$A_S = \text{source alpha} \quad C_L = \text{light map colour}$

$C_D \times C_S$ is not supported in PS2 hardware!

$(C_D - C_S) \times A_S$ is supported

$$C_D \times C_L = (C_D - C_S) \times A_S$$

$$A_S = \max(C_L.R, C_L.G, C_L.B)$$

$$C_S = C_D - \frac{(C_D \times C_L)}{A_S}$$

Lighting with textures

Base Texture (C_D)

A_S (hardware scale)

(C_D - C_S) x A_S

(C_D*)

(C_L)

(C_S)

(A_S)

C_S (hardware scale)


SSX
Terrain with Base Texture
© 2000-2001 Electronic Arts Inc.


SSX
Base Texture x Light Map
© 2000-2001 Electronic Arts Inc.

Object Rendering

SSX

- All objects dynamically lit
- 1 ambient + 3 directional
- Light directional matrix, color matrix
- Position, normal, STs (compressed)
- Object matrix (with scale)
- Double buffer VU1

Better methods

- Pre-calculate RGBs (memory .vs. quality)
- Use points and spots for dynamic lighting


SSX
Completed Real Time Scene
Show Help
© 2000-2001 Electronic Arts Inc.

Pipelines

Usually not given enough importance!

- Interface for content creation
- Garbage in garbage out (Game Engine)

Off the shelf package

- WYSIWYG
- lighting models

Give control to the artists

- Control (lights, cameras, animation, etc.)
- Fast in-game preview

Pre process everything you can

- Optimization
- Lighting
- Visibility

Future directions for PS2 Games?

Better system balancing

- Data management (DMA/Icache/Dcache)
- Texture management (page hits/ path3)
- Performance analyzer!
- VU0 usage
- Memory versus speed (streaming)

Broadcast quality

- Anti-aliasing, interlace flicker
- Depth of field
- Better lighting (Global illumination?)
- Soft shadows
- Occlusion systems

References

Advanced Animation and Rendering Techniques

Watt & Watt ISBN 0-201-54412-1

Real-Time Rendering

Moller & Haines ISBN 1-56881-101-2

Real-Time Rendering and Software Technology

Watt & Policarpo ISBN 0-201-61921-0

Computer Graphics: Principals and Practice-

Foley, vanDam, Feiner, Hughes ISBN 0-201-12110-7

www.ps2-pro.com (PS2 Dev Net)

Questions?

mrayner@ea.com