 KUID: 2856114

1. In the context of Vertex Buffer Objects (VBOs):

(a) (5) What type of data is stored in them? That is, for what purpose(s) are the data used?

~~VBO: unsigned int~~

5 store generic vertex attributes associated with primitive!

(b) (5) Where are they stored (i.e., CPU or GPU), and why are they stored there?

4 GPU

GPU can deal with many vertex data during one time, in order to decrease computations in CPU. + minimize CPU-GPU transfer time.

(c) (5) What shader stage(s) have direct access to the data in VBOs?

5 Vertex Shader
Vertex Fetch

(d) (5) For each shader that has direct access, does it have access to the entire VBO or some part of it? Explain by describing how the shader accesses "the entire VBO or some part of it".

1 Fetch - all VS - only one vertex via "in" variables

The entire VBO, ~~any~~ anonymous of buffer objects. GPU can extra many data ~~ea~~ each time, ~~VBO GPU don't know~~

We use GLBindBuffers(types, VBO); we bind the whole VBOs!

└ Not relevant

2. (5) What is the "vertex fetch" processor, and what role does it play during rendering?

3 On right track. Incomplete

Vertex fetch must know how to extract data information and vertex attribute in ~~GPU~~ VBO of GPU.

During rendering, vertex fetch is active and extract ~~execute~~ current VBO

3. Consider the following two OpenGL functions:

```
void glBufferData(GLenum target, unsigned int size, void* data,
    GLenum usage);

void glVertexAttribPointer(unsigned int index, int size,
    GLenum type, bool normalize, unsigned int stride,
    void* pointer);
```

Possible values for "target" that we have seen are GL_ARRAY_BUFFER and GL_ELEMENT_ARRAY_BUFFER. So far, we have only seen GL_FLOAT used for the "type" parameter.

(a) (5) For what purpose is glBufferData used? That is, what two things does it do?

~~GPU~~ data from CPU to GPU and ~~data~~ memory allocated!

5

(b) (5) Explain what the first three parameters to glBufferData specify.

5

GLenum target is type of ~~vertex A~~ vertices, like GL_ARRAY_BUFFER

unsigned int size : ~~the~~ vertices data's size (units: bytes, ~~s~~sizeof( ))

void* data : it is actual vertices data.

(c) (5) What does glVertexAttribPointer do? Specifically, how are its first three parameters used? What information do they provide, and to what pipeline process do they provide it?

5

tell vertex fetch how to extract data in VBO of GPU!

unsigned int index : vertex attributes (position, ...) (layout = 0.)
where data to go!

int size: Size of vertex attributes (~~vec~~ vec 2, 3, 4)

GLenum type : data type of vertex attributes (~~vec~~ types = float)

Vertex shader need the information!

4. We have discussed two general types of attributes: "per-primitive" and "per-vertex". They differ in several respects.

(a) (5) What is the definition of a primitive in the context of a "per-primitive" attribute?

primitive: a unit of geometry, like points, triangles, lines...
"uniform" use.
Can only be changed between calls to render primitives!

0

(b) (5) In what shader program(s) (i.e., in what shader pipeline stage(s)) are per-primitive attributes directly accessible?

Every shader pipeline
We only learn vertex shader and fragment shader, this two per-primitive attributes can directly accessible.

5

(c) (5) In what shader program(s) (i.e., in what shader pipeline stage(s)) are per-vertex attributes directly accessible?

Vertex Shader

5

(d) (5) If I want every vertex to have a different value for some per-vertex attribute, what mechanism or structure do I use?

We can make many VBOs to store different vertex attributes.

5

(e) (5) If I want each primitive to have a different value for some per-primitive attribute, how do I make that happen?

glUniform function calls during rendering time.
change uniform values when I want to have a different value!

5

5. (a) (5) There is one thing that the vertex shader is required to do for the OpenGL graphics pipeline. What is that?

Which one???

① gl_Position and change Model cordinates to Logical Devices

② must make sure positions of vertice data in ~~LD~~ OpenGL

③ Use "in" keyword

① is close: Must set gl_Position in LDS coordinates

(b) (5) In addition, the vertex shader often/typically does something else in the context of the pipeline. What and for what other pipeline process?

Vertex Shader also ~~can~~ has "out" keywords to ~~change~~ output data to next ~~step~~ in the pipeline.

5

6. For our 2D programs, we have studied 3 different coordinate systems.

(a) (5) What are the three?

Model coordinates ⟷ Logical Device (LDS) ⟷ pixel

5

(b) (5) In what ranges of coordinates in those systems are we primarily interested?

x, y ~~coordi~~ coordinates range [-1, 1] — LDS

MC - arbitrary

pixels 0.. windowDim-1

(c) (5) In our ModelView-Controller framework, which of the systems are used by:

ModelView: ~~VAO=VBOs~~, go pipelines from Vertex Fetch to Framebuffer to create white scenes.

MC - LDS

10

Controller: GLFW controller

LDS. pixels

GLFW functions: create window...
event loop
handle call back: display, keyboard...

7. In our ModelView-Controller framework:

(a) (5) Why is it important for each ModelView instance to report the bounding box that contains its Model Coordinate (MC) geometry?

10

SO Controller can track overall BB

look for the Xmax, Xmin, Ymax, Ymin each ~~new model to create~~ ~~and change them~~.

① Avoid distortion when we need to Aspect Ratio Preservation!

② when we have more-than one models, some models may be hidden ~~hided~~ by window ~~other~~ If we do not have bounding box.

(b) (5) What is the difference between the Controller's "overallMCBoundingBox" and the ModelView class' "mcRegionOfInterest"?

3

overall MC Bounding Box: compares all models to find X,Y max,min ~~inclu~~ entire scene ~~entire~~

mcRegionOfInterest: it is current ~~mode~~('s XY max,min to compare ~~last~~ models

~~In order to find XY min,max during current stag~~ models.

→ Current area of MC we wish to see on screen.

Least Squares Data Fitting

...tence, Ungueness, and Conditioning

...quares Problems

Outline