Shadow Framework

Alessandro Martinelli
alessandro.martinelli@unipv.it

20 Dicembre 2011

Shadow Framework

Shadow Framework:

- The Shadow Framework
- Fast Content Access through the Web
- Intrinsic Level of Detail
- SF Architecture and Versions

The **Shadow Framework**

The **Shadow Framework** is a CG **Rendering Framework** designed for **High Quality Contents**

- **Framework**: because its a set of Libraries and Tools to support **CG**
- **Shadow**: in **technical CG** the word **Shade** is often used in the contest of Programmable Graphics Hardware. Here the word **Shadow** is used to express an higher order level of **Graphics Hardware Programming**.

The **Shadow Framework**

The **Shadow Framework** is a CG **Rendering Framework** designed for **High Quality Contents**

- **Framework**: because its a set of Libraries and Tools to support **CG**
- **Shadow**: in **technical CG** the word **Shade** is often used in the contest of Programmable Graphics Hardware. Here the word **Shadow** is used to express an higher order level of **Graphics Hardware Programming**.

Why should it be used?

- Because it allows **modern modeling techniques** based on nowadays **standard GPU technology**:

The **Shadow Framework**

The **Shadow Framework** is a CG **Rendering Framework** designed for **High Quality Contents**

- **Framework**: because its a set of Libraries and Tools to support **CG**
- **Shadow**: in **technical CG** the word **Shade** is often used in the contest of Programmable Graphics Hardware. Here the word **Shadow** is used to express an higher order level of **Graphics Hardware Programming**.

Why should it be used?

- Because it allows **modern modeling techniques** based on nowadays **standard GPU technology**:
  - *designed to reduce scenarios data and allow* **fast access** *to 3D contents through the web*

How is this achieved?

The **Shadow Framework**

The **Shadow Framework** is a CG **Rendering Framework** designed for **High Quality Contents**

- **Framework**: because its a set of Libraries and Tools to support **CG**
- **Shadow**: in **technical CG** the word **Shade** is often used in the contest of Programmable Graphics Hardware. Here the word **Shadow** is used to express an higher order level of **Graphics Hardware Programming**.

Why should it be used?

- Because it allows **modern modeling techniques** based on nowadays **standard GPU technology**:
  - *designed to reduce scenarios data and allow* **fast access** *to 3D contents through the web*
- Because it allows a smoother definition of **Levels of Detail**

How is this achieved?

The **Shadow Framework**

The **Shadow Framework** is a CG **Rendering Framework** designed for **High Quality Contents**

- **Framework**: because its a set of Libraries and Tools to support **CG**
- **Shadow**: in **technical CG** the word **Shade** is often used in the contest of Programmable Graphics Hardware. Here the word **Shadow** is used to express an higher order level of **Graphics Hardware Programming**.

Why should it be used?

- Because it allows **modern modeling techniques** based on nowadays **standard GPU technology**:
  - *designed to reduce scenarios data and allow* **fast access** *to 3D contents through the web*
- Because it allows a smoother definition of **Levels of Detail**
  - Indeed, the same **SF model** describes itself tens or hundreds LoDs.

How is this achieved?

The **Shadow Framework**

The **Shadow Framework** is a CG **Rendering Framework** designed for **High Quality Contents**

- **Framework**: because its a set of Libraries and Tools to support **CG**
- **Shadow**: in **technical CG** the word **Shade** is often used in the contest of Programmable Graphics Hardware. Here the word **Shadow** is used to express an higher order level of **Graphics Hardware Programming**.

Why should it be used?

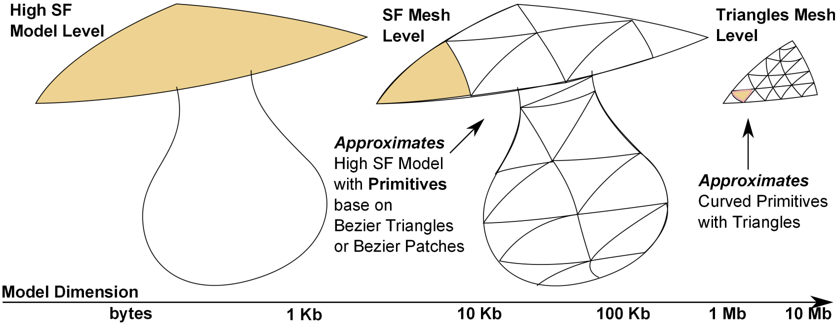
- Because it allows **modern modeling techniques** based on nowadays **standard GPU technology**:
  - *designed to reduce scenarios data and allow* **fast access** *to 3D contents through the web*
- Because it allows a smoother definition of **Levels of Detail**
  - Indeed, the same **SF model** describes itself tens or hundreds LoDs.
  - *this allows the same model to be* **adaptively rendered** *with different quality on* **devices with different capabilities**.

How is this achieved?

Two-Ways Hierarchical Modeling

Shadow Framework geometries have a 2 way hierachy:

- **SF Models** are based on very complex **mathematical surface models**.
- **SF Models** are approximated with **Curve Primitives**, which are tessellated into **Triangles**.



Two-Ways Hierarchical Modeling

Complex Surface Models

- An High Variety of Surface are supported.
- Few data for complex models.

Two-Ways Hierarchical Modeling

Complex Surface Models

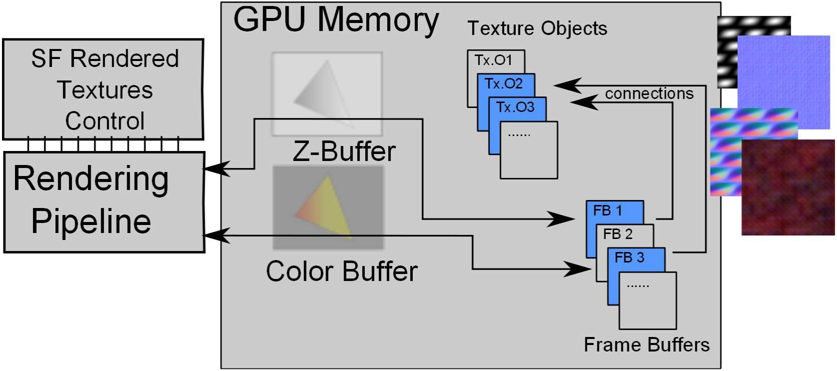
- An High Variety of Surface are supported.
- Few data for complex models.

Two Way Approximation

- **Complex Surfaces** are sampled in few points, because **Curved Primitives** gives a better approximation of any surface.
- **Complex Surfaces** to **Curved Primitives** approximation may be performed with generic GPU programming.
- **Curved Primitives** to **Triangles** tesselletion directly performed on **Graphics Hardware**

GPU-Side Precomputed Textures

- **SF Textures** are always **Rendered Textures**.
- **Rendered Texture** are **Procedural Textures** computed on the **GPU**.
- This is accomplished mostly with support of **Framebuffers**.



Fast Access to 3D Contents through the Web

- **Shadow Framework scenarios** are described with a little memory effort.
- This allows fast communication between web based applications.

Fast Access to 3D Contents through the Web

- **Shadow Framework scenarios** are described with a little memory effort.
- This allows fast communication between web based applications.

How **little** the memory effort is?

Fast Access to 3D Contents through the Web

- **Shadow Framework scenarios** are described with a little memory effort.
- This allows fast communication between web based applications.

How **little** the memory effort is?

Geometry Compression Level

Suppose we want to describe a *Sphere*

- With a Model named **Sphere**, its description will be **center+radius**, let's say it comes to be **16 bytes**.
- With a **III Order Bezier Triangle Mesh**, an high quality approximation will use **tens of vertices**, and the model will be **between 100 bytes and 1Kb**
- With a **Simple Triangle Mesh**, an high quality approximation will use thousends of triangles, and the model will be **between 10 Kb and 100 kb**

Fast Access to 3D Contents through the Web

- **Shadow Framework scenarios** are described with a little memory effort.
- This allows fast communication between web based applications.

How **little** the memory effort is?

Geometry Compression Level

Suppose we want to describe a *Sphere*

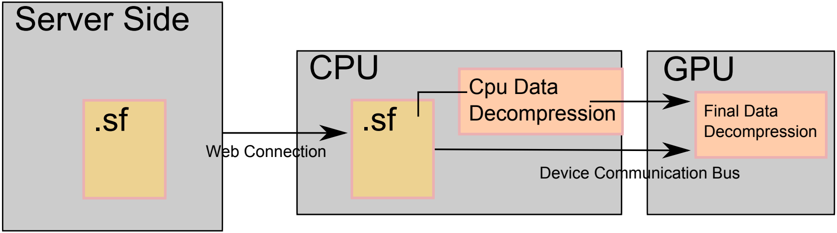
- With a Model named **Sphere**, its description will be **center+radius**, let's say it comes to be **16 bytes**.
- With a **III Order Bezier Triangle Mesh**, an high quality approximation will use **tens of vertices**, and the model will be **between 100 bytes and 1Kb**
- With a **Simple Triangle Mesh**, an high quality approximation will use thousends of triangles, and the model will be **between 10 Kb and 100 kb**

Texturing Compression Level

- Rendered Texture may vary in dimension from **some bytes** to even **tens of Kb** according to their complessity.
- A Texture Image, usually will be in the range **10k - 1Mb**
- **NOTE**: **Computer Graphics** often use *lossless* image formats!

Fast Access to 3D Contents through the Web: Decompression Pipeline

There's something like a **decompression system** into the ShadoFramework which open ShadowFramework data and extract **Triangle Meshes** and **Standard Textures** directly into the Rendering Pipeline.



The **Shadow Framework**: Intrinsic Level of Detail

Both **Rendered Textures** and **Hierarchical Geometries** have an intrinsic **Level of Detail**

- Approximation of **High Level Models** into **Curved Primitive Meshes** may be accomplished with different tessellation step.
- **Curved Primitive** may be tessellated into triangles with different tessellation steps.
- **Texture** may be rendered with **different resolutions**

In this way the same scenario may be rendered on **devices with different capabilities** keeping an high frame rate by **selecting the correct quality level** for this processes.

The **Shadow Framework**: version 1.0 and 2.0

Shadow Framework 1.0

- A Fixed set of **Models** and **Effects**.
- Based upon **OpenGL**, designed to support OpenGL.
- Frozen since **May 2010** at beta version on **www.shadowframework.com**

Shadow Framework 2.0

- Introduce a totally new ShadowFramework Pipeline.
  - Designed to be implemented with any **OpenGL 2.0 like** pipeline.
  - Designed to exploit future Graphics Hardware capabilities according to modern trends.
- No more fixed functionalities, new effects can be added to the high level **SF Pipeline** through **SF shading language**
- Actually, a Beta Version of OpenGL and OpenGL ES implementation of SF Pipeline is available, together with an Alpha versione of the Higher part of the Framework.

The **Shadow Framework 2.0** : Architecture

Based upon 2 main layers:

SF Graphics

A Graphics Module, using OpenGL, whose responsibilities are:

- **SFPipeline** : Shading and GPU programming
- **SFGraphics** : Rendering Curve Primitive Meshes, Managing Textures
- **SFMemory** : Keeping Geometries Buffers Data

Everything is **wrapped** with interfaces which hides the effective implementation, which may be based either on **OpenGL 2.0+, OpenGL ES 2.0 or WebGL**.

The **Shadow Framework 2.0** : Architecture

Based upon 2 main layers:

SF Graphics

A Graphics Module, using OpenGL, whose responsibilities are:

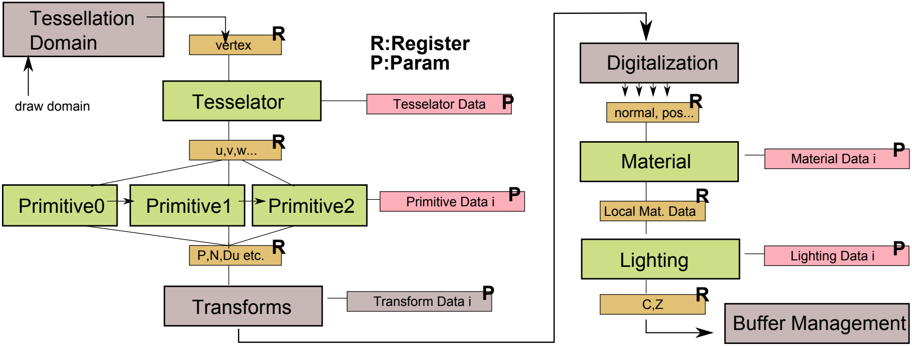
- **SFPipeline** : Shading and GPU programming
- **SFGraphics** : Rendering Curve Primitive Meshes, Managing Textures
- **SFMemory** : Keeping Geometries Buffers Data

Everything is **wrapped** with interfaces which hides the effective implementation, which may be based either on **OpenGL 2.0+, OpenGL ES 2.0 or WebGL**.

SF Core Level

- A Complex set of modules managing High Level Geometries and scenarios data.
- **SF Core** works using **SF Graphics** and is unaware of **OpenGL** implementation.

The **Shadow Framework** 2.0 Rendering Pipeline



The **Shadow Framework** 2.0 Rendering Pipeline

- **Tessellator** : tessellation functions applied directly on tessellation domain.
- **Primitive** : Curved Primitive Evaluation. More Primitive Data may be used on the same model, also for the same Channel (Position,Normal,Dus,Dvs,Texture Coordinates, etc.)
- **Transforms** : Camera and Scene transforms applied to the model (Fixed Functionality).
- **Materials** : apply the material and prepares lights data.
- **Light** : evaluate lights and define final Fragment Data.

Each (green) module can be assigned a specific program component. A set of **Program Components** build up a **Rendering Program**.

OpenGL implementation

On OpenGL implementation **Tessallator**, **Primitive** and **Transforms** are mixed into a **Vertex Shader**, and **Materials** and **Light** are mixed into **Fragment Shader**.

The **Shadow Framework** 2.0 Rendering Pipeline

- **Tessellator** : tessellation functions applied directly on tessellation domain.
- **Primitive** : Curved Primitive Evaluation. More Primitive Data may be used on the same model, also for the same Channel (Position,Normal,Dus,Dvs,Texture Coordinates, etc.)
- **Transforms** : Camera and Scene transforms applied to the model (Fixed Functionality).
- **Materials** : apply the material and prepares lights data.
- **Light** : evaluate lights and define final Fragment Data.

Each (green) module can be assigned a specific program component. A set of **Program Components** build up a **Rendering Program**.

OpenGL implementation

On OpenGL implementation **Tessallator**, **Primitive** and **Transforms** are mixed into a **Vertex Shader**, and **Materials** and **Light** are mixed into **Fragment Shader**.

The **Shadow Framework** 2.0 Rendering Pipeline : known issues (20-12-11)

- Tessellation support is partially missing
- No **SF control** on Buffers Manager (may be still controlled with OpenGL on OpenGL implementations)
- A little refactor may occur to improve Materials control.

... but the overall pipeline is working fine :)

SF2.0 Rendering Pipeline : Rendering Memory

SFArray

An array of the same kind of data. May be:

- a Vertices Array
- a Matrices Array
- a Primitive (faces) array
- a Structure Array (where structure are like C structure), used to keep uniform-like data.

SF2.0 Rendering Pipeline : Rendering Memory

SFArray

An array of the same kind of data. May be:

- a Vertices Array
- a Matrices Array
- a Primitive (faces) array
- a Structure Array (where structure are like C structure), used to keep uniform-like data.

SFArray are generated and accessed through the ShadowFramework pipeline Graphics

SF2.0 Rendering Pipeline : Rendering Memory

SFArray

An array of the same kind of data. May be:

- a Vertices Array
- a Matrices Array
- a Primitive (faces) array
- a Structure Array (where structure are like C structure), used to keep uniform-like data.

SFArray are generated and accessed through the ShadowFramework pipeline
Graphics
Why?

- **Maximize SF Pipeline** responsability on data storing and drawing.
- Allow OpenGL and platform specific optimization on how this data is stored and sent to pipeline.

SF2.0 Rendering Pipeline : Rendering Memory

SFArray

An array of the same kind of data. May be:

- a Vertices Array
- a Matrices Array
- a Primitive (faces) array
- a Structure Array (where structure are like C structure), used to keep uniform-like data.

SFArray are generated and accessed through the ShadowFramework pipeline Graphics
Why?

- **Maximize SF Pipeline** responsability on data storing and drawing.
- Allow OpenGL and platform specific optimization on how this data is stored and sent to pipeline.

Well working, even if not optimized.

SF2.0 Rendering Pipeline : Rendered Textures

SFRenderedTexture

A Rendering Process which draws on **user defined Buffers** which may be:

- **PlainBufferData**: support on rendering process
- **TextureData**: maybe exploited as texture once Rendering Process is complete

SF2.0 Rendering Pipeline : Rendered Textures

SFRenderedTexture

A Rendering Process which draws on **user defined Buffers** which may be:

- **PlainBufferData**: support on rendering process
- **TextureData**: maybe exploited as texture once Rendering Process is complete

SFTextureData

An **SFTextureData** may be:

- One of the results of a **RenderedTexture Process**
- An SFBitmap loaded into the pipeline. **SFBitmaps**:
    - May be generated with SFCore procedure
    - May be loaded from file

SF2.0 Rendering Pipeline : Rendered Textures

SFRenderedTexture

A Rendering Process which draws on **user defined Buffers** which may be:

- **PlainBufferData**: support on rendering process
- **TextureData**: maybe exploited as texture once Rendering Process is complete

SFTextureData

An **SFTextureData** may be:

- One of the results of a **RenderedTexture Process**
- An SFBitmap loaded into the pipeline. **SFBitmaps**:
  - May be generated with SFCore procedure
  - May be loaded from file .... but please don't do it!

SF2.0 Rendering Pipeline : Rendered Textures

SFRenderedTexture

A Rendering Process which draws on **user defined Buffers** which may be:

- **PlainBufferData**: support on rendering process
- **TextureData**: maybe exploited as texture once Rendering Process is complete

SFTextureData

An **SFTextureData** may be:

- One of the results of a **RenderedTexture Process**
- An SFBitmap loaded into the pipeline. **SFBitmaps**:
  - May be generated with SFCore procedure
  - May be loaded from file .... but please don't do it!

Issues:

- Not all Texture Parameters configurations have been tested.
- Some fix required to support DepthBuffers and Stencil Buffers.