JogAmp: 2D/3D & Multimedia across Devices

SIGGRAPH 2012 – Los Angeles Convention Center
August 7, 2012

Presented by:
Sven Gothel
Rami Santina
Xerxes Ranby
Wade Walker
Julien Gouesse
What is JogAmp?

JogAmp
JOGL – JOCL – JOAL ...
About US

- Open & Vendor Independent
- BSD License
- Java Graphics, Audio, Media & Processing
  High Performance Bindings
- One Stop Community Platform
  - SCM, Bugtracking, Build Server, Mailinglist/Forum,..
- Commercial Support
- http://jogamp.org
“Demoscene” production developed with JOGL

Total executable size including music+visuals must be <=4096 bytes (running length ~3 mins)

*Hartverdrahtet* placed 1st at Revision 2012 (worlds biggest demoscene event)

1:1 JOGL port ofc with sourcecode available: http://copypastaresearch.tumblr.com/
Hartverdrahtet Visuals in a Nutshell

- Single fragment shader (fullscreen billboard)
- Zero polygons! Analytical estimated surface of multidimensional IFS (fractal) volume
- Implements real-time raytracing (sphere-tracing based) with global illumination features (e.g. ambient occlusion)
- Everything is generated per pixel on-the-fly (no precalc)
- Postprocessing pass finishes the look adding volumetric lighting, noise and analog distortions
- Complete fragment shader including the raytracer, fractal, camera paths for 10 scenes and post-effects <1500 bytes
Sweet Home 3D / Java3D / Engines
NiftyGUI / Engines

Hello World
Panels and Hint (1.3)
Mouse Update (1.3)
Popup Menu (1.3)
Drag'n'Drop (1.2/1.3)
Textalign
Textfield
Tuer / Ardor3D / Engines
JOGL Android Binding

- http://www.youtube.com/watch?v=VHxtVT4tWjM
JOGL Embedded / R.-Pi
JOGL Embedded / ..
Cross Platform & Device: Use Case

Visual Project Control

C3D Studio/Planner
- Scenario Creation
- Data Integration
- ...

C3D Viewer
- Model Visualization
- Project Progress Update
- 4D Animation
- Report Generation
- Design Review

C3D Mobile
- BIM Model Visualization
- just-in-time progress update
- ...

http://c3d.com
C3D - Visual Project Controls
C3D - Visual Project Controls

Sample usecase: Colorize by Material Delivery Date, highlighting conflicts with plan...

Sample usecase: Visualize remaining activities to mark testpack as done
C3D - Visual Project Controls

Sample Usage: Generate Forman daily report and task list

C3D Mobile: Instead of a paper; generate a BIM model for each forman
Why JogAmp on Java?

• Availability:
  • Java, OpenGL, OpenCL, OpenAL, ..
  • Multiple Vendors
    – OpenJDK / IcedTea
    – Oracle JDK
    – IBM J9, ..
    – PhoneME
    – JamVM
    – CacaoVM
    – Dalvik
  • x86, arm, ppc, sh4, ..
  • GNU/Linux, Android, BSD, Mac OSX, Solaris/OpenIndiana, MS Windows
Why JogAmp on Java?

• Managed Code
  • Common API for
    – Windowing
    – GLDrawable / GLContext / GLSL
    – I/O, Resource Handling (Texture, Code, ..)
    – Rendering
  • OpenGL Pipelining / Debugging / Trace
  • Access to vast number of API / Middleware
JogAmp Continuity / Usage

- Usage [http://jogamp.org](http://jogamp.org)
  - Ardor3D
  - C3D Studio [http://c3d.com](http://c3d.com)
  - Elflight Engine
  - Processing
  - Gephi
  - NASA Worldwind
  - Java3D
  - ...
JogAmp Continuity / Maturity

• Maturity
  • Version 1
    – JSR-231
  • Version 2
    – OpenGL Profiles (ES 1+2, GL 2 + 3 + 4 )
    – Windowing Toolkit Abstraction
    – Continuity Build/Test Server http://jogamp.org/chuck/
    – 86 GlueGen + 278 JOGL Unit Tests

• Community Effort
  • Ports [FreeBSD, ARM-HF, ..]
  • Engine & Device Support
  • Bug Entries, Test Cases & Fixes
  • Code Reviews, Communication & General Help
JogAmp Deployment

- Preinstalled Bundles
  - Modularized JARs
  - Android APKs (modular, or all-in-one)
  - Maven
- Online / Cached
  - Automatic Native-JAR loading support
  - Applet
    - Classical
    - JNLP
  - Webstart (JNLP)
OpenGL Profiles
Windowing Toolkits

Native Window

Native Surface

X11 (Unix)  GDI (Windows)  Android  Coco (MacOSX)  SWT (SWT Canvas)  AWT (AWT Canvas)

GLX  WGL  EGL  CGL

GL
2011 – 2012 Enhancements

- GLMediaPlayer
  - Uses OpenMAX on Android via ICS's MediaPlayer / libstagefright
  - Uses libav/libffmpeg where available
    - Missing [OpenAL] audio output
  - Missing native implementation for Win32 / OSX
- Graph API for Curve & Text rendering via GPU
  - Experimental UI
- Mobile Bindings (Android Intel/ARM, Linux ARM)
- Stability
- NEWT AWT / SWT Enhancement
- Documentation & Tutorials
- Higher Community Participation
NEWT

• Seamless integration w/ native Windowing System
  • Multithreaded Access to Window Surface
  • Lock free event handling
  • Transparency, decoration and offscreen control
  • Screen Mode API (fullscreen, resolution & rotation)
  • X11, Win32, OSX, Android, OpenKD, .. implementation
  • AWT and SWT integration via native re-parenting
  • Desktop & Mobile
JOGL Android Binding

• Why?
  • Short Development Cycles
  • No device specific development
  • Supports NEWT (Multitouch, Surface, ...)
  • Same code compiled for all – almost Android agnostic.

• Deployment:
  • adb install jogl.apk
  • adb install myFancyapplication.apk
  • Manual Daisy Chained ClassLoader, if desired.
Graph API
Resolution Independent
Shapes and Curves
Resolution Independent Curve Rendering API

- Based on Paper:
- **NOT** Loop/Blinn
- Patent Free
- Can Render Bezier, Bsplines, NURBS
Resolution Independent Curve Rendering API

- Why?
  - Resolution Independent Text Rendering
  - GPU based - Fast
  - Seamless integration into Renderer (Scenegraph,...)
  - New User Interface – across devices

  - http://www.youtube.com/watch?v=Rqsu46ifMaw


Donec ut dolor et nulla tristique varius. Vivamus in lorem. Maecenas in ipsum ac justo tempus tempus.
JOGL Graph API

- Outline → OutlineShapes → GLRegion
- Renderer
  - RegionRenderer
  - TextRenderer (same as RegionRenderer)
    - Helper methods for texts and fonts.

```java
outline.addVertex(x, y, z, w, onCurve);

....

outlineShape.addOutline(outline);
outlineShape.addOutline(outline2);
region = GLRegion.create(outlineShape, getRenderModes());
region.render(gl, outlineShape,...);
```
JOGL Graph API

- Initializing:
  - Read Outlines (from font, svg, application, ...)
  - Modified Constrained Delaunay Triangulation
  - Generate Region

- Rendering:
  - VBO buffers
  - Realtime manipulation – weights
  - Transformation....
GPU based Resolution Independent UI

- Abstracted from the windowing toolkit
- Support multithreading
- Seamless integration into
  - A native window (HUD)
  - A custom Scenegraph (2D plane within 3D)
- High Quality rendering
- Super Fast
UI Requirements (WIP)

- Generic UI Rendering
  - Rendering shall be performed using native rendering TKs (JOGL, ..)
  - Render primitives on an offscreen 2D plane to be
    - integrated into a custom 3D scenegraph
    - rendered as a HUD.

- Generic User Input
  - Input events should be delegated from the custom scenegraph to the UI input module.
Using JOGL and Eclipse RCP in fluid dynamics research
My research

- Computational fluid dynamics
- Computational elastodynamics
- Soliton formation and interaction

- Lots of coding (simulator is ~25KLOC)
- Animated graphical display
- Requires interaction with the graphics to help invent and refine the algorithms
The Repeated Replacement Method: A Pure Lagrangian Meshfree Method for Computational Fluid Dynamics

Wade A. Walker*
Austin, Texas, United States of America

Abstract
In this paper we describe the repeated replacement method (RRM), a new meshfree method for computational fluid dynamics (CFD). RRM simulates fluid flow by modeling compressible fluids’ tendency to evolve towards a state of constant density, velocity, and pressure. To evolve a fluid flow simulation forward in time, RRM repeatedly “chops out” fluid from active areas and replaces it with new “flattened” fluid cells with the same mass, momentum, and energy. We call the new cells “flattened” because we give them constant density, velocity, and pressure, even though the chopped-out fluid may have had gradients in these primitive variables. RRM adaptively chooses the sizes and locations of the areas it chops out and replaces. It creates more and smaller new cells in areas of high gradient, and fewer and larger new cells in areas of lower gradient. This naturally leads to an adaptive level of accuracy, where more computational effort is spent on active areas of the fluid, and less effort is spent on inactive areas. We show that for common test problems, RRM produces results similar to other high-resolution CFD methods, while using a very different mathematical framework. RRM does not use Riemann solvers, flux or slope limiters, a mesh, or a stencil, and it operates in a purely Lagrangian mode. RRM also does not evaluate numerical derivatives, does not integrate equations of motion, and does not solve systems of equations.

Editor: Jörg Langowski, German Cancer Research Center, Germany
Received: September 7, 2011; Accepted: June 3, 2012; Published: July 6, 2012
Copyright: © 2012 Wade A. Walker. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
Funding: No current external funding sources for this study.
Competing Interests: The author has declared that no competing interests exist.
* E-mail: wwalker3@austin.rr.com
Research vs. production code

- Exploratory code, not production code
- Constantly changing and trying new things
- Need flexibility and interactivity

- Java provides garbage collection
  - Good for apps that are constantly being changed
- Eclipse RCP provides app framework
  - Good for writing workbench-like GUI apps
- But how to render interactive graphics?
Why JOGL?

- Easy hardware-accelerated graphics in Java
- Cross-platform (my simulator runs unaltered on Windows, Mac, and Linux)
- Runs fast, gives good interactivity
- Supported by great guys 😊
Demo simulator
JOGL on Embedded Devices

- Development Env:
  - Beagleboard / Pandaboard w/ ARM7l / PowerVR
    - Linux
    - Android
  - Platform based Unit tests
  - Continuous Integration with auto-builds.
  - Cross platform compilation/building
  - Utilizing HW accelerated GL if available (EGL/ES)
JOGL Android Binding

Details:

- Enhanced EGL binding
- Exposing GLES1 and GLES2 native profiles
- GL2ES1 and GL2ES2 profiles for Desktop/Mobile
- Using Android SDK/NDK
  - Requires SDK Level 9, Android 2.3 Gingerbread for NIO Surface access
- Tested with:
  - Pandaboard - PowerVR
  - Samsung Galaxy S2 – Arm/Mali
  - Samsung Galaxy S – PowerVR
  - Samsung Tablet / ASUS TF2 – Tegra2
  - ASUS TF3 - Tegra3
JOGL Android Binding

- Cross platform builds/tests with Linux host
- Scripts provided in source code repository
- NEWT Helper class (NewtActivity)
  - Android Surface / NEWT Window mapping
  - Android Input Event / NEWT translation
JogAmp's Ecosystem

- Middle and high level APIs
  - Scenegraphs: Ardor3D, Java3D, JMonkeyEngine, JReality, Aviatrix3D, 3DzzD, Avengina, Xith3D, MSG
  - UI frameworks: FengGUI, Nifty GUI
  - Visualization frameworks: LibGDX, Jzy3D, GLG2D, Gephi, …
  - Sound framework: Paul Lamb Sound Library

- Low level APIs & bindings
  - JOGL, JOCL, JOAL, JInput for JogAmp
Nifty GUI

- UI framework
- Layout in XML or Java
- Some build-in widgets, effects and styles
- Focused on easing controls creation rather than on providing tons of “standard” widgets
Java3D

- Object oriented and scenegraph based API
- Runs on top of JOGL
- Supports GLSL
- Spatial sound
Java3D

• Pros:
  • Quite easy to learn
  • Lots of tutorials and examples
  • Importers for some mainstream formats
  • Supports some “exotic” devices (multiple screen projectors, gloves, headsets, …)
Java3D

- Cons:
  - Very dependent on AWT (hard to port to NEWT)
  - Bad reputation (only minor maintenance loads for years, replaced by Prism in JavaFX)
  - Performance concerns (memory, speed)
  - Lacks lots of “common” features already implemented in other popular engines
Ardor3D

- Java based retained mode 3D engine
- Runs on top of JOGL, SWT OpenGL binding…
- Supports GLSL
- Skeletal animation
- Supports Android
- Hardware accelerated UI
- Terrain system (with geometry and texture clipmaps, level of details, …)
Ardor3D

• Pros:
  • Actively maintained
  • Most reliable JOGL based renderers
  • Abstracts rendering details but does not prevent you from extending its features with or without renderer independence
  • Render delegates used for legacy OpenGL code
  • Supports shaders (but still supports OpenGL 1.3)
  • Both community and paid support
Ardor3D

- Cons:
  - Focused on rendering (no sound, no physics, no networking, no state machines)
  - Lacks tutorials and very elaborated examples
  - Lacks importers (only Collada, OBJ and MD2)
  - Not yet any fully shader-based architecture (planned in Ardor3D 2.0)
  - No integrated game development environment
  - No build-in spatial partitioning
T.U.E.R

- (Graphically rudimentary) first person shooter
- Project started in October 2006
- Has used 4 different 3D engines
  - D3Caster (software rendering, raycasting)
  - My own engine (JOGL, optimized for flat mazes)
  - JMonkeyEngine 2 (with “my” custom JOGL renderer)
  - Ardor3D (with “my” JOGL 2 renderer)
- Focused on performance
T.U.E.R

- Relying on several third party libraries
  - JOGL 2.0
  - JOAL 1.1.3
  - JOrbis
  - Ardor3D 0.8
  - Paul Lamb's Sound Library
  - Fettle API (state machine framework)
T.U.E.R

- Main aspects about performance
  - Careful threading
    - Tick, update, render
    - No interruption in rendering code
    - No OpenGL context switch (when possible)
  - Careful use of native resources
    - Slicing of direct NIO buffers
    - Destruction of useless direct NIO buffers
T.U.E.R

• Main aspects about performance
  • Mesh optimization: merge of coplanar adjacent right triangles whose all 2D texture coordinates are canonical
  • Spatial partitioning
    – BSP trees (wip)
    – cells and portals (wip, only working on “flat” mazes)
T.U.E.R

- JFPSM, WYSIWYG FPS editor
  - 3D visualizer and editor
  - Designed for rapid prototyping
  - Focused on the editing of the game design by combining existing models and the packaging rather than on the modeling (Blender is better for that)
  - Allows to create “simple” 3D meshes from 2D maps and 3D patches
Q&A

- What's Next?
- Why is neither Swing nor AWT recommended?
- What are the supported IDEs?
Thank You

Rami Santina
Sven Gothel
Xerxes Ranby
Julien Gouesse
Wade Walker
Demoscene Passivist
Mark Raynsford
Michael Bien

... all the many contributors & users