INTERACTIVE GEOMETRY LAB

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Graphics Laboratory ETH Zurich

# Computer Graphics
# Exercise 5 - Curves and Surfaces

Handout date: 14.11.2014

Submission deadline: 05.12.2014, 14:00h

The goal of this homework is to familiarize yourself with B-Spline curves and subdivision surfaces. This homework consists of two parts: first, you will implement a simple cubic B-Spline curve editor, second you will implement 2 subdivision schemes for polygonal meshes.

You must work on this exercise **individually** as usual. You may submit your solution by:

- Emailing a .zip file containing your solution to `introcg@inf.ethz.ch` with both the *subject* and the *filename* as `cg-ex5-firstname_familyname`

The .zip file should contain:

- A folder named `source` containing your source code.
- A `README` file clearly describing what you have implemented for each module.

No points will be awarded unless your source code can:

- Run on the Firefox web browser on student lab PCs at CAB H 56 or 57 running Windows or Linux.

This exercise will be graded **during the exercise session on 05.12.2014**. You are required to attend, and those absent will receive 0 points. Grading will be conducted on the lab machines at **CAB H 56**. The students whose last name starts with A–K should come to the lab by 14:15, whereas those whose last name starts with L–Z should do by 15:00. This will alternate for next exercises. You are also required to email your solution before the submission deadline written above. Late submissions will not be considered, and will be given 0 points even if they have been graded during the grading session.

## Codebase

The codebase is written in JavaScript. The code should run on Firefox or Chrome web browser on all major platforms (Windows/Mac/Linux), but we only support and have tested the code with Firefox on Windows and Linux. Note that your solution should run out of the box on one of the two environments. You are encouraged to use the lab machines at CAB H 56 and H 57 to void any platform-specific complications.

The code template can be downloaded from the course web site. It implements the GUI and already gives you some basic functionality.

## B-Splines (10 points)

You are required to implement a cubic B-spline editor using the code base provided. Your editor should support the functionality described in the next sections. A screenshot of an example solution is shown in figure 1.

### User interface

The GUI consists of two windows. The top window shows your B-Spline curve editor. The basic implementation that you have to extend allows the user to add nodes or control points by clicking in the window. This will automatically also add knots in the knot vector that are displayed in the window below. The default knot sequence consists of integers starting at 0. The number of knots is always the number of control points plus four (however first and last knot values don't influence the curve). The number displayed near the knots is their index and *not* their value. The values of both the knots and control points can be modified using a mouse select/drag interface. Control points and knots cannot be deleted. Additionally, at the bottom of the knot window there is an additional active knot. The value of this knot will be used to illustrate the de Boor algorithm as illustrated in the image.

### Functionality

Given the UI you have to enhance the functionality of this codebase. In the editor window (top) you need to render the B-Spline curve corresponding to the control points and knot sequence specified by the user. The application has to be interactive, meaning that changes in the control points, knot vector or active knot should be reflected visually automatically.

In addition to the B-Spline, you have to render in this window, toggled by check boxes between the two windows, the construction lines of the de Boor algorithm corresponding to the active knot value as shown in the screenshot.

### Under the Hood

The code provided is structured over several files. The relevant files for this homework are the following:

1. `node.js`: Encapsulates the functionality of a control point.
2. `knot.js`: Encapsulates the functionality of a knot. Note the boolean `_isTimeKnot` marks the special active knot.
3. `curve.js`: Perhaps the most important of all, it encapsulates the B-Spline curve and its visualization. All rendering has to be done inside its draw member function. The class has references to all the necessary information: control points, knot vector, active knot, etc. The toggle variables for the various drawing modes are also available here. Inside the draw member function you have hints how to render points and lines and how to change the color using helper functions.
4. `bSplineTools.js`: An empty file that you can use to separate your B-Spline specific implementation from the drawing routine.

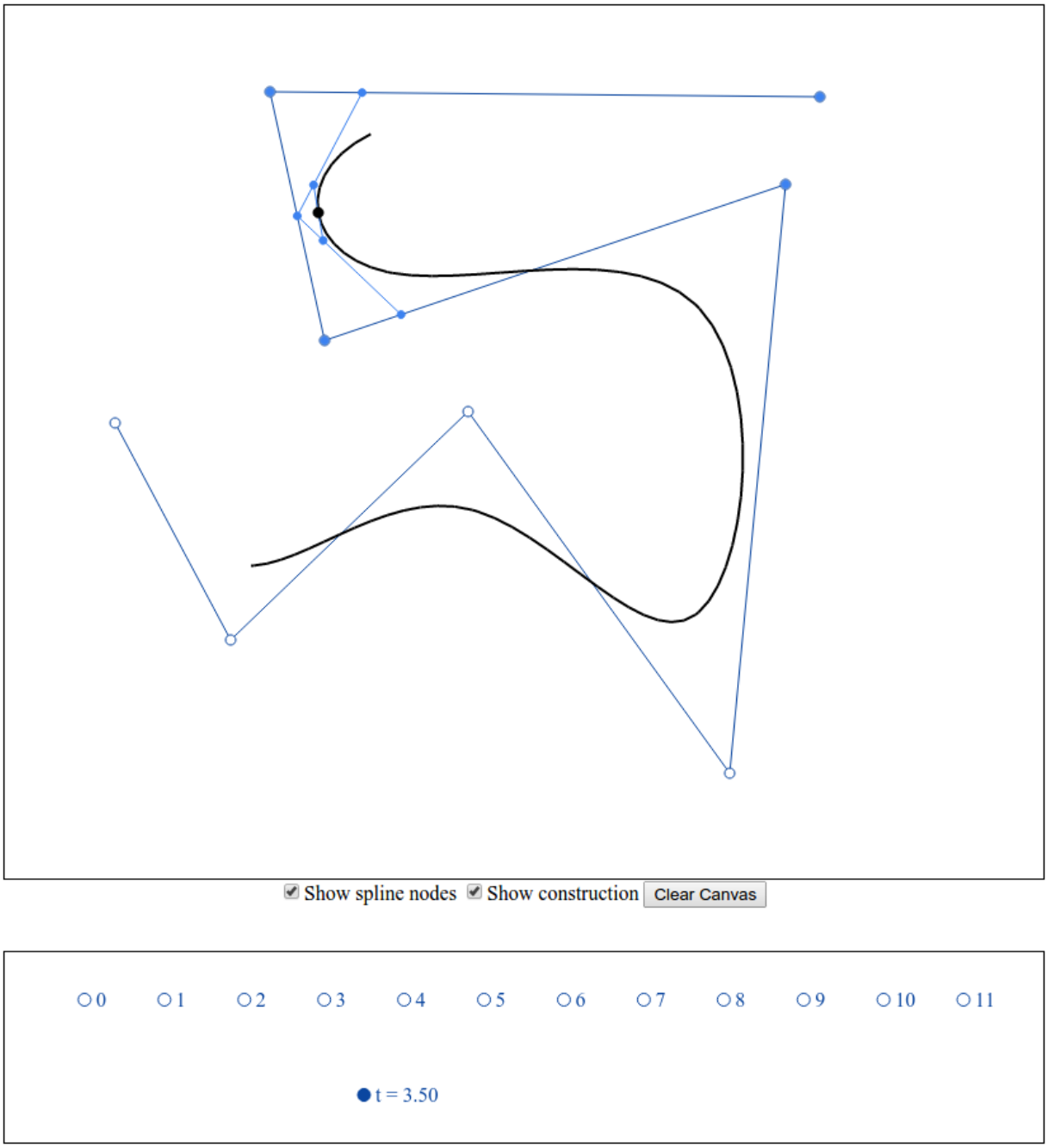


Figure 1: Screenshot of a sample solution to the B-spline curve.

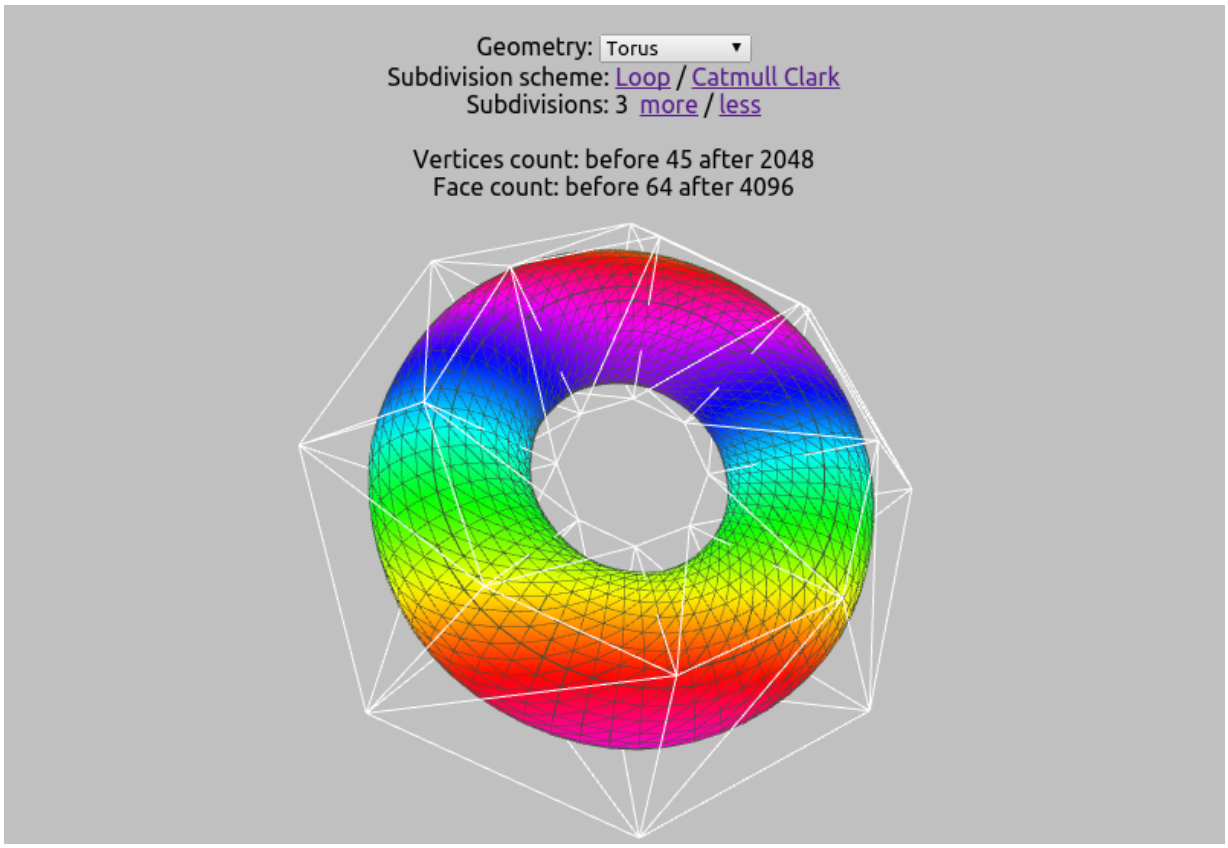


Figure 2: Screenshot of a sample solution to the subdivion surface.

## Subdivision Surfaces (10 + 10 points)

You are required to implement Loop Subdivision and Catmull-Clark schemes for a given mesh. A screenshot of an example solution is shown figure 2.

### User interface

When you open the codebase, you can see a mesh. Once you have solved the exercise a subdivided version of the wireframe mesh should also appear with colors. The whole GUI has already been written for you. It includes the mesh visualization with camera controls, as well as a simple interface to select a different mesh, the algorithm, and the number of subdivision iterations. It also give you simple statistics about the original and subdivided meshes.

### Under the Hood

The code provided is structured over several files. The relevant files for this homework are the following:

1. `subdivision.html`: Main page
2. `three.js`: WebGL library
3. `OrbitControls.js`: Camera controls

4. `OBJLoader.js` and `bigguy_0.obj`: OBJ file loader and an additional mesh

You shouldn't modify these files unless you know what you are doing.

There are two other files, `LoopSubdivision.js` and `CatmullClark.js`, which contain code templates. You need to complete the `smooth` prototype to implement the respective subdivision schemes. Useful functions and codes are already present, so you only have to create the new vertices and faces. Please refer to the code for more details. The parts you need to complete are marked with `TODO` within comments.