Intrinsic Curve properties visualizer

Daniel Aaron Salwerowicz, Christopher Kragebøl Hagerup

October 10, 2018

Introduction

In this project we aimed to visualize intrinsic properties of B-Spline curves (curve from now on) in 3D-space. These intrinsic curve properties are *curvature* and *torsion*. Curvature defines how fast a curve changes direction at any given point, whereas torsion defines how much the curve is "twisting" or "coming out of" the plane of curvature. As such we see that a curve cannot have torsion at a point if there is no curvature there. [1]

Methods

We have used GMlib to represent a curve in 3D space as well as representing its curvature using ovals (circles) and torsion using colours.

To represent curvature we transform the curvature measured with vector into a circle with radius equal length of the vector. Curvature at a given point is calculated using:

$$\kappa(t) = \frac{\left\| c^{'}(t) \times c^{''}(t) \right\|}{\left\| c^{'''}(t) \right\|^3}$$

As such these circles easily and intuitively convey the curvature of curve at any given point and how it changes over its length.

Torsion is represented with colours in HSV colour model, where hue is decided by the sign of torsion and saturation is decided by relative strength of torsion. Torsion itself is calculated by:

$$\tau(t) = \frac{\left(c^{'}(t) \times c^{''}(t) \right) \cdot c^{'''}(t)}{\left\| c^{'}(t) \times c^{''}(t) \right\|^2}$$

As such torsion is conveyed using colours and intesity to show its magnitude.

Here we will explain what we do to visualize these properties. First we create a `MyBSplineCurve` object that then creates a `MyVisualizer` instance and it in turn creates `CurveParam` objects that hold values relevant for visualization (see figure 1). These structs hold one `PCircle` that they are responsible for visualizing and updating. Each time the curve is changed it calls on updateParams method which in turn moves and updates circles.

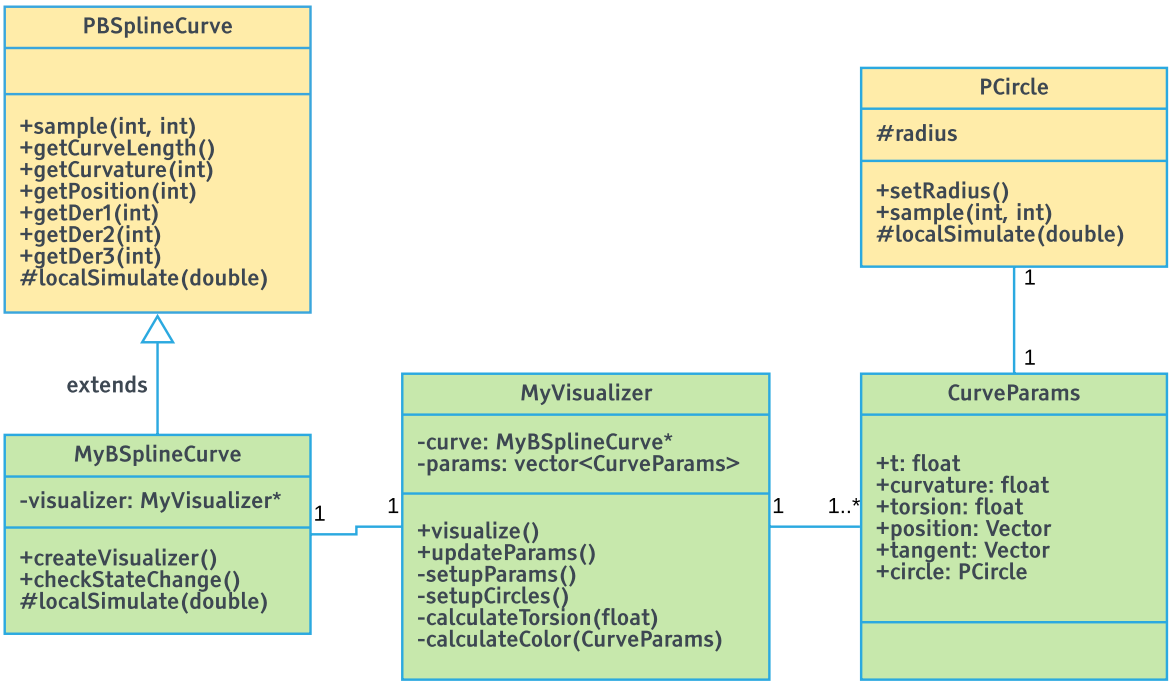

Figure 1: UML class diagram our program.

Results

We have ended up with a really adequate visualization program for curvature and torsion where user can easily see both of these properties. Visualization is really easy to inspect, transform and see how curvature and torsion change after manipulation. Figure 2 shows an example of curve simulated and visualized in our program. User can easily twist, bend and rotate the curve to inspect how distorions of it change its properties.

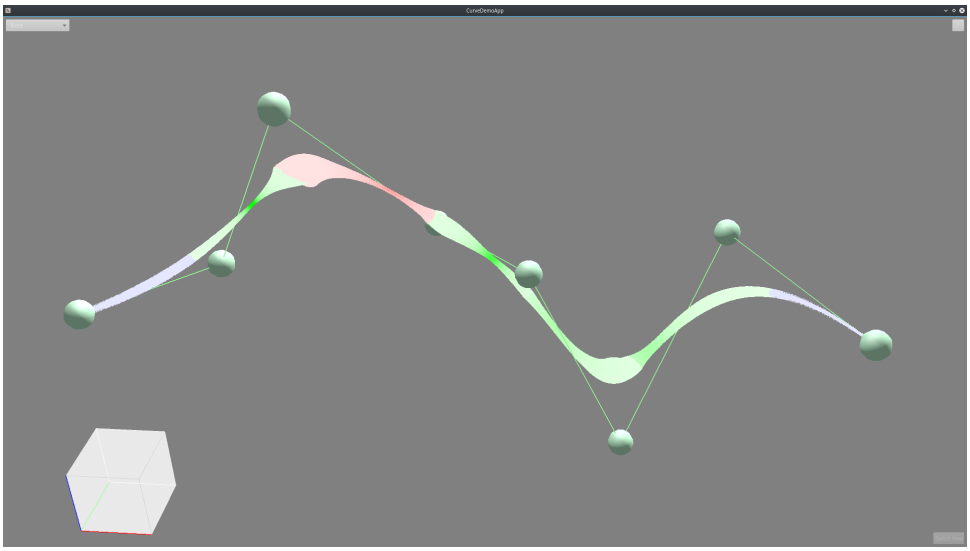

Figure 2: Visualization of curvature and torsion in our program.

Analysis

Based on our calculations and observations we conclude that this visualization program is quite accurate and represents the curve properties quite well. We have found out that in some instances it could've been beneficial to normalize the curvature values as to limit how big the circles can grow, but in most usecases this proved to be unnecessary, and it could easily be avoided by setting limitations on curve editing capabilities.

We have tried many different ways of displaying torsion by, but after discussion with Børre Bang we came to conclusion that this is the most readable option for displaying torsion. So hue is set by torsions sign, and saturation is set by relative value of torsion. Saturation is given by:

$$saturation = \frac{|\tau|}{|\tau_{max}|}$$

which will normalize the torsion values so that both negative and positive torsion values are set based on maximum torsion.

Discussion

We are quite happy with our visualizer, it is quite simple and easy to understand as well as it is lightweight and does not require use of shaders. It runs quite fast and lets user easily understand what he sees on the screen. It is also dynamic so each small edit made to curve is immediatly displayed on screen.

There are of course possibilities for further development and improvements in our visualizer. One of these would be to implement restrictions on curve editing, so we don't end up with enourmous circles that result from extremely high curvature. Another feature worth implementing would be possibility to visualize higher order derivatives of curve, that could be easily done by adding lines at various points in curve with length equal to magnitude of these derivatives.

Bibliography

[1] Jia, Y.-B. (2017). Arbitrary-speed curves. *Com S 477/577 Notes*.