CS52: Homework 4

Out: Nov 5. Due: Nov 19.

Problem 1: Keyframed Animation

In Assignment 3, we have defined a Bezier spline with keyframes to represent time variations of parameters. In particular, each segment of the spline is defined in the time interval $[t_k, t_{k+1})$; within this interval, the spline is controlled by the $\mathbf{q}_{4k}, \mathbf{q}_{4k+1}, \mathbf{q}_{4k+2}, \mathbf{q}_{4k+3}$ control points.

Suppose we are applying our keyframed spline to the position of an object.

(a) Which control point would you change to alter the object position at t_k? Which one would alter the object velocity at t_k?

(b) The previous splines does not enforce continuity at the segments endpoints. Is this useful or is it just an oversight? If the choice is a good one, give some example of real-world situations where you would have to break continuity to represent motion.

Problem 2: Deformations

(a) A twisting deformation can be expressed as a rotation around an axis of an angle that depends on the position of the point along the axis. In our case, we will rotate around the y axis of an angle $y\theta_{max}$. Given a point $(P) = (P_x, P_y, P_z)$, write the coordinates of the transformed point $\mathbf{P}_d$. You can assume that $P_y \in [0, 1]$.

(b) We would like to use skinning to express a similar deformation. To do this, we have set up two bones b_1 and b_2. b_1 is placed at $(0, 0, 0)$ while b_2 at $(0, 1, 0)$. During the animation b_1 will not be altered, while b_2 will be rotated an angle θ_{max} around y. We have set up the weights to be linearly interpolated, i.e. $w_1 = 1 - y$ and $w_2 = y$. Derive the transformation for the point $\mathbf{P}$.

(c) Comparing the transformed points in the two cases, comment on the "twist-collapse" problem inherent in skinning deformation.

Problem 3: Texturing

(a) We have introduce mip-mapping as a way to quickly compute texture averages under a pixel in the graphics pipeline. Would this be useful for raytracing? If so, which additional information would the ray hit need to provide (approximations are ok)?

(b) You are to use shadow mapping to compute shadows in an interactive application. The shadow lookups will happen at the fragment processing stage of the pipeline. Given the world-space position of a point $\mathbf{P}$, the world-to-clip space transform matrix for the light M_l and the depth texture $t(x, y)$ with $x, y, t \in [0, 1]$, write a closed-form expression to determine if $\mathbf{P}$ is in shadow.

(c) You are writing a racing car game. For speed consideration, you have decided to use textures to express the car details such as the paint color and the stickers on the car. In your game, cars have contact with each other and the world and you would like to provide some user feedback for that. In particular you have decided that for small scratches and skid marks, you would like to use additional textures. With memory and speed concerns in mind, how do you suggest to store this additional data (remember that these effects change rapidly at run time)? Would you try to update the original textures and create new ones? If creating new ones, what happens to the UV parameterization of the surfaces? Would you try to use new UV sets or projections? Describe the tradeoffs and assumptions of your design.

Problem 4: Comparing Raytracing to the Graphics Pipeline

We have to build a system and we are considering the use of raytracing and the graphics pipeline to render scenes with interesting lighting effects. The scene has l lights and o objects, r of which have reflections, and we want to generate images of n pixels.

For each of the following questions, give 3 answers corresponding to the graphics pipeline (using shadow mapping and cube environment mapping), a basic raytracer where each ray costs $O(o)$ intersections and an advanced raytracer where each ray costs $O(ln(o))$ intersections. To simplify the question, we only generate **one** ray for each pixel. You can assume that each ray will hit an object and r/o of all the rays will hit an object with reflection.

(a) How many times are objects accessed to generate the scene image with no lighting? How many times are objects accessed to generate the scene image with shadows but no reflection? How many times are objects accessed to generate the scene image with shadows and one-bounce reflections?

(b) If we only have one light, no shadows and no reflections, how many lighting computations are performed? For this assume that on average objects cover a fraction a of the total image area.

(c) Is the graphics pipeline hopelessly slow? Why is it in so much use? Motivate your answer with general reasoning and consider software/hardware architectural motivations too. Note

that some of this discussions are currently in progress in the research community.

Extra credit: Shadow Computations

(a) Shadow maps are much faster than raytraced shadows for shadow computation, but they lack in quality. Describe the main artifacts introduced by shadow mapping.

(b) Derive an approximated shadowing algorithm that combines shadow maps (for speed) and raytraced shadows (for quality).