Quiz 2

Due: 4:00 am, Thursday, December 1

Note:
1. Discussion is prohibited and you are required to work on the problem independently.
2. To get the credit, sufficient and appropriate intermediate steps are required in your answers.
3. Late submission will not be accepted.

Following the example of Humanoid robot in the lecture, design a left-child right-sibling binary tree for modeling a car. You need to write a pseudo code for specifying the nodes such as defining “m”, “f”, and “sibling” for the node. Note that the speed of the car should be related to the rotation speed of the wheels.

Hints: You can follow the example in Lecture 25.

Solution:
The coordinate system of the car is as shown in Fig. 1, with the origin at the center of the chassis. Assume the car is moving along the y-direction. The linear speed of the car is the same as the linear speed of wheels, which can be calculated given the angular velocity of the wheels as

\[ v = \text{w}_\text{angular} \times \text{wheel}_\text{radius} \]

Fig. 1 A general tree model for a car

We can model the car with a general tree model as Fig. 2, which can be converted to a left-child right sibling binary tree as Fig 3.

Fig. 2 A general tree model for a car
As discussed in Lecture 25, each tree node can be defined in a structure:

typedef struct treenode {
    mat4 m;
    void (*f)();
    struct treenode *sibling;
    struct treenode *child;
} treenode;

We need to perform a preorder traversal:

```c
void traverse(treenode* root) {
    if(root==NULL) return;
    mvstack.push(model_view);
    model_view = model_view*root->m;
    root->f();
    if(root->child!=NULL) traverse(root->child);
    model_view = mvstack.pop();
    if(root->sibling!=NULL) traverse(root->sibling);
}
```

---

//Pseudo code for specifying the nodes

```c
//properties of the car
float chassis_width, chassis_height, chassis_length;
float distance_fb_wheel;  //distance between the front and rear wheel
```

//motion
float w_angular;

// define the chassis
chassis.f = chassis; //drawing function for the chassis
chassis.m = translate(0, w_angular*wheel_radius*time, 0);
chassis.sibling = NULL;
chassis.child = & front_left_wheel;

//define the front left wheel
front_left_wheel.f = wheel; //drawing function for the wheel
//we only need to model the rotation of the wheel, while the linear motion is modeled through its relationship with the chassis
front_left_wheel.m = translate(0.5*chassis_width, 0.5*distance_fb_wheel, -0.5*chassis_height) * RotateX(w_angular * time);
front_left_wheel.sibling = & front_right_wheel;
front_left_wheel.child = NULL;

//define the front right wheel
front_right_wheel.f = wheel; //drawing function for the wheel
front_right_wheel.m = translate(-0.5*chassis_width, 0.5*distance_fb_wheel, -0.5*chassis_height) * RotateX(w_angular * time);
front_right_wheel.sibling = & back_left_wheel;
front_right_wheel.child = NULL;

//define the back left wheel
back_left_wheel.f = wheel; //drawing function for the wheel
back_left_wheel.m = translate(0.5*chassis_width, -0.5*distance_fb_wheel, -0.5*chassis_height) * RotateX(w_angular * time);
back_left_wheel.sibling = & back_right_wheel;
back_left_wheel.child = NULL;

//define the back right wheel
back_right_wheel.f = wheel; //drawing function for the wheel
back_right_wheel.m = translate(-0.5*chassis_width, -0.5*distance_fb_wheel, -0.5*chassis_height) * RotateX(w_angular * time);
back_right_wheel.sibling = NULL;
back_right_wheel.child = NULL;