EE 4702—GPU Programming

Midterm Exam Review

When / Where

Friday, 31 October 2025 CDT

Room 1218 PFT Hall (Here)

Conditions

Closed Book, Closed Notes

Bring one sheet of notes (both sides), 216 mm × 280 mm.

No use of communication devices.

Format

Several problems, short-answer questions.

Resources

Lecture “slides” used in class: `https://www.ece.lsu.edu/koppel/gpup/ln.html`

Solved tests and homework:. `https://www.ece.lsu.edu/koppel/gpup/prev.html`

It’s important to study the solutions.

Study Recommendations

Study this semester's homework assignments. Similar problems are likely to appear on the exam.

Exam problems written for students who have done the homework.

Note that doing the homework is not the same as looking at a solution.

Solve Old Problems—memorizing solutions **is not the same** as solving.

Following and understanding solutions **is not the same as** solving.

Use the solutions for brief hints and to check your own solutions.

Mathematics

Coordinates, Points, Vectors, Homogeneous Coordinates

Dot and Cross Products

Line / Plane Intercept

Matrix $\times$ Vector Product: How to, number of operations.

Transformations

Projections

Coordinate and Vector Classes

pVect, pCoor, pNorm, pMatrix

Matrix Constructors for Common Transformations

`pMatrix_Translate`, `pMatrix_Scale`, `pMatrix_Rotation`, `pMatrix_Frustum`.

Simple Physical Simulation.

Understand how world modeled.

Point masses, ideal springs, gravity field.

Time Step

Updating velocity and position.

Forces

Gravity.

Ideal spring.

Collision with platform.

Coordinate Spaces

Object, Eye, Clip, Window

CPU-Only Code

Rasterization

- Outer Loop: Primitives.
- Inner Loop: Fragments. (Iterate over barycentric coordinates.)

Ray Tracing

- Outer Loop: Pixels.
  - Cast a ray from eye through pixel . . .
- Inner Loop: Primitives (Triangles).
  - . . . and find closest intersection.

Barycentric Coordinates

Used to specify a point in a triangle.

Coordinates are denoted b_0, b_1, and b_2.

Some Properties

In the unit inteval: $0 \leq b_0 \leq 1, \quad 0 \leq b_1 \leq 1, \quad 0 \leq b_2 \leq 1.$

$b_0 + b_1 + b_2 = 1$

Can be used to form a two-level loop nest: (Code from `hw03.cc`)

```
for ( float b0=0; b0<=1; b0 += db0 )
  for ( float b1=0; b1<=1-b0; b1 += db1 ) {
      const float b2 = 1 - b0 - b1;
      // Compute coordinate in window space (wf) by interpolating using b0,b1,b2
      pCoor wf = b0*w0 + b1*w1 + b2*w2;
```

Primitive Topologies (Vertex Grouping)

Individual Triangles. (Vulkan `vk::PrimitiveTopology::eTriangleList`).

Triangle Strip. (Vulkan `vk::PrimitiveTopology::eTriangleStrip`).

Triangle Fan. (Vulkan `vk::PrimitiveTopology::eTriangleFan`).

Vertex Attributes

Object-space coordinate.

Color (material property).

Vertex normal.

Vulkan Primitives and Vertex Specification

Primitives (Primitive Topology)

Vulkan 1.3 Section 21.1 (`https://www.khronos.org/registry/vulkan/specs/1.3-extensions/html/chap21.html`)

`vk::PrimitiveTopology::eTriangleList`

`vk::PrimitiveTopology::eTriangleStrip`

`vk::PrimitiveTopology::eTriangleStrip`

`vk::PrimitiveTopology::eLineList`

`vk::PrimitiveTopology::eLineStrip`

`vk::PrimitiveTopology::ePointList`

Vulkan Storage Qualifiers (Storage Classes)

Storage Qualifiers

Used to specify *storage classes*.

Examples:

`uniform`: For declaring uniform variables. Need to know this for midterm.

`buffer`: For declaring storage buffers. For after midterm.

`shared`: Readable and writable by work group. Not covered before midterm

`in`: For declaring shader inputs. Need to know this for midterm.

`out`: For declaring shader outputs. Need to know this for midterm.

Vulkan Rasterization Rendering Pipeline

The Rasterization Stages Covered Here: Vertex, Geometry, Fragment

Fixed Functionality v. Programmable Stage. (See `https://www.ece.lsu.edu/koppel/gpup/2024/set-3-rend-pipe.pdf`)

Pipeline Stages

Know well for midterm exam: *vertex shader* and *fragment shader*.

Know vaguely for midterm exam: *geometry shader*.

A Pipeline Stage

Has a kind of *input* (Vertex, primitive, or fragment).

Has a kind of *output* (Vertex, primitive, or none).

Inputs are from CPU (vertex shader) or prior stage (geometry and fragment).

Outputs are to next stage (vertex, geometry) or frame buffer update (fragment).

Inputs from other shaders are user-defined.

Inputs from fixed functionality are pre-defined: *e.g.*, `gl_FrontFacing`.

Outputs to other shaders are user-defined.

Outputs to fixed-functionality are pre-defined: `gl_Position`.

Has a mandatory output.

Shader Programming

Programmable Shaders

Vertex, Geometry, Fragment.

For Each One:

Inputs, Outputs.

Conventional functionality.