15-462 Computer Graphics I

Midterm Examination

October 23, 2003

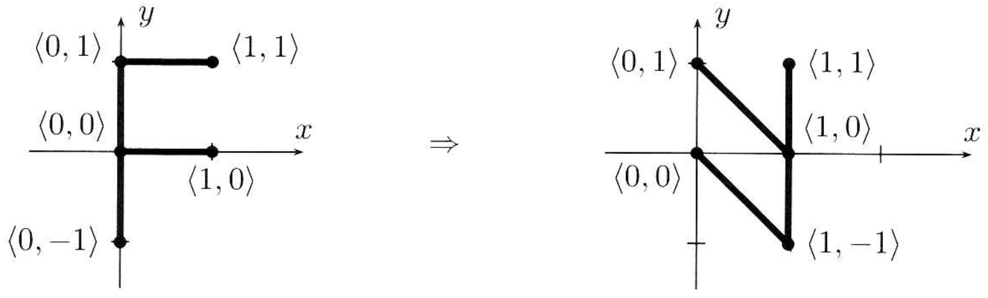
Name:	______________________________
Andrew User ID:	____________________

- This is a closed-book exam; only one double-sided sheet of notes is permitted.
- Write your answer legibly in the space provided.
- There are 12 pages in this exam, including 4 worksheets.
- It consists of 5 questions worth a total of 100 points.
- You have 80 minutes for this exam.

Problem 1	**Problem 2**	**Problem 3**	**Problem 4**	**Problem 5**	**Total**
20	**20**	**20**	**20**	**20**	**100**

1. 2D Transformations (20 pts)

Determine a 3x3 homogeneous matrix representation for the 2D affine transformation illustrated by



Let M represent the 3x3 matrix we wish to determine. Given the action of M on 3 independent homogeneous points, we can uniquely determine M. Consider the matrix of original points (as columns)

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

where we've chosen the three points (0,0), (0,-1) and (1,0). These points are transformed to

$$P' = MP = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Therefore we can determine the transformation by evaluating

$$\begin{aligned} M &= P'P^{-1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

We can verify the result on one of the other points. For example, we can correctly verify $(1,1)$ maps to $(0,0)$ as follows:

$$Mp = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

A second approach (that many people tried) is to try and guess a sequence of translations, rotations, shears, etc., that when multiplied will produce the correction transformation, M. While this is possible in this simple example, it is not always clear how to do this.

2. Weird Projections (20 pts)

We've considered one point perspective transformations, but it is also useful, e.g., in architectural drawings, to have multiple points of perspective. Recall that an example of a *one point perspective* transformation (in which one principle axis (z) pierces the projection plane) is

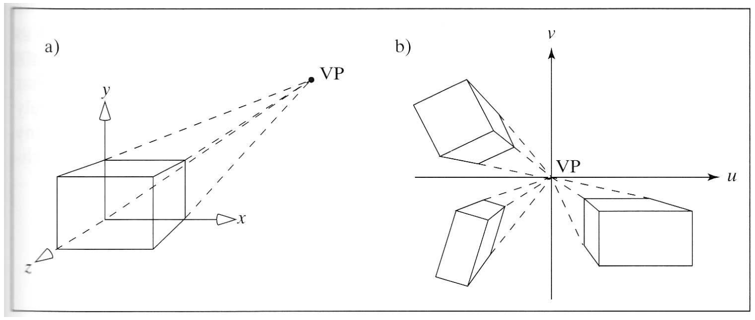
$$\mathbf{P^{1pt}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

The following projection matrix provides a *two point perspective* transformation,

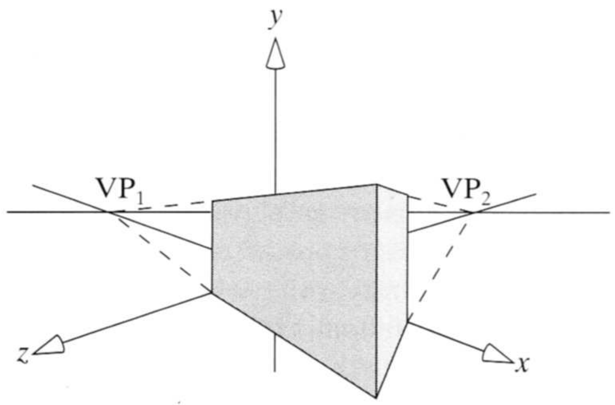
$$\mathbf{P^{2pt}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{\sin(q)}{d} & 0 & \frac{\cos(q)}{d} & 0 \end{bmatrix}$$

Describe with words and a diagram what this particular projection does. Why is it called a two point perspective transformation? What is the effect of the q parameter?

The usual perspective has one vanishing point, and the transformation $\mathbf{P^{1pt}}$ can be used to produce images such images. The following image shows (left) a one-point perspective scenario similar to $\mathbf{P^{1pt}}$, and (right) a projected image with a vanishing point at (0,0).



One the other hand $\mathbf{P^{2pt}}$ produces a scenario similar to the following image:

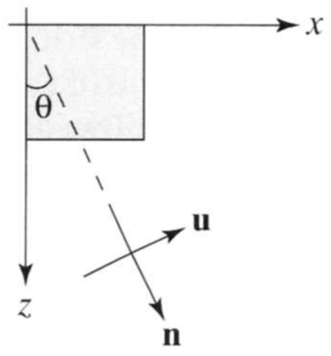


Note the two finite perspective vanishing points, and hence the name "two point perspective." Historically, perspective projections are categorized by the number of *finite vanishing points*. Alternately, you can categorize them by the number of principal projection axes that are not perpendicular to the view direction vector, $\mathbf{n}$, which is the same as the number of principal perspective axes that pierce the view plane.

The effect of the q parameter is to transition between one- and two-point perspective, by effectively "rotating the perspective effect." Note that if $q = 0$, then $\mathbf{P^{2pt}}(\mathbf{q} = \mathbf{0}) = \mathbf{P^{1pt}}$, so we're back to one-point perspective. By increasing q from zero, we gain the two-point perspective by effectively moving the x-axis vanishing point in from infinity to a finite location. In fact, $\mathbf{P^{2pt}}$ is just a rotated version of $\mathbf{P^{1pt}}$:

$$\mathbf{P^{2pt}} = \mathbf{R_y}(-\mathbf{q})\mathbf{P^{1pt}}\mathbf{R_y}(\mathbf{q}).$$

The following figure illustrates the camera direction $\mathbf{n}$ whose angle θ is akin to our q angle:



You can also compute the location of the vanishing points by taking points at infinity on the X, Y, and Z axes, and see how they are mapped when $\mathbf{P^{2pt}}$ is applied. In homogeneous coordinates, a point at infinity is represented by placing zero in the W coordinate of a point vector. So, a point at infinity on the X axis is represented as $(1, 0, 0, 0)'$. When we apply $\mathbf{P^{2pt}}$ to this point, we get $(1, 0, 0, sin(q)/d)'$. In non-homogeneous coordinates, this point becomes $(d/sin(q), 0, 0)'$. So, the vanishing point for X lies at the above coordinates. Similarily, the vanishing point for the Z axis lies at $(0, 0, d/cos(q))'$. However, when we apply $\mathbf{P^{2pt}}$ to a point at infinity on the Y axis we get (in homogeneous coordinates) $(0, 1, 0, 0)'$, and so this point remains at infinity. Since we are in a two-point projection system, it makes sense that only two axes (X and Z) map to vanishing points, while one still approaches infinity.

Further information on multi-point perspective can be found on Brian Barsky's page:
`http://www.cs.berkeley.edu/~barsky/perspective.html`

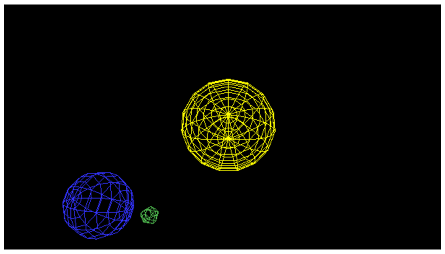
3. OpenGL & Hierarchies (20 pts)

Describe what you think the following piece of `display()` code does. Feel free to mark up the code and place coherent comments in the right margin. (Hint: It's a classic hierarchical system.):

The example animates an Earth/Moon/Sun system. Commented code follows:

```
...
// Clear the current matrix (Modelview)
glLoadIdentity();

// Back off eight units to be able to view from the origin.
glTranslatef ( 0.0, 0.0, -8.0 );

// Rotate the plane of the elliptic
// (rotate the model's plane about the x axis by fifteen degrees)
glRotatef( 15.0, 1.0, 0.0, 0.0 );

// Draw the sun-- as a yellow, wireframe sphere
glColor3f( 1.0, 1.0, 0.0 );
glutWireSphere( 1.0, 15, 15 );

// Draw the Earth
// First position it around the sun
//Use DayOfYear to determine its position
glRotatef( 360.0*DayOfYear/365.0, 0.0, 1.0, 0.0 );
glTranslatef( 4.0, 0.0, 0.0 );
glPushMatrix();// Save matrix state
// Second, rotate the earth on its axis.
//Use HourOfDay to determine its rotation.
glRotatef( 360.0*HourOfDay/24.0, 0.0, 1.0, 0.0 );
// Third, draw the earth as a wireframe sphere.
glColor3f( 0.2, 0.2, 1.0 );
glutWireSphere( 0.4, 10, 10);
glPopMatrix();// Restore matrix state

// Draw the moon.
//Use DayOfYear to control its rotation around the earth
glRotatef( 360.0*12.0*DayOfYear/365.0, 0.0, 1.0, 0.0 );
glTranslatef( 0.7, 0.0, 0.0 );
glColor3f( 0.3, 0.7, 0.3 );
glutWireSphere( 0.1, 5, 5 );
...
```

One snapshot of rendered animation is



4. **Appearance Modeling (20 pts)**

1. **Mipmapping (6 pts):** Derive the maximum amount of additional memory required to mipmap a very large square image?

   Mipmapping recursively blurs and downsamples an image so that in addition to the original image, we must also store images that are one quarter the size of the image they were generated from. The total memory required (in units of the original image's memory size) is

   $$1+\frac{1}{4}+\left(\frac{1}{4}\right)^2+\ldots=\frac{1}{1-\frac{1}{4}}=1+\frac{1}{3}$$

   so that no more than $\frac{1}{3}$ additional memory is required.

2. **Shading (7 pts):** Describe the difference between Gouraud shading and Phong shading?

   Gouraud shading computes vertex lighting using the Phong lighting equation and a normal computed by averaging the normals of all adjacent faces, and then interpolates vertex colors accross the polygon in the raster stage. On the other hand, Phong shading interpolates vertex normals across the polygon, and then uses these normalized normal values to evaluate the Phong lighting model at the per-pixel level. (Note that Phong shading is more expensive than Gouraud shading.)

3. **Frequency-dependent lighting (7 pts):** Our textbook Phong lighting model includes several user-specified constant parameters, e.g., diffuse reflectance, k_d. However, it can be useful to allow these parameters to have frequency dependence. Describe one visual effect this might afford?

   SKIPPED. (Everyone given 7/7.)

5. Kochanek-Bartels Splines (20 points)

In 1984 D. Kochanek and R. Bartels introduced a new type of spline to give animators more control over keyframe animation. These popular splines are nothing more than Hermite curves and a handful of formulas to calculate the tangents. Given a sequence of points, $\ldots, P_{i-1}, P_i, P_{i+1}, P_{i+2}, \ldots$, the Hermite spline $P(u)$ connecting the two points P_i and P_{i+1} can be written as

$$P(u) = (u^3 u^2 u 1) \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} P_i \\ P_{i+1} \\ T_i \\ T_{i+1} \end{pmatrix}$$

where T_i and T_{i+1} are the tangent vectors at the points. The two tangents that they chose on either side of a point (for each adjacent spline) were as follows: the incoming (left) tangent to a point was

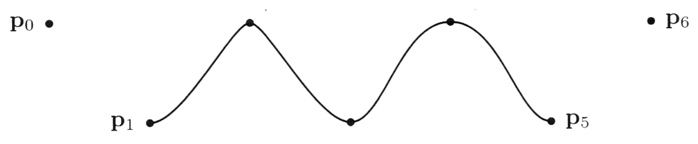
$$TS_i = \frac{(1-a)(1-b)(1+c)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1+b)(1-c)}{2}(P_{i+1} - P_i)$$

and the outgoing (right) tangent to a point was

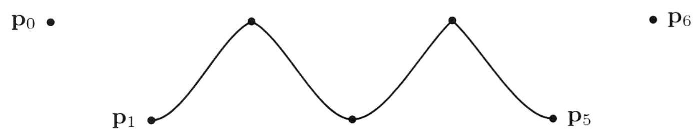
$$TD_i = \frac{(1-a)(1+b)(1+c)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1-b)(1-c)}{2}(P_{i+1} - P_i).$$

These splines are also referred to as **TCB-Splines** because they provide control over three important behaviors:

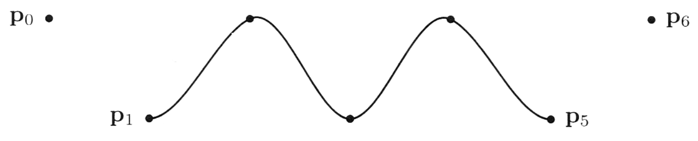
- **Tension:** How sharply does the curve bend?
- **Continuity:** How rapid is the change in speed and direction?
- **Bias:** What is the direction of the curve as it passes through the keypoint?



The effects of the tension parameter.



The effects of the continuity parameter.



The effects of the bias parameter.

1. **TCB Matching (10 pts):** Match the above parameters to the TCB properties, and provide some brief evidence (e.g., words, or using a diagram).

   The parameters match in order: a=Tension, b=Continuity, c=Bias. Their effects are illustrated with numbers here:

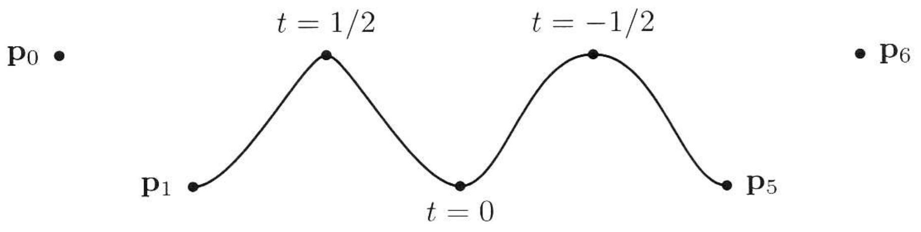


Figure VII.26. The effects of the tension parameter.

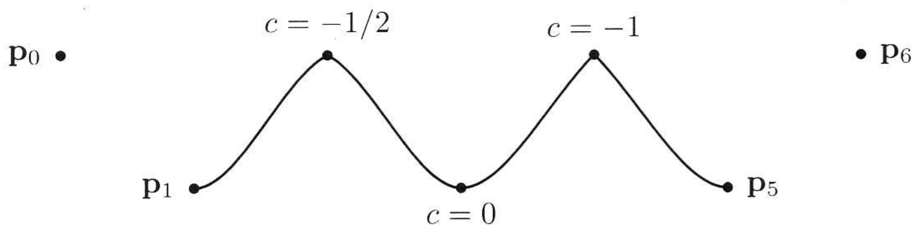


Figure VII.27. The effects of the continuity parameter.

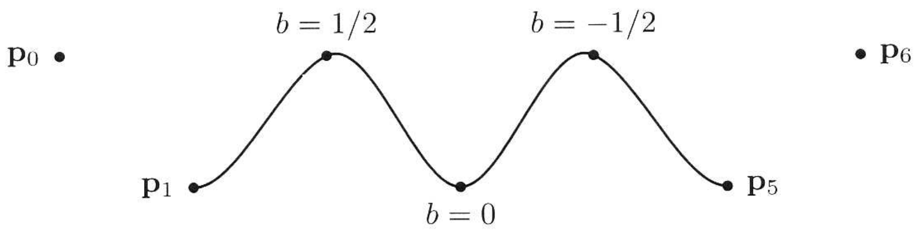


Figure VII.28. The effects of the bias parameter.

   What follows is a very detailed discussion taken from *3D Computer Graphics: A mathematical approach with OpenGL, by Sam Buss, Cambridge University Press, 2003.*

The tension parameter, denoted t, adjusts the tightness or looseness of the curve. The default value is $t = 0$; positive values should be less than 1 and make the curve tighter, and negative values make the curve looser. Mathematically, this has the effect of setting

$$D\mathbf{q}_i^- = D\mathbf{q}_i^+ = (1-t)\left(\frac{1}{2}\mathbf{v}_{i-\frac{1}{2}} + \frac{1}{2}\mathbf{v}_{i+\frac{1}{2}}\right),$$

that is, of multiplying the derivative by $(1-t)$. Positive values of t make the derivative smaller: this has the effect of making the curve's segments between points $\mathbf{p}_i$ straighter and making the velocity of the curve closer to zero at the points $\mathbf{p}_i$. Negative values of t make the curve looser and can cause it to take bigger swings around interpolation points. The effect of setting tension to $1/2$ and to $-1/2$ is shown in Figure VII.26.

The continuity parameter is denoted c. If $c = 0$, then the curve is C^1-continuous; otherwise, the curve has a corner at the control point $\mathbf{p}_i$ and thus a discontinuous first derivative. The mathematical effect of the continuity parameter is to set

$$\begin{aligned} D\mathbf{q}_i^- &= \frac{1-c}{2}\mathbf{v}_{i-\frac{1}{2}} + \frac{1+c}{2}\mathbf{v}_{i+\frac{1}{2}} \\ D\mathbf{q}_i^+ &= \frac{1+c}{2}\mathbf{v}_{i-\frac{1}{2}} + \frac{1-c}{2}\mathbf{v}_{i+\frac{1}{2}}. \end{aligned}$$

Typically, $-1 \le c \le 0$, and values $c < 0$ have the effect of turning the slope of the curve towards the straight line segments joining the interpolation points. Setting $c = -1$ would make the curve's left and right first derivatives at $\mathbf{p}_i$ match the slopes of the line segments joining $\mathbf{p}_i$ to $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$.

The effect of $c = -1/2$ and $c = -1$ is shown in Figure VII.27. The effect of $c = -1/2$ in this figure looks very similar to the effect of tension $t = 1/2$ in Figure VII.26; however, the effects are not as similar as they look. With $t = 1/2$, the curve still has a continuous first derivative, and the velocity of a particle following the curve with u measuring time will be slower near the point where $t = 1/2$. On the other hand, with $c = -1/2$, the curve has a "corner" where the first derivative is discontinuous, but there is no slowdown of velocity in the vicinity of the corner.

The bias parameter b weights the two average velocities $\mathbf{v}_{i-\frac{1}{2}}$ and $\mathbf{v}_{i+\frac{1}{2}}$ differently to cause either undershoot or overshoot. The mathematical effect is

$$D\mathbf{q}_i^- = D\mathbf{q}_i^+ = \frac{1+b}{2}\mathbf{v}_{i-\frac{1}{2}} + \frac{1-b}{2}\mathbf{v}_{i+\frac{1}{2}}.$$

The curve will have more tendency to overshoot $\mathbf{p}_i$ if $b > 0$ and to undershoot it if $b < 0$. The effect of bias $b = 1/2$ and bias $b = -1/2$ is shown in Figure VII.28.

The tension, continuity, and bias parameters can be set independently to individual interpolation points or uniformly applied to an entire curve. This allows the curve designer to modify the curve either locally or globally. The effects of the three parameters can be applied together.

2. **Convex hull (10 pts):** Do TCB-Splines have the convex hull property?

   No. Hermite splines do not lie inside their control polygon, and therefore the TCB-Splines do not have the convex hull property. This is somewhat of a trick question because Hermite splines have user-specified tangents that do not contribute to the control polygon; unlike Bezier splines which have 4 points per spline segment (and use points to approximate tangents), Hermite splines only have two points (and two tangents). Note that the only way for a single cubic Hermite segment to lie within it's two control points is if it is a straight line, i.e., a linear curve.

   You could also see that TCB-Splines from the "bias figure" do not even lie within the convex hull of the set of all control points, so certainly TCB-Splines do not have the convex hull property.