CSC418 Assignment 1 Report

Soon Chee Loong, 999295793, c4soonch

10th October 2014

This code was written entirely by my own. However, I did refer to OpenGL online documentation as well as OpenGL code provided online to better understand how to work with OpenGL. A starter code was also provided to me by Prof. Kyros to help handle compilation and initialization of OpenGL for Assignment 1.

Design

I choose to program using the Object Oriented Programming Methodology as it will results in cleaner, flexible, and adaptable code. The drawback is that it took a lot of time to complete Assignment 1. The list of Objects, which I call Component in my code are given below. The commented code in Figure 1 shows the hierarchy in which the Components of each part of the penguin are connected.

```
// ----------- Penguin components defined ----------------------------------------------------------------------------------------
// Final Kinematics, from root to leaves
// Notation: a->b means  'a' has 'b' in its iterator list
// Current Progress: penguin->zoomer-> bodyJoint -> Body -> headJoint     -> Head     -> mouthJoint     -> Mouth
//                          ->translator                 -> handJoint     -> Hand
//                                                       -> leftLegJoint  -> LeftLeg  -> leftFeetJoint  -> leftFeet
//                                                       -> rightLegJoint -> RightLeg -> rightFeetJoint -> rightFeet
// ------------------------------------------------------------------------------------------------------------------------------
```

Figure 1: Code shows the connection between the various entities of the penguins.

Basically, a single Penguin object is the virtual base class of all Components. It contains an iterator

of Components to be able to store all the various objects such as Torso, Leg, Head, Mouth, and Feet in its iterator.

This design utilizes polymorphism and calls each different object to execute its own Display() (written as Update() in code) method.

Although the transformation between each component's coordinates are handled by OpenGL,

I had to compute suitable relative positions between these objects when drawing them.

For example, I had to compute postions to allow the top of the Circular Torso to be placed

slightly below the bottom of the Rectangular Head. Also, I needed to make sure that the objects

are connected in proper order such that they are drawn in correctly. Improper ordering of the

objects in the connection can result in the Torso output overwriting the output of the Leg.

This looks physically incorrect in reality.

Components are added to the existing connection using the AddConnection() method that is available from the virtual base class, Entity.

Transformation is done neatly by including a Joint object between different Components.

As the output of a Penguin-liked Object is a specific one, the Joint Object is used

to handle all specific computations of transformation between different Component's coordinate frame

such that the final output appears like a Penguin. This allows each component to execute its methods

within its own coordinate frame without realizing that it is part of a larger Entity, which is the Penguin.

Available Objects are:

`Rectangle`
: Draws fixed size rectangles.

`Circle`
: Draws fixed size circles.

`Color`
: Changes the current color.

`Zoomer`
: Uniform scale of the model view matrix.

`Translator`
: Translate the model view matrix.

An advantage of this design is that it is very easy to specify the structure and hierarchy of the operations. For example, the following code:

```
Entity Hand;
Hand.AddComponent(new Color(0.5, 0.5, 0.5));
Hand.AddComponent(new Rectangle(0.5, 0.5, 0.5));
Entity Body;
Body.AddComponent(new Joint(x, y, angle, arm));
Body.Update();
```

This results in a Hand Component including a Color, Rectangle, and Joint in its iterator. When the Component that points to Hand calls Hand to Update(), Hand iteratively calls the Components that it points to Update() as well.

The penguin has the following parts:

Body/Torso

The main entity that every other Component is attached to. This has two degrees of freedom for translation.

Head

The Head has the eyes and the upper mouth which moves with the head. The Head has one degree of freedom to rotate about the Body it is connected to.

Mouth

The Mouth is connected to the Head with a joint, allowing the Mouth to be translated vertically. This simulates the Penguin opening and closing its mouth.

Left and Right Legs

The Legs are connected to the Body with joints. The joints provide a degree of freedom each to rotate each Leg.

Left and Right Feet

The Feets are attached to their respective Legs with joints, each providing a degree of freedom for rotation of the Feet.

Transformations

Translations
All translations are done by `Joints and Translator`.
Rotations
All rotations are also done by `Joints`.
Scales
Only Uniform Scale is needed for the Penguin, as it does not physically look realistic for the Penguin to shear in any direction. It is called `zoomer` in code.

Shapes

Rectangles
Rectangles are drawn by using the Polygon with for vertices.
Circles
Circles are drawn by using the parametric equation of circles. Each circle has a `smoothness` that specifies the number of points used to draw the circle. The points smoothness increase with the size of the circle.

Results

I successfully completed the entire Assignment. The Penguin is hardcoded to oscillate all its joins simultaneously like a pendulum. Small circles are drawn to depict the locations of the rotary joints. Finally, the output Penguin does look like a cute Penguin. There are the required 9 degrees of freedom to control using the spinners.