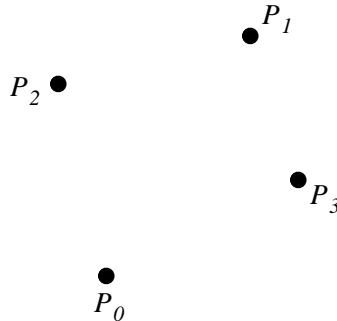


Questions 1

Question Q3-1: (Solution, p 2) Give an example of something that can easily be accommodated in ray tracing but not with the faster graphics pipelining technique.

Question Q3-2: (Solution, p 2) Sketch a Bezier curve between the below points. Show some intermediate work.



Question Q3-3: (Solution, p 2) Both cubic B-splines and Bezier curves are approximation curves, although Bezier curves are computationally simpler. What is the advantage of cubic B-splines?

Question Q3-4: (Solution, p 2) Explain the Monte Carlo technique for computing the integral of a function, and explain how the concept applies also to graphics.

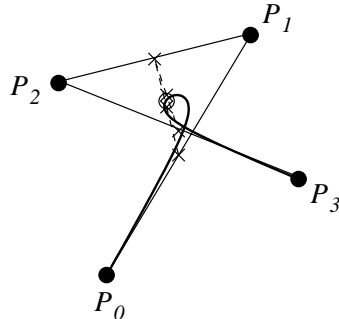
Question Q3-5: (Solution, p 2) Explain the basic technique underlying the automatic detection of beats within a piece of recorded music.

Question Q3-6: (Solution, p 2) How is an octree useful for detecting collisions?

2 Solutions

Solution Q3-1: (Question, p 1) There are several possible answers. Reflection and refraction are two of the most important. Another is adding shadows, which can only be added to the graphics pipeline with great difficulty but which is easily supported within ray tracing. Also, ray tracing is adapts more easily to a wide variety of curved surfaces.

Solution Q3-2: (Question, p 1)



For my work here, I computed the midpoint using the technique shown in class. I also used the tangents based on the lines between the control points.

Solution Q3-3: (Question, p 1) Suppose we have a curve between many control points. With Bezier curves, if we move any control point, that movement will affect the entire curve; with a cubic B-spline, however, moving a control point only affects a small segment of the overall curve.

Solution Q3-4: (Question, p 1) Monte Carlo integration works by taking the average of a sample of function values. Since the expected value of a random sample is proportional to the integral, the average is highly likely to be close to the integral.

This applies in graphics, too, where the illumination each point receives is an integral over the amount of light coming in along each ray on a sphere. We can apply Monte Carlo integration to approximate this by evaluating the light on a random sample of rays and averaging.

Solution Q3-5: (Question, p 1) We maintain a buffer remembering the total energy for each second, and we identify as a *beat* any segment of roughly $1/40$ seconds whose average energy is significantly above the average energy for the entire buffer. By *significantly above*, I mean about 50% more than the buffer's average energy.

Solution Q3-6: (Question, p 1) A straightforward collision detection system would need to check each object against all other object to see whether it collides with any of them. This is a $O(n^2)$ approach. With the objects stored within an octree, we would only need to check for collisions between objects intersecting the same cube of the octree; as a result, the amount of time taken is likely to be closer to $O(n)$.