

# CSE 167, Winter 2003: Assignment 2 - Fireworks

Due date: February 5, 3pm

Part 1 points: 75

Part 2 (Extra credit) points: 20

In this assignment you will be drawing 2 separate paths for fireworks using Bezier curves, implementing an interactive user interface so that a user can trace the path with a ball, and then exploding fireworks when the ball reaches the top.

## Part 1: Drawing interactive curved paths.

### 1. Drawing Bezier curves

The first functionality is being able to draw Bezier curves which will determine the path of an interactive object (yellow dot). The curve drawing has to use explicit curve evaluation as discussed in class. You may NOT use the OpenGL evaluators.

### 2. Implementing interactive functionality: Picking & Dragging

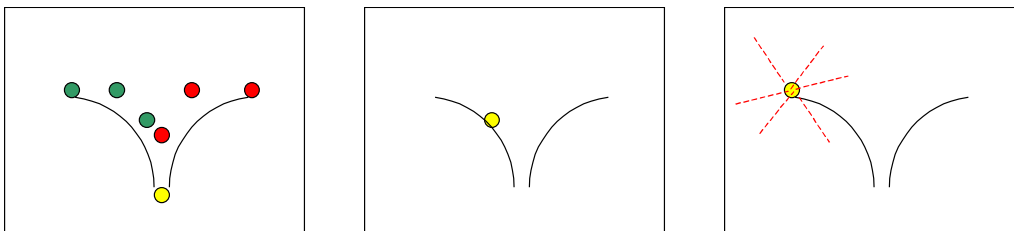
The second functionality to implement is allowing the user to pick points in the screen, such as control point locations and objects. Also, you will need to interpret user gestures such as a pressed mouse button, followed by a drag, and a released button.

### 3. User Interface State

Third, in order to process events in the right context, you will need to maintain a state in your program. For example, a user click in a particular state can mean “place control point here” vs “pick this object”.

### Details:

When the application starts, you should display a yellow dot close to the bottom edge of the window. The subsequent actions by the user will drive your application. The first thing the user can do is define curves by selecting control points.



The first diagram shows how the initial curves should look, and the control points for the curves. The yellow dot is a shared control point between the two curves. The green control points are for the left curve and the red control points are for the right curve. Note, you don't have to have curves that look exactly like I've drawn, but they should be something similar. If either of the curves is defined, the user should be able to pick any of the control points and move them around. When the control points are moved, the curve should change accordingly.

The second diagram shows that the yellow dot can be moved by the user, but it is constrained to follow the curve. So you have to decide which curve is closest to the user's

mouse movements, and follow that one up. When the yellow dot has been picked and is being moved, the control points are hidden.

The third diagram shows that when the yellow dot reaches the top, you should display something resembling fireworks. You can be creative here and draw whatever you want at the top. The fireworks stay displayed at the top of the curve until the user changes that particular curve. If the user moves the yellow dot over the second curve, the same behavior is repeated.

Steps for Part 1:

1. Initialize the window and draw the yellow dot.
2. Create a menu with the following options:
  - a. Draw curve 1: when this is selected, the next three user picks determine the control points for curve 1. If the user selects this when a curve 1 already exists, the new curve replaces the old one.
  - b. Draw curve 2: when this is selected, the next three user picks determine the control points for curve 2. If the user selects this when a curve 2 already exists, the new curve replaces the old one.
  - c. Exit: exit your application when this is selected
3. Implement a picking callback function which
  - a. Detects that the user should now pick control points (either for curve 1 or curve 2) and registers the position of the control points
  - b. Otherwise, after both curves are defined, it should detect when the yellow dot is picked. That will signal that the user is ready to move the yellow dot up one of the curves.
  - c. Detects if any of the control points is picked.
4. Implement a mouse motion callback which
  - a. Detects when the yellow dot can be moved
  - b. When the yellow dot is first picked, the next direction that the mouse moves, determines which curve the dot will move along (either curve 1 or curve 2)
  - c. After that, all subsequent motions of the mouse advances the yellow dot until it reaches the top. The yellow dot returns to its original position (at the bottom) when the mouse button is released.
  - d. If one of the control points is picked, the subsequent mouse movements translate the control point to the new location. When the mouse button is released the control point stays in the last moved position.
5. Implement a drawing function that takes as input 4 control points and draws a smooth Bezier curve corresponding to the control points.
6. Implement a drawing function to draw some primitive resembling fireworks at the top of either (or both) curve(s).
7. Implement the viewport and gluOrtho2D transformations such that when the window is resized, all primitives are scaled proportionately.

### **Extra credit: Adding interactive rotations & recursive subdivision**

1. **Interactive rotations (5 points):** Detect when the user has picked one of the curves and allow the subsequent mouse movements to rotate the curve. The curve should still remain pinned to the yellow dot at one end.
2. **Recursive subdivision (15 points):** Add an additional menu item called "Subdivide curve". Then allow the user to select one of the curves at the next pick. If a curve is selected, subdivide the curve into two Bezier curve segments. The top half of the original curve should be 1 new curve segment and the bottom half should be another curve segment. All 7 control points (one of them is the yellow dot) of the two new segments should be now displayed. The user can now select any one of the control points and manipulate the curve. Constrain control point displacement such that the 2 new segments do not separate, and also keep the 2 new curve segments G1 continuous (share tangent at boundary). Note: the fireworks are still displayed at the top of the entire curve, not at the top of each curve segment.

Note: if you are planning to do the extra credit, ensure you first have a working version of the required assignment. Save the working version in a separate project and then embark upon the extra credit. If you can't demonstrate a working version of the required part, you will not get full credit for it.

Submission:

1. Report explaining the algorithm, description of functions, and any other implementation details that explain your code.
2. Entire Visual C++ project directory including source files, header files and the compiled executable.

Submission process: WebCT based assignment turn in functionality should be working. Details of submission process will be posted to the mailing list on 1/31/2003. The assignment will not be accepted via email.

Late penalty will be applied to assignments turned in after the time stated above.