CS 348C | **Fall 1995-6**
Assignment 3 | **Apostolos Lerios**

Due 12/7/95, 2:30p.

Notes

1. When you read this line, and provided you are taking the class for 3 units, please send mail to `tolis@cs.stanford.edu` and let me know that you have picked up the homework.
2. World Wide Web (WWW) documents referenced in this assignment can all be found under the course hierarchy

   `http://graphics.stanford.edu/courses/cs348c-95-fall/`

3. Please read the WWW page `requirements.html` for policies and regulations regarding your homework.
4. All uses of the word "patch" in this assignment refer unambiguously to a tensor-product patch, the reason being that `SoftImage` supports only such patches.
5. The `SoftImage 3D` **Undo** command is under the **History** menu cell. Also, the **Show→Point** and **Show→Line** commands may be helpful, as they cause `SoftImage 3D` to render the control points and polygons (or nets) of splines.

Preparation: `SoftImage` setup.

This assignment requires you to do some elementary modeling using `SoftImage 3D`, the `SoftImage` integrated system for 3D modeling, animation, and rendering. Before you start working on the assignment, set up your Sweet Hall account for use with `SoftImage`. See the WWW page

`software/softimage.html`

for a brief overview of `SoftImage`, and for detailed information on setting up your accounts.

If you have an account on the machines of the CIS computer graphics laboratory, you may use that account, instead.

Problem 1: `SoftImage 3D` Tutorial. *(50 points)*

In this problem, you will complete lessons 1, 3, 6 and 7 of the `SoftImage 3D` tutorial. While you are only required to do the indicated lessons, you are encouraged to complete the tutorial, doing all lessons in sequence. Doing so will not take too much additional time, and it will

- provide you with a good overview of `SoftImage`, which you might use to produce the final renderings for your project, as well as
- help you complete the assigned lessons faster and without frustration. This is because, to some extent, lessons build on one another.

The `SoftImage 3D` tutorial is available at the teaching laboratory in Sweet Hall, in a box labeled **CS 348C**, with individual lessons separately bound. A second copy is available from the course coordinator for very short-term loans.

Here are the five parts of this problem:

1. *(12 points)* Do lesson 1 of the `SoftImage 3D` tutorial. Save your work, naming your scene **`head`**, and using **`quickstart`** as prefix.
2. *(12 points)* Do lesson 3 of the `SoftImage 3D` tutorial. Save your work, naming your scene **`tunnel`**, and using **`quickstart`** as prefix.
3. *(12 points)* Do lesson 6 of the `SoftImage 3D` tutorial. Skip steps 7 through 9 if you encounter problems starting up the mouse channel driver. Save your work, naming your scene **`hall`**, and using **`quickstart`** as prefix.
4. *(12 points)* Do lesson 7 of the `SoftImage 3D` tutorial. Save your work, naming your scene **`jar`**, and using **`quickstart`** as prefix.
5. *(2 points)* List four *conceptually distinct* ways in which you used splines (curves or surfaces) while doing lessons 1, 3, 6, and 7 of the `SoftImage 3D` tutorial. "Conceptually distinct" tasks roughly correspond to different menu cells.[1]

Problem 2: Degree-Raising Rational Curves. *(15 points)*

We saw in class that the Bernstein form of a degree 2 planar rational curve $\bar{\mathbf{q}}(t)$ is

$$\bar{\mathbf{q}}(t) = \frac{w_1\bar{\mathbf{b}}_1 B_1(t) + w_2\bar{\mathbf{b}}_2 B_2(t) + w_3\bar{\mathbf{b}}_3 B_3(t)}{w_1 B_1(t) + w_2 B_2(t) + w_3 B_3(t)},$$

where

$$\bar{\mathbf{b}}_1 = (x_1, y_1) \quad \bar{\mathbf{b}}_2 = (x_2, y_2) \quad \bar{\mathbf{b}}_3 = (x_3, y_3)$$

are the Bézier points of the rational curve, w_1, w_2, and w_3 are their weights, and

$$B_1(t) = (1-t)^2 \quad B_2(t) = 2(1-t)t \quad B_3(t) = t^2$$

are the Bernstein polynomials for quadratics. We also saw that the rational curve $\bar{\mathbf{q}}(t)$ can be thought of as the projection of the 3D generating curve

$$\mathbf{q}(t) = \mathbf{b}_1 B_1(t) + \mathbf{b}_2 B_2(t) + \mathbf{b}_3 B_3(t),$$

[1] A menu cell is one of the buttons on the vertical edges of the `SoftImage 3D` screen. Menu cells activate popup menus, whose individual entries are *not* called menu cells.

where

$$\mathbf{b}_1 = \begin{pmatrix} w_1 \\ x_1 w_1 \\ y_1 w_1 \end{pmatrix} \quad \mathbf{b}_2 = \begin{pmatrix} w_2 \\ x_2 w_2 \\ y_2 w_2 \end{pmatrix} \quad \mathbf{b}_3 = \begin{pmatrix} w_3 \\ x_3 w_3 \\ y_3 w_3 \end{pmatrix}.$$

Finally, we considered a rational curve $\bar{\mathbf{q}}(t)$ such that

$$\begin{aligned} w_1 &= 1 \qquad & (x_1, y_1) &= (1, 0) \\ w_2 &= e \qquad & (x_2, y_2) &= (1, 1) \\ w_3 &= 1 \qquad & (x_3, y_3) &= (0, 1), \end{aligned}$$

and found the value of e so that the curve is a circle:

$$e = \frac{1}{\sqrt{2}}.$$

In this problem, we will derive a degree 3 representation of the unit circle, and model the result in `SoftImage 3D`.

1. *(3 points)* Find $\mathbf{q}(t)$, the generating curve of $\bar{\mathbf{q}}(t)$.

2. *(4 points)* $\mathbf{q}(t)$ is a quadratic 3D curve. Degree-elevate it to a (degenerate) cubic 3D curve $\mathbf{c}(t)$. Express $\mathbf{c}(t)$ in terms of the Bernstein polynomials for cubics. Show your work.

3. *(3 points)* Project $\mathbf{c}(t)$ on the $w = 1$ plane, in order find a degree 3 representation $\bar{\mathbf{c}}(t)$ of the circle; write $\bar{\mathbf{c}}(t)$ in Bernstein form. Compute the coordinates of the Bézier points of $\bar{\mathbf{c}}(t)$, and their weights, to four decimal places. Show your work.

4. *(5 points)* Use `SoftImage 3D` to verify your results. Follow the steps below:

   - Draw a very fine linear approximation to the unit circle, using the **Get→Primitive→Circle** command, with 400 steps.
   - Draw a cubic NURBS with four control points, using the **Draw→Curve→NURBS** command. The NURBS you just drew consists of a single segment. This segment is part of a single rational curve whose shape is controlled by its Bézier points; these points coincide with the NURBS de Boor points.
   - Refine the rational curve rendering, using the **Info→Selection** command to change the number of steps to 100.
   - Set the positions and weights of the Bézier points to the values you computed, using the **Edit→Coordinate** command. `SoftImage 3D` should draw a quarter of a circle, as it only plots the curve segment corresponding to a narrow range of parameter values.

- Zoom as close to the rational curve as you can, and pan the camera along the curve's full length: the curve should perfectly overlap the circle.
- Save your work, naming your scene `circle`, and using `custom` as prefix.

Problem 3: Subdividing Rational Patches. *(15 points)*

In this problem, we will create a rational patch and then split it into two patches. While polynomial patches would also work for this problem, the `SoftImage 3D` NURBS interface offers interactive dialog boxes: these dialog boxes update the drawings immediately after you modify parameter values, thus allowing you to preview intended changes before committing to them.

1. *(3 points)* Consider a cubic rational patch. By some algorithm, we split this patch into two parts, along a domain line of constant v, thus obtaining a spline surface with two patches. Assume that we performed the split using some algorithm which guaranteed that the spline patches will always meet with C^2 continuity. How many control points does the spline surface have? Justify your answer.

2. *(2 points)* What if the split took place in a manner such that only C^1 continuity was guaranteed between adjacent patches? What about C^0? Justify your answers.

3. *(10 points)* Use `SoftImage 3D` to verify your results. Follow the steps below:

   - You cannot create a rational patch directly in `SoftImage 3D`. The way to model a single patch is by first creating a sphere as a cubic NURBS surface, with 4 steps in each direction (u and v); to this end, use the **Get→Primitive→Sphere** command.
   - The sphere is a closed surface. Open the surface along u, while preserving its shape, using the **Draw→Open/Close** command.
   - Cut the resulting two-patch spline into two pieces, the cutting line placed at $v = 1$; to this end, use the **Draw→Cut** command.
   - You now have isolated a single rational patch. Delete all other scene models.
   - Fix the shading of the patch, using the **Effect→Inverse** command.
   - Pull apart the four overlapping control points at the pole of the ex-sphere.
   - Use the **Info→Selection** command to change the number of steps[2] in both directions to 10.

[2] Why this parameter is given the same name as the number of patches in the sphere creation dialog box remains a mystery to me. "Steps", here, refers to the number of subdivisions made along each domain axis while triangulating the patch for rendering purposes.

- Duplicate the patch three times, using the **Duplicate→Repetition** command.
- Subdivide each of the three copies along a domain line of constant v, whose image splits the patch in two halves, approximately; to this end, use the **Effect→Subdivision** command. Perform the three subdivisions using, respectively, C^0, C^1, and C^2 continuity.
- Use the **Info→Selection** command to calculate the count of control points for each NURBS surface generated by the above subdivisions. You should get the same answer as in the previous parts of this problem.
- Move around the control points of each NURBS, in such a way that, in the shaded renderings, it is easy to visualize the different degrees of continuity between adjacent patches; for example, create a crease for the C^0 NURBS.
- Save your work, naming your scene `subdivided`, and using `custom` as prefix.

Problem 4: Merging Rational Curves. *(15 points)*

In the previous problem, we created a rational patch and then split it into two patches. Now we will merge two rational curves into a NURBS curve. As before, while B-spline curves would also work for this problem, we prefer NURBS because the `SoftImage 3D` NURBS interface offers interactive dialog boxes.

1. *(10 points)* Follow the instructions below to create two rational curves in `SoftImage 3D` and merge them:

    - Create two arbitrary planar cubic rational curves and refine their rendering resolution, by following steps similar to the ones described in problem 2. Place the last point of the first curve and the first point of the second curve near, but not on top of, each other.
    - Create a null, using the **Get→Primitive→Null** command. Make both curves children of the null, using the **Parent** menu cell.
    - Select the tree rooted at the null, and duplicate it (thus copying the curve pair) five times, using the **Duplicate→Repetition** command.
    - Merge the two adjacent curves of each pair, using the **Draw→Merge Curves** command. For each blend, use a different type in the dialog box.
    - Save your work, naming your scene `merged_curves`, and using `custom` as prefix.

2. *(5 points)* Assuming the two original curves were drawn for the parameter intervals $[0, 1]$ and $[2, 3]$, match each of the first three splines — the ones

generated using the first three blend types — to their *most probable* knot sequence:

0	0	0	1	1.33			1.67	2	3	3	3
0	0	0	1	1	1	2	2	2	3	3	3
0	0	0	1	1			2	2	3	3	3

Justify your answers. *Hints:* examine the control points of the original curves and the generated splines; their count and positions, as well as the continuity of the blend, should provide enough information to answer the question. If you get stuck, you might want to repeat this experiment using surfaces, and merging rational patches: the reason is that `SoftImage 3D` draws the embedded curves along patch boundaries, and thus you can easily tell how many patches comprise a given NURBS surface. If you repeat the experiment, save your work, naming your scene `merged_surfaces`, and using `custom` as prefix.

Problem 5: NURBS in `SoftImage 3D`. *(5 points)*

Browse through the `SoftImage 3D` *User's Guide*, *Reference 1* binder, Model module section (pages 2-1 to 2-235), and complete the following tasks:

1. *(1 point)* Identify four commands which, for no apparent reason, the user may perform on what the *User's Guide* calls splines (see page 2-4), but not on NURBS.

2. *(1 point)* What is wrong with the implementation of the command described on page 2-98? *Hint:* try trimming a NURBS surface in `SoftImage 3D`. *This is a trick question.*

3. *(1 point)* By properly setting the weights of their de Boor points, NURBS curves can perfectly model conics, such as circles, and NURBS surfaces can perfectly model quadrics, such as spheres. What values does `SoftImage 3D` pick for the de Boor point weights of a NURBS circle (or sphere)? Use the **Get→Primitive→Circle**, **Get→Primitive→Sphere**, and **Info→Points** commands to answer this question. Are the resulting circle or sphere exact?

4. *(1 point)* Is there a good reason that splines have to be drawn counterclockwise while NURBS have to be drawn clockwise for surface extrusion (page 2-132) or skinning (page 2-146)? *Hint: this is a trick question.*

5. *(1 point)* Is "NURB" the singular of "NURBS" (page 2-18)?

6. *(0 point)* Why did Microsoft corporation buy SoftImage Inc. for such a cheap price? Just kidding...

Conclusion: Turning in Your Work.

After completing the assignment, you must turn in both your written answers and your `SoftImage` models. You should turn in your written work in class. Written answers are required for

- Problem 1, part 5.
- Problem 2, parts 1, 2, and 3.
- Problem 3, parts 1 and 2.
- Problem 4, part 2.
- Problem 5, parts 1, 2, 3, 4, and 5.

In order to submit your `SoftImage` models, follow the instructions in the file

`/usr/class/cs348c/submissions/soft/README`

on the Sweet Hall machines.

Good luck with your project!