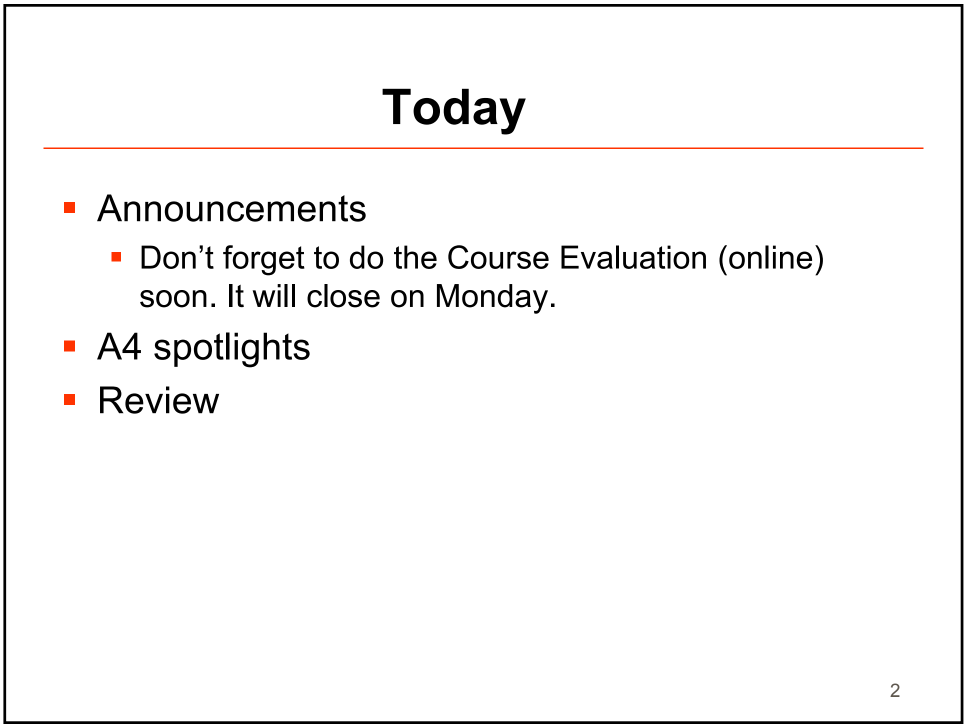
314 review

Dinesh K. Pai



Today

- Announcements
  - Don’t forget to do the Course Evaluation (online) soon. It will close on Monday.
- A4 spotlights
- Review

Assignment 4 spotlights

- Usual caveats apply… this is just a sampling, not necessarily the “best”, etc.
- Assignments graded on Friday were not included, since there was no time to do so.

Course recap

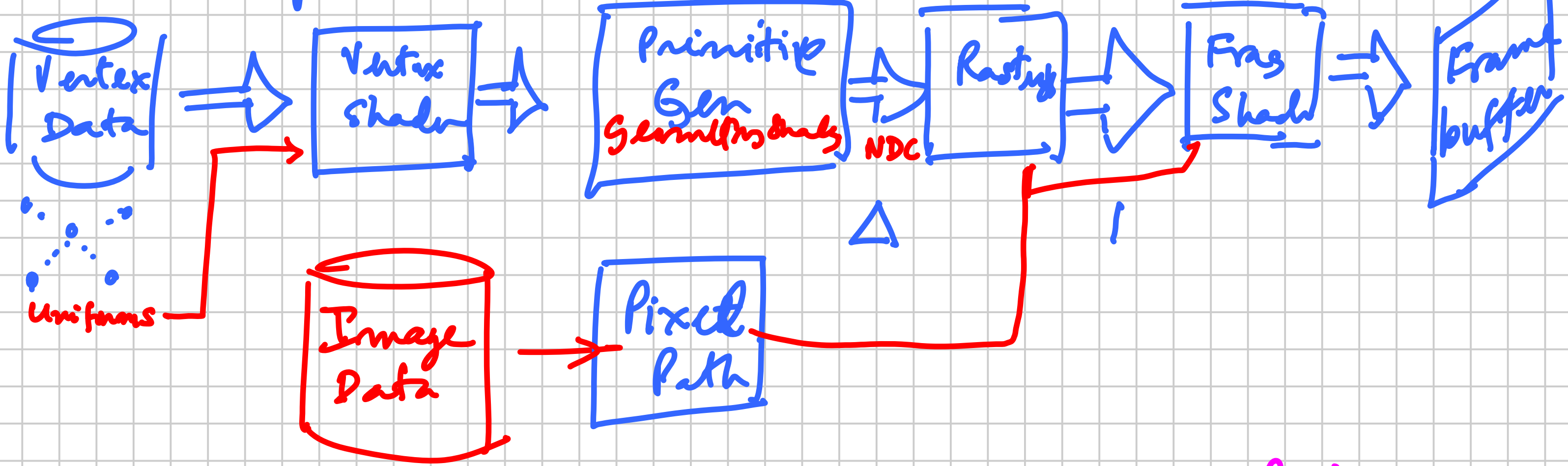
Significant Recent Changes to 314

- Computer graphics using a modern, shader-based, approach (from Jan 2014)
  - This is the state of the art in interactive graphics, for OpenGL and DirectX, also WebGL and OpenGL ES
- All assignments using Three.js and WebGL
  - Simplifies setup, experimentation, and deployment
- A new 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

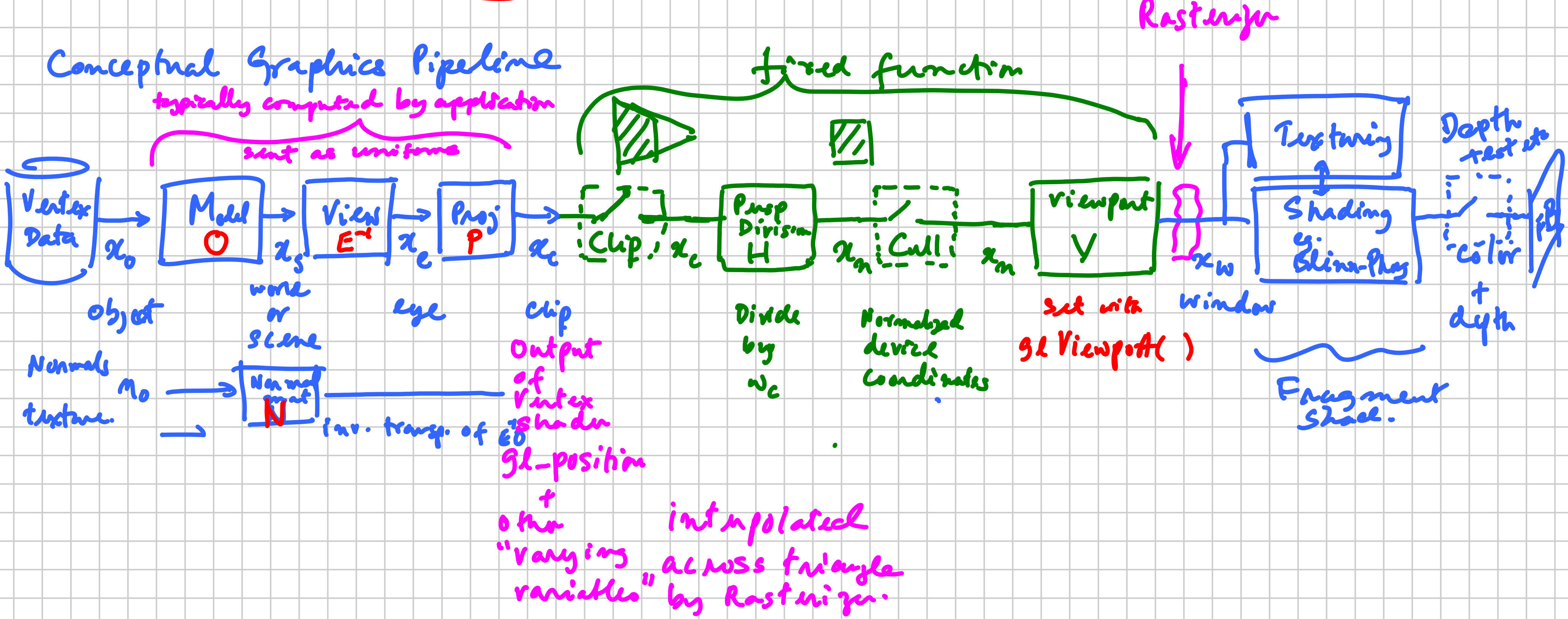
- Rather than fast forward through the course, will try to provide big picture, now that you know the most important pieces
- Will use the WebGL and Conceptual Graphics Pipelines to highlight key points

Pipelines

OpenGL Pipeline



Conceptual Graphics Pipeline

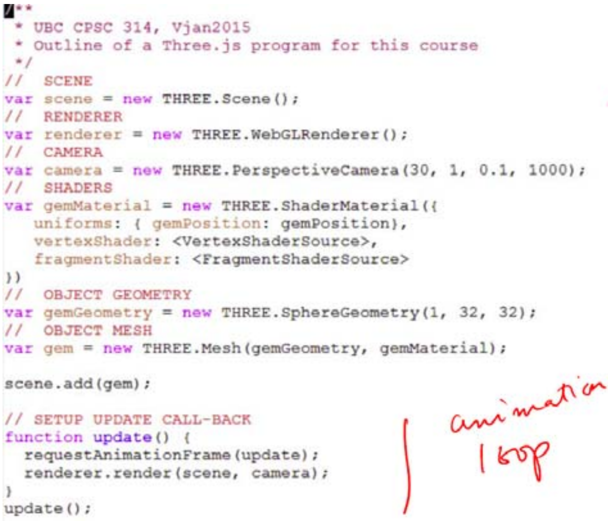


Other topics to know

OpenGL/WebGL basics

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

Client-side Programming, with Three.js



```
/**
 * UBC CPSC 314, Vjan2015
 * Outline of a Three.js program for this course
 */
//  SCENE
var scene = new THREE.Scene();
//  RENDERER
var renderer = new THREE.WebGLRenderer();
//  CAMERA
var camera = new THREE.PerspectiveCamera(30, 1, 0.1, 1000);
//  SHADERS
var gemMaterial = new THREE.ShaderMaterial({
   uniforms: { gemPosition: gemPosition},
   vertexShader: <VertexShaderSource>,
   fragmentShader: <FragmentShaderSource>
})
//  OBJECT GEOMETRY
var gemGeometry = new THREE.SphereGeometry(1, 32, 32);
//  OBJECT MESH
var gem = new THREE.Mesh(gemGeometry, gemMaterial);

scene.add(gem);

// SETUP UPDATE CALL-BACK
function update() {
  requestAnimationFrame(update);
  renderer.render(scene, camera);
}
update();
```

animation loop

Client-side Programming, with Three.js

- Understand the structure of a Three.js program
- Know useful Three.js functions
  - Setting up the SceneGraph
  - Communicating with the WebGL server using ShaderMaterial
    - Uniforms
    - Loading Vertex and Fragment shaders
    - Loading Textures with ImageUtils.loadTexture(), and passing them to shaders
  - Useful Matrix4 functions
    - lookAt, makePerspective, etc.

Representing POINTS using vector, affine, and projective spaces

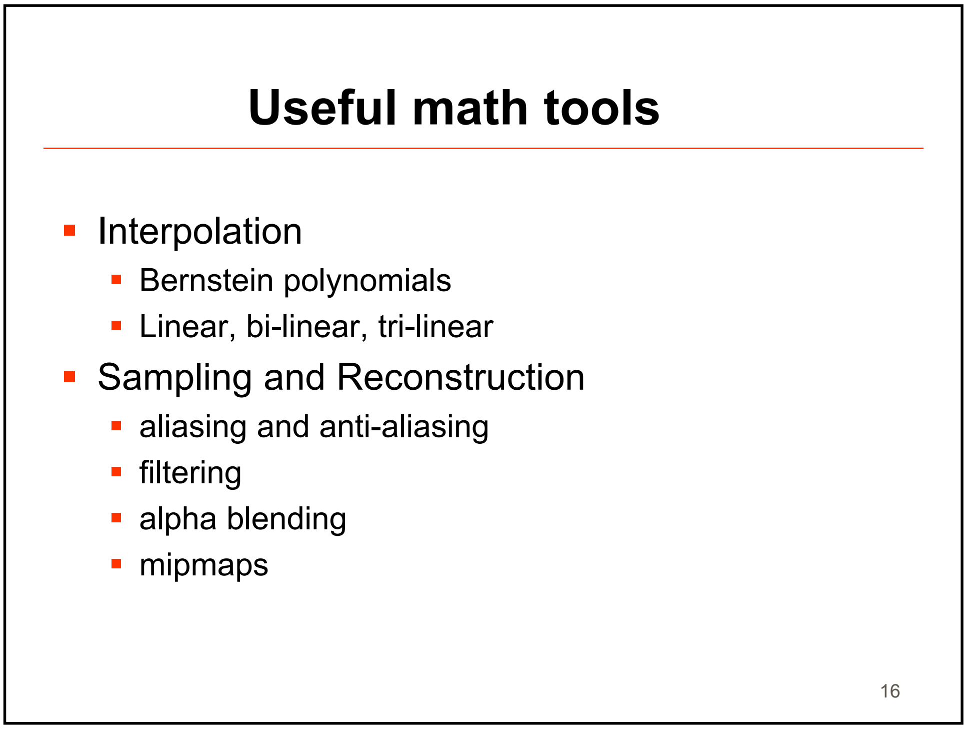
- notation
- frames: coordinates are not just numbers, they are with respect to a frame
- homogeneous transformation matrices
- interpret a sequence of transformations
- normal matrix

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

15



Useful math tools

- Interpolation
  - Bernstein polynomials
  - Linear, bi-linear, tri-linear
- Sampling and Reconstruction
  - aliasing and anti-aliasing
  - filtering
  - alpha blending
  - mipmaps

Thanks!

Have a great summer