

[MATH2605] Project 2

Interpolation of Position and Orientation

Introduction

From Chapter 5, we learned that we can compute a rotation matrix from rotation axis and angle. We also learned that we can compute the rotation axis and angle from a rotation matrix. Thus, rotation axis/angle can be converted to rotation matrix and vice versa. This project is to apply these conversion formula to a 3D graphics problem.

Using rotation matrix (or rotation angle and axis) to represent the orientation of a graphical object is essential in computer graphics (You may learn such thing in CS3451). Therefore, we choose it as the specific application of the rotation matrix in this project. Although representation of orientation is not hard, it is not in the scope of this class. Therefore, I do not explain it. Instead, I provided all the source code for the graphical issues such as setting the orientation of object, changing camera position and orientation, etc.

Background

Consider an object moving (translating and rotating) in a 3D space. The center of the object $\mathbf{x}(t)$ is changing as a function of time. The orientation of the object is also a function of time and is represented by a rotation matrix $\mathbf{R}(t)$.

Suppose that $\mathbf{x}(0), \mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(n)$, $\mathbf{R}(0), \mathbf{R}(1), \mathbf{R}(2), \dots$, and $\mathbf{R}(n)$ are given. We would like to first compute the positions and orientations in the middle, i.e., compute $\mathbf{x}(0.5), \mathbf{x}(1.5), \mathbf{x}(2.5), \dots, \mathbf{x}(n-0.5)$, $\mathbf{R}(0.5), \mathbf{R}(1.5), \mathbf{R}(2.5), \dots$, and $\mathbf{R}(n-0.5)$. The easiest way will be the linear interpolation:

$$\begin{aligned}\mathbf{x}(0.5) &= (\mathbf{x}(0) + \mathbf{x}(1))/2 \\ \mathbf{x}(1.5) &= (\mathbf{x}(1) + \mathbf{x}(2))/2 \\ \mathbf{x}(2.5) &= (\mathbf{x}(2) + \mathbf{x}(3))/2 \\ &\dots \\ \mathbf{x}(n-0.5) &= (\mathbf{x}(n-1) + \mathbf{x}(n))/2\end{aligned}\tag{1}$$

Since the rotation matrix cannot be directly interpolated by computing $(\mathbf{R}(0) + \mathbf{R}(1))/2$, we compute the rotation angle $\theta(t)$ and the rotation axis $\mathbf{u}(t)$ of $\mathbf{R}(t)$. From these, we compute the vector $\vec{\phi}(t) = \theta(t)\mathbf{u}(t)$, and then we interpolate $\vec{\phi}(t)$ by

$$\begin{aligned}\vec{\phi}(0.5) &= (\vec{\phi}(0) + \vec{\phi}(1))/2 \\ \vec{\phi}(1.5) &= (\vec{\phi}(1) + \vec{\phi}(2))/2 \\ \vec{\phi}(2.5) &= (\vec{\phi}(2) + \vec{\phi}(3))/2 \\ &\dots\end{aligned}\tag{2}$$

The rotation matrix $\mathbf{R}(t)$ can be computed from $\vec{\phi}(t)$ for $t = 0.5, 1.5, 2.5, \dots$

However, the linear interpolation will produce a path that is not smooth. One method that produces smooth path is

the 4-point interpolation that uses the following rule:

$$\begin{aligned}
\mathbf{x}(0.5) &= (-\mathbf{x}(0) + 9\mathbf{x}(0) + 9\mathbf{x}(1) - \mathbf{x}(2))/16 \\
\mathbf{x}(1.5) &= (-\mathbf{x}(0) + 9\mathbf{x}(1) + 9\mathbf{x}(2) - \mathbf{x}(3))/16 \\
\mathbf{x}(2.5) &= (-\mathbf{x}(1) + 9\mathbf{x}(2) + 9\mathbf{x}(3) - \mathbf{x}(4))/16 \\
\mathbf{x}(3.5) &= (-\mathbf{x}(2) + 9\mathbf{x}(3) + 9\mathbf{x}(4) - \mathbf{x}(5))/16 \\
\mathbf{x}(4.5) &= (-\mathbf{x}(3) + 9\mathbf{x}(4) + 9\mathbf{x}(5) - \mathbf{x}(6))/16 \\
&\dots \\
\mathbf{x}(n-2.5) &= (-\mathbf{x}(n-4) + 9\mathbf{x}(n-3) + 9\mathbf{x}(n-2) - \mathbf{x}(n-1))/16 \\
\mathbf{x}(n-1.5) &= (-\mathbf{x}(n-3) + 9\mathbf{x}(n-2) + 9\mathbf{x}(n-1) - \mathbf{x}(n))/16 \\
\mathbf{x}(n-0.5) &= (-\mathbf{x}(n-2) + 9\mathbf{x}(n-1) + 9\mathbf{x}(n) - \mathbf{x}(n+1))/16
\end{aligned} \tag{3}$$

Of course, $\vec{\phi}(0.5), \vec{\phi}(1.5), \dots, \vec{\phi}(n-0.5)$ can be computed in the same way:

$$\begin{aligned}
\vec{\phi}(0.5) &= (-\vec{\phi}(0) + 9\vec{\phi}(0) + 9\vec{\phi}(1) - \vec{\phi}(2))/16 \\
\vec{\phi}(1.5) &= (-\vec{\phi}(0) + 9\vec{\phi}(1) + 9\vec{\phi}(2) - \vec{\phi}(3))/16 \\
\vec{\phi}(2.5) &= (-\vec{\phi}(1) + 9\vec{\phi}(2) + 9\vec{\phi}(3) - \vec{\phi}(4))/16 \\
\vec{\phi}(3.5) &= (-\vec{\phi}(2) + 9\vec{\phi}(3) + 9\vec{\phi}(4) - \vec{\phi}(5))/16 \\
\vec{\phi}(4.5) &= (-\vec{\phi}(3) + 9\vec{\phi}(4) + 9\vec{\phi}(5) - \vec{\phi}(6))/16 \\
&\dots \\
\vec{\phi}(n-2.5) &= (-\vec{\phi}(n-4) + 9\vec{\phi}(n-3) + 9\vec{\phi}(n-2) - \vec{\phi}(n-1))/16 \\
\vec{\phi}(n-1.5) &= (-\vec{\phi}(n-3) + 9\vec{\phi}(n-2) + 9\vec{\phi}(n-1) - \vec{\phi}(n))/16 \\
\vec{\phi}(n-0.5) &= (-\vec{\phi}(n-2) + 9\vec{\phi}(n-1) + 9\vec{\phi}(n) - \vec{\phi}(n+1))/16
\end{aligned} \tag{4}$$

From these, the rotation matrices $\mathbf{R}(0.1), \mathbf{R}(1.5), \mathbf{R}(2.5), \dots$ can be computed. Once this step is performed, we have positions and orientations at the middle of two consecutive positions and orientations. Therefore, we have about twice more positions $\mathbf{x}(0), \mathbf{x}(0.5), \mathbf{x}(1), \mathbf{x}(1.5), \mathbf{x}(2), \mathbf{x}(2.5), \dots, \mathbf{x}(n-1.5), \mathbf{x}(n-1), \mathbf{x}(n-0.5), \mathbf{x}(n)$, and orientations $\mathbf{R}(0), \mathbf{R}(0.5), \mathbf{R}(1), \mathbf{R}(1.5), \mathbf{R}(2), \mathbf{R}(2.5), \dots, \mathbf{R}(n-1.5), \mathbf{R}(n-1), \mathbf{R}(n-0.5), \mathbf{R}(n)$.

We can repeat this process to compute $\mathbf{x}(0.25), \mathbf{x}(0.75), \mathbf{x}(1.25), \mathbf{x}(1.75), \dots$ and $\mathbf{R}(0.25), \mathbf{R}(0.75), \mathbf{R}(1.25), \mathbf{R}(1.75), \dots$. To reduce a possible confusion, I show that $\mathbf{x}(1.75)$ is computed as

$$\mathbf{x}(1.75) = (-\mathbf{x}(1) + 9\mathbf{x}(1.5) + 9\mathbf{x}(2) - \mathbf{x}(2.5))/16 \tag{5}$$

If we repeatedly apply this process that adds positions and orientations in the middle, we can produce positions and orientations densely sampled over time.

What you have to do

In the *interpolation.pde* file, there are two empty subroutines *phiToR* and *RTophi*.

```
float [] [] phiToR(float phi [])
{
}
```

phiToR computes \mathbf{R} from $\vec{\phi}$, and then returns the computed \mathbf{R} . Complete this subroutine. Note that $\phi = \theta \mathbf{u}$. Use the equation (6.2).

```
float [] RTophi(float R [] [])
{
}
```

RTophi computes $\vec{\phi}$ from \mathbf{R} , and then returns the computed $\vec{\phi}$. Complete this subroutine. You can compute θ from the equation (5.8) and the rotation axis \mathbf{u} from (6.8), and then compute $\vec{\phi} = \theta\mathbf{u}$.

There are four incomplete lines in *subdivide()* function of *Model.pde* file. Complete them. This should be trivial as far as you know the 4-point rule.

Finally, create another copy of the whole project (You may name it *Interpolation2*), and then modify *subdivide()* to directly interpolate \mathbf{R} by

$$\begin{aligned}
 \mathbf{R}(0.5) &= (-\mathbf{R}(0) + 9\mathbf{R}(0) + 9\mathbf{R}(1) - \mathbf{R}(2))/16 \\
 \mathbf{R}(1.5) &= (-\mathbf{R}(0) + 9\mathbf{R}(1) + 9\mathbf{R}(2) - \mathbf{R}(3))/16 \\
 \mathbf{R}(2.5) &= (-\mathbf{R}(1) + 9\mathbf{R}(2) + 9\mathbf{R}(3) - \mathbf{R}(4))/16 \\
 &\dots \\
 \mathbf{R}(n-1.5) &= (-\mathbf{R}(n-3) + 9\mathbf{R}(n-2) + 9\mathbf{R}(n-1) - \mathbf{R}(n))/16 \\
 \mathbf{R}(n-0.5) &= (-\mathbf{R}(n-2) + 9\mathbf{R}(n-1) + 9\mathbf{R}(n) - \mathbf{R}(n))/16
 \end{aligned} \tag{6}$$

Add some discussions on what went wrong in the web site.

Extra credits

The followings are some ideas.

- Show the four initial locations and orientations in a different color. Show the animation of the object. For the animation, you may call *subdivide* a few more times from *setup* function in *Interpolation.pde*.
- In the *InitializePositionAndOrientation* subroutine in the *Model.pde* file, the initial locations of the object is defined. Modify them and observe how objects change positions. Check if there is any strange movement.
- Implement the linear interpolation (1) and (2). Compare with the 4-pt rule. You may capture the images of the two methods, and upload them in the web site.
- Any other improvements.

Submission

Create a web site to post your applets. Add some description if you have done some extra work. Send the link to the web site and the zipped source code via email. The title of the email should be *[Math2605, Project2] YOUR FULL NAME*.

The project (except for the extra credit) is due on Apr 9th. The extra credit part can be submitted to me via email no later than Apr. 28th. Please have the email titled *[Math2605, Project2, Extra credit #1] YOUR FULL NAME*, *[Math2605, Project2, Extra credit #2] YOUR FULL NAME*, etc. In the email or in the web site, please indicate what you have done. Alternatively, you may visit me during my office hour and show me what you have done.