GameDevelopers Conference

graphicREMEDY

Advanced OpenGL Debugging & Profiling with gDEBugger

Yaki Tebeka

Avi Shapira

- Graphic Remedy

Why is OpenGL debugging difficult?

- The application views the graphic system as a "black box"
  - You cannot put a breakpoint on an OpenGL function
  - You cannot watch OpenGL state variable values
  - You cannot view allocated graphic objects
- A render context is a huge state machine
- Commonly used features use a lot of state variables (material, textures, etc)
- OpenGL is a low level API: a few thousand calls per frame
- OpenGL error model

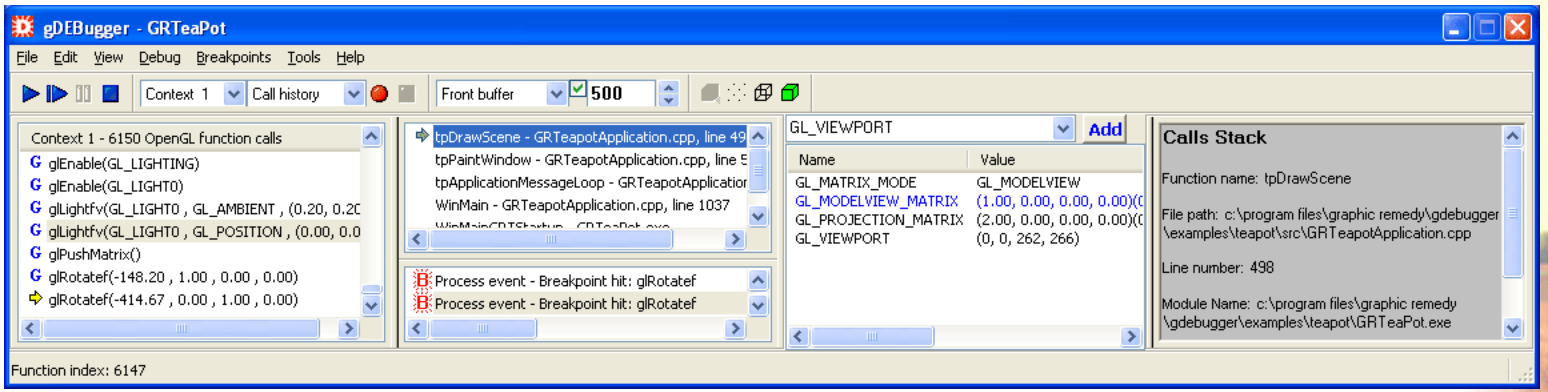
Use an OpenGL debugger!

- An OpenGL debugger transforms your debugging task into a “white box” model
  - Lets you watch OpenGL state variable values
  - Lets you put a breakpoint on an OpenGL function
  - Lets you view allocated objects
  - Breaks automatically when an OpenGL error occurs
  - View the application call stack and source code when the application breaks
  - Displays the OpenGL calls log
- ➢ Saved debugging time
- ➢ Improved product quality

GUI Overview

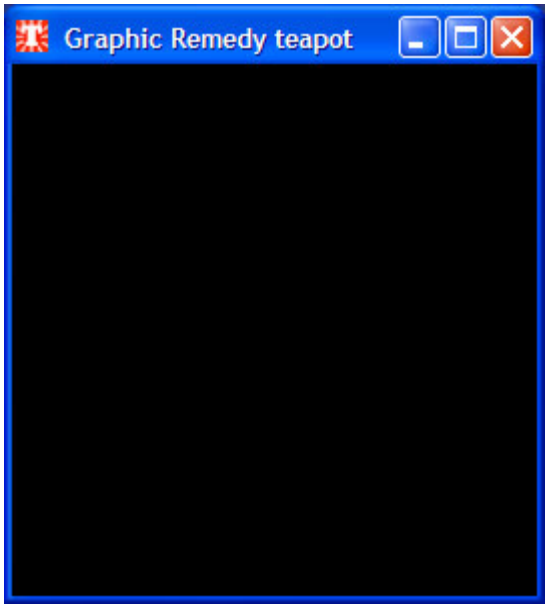
The gDEBugger GUI is designed to serve graphic applications:

- **Small footprint**
- **Customizable views**
- **Viewers**
- **Toolbars**
- **Always-on-top mode**



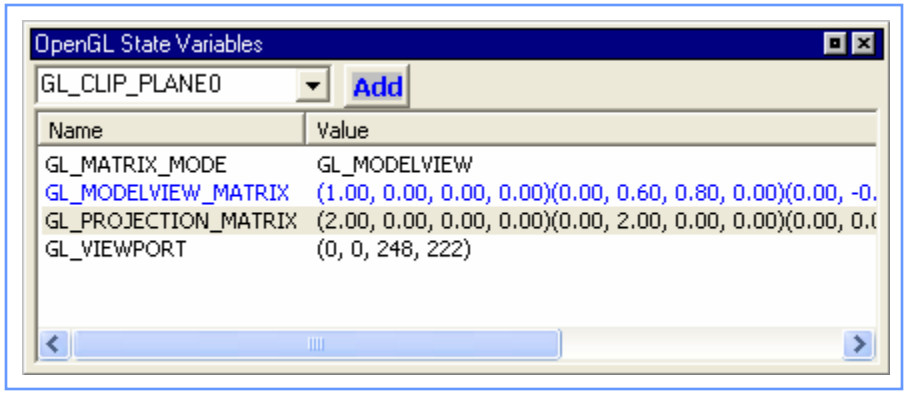
Case Study 1

The rendered object does not appear on the screen.



Remedy 1

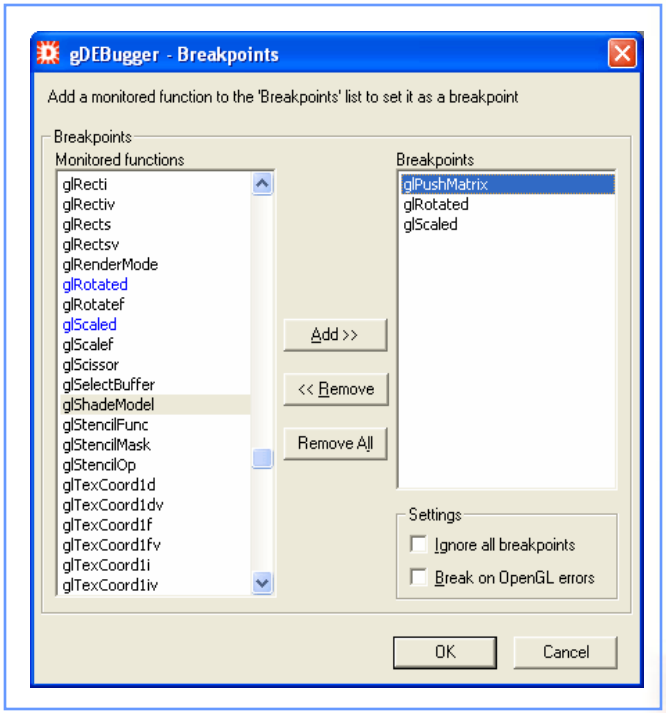
- Use the State Variables view:
  - Break the application run when the relevant object is being rendered
  - Watch the values of the state variable associated with the described problem



Remedy 1 - cont

Breakpoints

- Place breakpoints at the relevant state change functions
- View the call stack that changes the state variable associated with the described problem



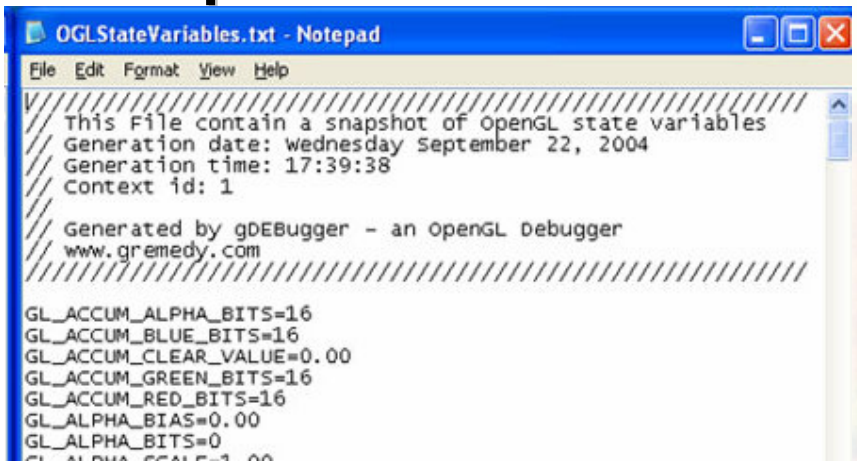
Case Study 2

One of the rendered objects material is incorrect. This happens in a certain mode / scenario.

Remedy 2

If the problem occur only in a certain mode or situation:

- Break the application run when things are ok
- Take a snapshot of the entire state machine
- Break the application run when things go wrong
- Take another snapshot of the state machine
- Compare the two snapshots



Case Study 3

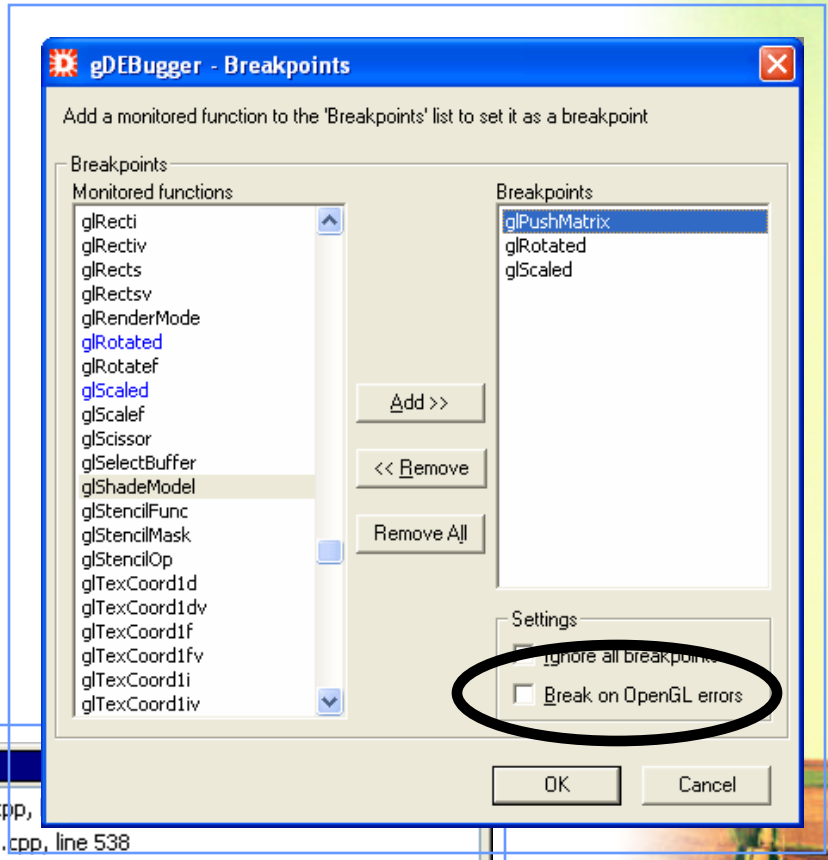
Locate OpenGL errors

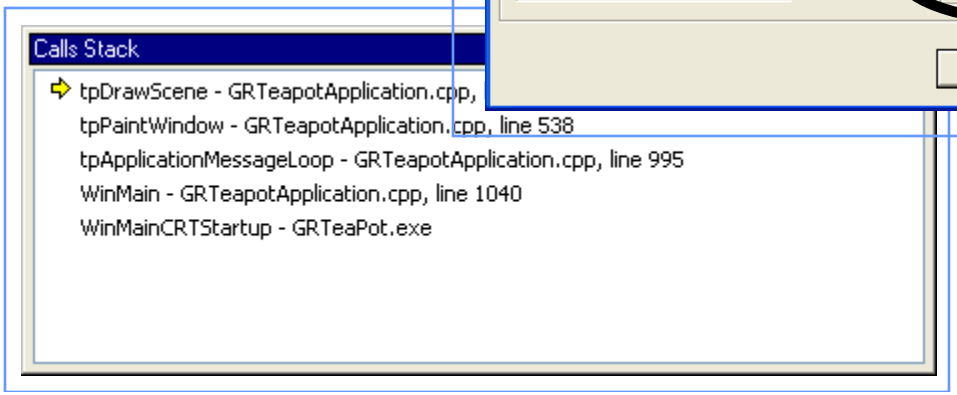
- Its easy to test for errors, but much harder to locate them!

```
GLenum openGLError = glGetError();
if (openGLError != GL_NO_ERROR)
{
      assert(0);
}
```

Remedy 3

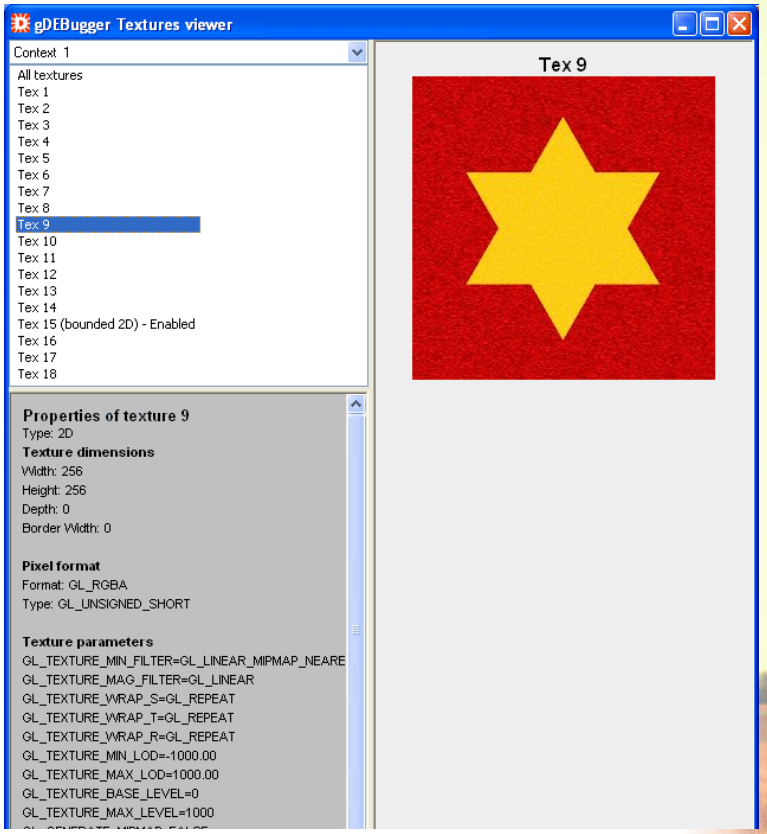
- Use the “Break on OpenGL error” option
- View the application call stack and source code at the error location





Texture viewer

- Helps find “Texture related” problems
- Lets you view textures allocated by the debugged application
- Displays texture properties
- Updates in real time
- Textures data can be saved to disk as an image



GL_GREMEDY_string_marker

- A typical OpenGL calls log contains an endless log of function calls
- This extension allows you to mark log segments and make the log more readable

```
glBindTexture(GL_TEXTURE_2D, 2) [Context 1 - Texture 2: ]
glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, (1.00, 1.00, 1.00, 0.00))
glEnable(GL_TEXTURE_2D)
glColor3ub(48, 64, 176)
glDisable(GL_CULL_FACE)

Marker: Drawing the object: #20

glDisable(GL_DEPTH_TEST)
glCallList(1)
glEnable(GL_DEPTH_TEST)
glEnable(GL_CULL_FACE)


glBindTexture(GL_TEXTURE_2D, 1) [Context 1 - Texture 1: ]
glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, (0.10, 0.10, 0.10, 0.00))
```

OpenGL Function Calls History

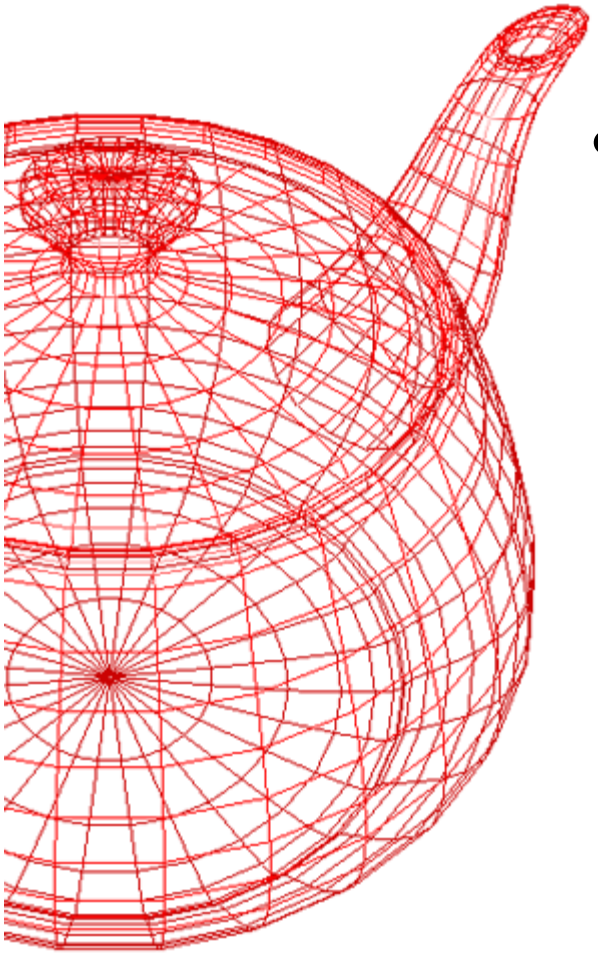
Context 1 - 5 OpenGL function calls

- G glClearColor(0.00, 0.00, 0.00, 1.00)
- G glClear(0x4100)
- glStringMarkerGREMEDY(Setting up material)
- G glShadeModel(GL_SMOOTH)
- glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, (1.00, 0.00, 0.00, 1.00))

Interactive mode

- Lets you view your graphic scene as it is being rendered by forcing OpenGL to draw directly into the front buffer.
- Lets you slow down OpenGL.
- Enables breaking the app run when the desired object is drawn.

Forced raster mode



- Lets you force OpenGL polygon raster mode.
  - GL_POINT
  - GL_LINE
  - GL_FILL

Why is OpenGL profiling difficult?

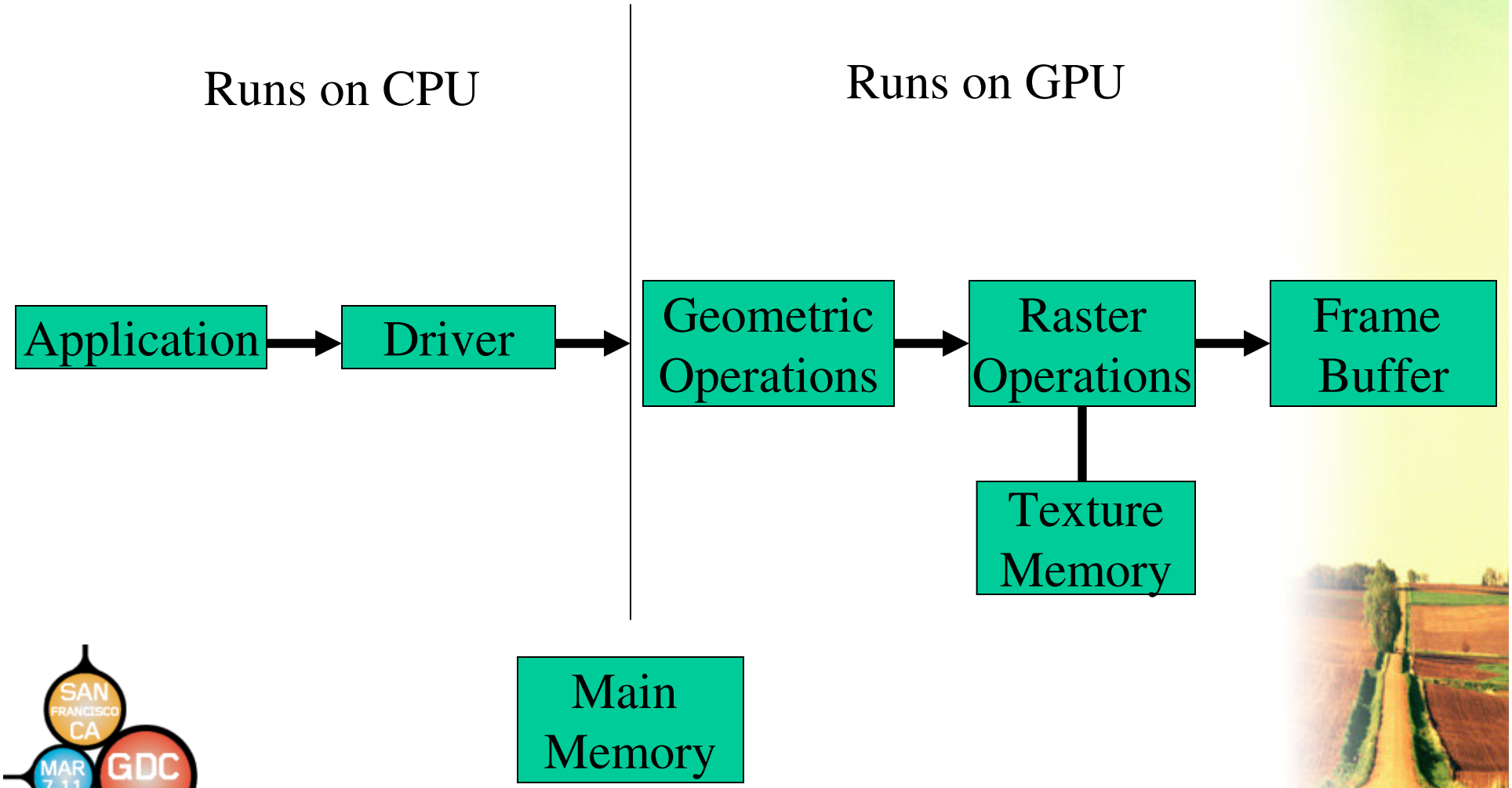
- The graphic system works as a pipeline: the slowest stage sets the pace
- Lots of potential bottlenecks: geometric operations, raster operations, CPU, bus, etc.
- The graphic system is highly parallel

Are you doing the right things?

- Check if there are OpenGL errors
- View your application's OpenGL calls log
- Disable vertical sync

Fix your code before profiling it!

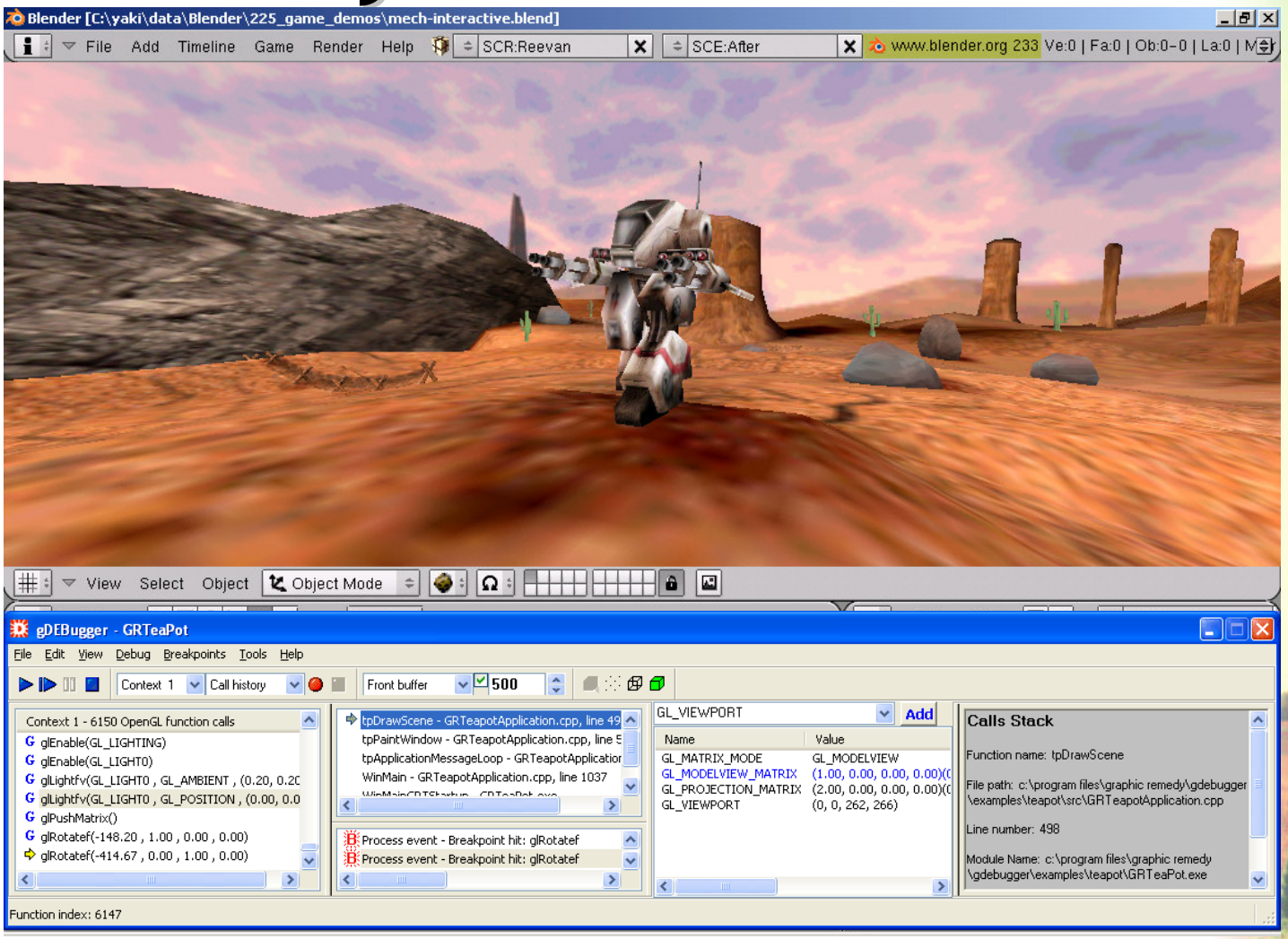
The Graphic Pipeline (2000 Miles View)



Finding Performance Bottlenecks

- Enter “Profiling Mode“ and turn on the FPS heads-up display
- “Turn off” the pipeline stages one after the other
- If performance improves when turning off a certain stage, you have probably found a bottleneck

Case Study


Blender [C:\yaki\data\Blender\225_game_demos\mech-interactive.blend]
File Add Timeline Game Render Help
SCR:Reevan
SCE:After
www.blender.org 233 Ve:0 | Fa:0 | Ob:0-0 | La:0 | M
View Select Object Object Mode
gDEBugger - GRTeaPot
File Edit View Debug Breakpoints Tools Help
Context 1 Call history
Front buffer 500
Context 1 - 6150 OpenGL function calls
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
glLightfv(GL_LIGHT0 , GL_AMBIENT , (0.20, 0.2
glLightfv(GL_LIGHT0 , GL_POSITION , (0.00, 0.0
glPushMatrix()
glRotatef(-148.20 , 1.00 , 0.00 , 0.00)
glRotatef(-414.67 , 0.00 , 1.00 , 0.00)
tpDrawScene - GRTeapotApplication.cpp, line 49
tpPaintWindow - GRTeapotApplication.cpp, line 5
tpApplicationMessageLoop - GRTeapotApplicatior
WinMain - GRTeapotApplication.cpp, line 1037
Process event - Breakpoint hit: glRotatef
Process event - Breakpoint hit: glRotatef
GL_VIEWPORT
Add
Name
Value
GL_MATRIX_MODE
GL_MODELVIEW
GL_MODELVIEW_MATRIX
GL_PROJECTION_MATRIX
GL_VIEWPORT
(0, 0, 262, 266)
Calls Stack
Function name: tpDrawScene
File path: c:\program files\graphic remedy\gdebugger
\examples\teapot\src\GRTeapotApplication.cpp
Line number: 498
Module Name: c:\program files\graphic remedy
\gdebugger\examples\teapot\GRTeaPot.exe
Function index: 6147

CPU / BUS bottleneck

- A typical application usually does more than just call OpenGL: physics, collision detection, LOD switch, etc.
- Isolate the GPU - ignore all render commands
- Application performance should improve

Texture fetch bottleneck

- Getting textures into the texture cache can be an expensive operation
- Force a simple 2X2 stub texture that always fits in the textures cache
- Application performance should improve

Fragment / Vertex shaders bottleneck

- Complicated Fragment / Vertex shaders can be expensive
- Force a simple Fragment / Vertex shader program
- Application performance should improve

gDEBugger v1.4 OpenGL Support

- OpenGL 2.0 standard
- The following extensions:

ARB_fragment_shader
ARB_shader_objects
ARB_shading_language_100
ARB_vertex_program
ARB_vertex_shader
ATI_fragment_shader
ATI_text_fragment_shader
EXT_vertex_shader
GL_ARB_depth_texture
GL_ARB_occlusion_query
GL_ARB_shadow
GL_ARB_shadow_ambient
GL_ARB_texture_border_clamp
GL_ARB_texture_compression
GL_ARB_texture_compression
GL_ARB_texture_cube_map
GL_ARB_texture_cube_map
GL_ARB_texture_env_add
GL_ARB_texture_env_combine
GL_ARB_texture_env_crossbar
GL_ARB_texture_env_dot3
GL_ARB_texture_mirrored_repeat
GL_ARB_texture_non_power_of_two
GL_ARB_vertex_blend
GL_ARB_vertex_buffer_object
GL_EXT_bgra
GL_EXT_blend_logic_op
GL_EXT_blend_minmax
GL_EXT_blend_subtract
GL_EXT_packed_pixels
GL_EXT_texture
GL_EXT_texture3D
GL_GREMEDY_string_marker
GL_HP_occlusion_test
GL_NV_fragment_program
GL_NV_fragment_program_option
GL_NV_fragment_program2
GL_NV_occlusion_query
GL_NV_primitive_restart
GL_NV_texgen_reflection
GL_NV_texture_shader3
GL_NV_vertex_program
GL_NV_vertex_program1_1
GL_NV_vertex_program2
GL_NV_vertex_program2_option
GL_NV_vertex_program3
GL_SGIS_generate_mipmap
GL_SGIS_texture_border_clamp
GL_SGIS_texture_edge_clamp
GL_SGIS_texture_lod
GL_SGIS_texture_select
GL_SGIX_depth_texture
GL_SGIX_interlace
GL_SGIX_shadow
GL_SGIX_shadow_ambient
WGL_I3D_genlock

New Versions Will Include

- Performance Performance Performance:
  - OpenGL function calls statistics: calls counters, spent time counters, etc
  - Track allocated OpenGL resources
  - Rendered primitive statistics
  - Primitive draw Batch sizes
- Buffer viewer: view pbuffers, FBOs, depth buffer, etc
- Shader source code viewer & Editor
- Disable a given extension/s

Questions?

- info [at] gremedy.com

www.gremedy.com

* We will be available in the exhibition shop-floor at the NVIDIA booth