Hardware-Accelerated Free-Form Deformation

Clint Chua and Ulrich Neumann

Integrated Media Systems Center

University of Southern California

Overview

- Motivation
- Background of FFDs
- OpenGL and the Proposed System
- Optimizations
- OpenGL Command Extensions

What is Deformation?

- Alteration of shape or dimension
  - elastic deformation vs. rigid body transformation
- Types of deformation
  - pre-computed (keyframed) vs. dynamic

Motivation

- Need for a standard deformation API
  - Integrated commands vs. custom code
- Integrate hardware components for real time dynamic deformation
  - Auxiliary components are simple and can be reused

State of the Art

- Examples of two commercial products that have hardware features that support deformation
  - Sony Playstation 2
    - 10 Floating Point Multiply Accumulate units
    - Kutaragi, et.al. ISSCC’99 pp. 256-259
  - ATI Radeon GPU
    - “Keyframe interpolating mechanism”
    - http://www.ati.com

Criteria for Our System

- Wanted to use a well known deformation system
- Needed to be detached from physically-based models
  - Geometric vs. Physically-based deformation
- Easily integrated into an existing graphics pipeline

Free-Form Deformation

- Well-known geometric deformation
  - Integrate into a graphics API
  - Sederberg and Parry, Siggraph 1986
- Three different varieties
  - Original, parallelpiped (Sederberg and Parry)
  - Extended, regular solids (Coquilart) -- our method
  - Arbitrary, arbitrary topology (MacCraken and Joy)

FFD Properties

- Applicable to nearly any type of geometric models
  - triangle patches to parametric surfaces
- Surface continuity and volume preserving
- Can be applied hierarchically for local and global deformation

FFD Background

- The “Jello” analogy
- The mathematical view

$$q_{i,j,k}(s,t,u) = \sum_{l,m,n=0}^{3} P_{i+l,j+m,k+n} B_l(s) B_m(t) B_n(u)$$

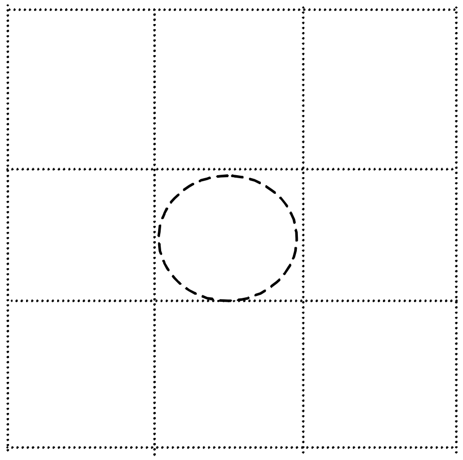
- where *q* is the deformed point, *B* is the B-spline basis functions and (*s,t,u*) are the parameterized coordinates

How to Use an FFD (Step 1)

- Embed the object points in the deformation region
  - Involves a Newton-Raphson process to solve:

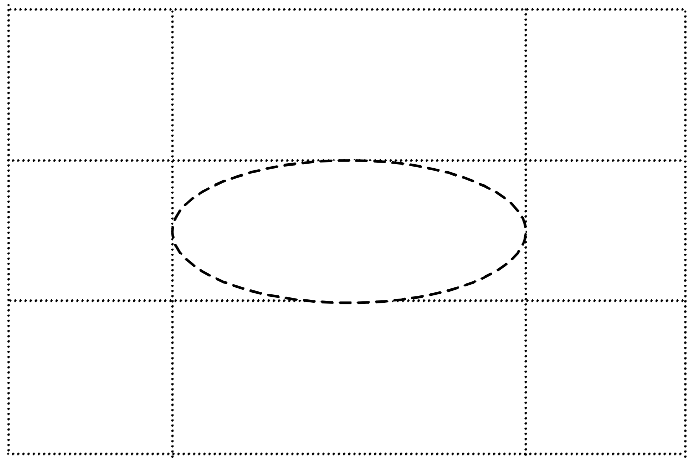
$$\sum_{l,m,n=0}^{3} P_{i+l,\,j+m,k+n} B_l(s) B_m(t) B_n(u) - X_{i,\,j,k} = 0$$

- This results in a set of parameterized coordinates for the object

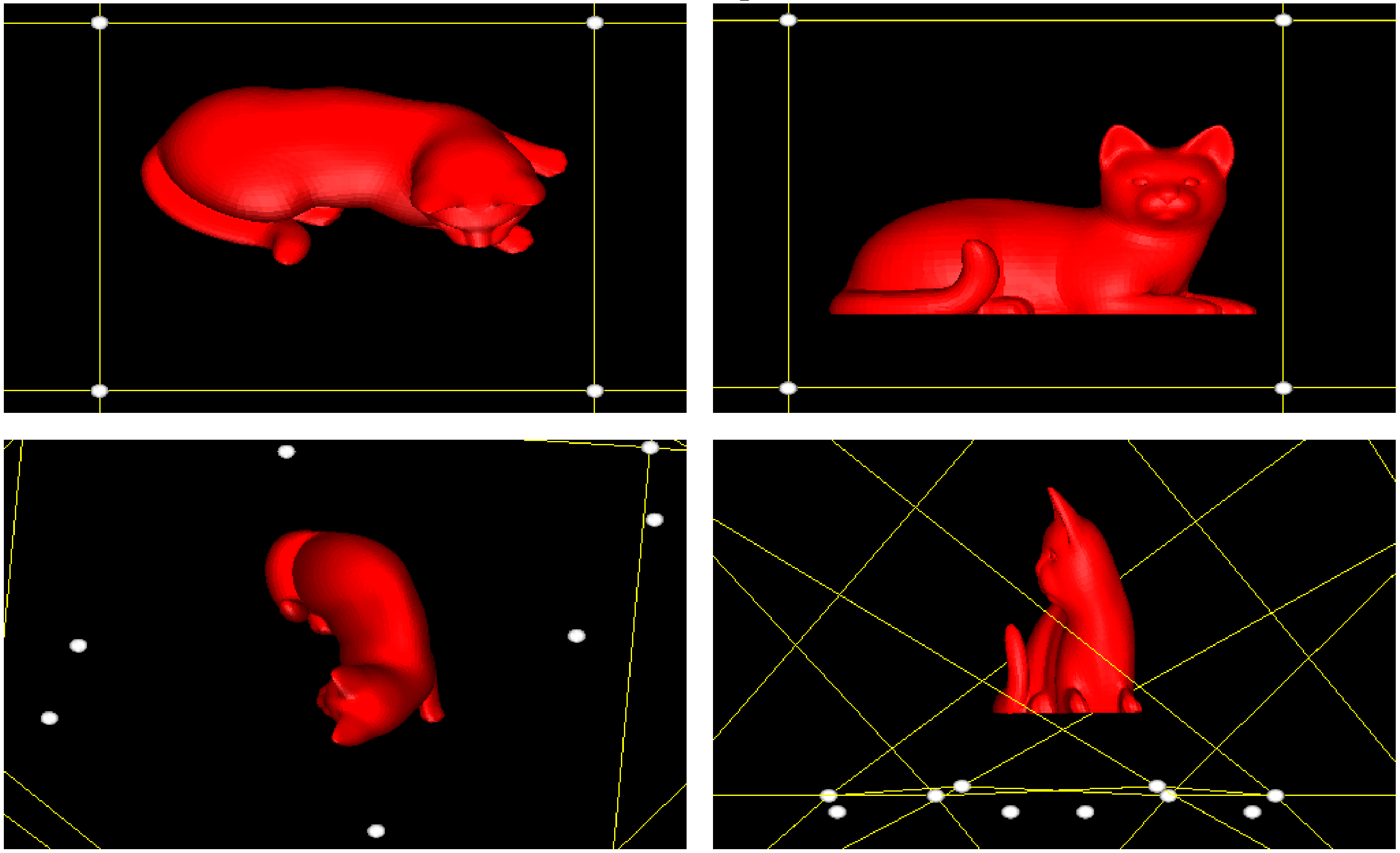


How to Use an FFD (cont’d)

- Move the control points to deform the region of space
- Recalculate the new positions of the points based on the new control point locations

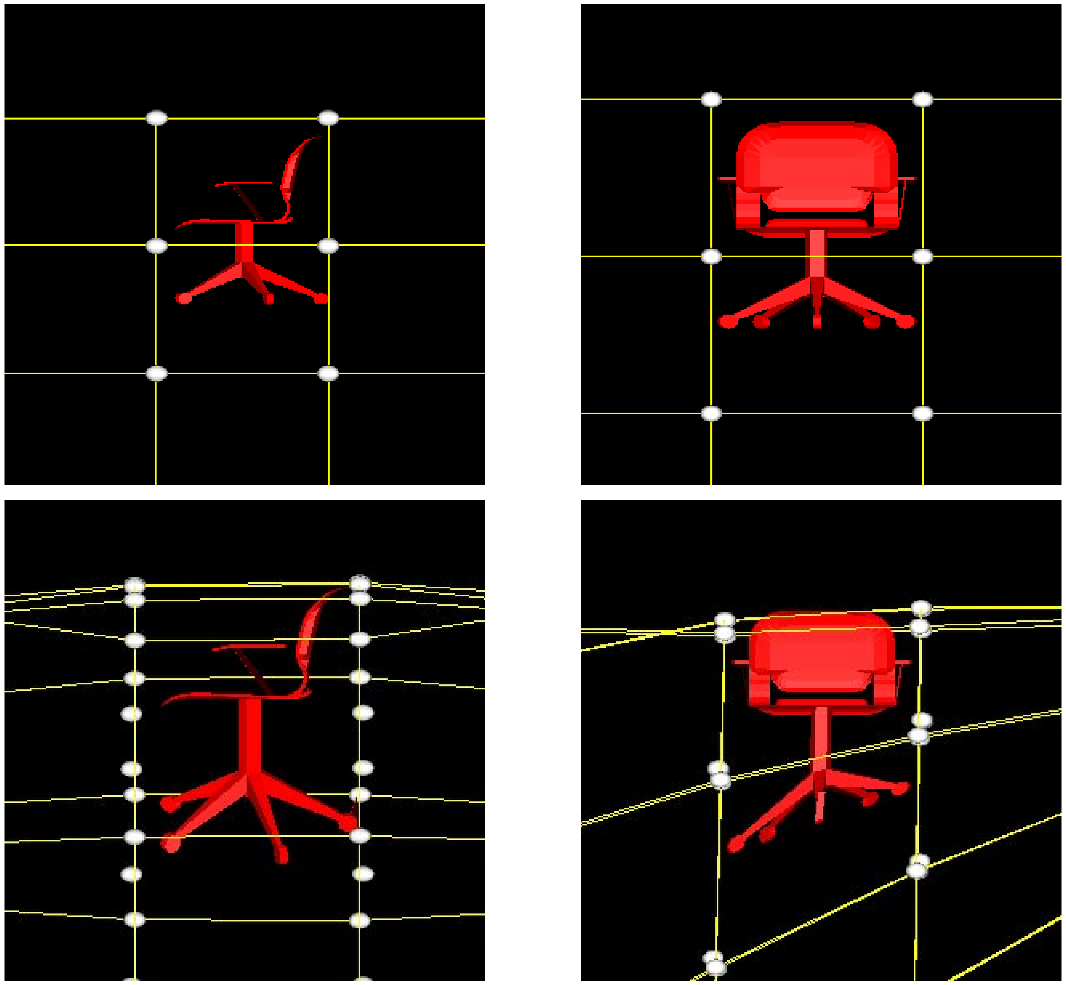


Example



7340 vertices and 14348 triangles

Another Example



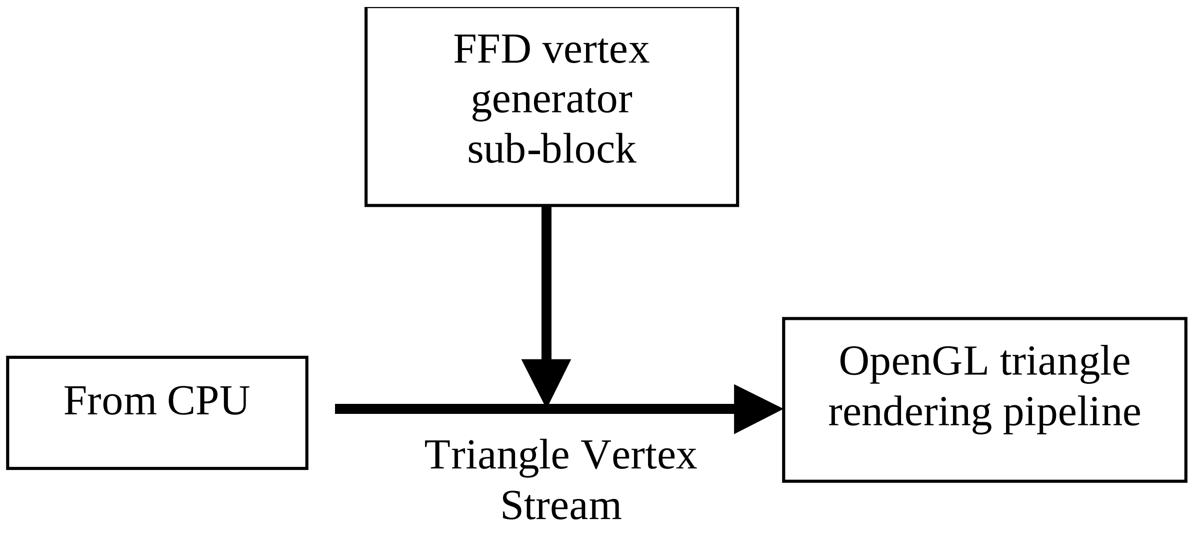
1022 vertices and 1232 triangles

OpenGL and Our System

- Similarities between FFD and OpenGL spline rendering system
  - bivariate vs. trivariate hyperpatches
- OpenGL has a polynomial evaluator system
- OpenGL can be extended to accommodate the proposed system

System Overview

- Need to add a component to the OpenGL rendering system:


FFD vertex generator sub-block
From CPU
Triangle Vertex Stream
OpenGL triangle rendering pipeline

1) Parameterized local coordinates are fed into the polynomial evaluators

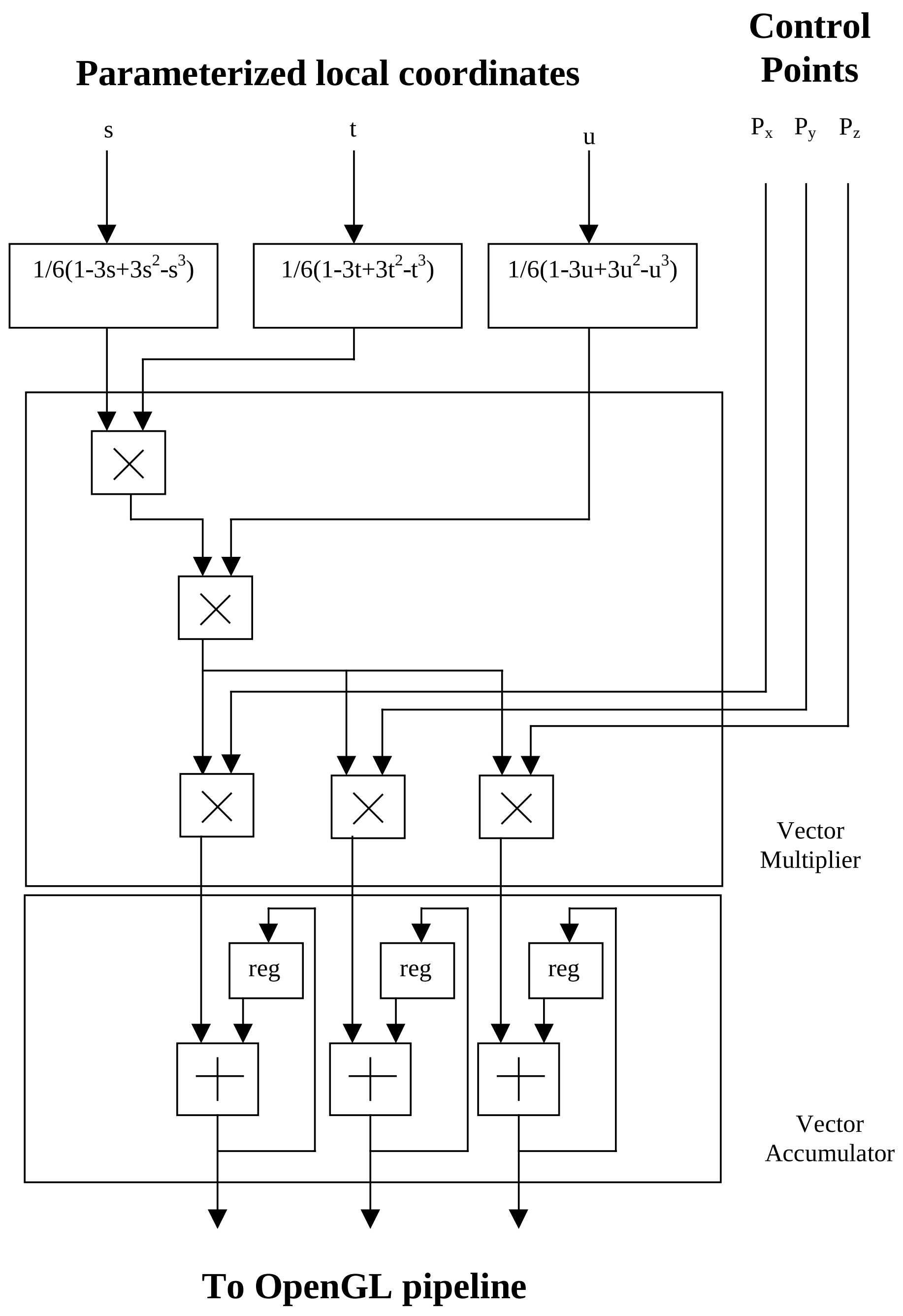
2) Basis functions are evaluated

3) Results are fed into the vector multiplier with the appropriate control points

4) The vector multipliers results are accumulated in the vector accumulator

5) This process is done for each parameterized local coordinate with all 64 control points

6) After all 64 loops have been accumulated, the result is passed to the OpenGL pipeline



Optimizations

- Data Interleaving and DMA
  - Proper data arrangement so that the sub-block is continually operating
- Control Point Cache
  - Control Points are reused and may benefit from a cache

Polynomial Coefficient Register File

- Recall that the B-spline basis functions are:

$$
\begin{aligned}
B_0(u) &= \tfrac{1}{6}(1-3u+3u^2-u^3) \\
B_1(u) &= \tfrac{1}{6}(4-6u^2+3u^3) \\
B_2(u) &= \tfrac{1}{6}(1+3u+3u^2-3u^3) \\
B_3(u) &= \tfrac{1}{6}u^3
\end{aligned}
$$

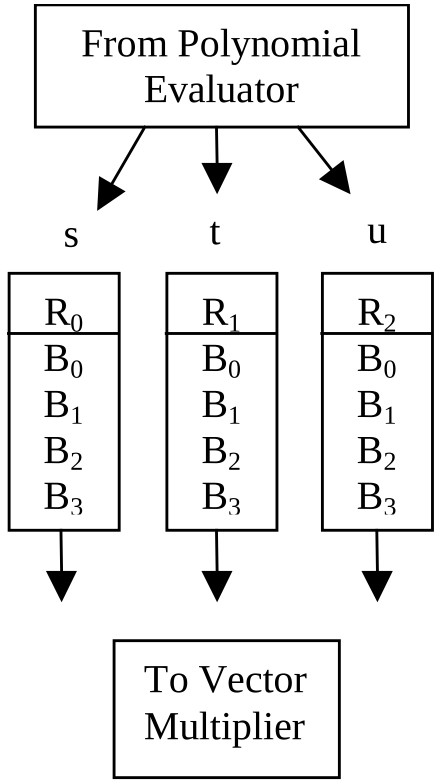
- A polynomial can be represented as a vector of coefficients
- Store all 4 coefficient vectors then load the appropriate vector when needed by the polynomial evaluator

Variations of Implementation

- Basic System
  - Single sub-block is the simplest but slowest
- Fully Parallel System
  - 64 sub-blocks operating in parallel
- Iterative Tree System
  - Compromise between both extremes

Combination Generator

- Since the B-spline basis are constantly being re-evaluated, the system may benefit from evaluating all unique results and store them in a register file



OpenGL Extensions

- OpenGL intrinsic vs. GLU
- Proposed GLU API that mimics the GLU API for NURBS rendering
  - **gluNewFFDRenderer** - generate a new FFD object to refer to when rendering
  - **gluFFDProperty** - change common properties like line width, etc.
  - **gluFFDCallback** - callback used to monitor the progress of rendering and error monitoring
  - **gluFFDVolume** - call used to generate the deformed surface. You pass in the array of control points and the parameterized local coordinates

Conclusions

- FFD is a well-known and flexible geometric deformation technique
- Shown how FFD is integrated into OpenGL as a standard deformation API
- Shown how FFD can be hardware accelerated and integrated into the existing OpenGL pipeline.

Acknowledgements

- This research has been funded by the Integrated Media Systems Center, A National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152
- Special Thanks to Dr. Tony De Rose and James Gain