Computer Animation2: Motion Capture Interpolation

Name:

Coding Environment: macOS Catalina

<Describe what I have implemented>

1. Implemented three interpolation schemes successfully. (interpolator.cpp)  
   Bezier Euler, Linear Quaternion, Bezier Quaternion
2. Report of comparing four interpolation methods (which shows below)
3. Plot four graphs (Data and graphs are stored in Plot folder)
4. Three videos (Using iMovie to convert ppm to mp4)

<Extra>

All screenshots are saved in Extra folder.

1. Analyzing the computation time of the different interpolation techniques.
2. Supporting keyframes that are non-uniform in time. (set N = 0 during interpolation)
3. Adding background music to three videos.

Detail will be shown in this report.

Basic implementation and analysis.

1. Plot

A close up of a map

Description automatically generated

1. A close up of a map

   Description automatically generated

3.

A close up of a map

Description automatically generated

A picture containing sitting, black, white

Description automatically generated4.

According to four graphs, I found that Bezier Quaternion has the best performance.

The curve for linear Euler is the sharpest one among four interpolation methods. It takes same ‘v’ for all time in one interpolation. When applying it to martial art and dance, the avatar shows unstable movement when it rotates or turns around to a different direction.

The curve for Linear Quaternion works a little bit of smoother than Linear Euler. It is closer to the input curve, but it still has sharp angles which can be seen in graph2. During testing dance animation, it works better linear Euler, the avatar will not shake that often when it jumps and rotates.

The curve for Bezier Euler is much smoother than linear Euler. It can avoid sharp angles. Applying this method to the dance avatar, it will shake twice when it jumps and does rotation, but it looks better than two linear methods.

The curve for Bezier Quaternion has the best performance according to four graphs. It will not shake or drop down immediately. It follows the input curve best.

A screenshot of a cell phone

Description automatically generatedHere are the screenshots for analyzing computation time.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedTime consuming: Bezier Quaternion > Bezier Euler > Linear Quaternion > Linear Euler

According to all analysis, I totally understand why Unity has to use quaternion to do calculation. When I first used quaternion in Unity, I think it is annoying to do the translation. I thought it was the problem of C# that it did not provide easier computational way. Quaternion works much better than xyz rotation, and it can also avoid gimbal lock problem. Bezier Euler solve the awkward sharp movement comparing with linear interpolation. Bezier method need more computation time because it has to use four control points. Quaternion method needs more computation time because it has to do conversion from Euler to Quaternion and back.

I faced and solve two interesting bugs during implementing Linear Quaternion method, and I has deeper understanding of quaternion.

The first one is I did not read the instruction carefully. I do not know the starter code already has conversion for Quaternion to Euler and back. I wrote my own by reading Shoemake’s paper, and I did not realize it needs to contain so much constrains during conversion, until I cannot run it successfully.

The second one is I forget to remind myself Quaternion q and -q has the same rotation matrix, which means I need to find the nearest one to get my control point right. I also need to check if the start point and end point are at the same position, otherwise I will get ‘nan’ value by using SLERP interpolation.

After finding these difficulties, I finally finished this assignment. I feel excited to figure it out by reading notes and paper.

Besides computation time analysis, I also implemented interpolation by not using uniform keyframe. It can be test by setting N = 0. It will generate a random number and set this number as the interpolation number. I set this random number range from 5 to 19. I applied this function to two methods Linear Euler and Linear Quaternion. The reason for I only apply this function to these two methods is that Bezier method sets 1/3 as the coefficient during calculating control points. It does not make sense to use the same coefficient with different interpolation number. It works pretty well in two methods.

Here is the input and (partial) output sample.

A close up of text on a black background

Description automatically generated

At last, I added background music in three videos. I hope you can have fun while playing them around.