Object Manipulation with Hierarchical Frames

Due: Friday, March 23, 2018 (11:59 PM)

1 Overview

In this homework assignment, you will implement a 'baby mobile' composed of a number of objects. A mobile is a type of kinetic sculpture constructed to take advantage of the principle of equilibrium, and it is popular in the nursery or a baby room (https://en.wikipedia.org/wiki). Your baby mobile should contain some variety of objects and rotate around its parent node in constant speed. You should compose a hierarchical frame for the mobile objects, and apply transformation to these objects for the animation.

When you execute your program, the mobile should stay still. You should implement a proper keyboard interaction to activate the animation - i.e., for changing the speed of rotation and selecting the center of rotation or an object moving.

There will be the creativity points for your own feature to add. One of the options is to simulate physics on this mobile considering that each object has some weights which affects the height of each object automatically changing.

2 Define primitive geometries

In common/geometry.hpp, you will see the definition of a colored cube in the InitDataCube. To define a baby mobile, we ask you to implement on your own to create more geometric primitives other than a cube such as a sphere, a pyramid, a cylinder, or a circular cone. You should have at least three different types of primitives for building up the mobile objects. In the document, you should write up how you define these objects in OpenGL.

3 Complete get_linear and get_translation

Implement the functions get_linear() and get_translation() in common/affine.h. The purpose of these functions factorizes a given affine transform A into linear part (L) and translation part (T). Note that GLM specifies its matrix in column-major order. In your implementation of the functions, your element should be assigned in column-major order.

4 Implement hierarchical frames

You should represent the mobile in hierarchical frames. If the rigid-body transformation is applied to a certain node, its children should be affected by the changes in the parent model transformation. In this assignment, you should implement data structure for handling hierarchical rendering of your models such as Node class. If you finish this step, the mobile

structure is rendered in the 3D scene. To show the connection between a parent and a child object, please draw a line between them. You can use GL_LINES..

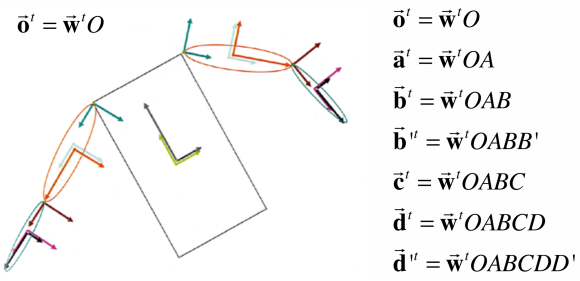


Figure 1: Hierarchical frame for the arms

5 Object manipulation with keyboard interaction

For manipulating objects defined hierarchically, you should track current manipulation node as global variable. The current manipulating node can be changed by keyboard which is handled by keyboard callback. Use 'o' key in your keyboard so that pressing the 'o' key cycles through the different node in the mobile. Moreover, you should visualize current manipulating node, for example, by drawing the selected object in wireframe.

The following is an example code for drawing the wireframe of your model instance.

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); // draw wireframe
model_instance->draw();
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // draw filled models again
```

In the initial state of program, the rotation speed of node is zero degree per second. In your keyboard callback, you should change its speed by pressing 'N' is minus 10-degree per second. 'M' is plus 10-degree per second, and 'R' is reset the rotation speed of selected node.

6 Creativity

Upto this point, the interface that you have made for this homework provides you the basic functions for generating 3D models and constructing a baby mobile interacting with a keyboard.

For your creativity points will be given mainly for modeling, transformation (motion) and vertex shading this time. The followings are some suggestions.

1. Modeling: you may compose your node with a collection of primitives (e.g. by attaching two cubes as one node.).
2. Transformation (motion): You can implement physics simulation for your mobile. Each node has a mass property so that the changes in the mass property cause the

transform changes in each rod. To have some realism, the changes are continuously (progressively) applied until reaching to intended level. That is, the user cannot manipulate a node until the transformation is fully applied.)

3. Vertex shading: Setting an interesting color for the objects. You need to explain why the coloring is interesting.

Note that you do not need to do any extensive part in fragment shader for this homework.

7 Documentation

It is important to explain what you have done to meet each specification for the homework. Please do not just cut and paste your code.

1. Code: Please write one-line comment for your code, such as class member variable and class member function.

2. PDF: In your write-up, you should write how you define each geometric primitives in common/geometry.hpp. Moreover, you should write how the transformation is applied in your baby mobile.

8 Specification & Grading

Specification	Grades
1. Define three geometric primitives	20
2. Affine transform factorization	10
2.1 get_linear (5)	
2.2 get_translation (5)	
3. Define hierarchical frames	20
4. Object manipulation using auxiliary frame	10
5. Correct Implementation for Keyboard Callbacks	5
6. Documents	10
7. Creativity	25
Total	100

Table 1: Specification and Grading

Sheet Date

Supplements.

Auxiliary frame.

Let's define two objects frame $\vec{a} = \vec{\omega} T_a R_a$, $\vec{b} = \vec{\omega} T_b R_b$.

$T_a R_a = A$, $T_b R_b = B$ are their rigid-body transformation.

We can visualize these two frames.

If we want to apply a transformation to b_y with respect to $\vec{a}$,

We can apply those matrices in step-by-step.

① Move $\vec{b}$ to $\vec{\omega}$. ② apply inverse of R_a to apply transformation

③ apply transformation R, ④ apply R_a again, ⑤ Move to $\vec{b}$ position

So the multiplication is $T_b R_a R R_a^{-1} T_b^{-1}$

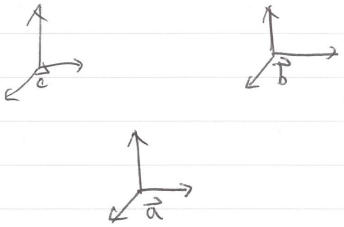
We call it auxiliary frame because we want the object's rotation ① about the object's origin ② about the related frame (ex. $\vec{a}$).

$\overrightarrow{\text{auxiliary}} = \vec{\omega} T_b R_a = \vec{\omega}\,\text{Auxiliary} = \vec{c} = \vec{\omega} C$

$$\begin{aligned}\vec{b} &= \vec{\omega} B \\ &= \vec{c} C^{-1} B \\ &= \vec{c} R \cdot C^{-1} \cdot B \\ &= \vec{\omega} \cdot C \cdot R \cdot C^{-1} \cdot B\end{aligned}$$

$C = T_b R_a$, $C^{-1} = R_a^{-1} T_b^{-1}$,

* Hierarchy frame.



$\vec{a} = \vec{\omega} A$

$\vec{b} = \vec{a} B = \vec{\omega} A \cdot B$

$\vec{c} = \vec{a} C = \vec{\omega} A \cdot C$

⋮

If I want to apply transform $\vec{b}$, $\vec{c}$ w.r.t $\vec{a}$?

⇒ use "auxiliary frame"

KAIST A4 FILE NOTE

Figure 2: Supplements