SpiderGL

3D Graphics for Next-Generation WWW

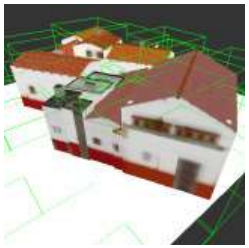
Marco Di Benedetto

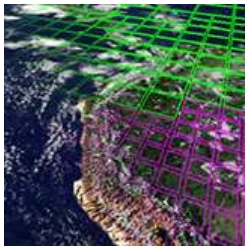
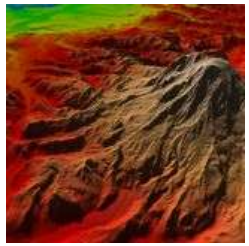
Visual Computing Lab – ISTI – CNR Pisa, Italy

http://vcg.isti.cnr.it

http://spidergl.org

SpiderGL : Overview



- JavaScript 3D Graphics library which relies on WebGL for realtime rendering

- Helps close the gap between web developers interested in 3D Graphics and experienced CG programmers by offering web-friendly API and tools for 3D web development

- Provides typical 3D graphics structures and algorithms to developers
- Linear algebra, geometry, visibility culling, fast mesh rendering, multiresolution, asynchronous content loading, UI, …

- Philosophy: ease the development of 3D applications without introducing unnecessary abstraction layers nor preventing low level access
- Procedural Core, scene graph can be built on top
- Seamless integration into existing code

SpiderGL and WebGL

- Low-Level Constructor Functions
- Utility routines to ease the creation of all WebGL resources (programs, textures, framebuffers, …)
- Highly configurable with *options* parameters to override defaults
- Reflection used to populate structures (uniforms locations, channels bits, ...)

- High-Level Wrappers
- Provide a more Object Oriented usage paradigm
- Interact seamlessly with native low-level WebGL handles

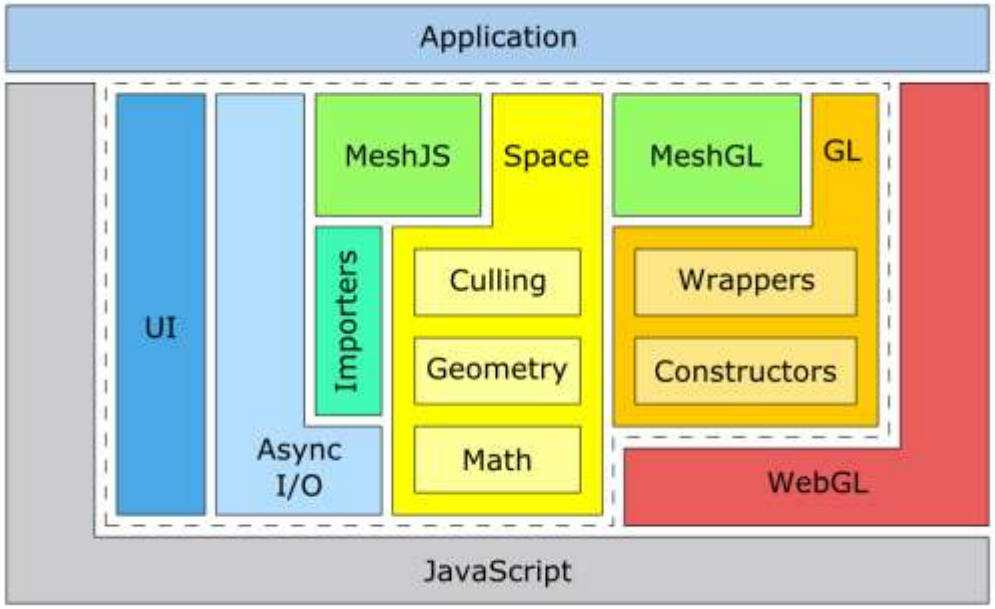
- Smart Datasource Bindings
- Do *not* impose naming convention on vertex attributes, program uniforms, samplers
- Correspondences are set via *mapping* parameters: JavaScript objects of the form

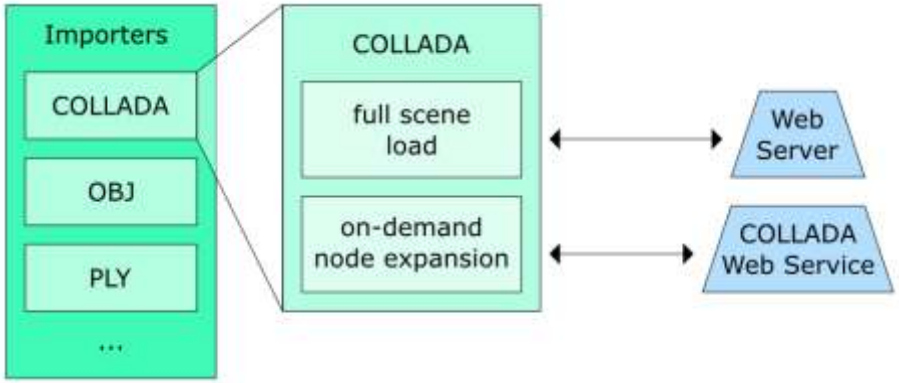
```
var vertexStreamMapping = {
    meshVertexAttribName1 : "vertexShaderAttribName1",
    meshVertexAttribName2 : "vertexShaderAttribName2", ...
};
```

allow decoupling of entities which take part in the rendering process (meshes - shaders, application state – program uniforms, …)

- Asynchronous Data Loading and Caches
- Optional texture/geometry managers for on-demand data fetching and storage

SpiderGL : Architecture





- GL : WebGL Objects
  - Utility constructor functions
  - Object Wrappers
- MeshGL: Renderable 3d Model
  - WebGL buffers container
- MeshJS: Editable 3D Model
  - Suitable for geometry processing
- Space:
  - Math: base math, linear algebra
  - Geometry: geometic objects & algorithms
  - Culling: visibility culling objects & algorithms
- Async I/O: asynchronous content loading
  - Http requests, priority queues…
  - Mesh importers
- UI: User Interface facilities
  - GLUT-like event handling
  - Interactors (trackball, camera, …)

SpiderGL and COLLADA

- Like the rendering core, assets are at the foundations of a graphics application
- Geometry, materials, hierarchical relationships, animation, …

- COLLADA can handle *all* of them (and much more) with a simple, human-readable structure
- Content-rich scenes can be decomposed in shader database, geometry database, rendering technique database and *links* among them
- No need to load the whole scene at once: exploit asynchronous xml http requests to fetch the scene subset you need, when you need them

- SpiderGL handles meshes as raw containers of vertex attributes and connectivity information
- → fits naturally with COLLADA geometry representation
- No assumption on attributes type / dimensionality
- Default bindings are easily created with *mapping* parameters

- SpiderGL – COLLADA Roadmap:
- WebGL compatibility is enforced → implement the COLLADA OpenGL|ES 2.0 profile
- 1st Phase: 3D model geometry import, preliminary shader db [Spring 2010]
- 2nd Phase: Rendering Techniques [Summer 2010]
- 3rd Phase: Skinning and Animation [Fall 2010]

Links

- http://spidergl.org
- http://vcg.isti.cnr.it

- http://www.khronos.org/webgl
- http://www.khronos.org/collada

Contact:

Marco Di Benedetto

Researcher at Visual Computing Lab – ISTI,

National Research Council (CNR), Pisa, Italy

mailto: marco.dibenedetto@isti.cnr.it