314 review

Dinesh K. Pai

1

Announcements

- Don't forget to do the Course Evaluation (online) this week. It will close on Friday, April 11 at 11:59 PM.

2

Exam Format

- The exam will be similar to midterms, but longer. Closed book, closed electronic device (laptops, phones, etc. should be out of sight).
- 150 marks (in 150 minutes)
- Three types of questions
  - small questions (fill in the blank, many choices given) "Can you recognize the concepts?"
  - direct questions (write down short answer) "Do you understand the concepts?"
  - problem solving questions "Can you use your knowledge in a new situation?"

3

Exam Format

- The first two question types are meant to be easy. Try to go through them quickly, so that you have time to think about the problem solving questions at the end.
- Some questions may have multiple parts that build on one another. You can get credit for later parts if you show your steps

4

Exam Format (changes)

- New: you may be asked what a small piece of code does.
  - This will be similar to code you have seen in your assignments
- New(ish): You may be asked to write small program fragments. Exact syntax is not important, but conceptual understanding is. E.g., you should know different types of data you can pass to shaders, and how to do that from an OpenGL program. Straightforward if you understood what you did in the assignments.

5

Exam Preparation

- Review lecture notes, text, and assignments
- Everything covered in lecture could be on the exam
- Everything covered in listed textbook chapters could be on the exam
- I will provide some practice problems
- Extra office hours: 22nd, 23rd, 24th, 25th 1-2pm in 005 lab.

6

Textbook reading

- Read ALL of Chapters 1-18 and Appendix A, except as noted below
  - Skip all of Chapters 7,8,13
  - Ch 2: skip Eq. 2.5
  - Ch 5: skip 5.4
  - Ch 9: skip 9.3
  - Ch 11: skip 11.2.1 <- change
  - Ch 10: skip 10.3.2, 10.3.3
  - Ch 12: skip 12.2, 12.4
  - Ch 18: Understand concepts. No need to memorize the resampling equation. See lecture notes.

7

Course recap

8

General Changes This Year

- We introduced computer graphics using a modern, shader-based, approach
  - This is now the standard practice, for both OpenGL and DirectX, also WebGL and OpenGL ES
  - A significant change from previous offerings of 314
- A new required textbook, made available online for free from UBC library
  - Tried to stay close to the textbook to make it easier to review material
  - But some changes (e.g., better notation) and additions (e.g., interpolation) as needed

9

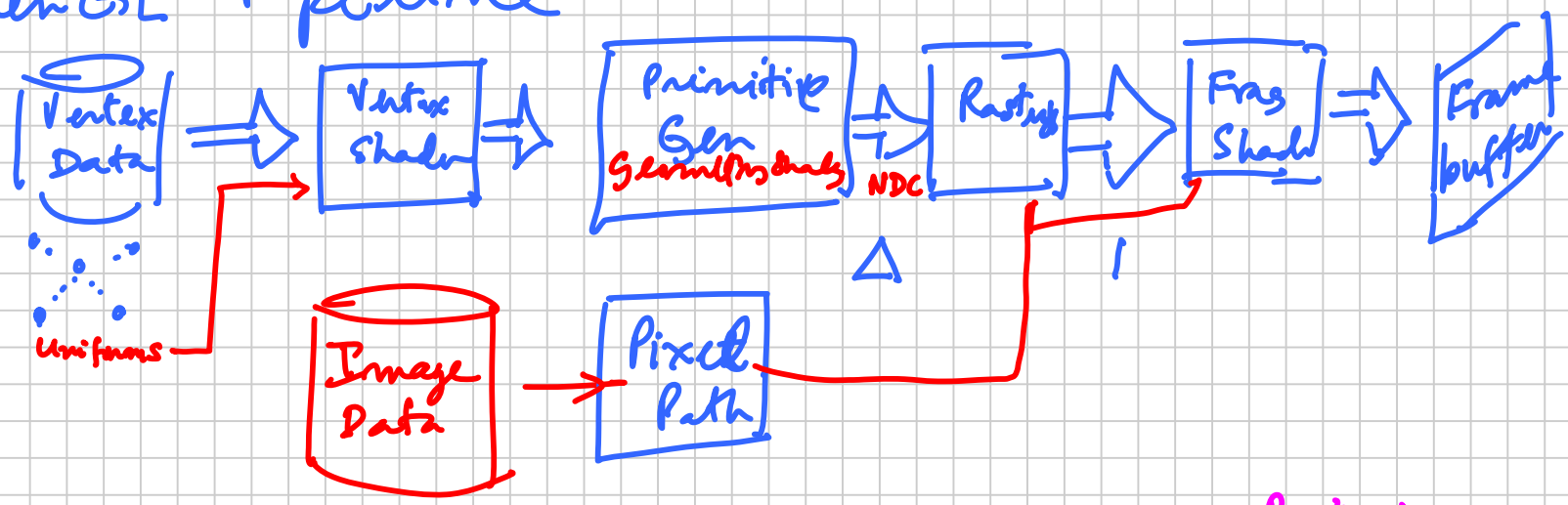
- Rather than fast forward through the course, will try to provide big picture, now that you know the most important pieces
- Will use the OpenGL and Conceptual Graphics Pipelines to highlight key points

10

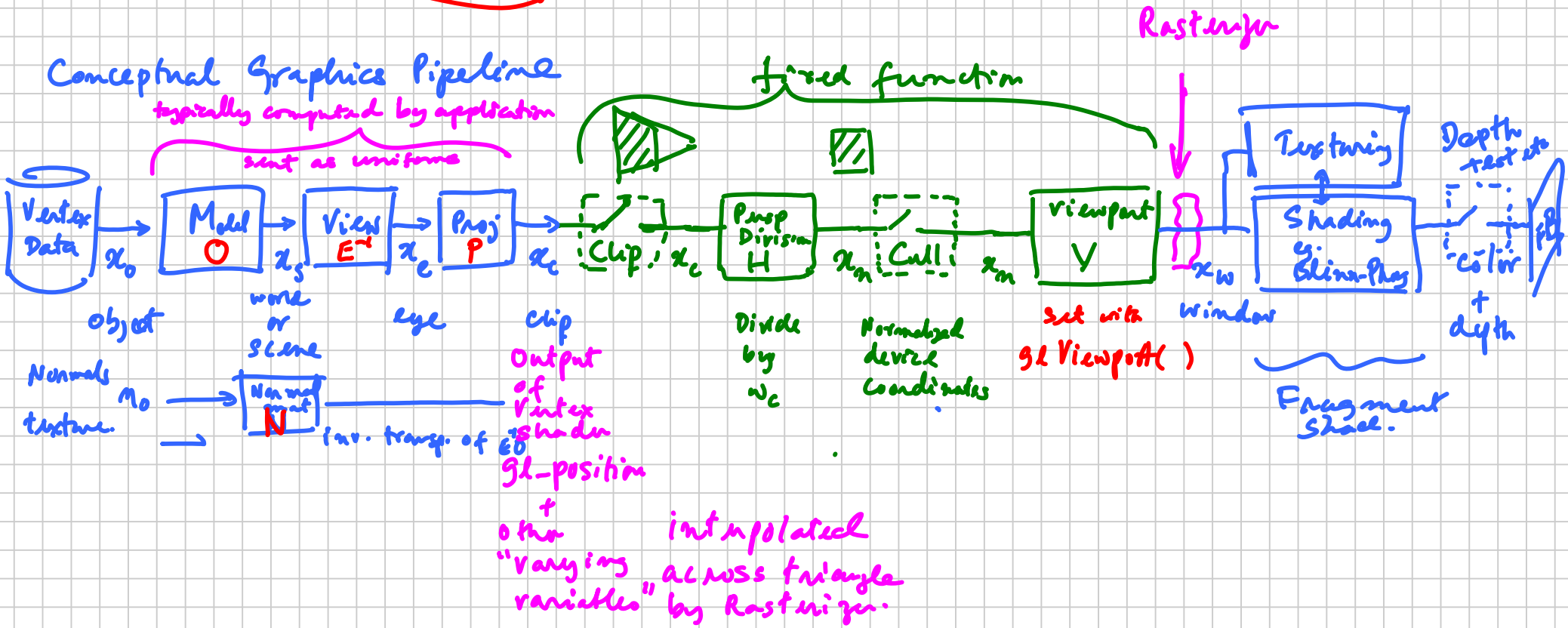
Pipelines

Note Title 2014-04-11

OpenGL Pipeline



Conceptual Graphics Pipeline



OpenGL basics

- client server model
- programmable pipeline
- Shaders: vertex and fragment
- useful data types and qualifiers
  - (vec4, mat4,...; uniform, in, out, layout)
- useful GLSL functions
  - matrix vector algebra, reflect, normalize, …

Client-side OpenGL

- useful GL functions
  - Should know what vertex buffer object and vertex array objects are
  - Communicating with server: glUseProgram, glUniform<type>, glBindBuffer, glBufferData, glVertexAttribPointer, glBindVertexArray, etc.
  - Drawing: glDrawArrays() or glDrawElements()
  - glClear
  - glViewport
- useful GLM functions
  - lookAt, perspective

inverse of what's in book

13

Representing POINTS using vector, affine, and projective spaces

- notation
- frames: coordinates are not just numbers, they are with respect to a frame
- homogeneous transformation matrices
- interpret a sequence of transformations
- normal matrix

14

Homogeneous transformations of points

- General: a "space" == coordinates + legal transformations of coordinates
- vector: linear transformations: rotation, reflection, scaling (about origin)
- affine: linear + translation
- projective: affine + central projection

add 1 coord + 4x4 transform

15

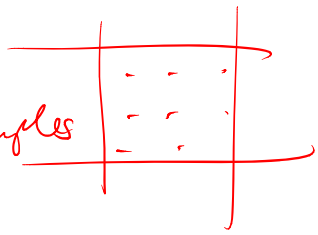
Useful math tools 1: Interpolation

- Bernstein polynomials
- partition of unity
- Bezier curves
- splines

16

Useful math tools 2: Sampling and Reconstruction

- aliasing and anti-aliasing
- filtering by averaging samples
- alpha blending
- mipmaps



17

Thanks!

Have a great summer

20