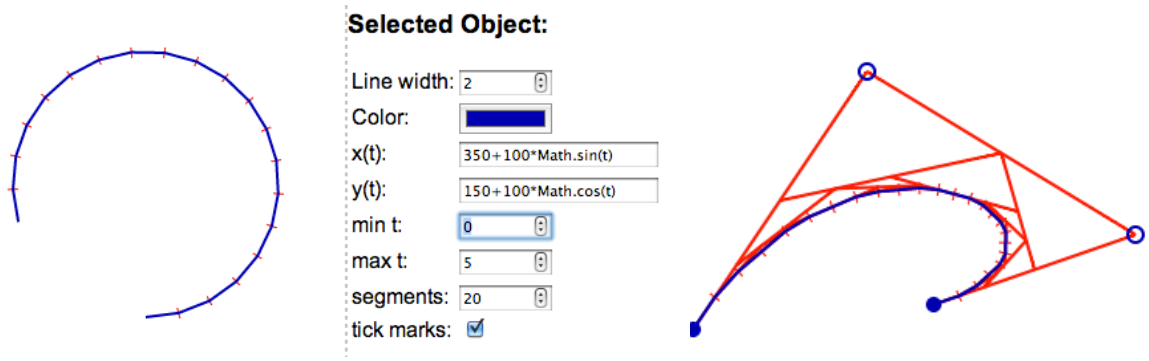
Computergrafik 2 / Aufgabe 1.2

Parametrische Kurven und Bézierkurven



Links: Beliebige parametrische Kurve mit "tick marks".
Rechts: Mittels Unterteilung dargestellte Bézier-Kurve und zugehörige Kontrollpolygone.

Lernziele / Motivation

Bei diesen Aufgaben lernen üben Sie die Darstellung allgemeiner parametrischer Kurven und kubischer Bézier-Kurven. Zusätzlich implementieren Sie ein adaptives Verfahren für Bézier-Kurven, welches die Kurve dort weiter unterteilt, wo der Darstellungsfehler eine bestimmte Grenze überschreitet.

Vorbereitung

Zur Lösung dieser Aufgabe erweitern Sie den Code aus Aufgabe 1.1.

Aufgabenstellung

1. Implementieren Sie ein Objekt `ParametricCurve`, welches in der Lage ist, beliebige parametrische Kurven im Canvas darzustellen. Der Benutzer soll die Formel, mittels derer jeweils die x- und y-Koordinate für ein beliebiges `t` berechnet wird, als Text eingeben können (Siehe Screenshot oben auf dieser Seite). Neben der Berechnungsformel soll der Benutzer zwei Parameter `t_min` und `t_max` eingeben, die den Definitionsbereich der Kurve angeben. Die Anzahl der Liniensegmente, mit denen die Kurve angenähert wird, soll ebenfalls vom Benutzer festlegbar sein.
    a. Zur Umsetzung der Formel-Berechnung dürfen Sie die gefährliche JavaScript-Funktion `eval()` in dieser Aufgabe explizit verwenden. *Fangen Sie jedoch Syntax-Fehler des Benutzers so ab, dass das Programm weiterhin funktioniert und der Benutzer den Fehler korrigieren kann.*
    b. Ihr `ParametricCurve`-Objekt benötigt wie schon der `Circle` die Methoden `draw()` und `isHit()`. Die Methode `createDraggers()` sollte eine leere Liste zurückliefern.
    c. Siehe Implementierungshinweise weiter unten auf diesem Blatt.

2. Implementieren Sie ein Objekt `BezierCurve`, welches basierend auf vier gegebenen Kontrollpunkten eine kubische Bézier-Kurve darstellt. `BezierCurve` soll auf Komponenten von `ParametricCurve` aufbauen. Refaktorieren Sie `ParametricCurve` ggf. so, dass Sie auf vernünftige Weise Code wiederverwenden können.
    a. Verwenden Sie `PointDragger` zur Manipulation der Kontrollpunkte.
    b. Wenn das Objekt im Canvas selektiert ist, soll neben den vier Kontrollpunkten auch das Kontrollpolygon der Kurve visualisiert werden. Implementieren Sie dazu einen neuen Dragger-Typen, der lediglich das Polygon zeichnet und selbst keine Interaktivität implementiert.

Optionale Zusatzaufgaben

3. Visualisieren Sie die Punkte, an denen die Kurve ausgewertet wurde, durch kleine Striche ("tick marks"), die senkrecht zur Kurve verlaufen. Machen Sie die Visualisierung dieser Striche im UI ein- und ausschaltbar.
4. Implementieren Sie die Darstellung der Bézierkurve zusätzlich mittels adaptiver Unterteilung durch den de-Casteljau-Algorithmus.
    a. Eine maximale Unterteilungstiefe soll vorgebbar sein.
    b. Ein maximaler Fehler (Abstand, Krümmung) soll vorgebbar sein.
    c. Die Elemente der bei der Unterteilung konstruierten Kontrollpolygone sollen optional sichtbar zu machen sein.

Implementierungshinweise

Zusätzlich zu den Hinweisen aus Aufgabe 1.1 hier noch folgende Tipps:

- Checkboxen werden in HTML mittels `<input type="checkbox">` erzeugt. Sie enthalten ein Attribut namens `checked`, welches entweder undefiniert oder auf den Wert "checked" gesetzt ist.
- In 2D ist eine Normale zu einem Richtungsvektor `[x,y]` durch `[-y,x]` gegeben.

Abgabe

Die Abgabe der gesamten Aufgabe 1 (1.1 + 1.2) soll via Moodle bis zu dem dort angegebenen Termin erfolgen. Verspätete Abgaben werden wie in den Handouts beschrieben mit einem Abschlag von 2/3-Note je angefangener Woche Verspätung belegt. Geben Sie bitte pro Gruppe jeweils nur eine einzige `.zip`-Datei mit den Quellen Ihrer Lösung ab.

Demonstrieren und erläutern Sie dem Übungsleiter Ihre Lösung *in der nächsten Übung nach dem Abgabetag*. Die Qualität Ihrer Demonstration ist, neben dem abgegebenen Code, ausschlaggebend für die Bewertung! Es wird erwartet, dass alle Mitglieder einer Gruppe anwesend sind und Fragen beantworten können.