Conceptual Question:

1. In C++, what is a virtual function and how does it relate to object polymorphism? Say I have a class Base and a class Derived that inherits from base. Both classes implement a non-virtual function called func(). If one invokes func() by dereferencing a pointer to a Base that actually points to an instance of Derived, how does the compiler know which implementation of func() to call?

Answer:

a virtual function or virtual method is an inheritable and overridable function or method for which dynamic dispatch is facilitated.

In this case, the compiler calls the func() of Base class.

But if the func() is a virtual function, the compiler will call func() of the Derived class.

1. The ShaderProgram class has several member variables of type int, such as attrPos and unifModel. What do these variables represent? How are they given values in the first place?

int attrPos; // A handle for the "in" vec4 representing vertex position in the vertex shader

int attrNor; // A handle for the "in" vec4 representing vertex normal in the vertex shader

int attrCol; // A handle for the "in" vec4 representing vertex color in the vertex shader

int unifModel; // A handle for the "uniform" mat4 representing model matrix in the vertex shader

int unifModelInvTr; // A handle for the "uniform" mat4 representing inverse transpose of the model matrix in the vertex shader

int unifViewProj; // A handle for the "uniform" mat4 representing combined projection and view matrices in the vertex shader

int unifColor; // A handle for the "uniform" vec4 representing color of geometry in the vertex shader

The given values in the first place:

attrPos(-1), attrNor(-1), attrCol(-1),unifModel(-1), unifModelInvTr(-1), unifViewProj(-1), unifColor(-1)

1. In the OpenGL Shading Language (GLSL), what is a uniform variable? What is an "in" variable? What is an "out" variable? How does a vertex shader pass data to a fragment shader?

Uniforms are constant across all instances of a shader program.

Ins/Outs differ across instances, Ins are more like Input and the Outs are like output.

Code Requirements

4.1 Scene Graph Classes:

I create a class Node:public QTreeWidgetItem and TranslateNode, RotateNode, ScaleNode which inherit class Node. The details can be found in mygl.h(Line 23~60) and mygl.cpp(Line 10~151)

4.2 Scene Graph Traversal:

I made Traverse function as a member function of class MyGL, you can find it at mygl.h and mygl.cpp(Line 156~173), I called this function at function paintGL() in mygl.cpp(Line 354).

4.3 Polygon Primitives with Vertex Buffer Objects:

You can find the implement of the cube at /scene/cube.h and /scene/cube.cpp and I used it to built the Character.

4.4 Character:

mygl.cpp(Line 231~261), I built the Body Node, Head Node, Limb01 Node and Limb11 Node. You can see it at running windows.

Also the Head and Limbs inherit the Body Node, the Limb11 inherit Limb01, which enables them move/rotate/scale with body.

To enable Head and Limb01, Limb11 to rotate or change by their own pivots and joints, I implement a function called Rotate\_Limb, you can find it in mygl.cpp(Line 196~202)

4.5 Fragment shader: Lambertian reflection:

Lambert.frag.glsl(Line 34~35)

4.6 Qt GUI: QTreeWidget:

The implement can be found in the running windows, So I don’t have to explain where my code is(There are everywhere).

Please ignore the color-changing of the cone and pipe, I just draw them at paintGL function to show the result of the 5.1 and didn’t care much about their color attributes.

Extra Credit:

5.1 Additional polygon:

Cone(/scene/cone.h and /scene/cone.cpp)

Pipe(/scene/pipe.h and /scene/pipe.cpp)

5.3. Animation:

I realized the animation of Translate, Rotate, Scale. Although it can only deal with the positive input, the negative is the same method and principle ,so I just didn’t write. (Using QTimer)

I added 2 variables to class Node, they are Animation(the matrix of translate and scale) and Animation\_Rotation(the matrix of rotation).

I also added 5 variables to class MyGL, they are QTimer \*timer, AniTrans, AniRot, AniSca, AniNode and 2 functions SetTimer() and setAnimation and 1 slot OnTimerOut(). You can find their codes in mygl.h and mygl.cpp.

I called the animation in function initializeGL() in mygl.cpp(Line 285~289) and you can see the result on the running windows.

My email address is [jiaww@seas.upenn.edu](mailto:jiaww@seas.upenn.edu), If you have any questions on my homework, just let me know and I could explain it as soon as I can.