Computeranimation

186.139 - SS 2020

Assignment 2: Interpolation

The goal of this assignment is to understand the basics of spline interpolation.

Configurations:

Windows:

- Install Visual Studio 2019 or 2017 (the Community Edition is free)
  - https://visualstudio.microsoft.com/de/vs/community/
- Download the Windows SDK 10.0.18362.0 (this should be the most current one)
  - https://developer.microsoft.com/de-de/windows/downloads/windows-10-sdk/
- The prebuilt libraries can be downloaded from TUWEL
  - Download the CommonFiles package from TUWEL for your version of Visual Studio
  - This folder contains an include and lib folder
  - Copy these two folders into each exercise package
- Run the project – Debug – x64
  - 141 for VS 2017, 142 for VS 2019, SDK 10.0.18362.0 (this should be default)

If this does not work you may need to build the libraries for yourself. Contact me via discord! I can share some useful advice ;)

OS X (no Discord tutor support):

Get the libcinder Assignment package (Planetarium) from Tuwel

Get libcinder: https://libcinder.org/download

Linux (no Discord tutor support):

You will have to compile cinder for yourself. Follow one of the guides from here: https://github.com/cinder/Cinder/wiki/Cinder-for-Linux

Get the libcinder Assignment package (Planetarium) from Tuwel

Assignment:

Each of the tasks will reward you with a certain amount of points. To pass each exercise, you will need at least 50% of the total points of each exercise. You may use the provided Assignment package, which contains some code to help you learning cinder, or start your own entirely. If you are unsure how to do something with libcinder, you can check out the samples provided by cinder.

Hand in:

- A short video of your program (maximum of 60 seconds), showing each task.
- The source code.

Exercise 2: Interpolation

This exercise is about Interpolating values between control points and then moving an object along the coordinates. There are comments in the code in the form of "//todo TASK1" where you can add your code.

There is some basic functionality already implemented. Use your right mousebutton to rotate the camera, mousewheel zooms and middle mouse button moves the camera.

You can add control points and switch between modes through the interface (notice that according to the selected mode there are different manipulators available for the control points). Each function in Tasks 1 to 4 should return a std::vector<vec3> of interpolated positions (see the code).

Chapter 3 in the lecture book explains the basics. For the relevant math have a look at appendix B!

Task 1 – Linear Interpolation (2 points)

Files: splines.cpp & splines.h Function name: GetInterpolatedLine(float interval)

Linearly interpolate between control points.

- **Chapter 3 in the lecture book explains the basics. For the relevant math have a look at appendix B**.
- There are a few mistakes in the provided figures, here are some corrections:
  - B.72: The second line should be $3u^2$ instead of $3u^3$.
  - B.74: The vector B repeats one element, simply ignore the duplicate.

Tasks 2, 3 and 4 are very similar.

- **Complete any of Tasks 2, 3 or 4 to get 4 Points,**
- **2 of them for 6 Points**
- **or all 3 for 8 Points.**

Task 2 – Hermite Interpolation

Files: splines.cpp & splines.h Function name: GetParabolaInterpSpline(float interval)

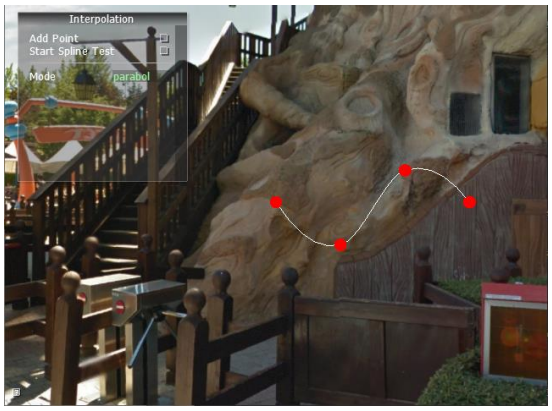
Implement Hermite splines.

- A common oversight: You need to add the position of the control point to the (relative) position of the tangent (the book assumes absolute positions of the tangent vectors).

Task 3 – Blended Parabolas

Files: splines.cpp & splines.h Function name: GetParabolaInterpSpline(float interval)

Implement Blended Parabolas. You do not need to include the end condition handling. (The start and end point do not need to be connected)



Task 4 – Bezier Splines

Files: splines.cpp & splines.h Function name: GetBezierInterpSpline(float interval)

Implement Cubic Bezier spline interpolation

Task 5 – Interpolate along the Splines
(1 point + 1 point if the speed is constant – show this in your video)

Files: InterpolationApp.cpp Function names: runSplineTest(), splineTestUpdate()

Implement a Cube (or any shape) to move along the previously generated splines. The distance travelled should be at a constant speed. (Linearly interpolate between each of the positions returned by the previous tasks)

It does not need to be numerically accurate!