CMPSCI 373 (S20) Homework 3: Curves

Overview

In this homework you will implement algorithms to compute Bezier, B-Spline, and Hermite curves. The starter code can then generate and render surfaces of revolution using the curves you compute.

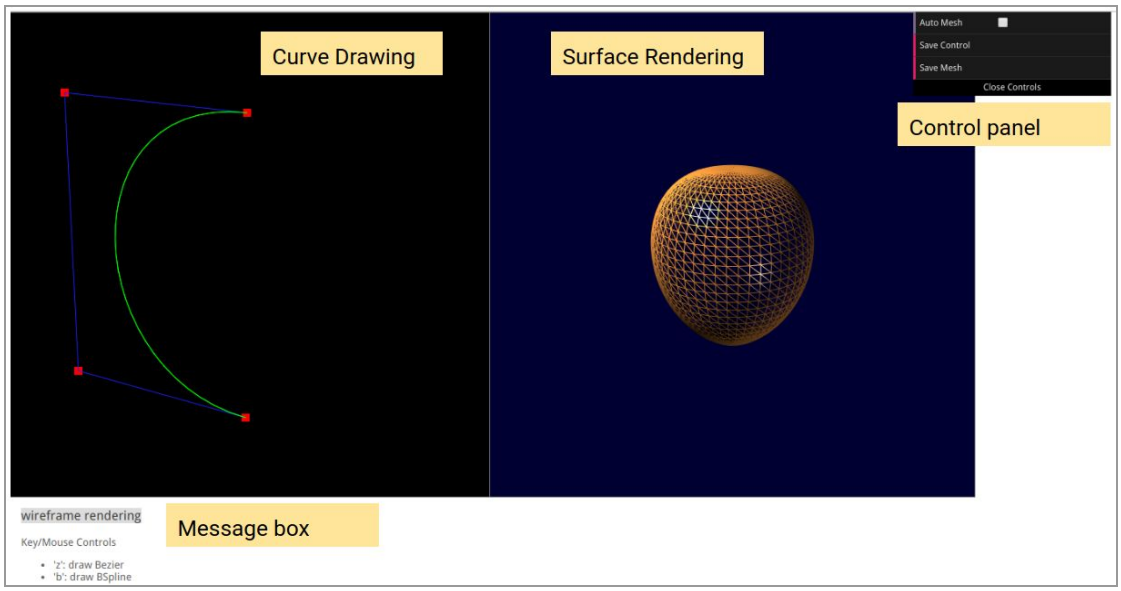
Starter Code

- **Download** the starter code from Piazza, and **unzip** it. The starter code consists of the following:
  - `index.html` defines the main HTML elements, including two canvases (one for curve drawing, one for surface rendering) and it includes other .js files.
  - `three.min.js/dat.gui.min.js/FileSaver.js/OBJExporter.js` are utility libraries for WebGL rendering, GUI creation, file saving, and exporting surface mesh to .obj files.
  - `curves.js` contains the main code of this homework.
  - `preset.js` contains predefined sets of control points for testing and debugging.

- Open `index.html` in a browser, you should see two canvases: the one on the left is for **curve drawing**, and on the right is for **surface rendering**. There is a floating **control panel** on the top-right corner, and a **message box** under the curve drawing window, which shows the most recent action. The main key/mouse controls are documented at the bottom of the webpage.
  - The page automatically loads the 1st predefined control point set, which has 4 points. Each point indicated by a **red** square.
  - Use your mouse to drag and move any control point.
  - **`CTRL+Click`** (or ALT+Click, or COMMAND+Click) to append a new control point.
  - **`SHIFT+Click`** to delete an existing control point.
  - Press **`e`** to erase all control points.
  - Press number keys (**`1 to 6`**) to load a predefined control point set.
  - Press **`z/b/m`** to switch between Bezier, BSpline, Hermite curve (the resulting curve will show up in **green** **once you have completed** the corresponding code).
  - Press **`=/-`** to increase or decrease the number of segments (**`nSegment`**).
  - Press **`l/c`** to toggle show/hide control lines, and close/open curve (applicable to B-Spline and Hermite only)
  - Press **`s`** to generate surface of revolution. At the surface rendering window, you can use your mouse left button to rotate the mesh (vertically) and right button to zoom in/out.
  - Press **`a/left_arrow/right_arrow`** to toggle show/hide rotation axis, and move the axis.
  - Press **`w/f`** to toggle wireframe vs. solid rendering, and flat vs. smooth shading.
  - On the floating control panel, check **`Auto Mesh`** to keep the mesh automatically updated whenever you make a change to the curve. Click **`Save Control`** to save a copy of the control points to a .js file; and **`Save Mesh`** to save the current mesh to a .obj file.

The next page shows a snapshot of the homework (again, the green curve will only show up after you have completed the code).

CMPSCI 373 (S20) Homework 3: Curves



Next, open `curves.js` in a code editor, and take a careful look at the entire code.

- This is **the only file** you need to modify. It has provided necessary code for drawing and modifying control points, handling mouse/keyboard controls, loading predefined control point set, and generating and rendering surfaces of revolution.
- **Your task** is to complete the three functions that compute **Beizer**, **BSpline** and **Hermite** curves. These functions are all marked by **// ===YOUR CODE STARTS HERE===** comment. You are allowed to create additional functions if you need, but you **should NOT** include any other Javascript library; also **Do NOT** change the name of any existing function in the starter code.
- For each of the three functions:
  - The input control points are all stored in **`controlPts`** array, which is a global variable. Each element in the array has **`x`** and **`y`** coordinates of a point, and **`dx`**, **`dy`** which define the tangent vector of the point (only useful in Hermite). For example, for the i-th point you can access them by **`controPts[i].x, controPts[i].y, controPts[i].dx, controPts[i].dy`**. This array is **read-only** and you should **NEVER make changes to this array** in any way.
  - You will compute and append curve points to the **`curvePts`** array. The starter code then calls **`drawCurve()`** at the end of each function to plot the curve this array. In Javascript, you can define an object, such as a 2D point, by using the JSON notation as:
    `{'x': 5.0, 'y': 6.0}`
    This creates an object containing elements x, y, with values 5.0 and 6.0 respectively. You can append such a point to **`curvePts`** array by **`curvePts.push({'x':5.0, 'y': 6.0});`**
- Specific requirements for each curve:
  - The **`nSegment`** global variable defines how many line segments you should use to approximate the curve. In other words, sample the curve at **`(nSegment+1)`** points (which result in **`nSegment`** line segments when plotted). For **Bezier** curve, you should use **all control points** and the recursive formulation we covered in class to compute a single curve, sampled at **`(nSegment+1)`** curve points.

- For **BSpline** curve, compute a piecewise curve using **every four adjacent control points**. Normally, the curve is assumed to be open, so if there are `N` control points, you code should generate (`N-3`) pieces of curves, each sampled at `(nSegment+1)` points. However, if the `closedCurve` variable is `true`, you should close the curve (by looping back to the beginning of the control point array, as if it's a circular queue), and hence the close curve will contain `N` pieces, each again sampled at `(nSegment+1)` curve points. Refer to lecture slides in-class demo for details.
- For **Hermite** curve, compute a piecewise curve using **every two adjacent control points**. Similar to BSpline, each piece should be sampled at `(nSegment+1)` curve points; and the curve can be open or closed depending whether `closedCurve` is `true` or `false`. Additionally, the formulation of Hermite curve requires a tangent vector at each point, defined by the `dx`, `dy` elements in the `controlPts` array.

  On the curve drawing window, when you switch to **Hermite** curve (by pressing '**m**' key), you will notice that each control point has an associated circular point (called **target point** in the code). The subtraction of target point from its associated control point defines the tangent vector. You can move the target point in order to change the tangent vector. The handling of target points and computation of tangent vectors are already implemented for you. One of the predefined control point sets (the cup example) does not come with tangent vectors; the `computeHermite` function checks the existence of tangent vectors and only proceeds if they are provided.

Surface of Revolution

The starter code has implemented surface of revolution, to demonstrate how the curves you generated can be used to compute a smooth surface. At any time, press the '**s**' key or **space** bar to produce a surface from your current curve. Press '**w**' and '**f**' keys to toggle wireframe and flat shading modes. Press '**a**' to show/hide the rotation axis in the curve drawing window (it shows up as a red vertical line), and **left/right** arrow keys to move the axis. You can also leave the **Auto Mesh** checkbox on to allow continuous update of the surface (but keep in mind this may considerably slow down the interaction speed). When you are satisfied with the surface, you can click **Save Mesh** on the control panel to export the surface to a .obj file. The .obj file can then be imported to many 3D viewers (such as MeshLab) for further editing, or sent to 3D printer to produce a physical copy of the object.

Loading and Saving Predefined Control Point Set

In the starter code, we have provided 6 sets of predefined control points. Press the number keys (1 to 6) to switch between them. These serve as test datasets for you to quickly check and debug your code. Feel free to generate additional control points of your own design. When you are satisfied with the result, you can click **Save Control** on the control panel to download them as a file, open the file, and insert the data into the `preset.js` file, so that it becomes another predefined control set. This allows you to preserve the control points you have created.

CMPSCI 373 (S20) Homework 3: Curves

Grading (28 points)

- [7 pts] Correctly implement **computeBezier** function.
- [9 pts] Correctly implement **computeBSpline** function, including both open and closed curves.
- [8 pts] Correctly implement **computeHermite** function, including both open and closed curves.
- [4 pts] Creative objects: use your implementation to generate **two** novel surfaces of revolution of your own design, save them as **.obj** files, and attach them in the submission file. These must be of **your own design** and **CANNOT** be using one of the provided control points. Many objects we see in the world are rotationally symmetric, such as cup, bowls, lamps etc. Use your creativity and imagination. Each creative object you make is worth 2 points.
- (Optional) if you would like to 3D print your creative objects, contact the instructor directly.

Submission

Zip your homework 3 folder (including all files, and creative objects) and submit in Gradescope.

Sample Results

Sample results using the first predefined control point set are given below.