CUDA-OpenGL interoperability

Visualization

Sergiu Oprea
Albert García García

soprea @ dtic.ua.es
agarcia @ dtic.ua.es

Contents

Introduction

OpenGL

CUDA + OpenGL interop

Example

Introduction

Computer graphics applications

Visualization

- Graphics-oriented hardware
- Use CUDA to perform calculations and then display visual results
- Efficient compute-visualization pipeline
  - No extra memory movements involved
  - Unlocks CPU cycles for other application tasks

OpenGL

OpenGL

Open Graphics Library

- OpenGL is an interface for the graphic hardware programmer
- API:
  - Primitives: points, lines and polygons
  - Properties: colors, lighting, textures
  - View: camera position and perspective

OpenGL

Example

```
glBindTexture( GL_TEXTURE_2D, textureID);
glColor3f(1.0f,0,0);
glBegin(GL_QUADS);
  glTexCoord2i( 0, h);
  glVertex3f(0,0,0);

  glTexCoord2i(0,0);
  glVertex3f(0,1.0f,0);

  glTexCoord2i(w,0);
  glVertex3f(1.0f,1.0f,0);

  glTexCoord2i(w,h);
  glVertex3f(1.0f,0,0);
glEnd();

SwapBuffers(hDC);
```

OpenGL is *state-based*, parameters are sticky.

CUDA + OpenGL interop

CUDA + OpenGL

Interoperability

- CUDA for calculations, data generation, image manipulation,...
- OpenGL to draw the pixels & vertices on the screen
- Efficient interaction/coordination
  - Share data via a common memory in the framebuffer

CUDA + OpenGL

Interoperability

- CUDA C uses memory management techniques similar to those of C (malloc, pointers,...)
- OpenGL stores data in generic and abstract buffers called buffer objects.
- The interaction between CUDA/OpenGL is based on a simple concept:
  - Map/Unmap an OpenGL buffer for CUDA interoperability

Example

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

1. Create a GL window
2. Create a GL context
3. Configure the GL viewport and coordinate system
4. Create a CUDA context
5. Generate one or more GL buffers to share with CUDA
6. Register buffers on CUDA

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

1. Create a window (OpenGL)
   - Every S.O. does it differently, in WIN 32 it is:
     - CreateWindowEx() Win32
     - It needs a Windows HDC:

```
HDC hDC;
hDC=GetDC(hWnd)
```

   - We can use GLUT/GLFW to manage the creation/destruction of windows.

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

2. Create a GL context

```
// create a wGL rendering context
HGLRC hGLRC;
hGLRC=wglCreateContext(hDC);
//  Enable rendering context
wglMakeCurrent(hDC,hGLRC);
// load OpenGL extensions to use buffers
glewInit();
```

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

3. Configure the GL viewport and the coordinate system

```
// configure viewport size
glViewport(0, 0, width, height);
// setup projection mode
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0,1.0f,0,1.0f,-1.0f,1.0f);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

3. Configure the GL viewport and the coordinate system

```
……
    // Enable depth test
    glEnable(GL_DEPTH_TEST);
    // Clean color and viewport
    glClearColor(1.0f, 1.0f, 1.0f, 1.5f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

4. Create a CUDA context
   - OpenGL context must be created first
   - Create a CUDA context:
     - Driver API: Use **cuGLCtxCreate()** instead of cuCtxCreate()
     - Runtime API: use **cudaGLSetGLDevice()** before using the GL API
   - The CUDA/OpenGL interaction is defined in :
     - cudagl.h for the Driver API
     - cuda_gl_interop.h CUDA C Runtime

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

5. Create OpenGL buffers

```
GLuint bufferID;
// buffer ID
glGenBuffers(1,&bufferID);
// set UNPACK for the current buffer
glBindBuffer(GL_PIXEL_UNPACK_BUFFER, bufferID);
// memory assignment
glBufferData(GL_PIXEL_UNPACK_BUFFER, width * height * 4,
NULL, GL_DYNAMIC_COPY);
```

CUDA + OpenGL

Example

Steps to configure OpenGL with CUDA:

6. Registrar los buffers en CUDA

- Driver API:
  - cuGLRegisterBufferObject(GLuint bufferobj);
  - cuGLUnregisterBufferObject(GLuint bufferobj);
- Runtime API:
  - cudaGLRegisterBufferObject(GLuint bufObj);
  - cudaGLUnregisterBufferObject(GLuint bufObj);

CUDA + OpenGL

Example

Map/Unmap

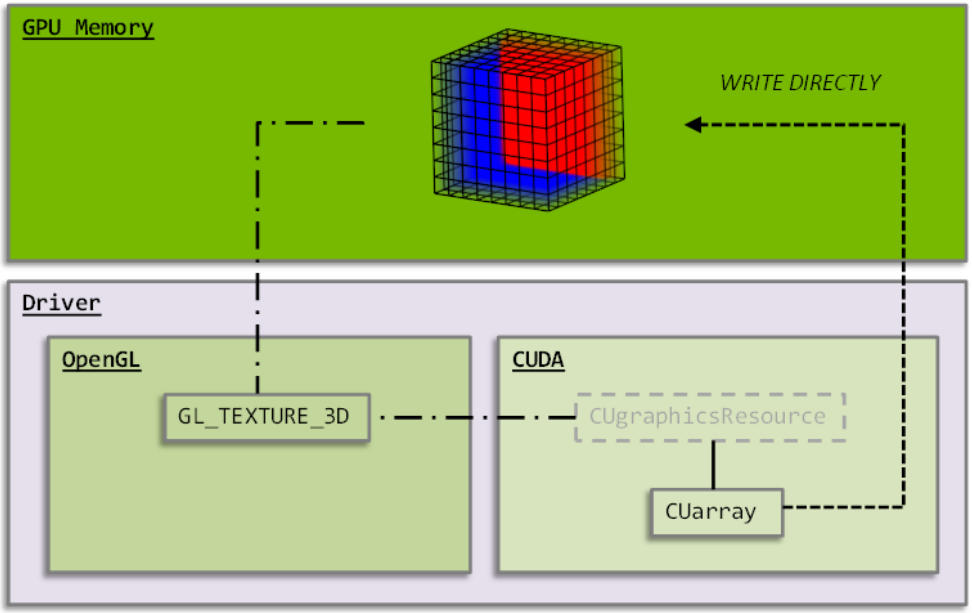
```
// map
float4 *dptr;
cudaGraphicsMapResources(1, vbo_resource, 0);
size_t num_bytes;
cudaGraphicsResourceGetMappedPointer((void **)&dptr, &num_bytes,
                                     *vbo_resource));
// cuda kernel
launch_kernel(dptr, mesh_width, mesh_height, g_fAnim);

// unmap
cudaGraphicsUnmapResources(1, vbo_resource, 0);

// draw
Draw()
```

CUDA + OpenGL

Example Map/Unmap



https://github.com/nvpro-samples/gl_cuda_interop_pingpong_st

CUDA-OpenGL interoperability

Visualization

Thanks for your attention!

Sergiu Oprea
Albert García García

soprea @ dtic.ua.es
agarcia @ dtic.ua.es