Solutions to the Take-Home Midterm Quiz

Solution 1:

(a) `glutMouseFunc`: Invoked when a mouse button is depressed or released.
(b) `glutMotionFunc`: Invoked when the mouse is dragged while button is depressed.
(c) `glutPassiveMotionFunc`: Invoked when the mouse is moved, irrespective of whether any button is pushed.

Solution 2:

- Storytelling and narrative involves the storyline, dramatic effect and motivation, involvement. (E.g., Evil zombies invade our fair city, and my job is to kill all the zombies.)
- Core mechanics involves the basic rules of play. (E.g., Shooting and hitting a zombie decreases its strength by 20%.)
- Interactivity involves how the player perceives the world and how he/she acts within it. (E.g., The mouse is used to aim the anti-zombie gun, and hitting the space-bar fires it.)

Solution 3:

(a) The first command applies the transformation to the current node relative to this node’s local coordinate system. The second applies the transformation relative to the world’s coordinate system.

(b) (ii): As with any operation applied to a node in a scene graph, the transformation applies to this node and all its descendants. (The fact that it is done relative to the world coordinate system affects the actual translation, but not the nodes that are affected.)

(c) Let $C$ denote the node’s current associated transformation. To implement the first command, we can create a $4 \times 4$ homogeneous translation matrix $T$ (whose diagonal entries are $(100, 10, 0, 1)$) then premultiply this times $C$, making the result the new transformation for the node.

To implement the second command, let $M$ be the matrix that performs a change of coordinate transformation from the node’s coordinate system to the world’s coordinate system. To perform the transformation relative to the world coordinate system, we would first convert to the world coordinates, then apply $T$, and then convert back. Thus, we would premultiply $C$ times the matrix $M^{-1}TM$, making the result the new transformation for the node. (Presumably, $M$ would need to be updated in the process.)

Solution 4: Throughout, we will represent the quaternion $(s, x, y, z)$ with the scalar-vector notation $(s, \mathbf{u})$, where $\mathbf{u} = (x, y, z)$.

(a) By the definition in class, the quaternion that encodes a rotation by angle $\theta$ about unit vector $\mathbf{u}$ is $(\cos(\theta/2), (\sin(\theta/2))\mathbf{u})$. In this case $\mathbf{u} = (0, 1, 0)$ and so we have

$$
q = \left\langle \cos \frac{\pi}{4}, \sin \frac{\pi}{4} \right\rangle (0, 1, 0) = \left\langle \cos \frac{\pi}{4}, 0, \sin \frac{\pi}{4}, 0 \right\rangle = \left\langle \frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 0 \right\rangle.
$$

(b) Recall that the conjugate of $q$, denoted $q^*$, is formed by negating the vector part (the last three coordinates). That is, $q^* = (1/\sqrt{2}, 0, -1/\sqrt{2}, 0)$. Since $q$ is of unit length, it follows that $q^{-1} = q^*$. However, let us work it out in detail. Recall from class that if $q = (s, x, y, z)$ then $qq^* = (s^2 + x^2 +$
$y^2 + z^2$). (This can be proved more tediously by applying the definition of quaternion multiplication.)

Therefore, by definition of inverse given in class we have

$$q^{-1} = \frac{q^*}{qq^*} = \frac{q^*}{\frac{1}{2} + \frac{1}{2} + 0} = \left\langle \frac{1}{\sqrt{2}}, 0, \frac{-1}{\sqrt{2}}, 0 \right\rangle,$$

as desired.

(c) Let $v = (0, 2, 1)$, implying that $p = \langle 0, v \rangle$. Let $u = (0, 1, 0)$. Recall that the quaternion product of $(s, u) \cdot (t, v) = \langle st - (u \cdot v), sv + tu + (u \times v) \rangle$. Applying this we have

$$pq^{-1} = \langle 0, v \rangle \left\langle \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}} u \right\rangle$$

$$= \left\langle 0 \cdot \frac{1}{\sqrt{2}} - \left( v \cdot \left( \frac{-1}{\sqrt{2}} u \right) \right), 0 \cdot \left( \frac{-1}{\sqrt{2}} u \right) + \frac{1}{\sqrt{2}} v + \left( v \times \left( \frac{-1}{\sqrt{2}} u \right) \right) \right\rangle$$

$$= \left\langle \frac{1}{\sqrt{2}} (v \cdot u), \frac{1}{\sqrt{2}} (v - \frac{1}{\sqrt{2}} (v \times u)) \right\rangle = \frac{1}{\sqrt{2}} \langle (v \cdot u), v - (v \times u) \rangle.$$

Since $u = (0, 1, 0)$ and $v = (0, 2, 1)$, we have $(u \cdot v) = 2$ and $(v \times u) = (-10, 0)$. Thus we have

$$pq^{-1} = \frac{1}{\sqrt{2}} \langle 2, (0, 2, 1) - (-1, 0, 0) \rangle = \frac{1}{\sqrt{2}} \langle 2, (1, 2, 1) \rangle = \left\langle \frac{2}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{2}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\rangle,$$

as desired.

(d) Let $w = (1, 2, 1)$, so that

$$pq^{-1} = \left\langle \frac{2}{\sqrt{2}}, \frac{1}{\sqrt{2}} w \right\rangle.$$

We pre-multiply by $q$, which yields

$$q(pq^{-1}) = \left\langle \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} u \right\rangle \left\langle \frac{2}{\sqrt{2}}, \frac{1}{\sqrt{2}} w \right\rangle$$

$$= \left\langle \frac{1}{\sqrt{2}} (1, u), \frac{1}{\sqrt{2}} (2, w) \right\rangle = \frac{1}{2} \langle 1, u \rangle \langle 2, w \rangle$$

$$= \frac{1}{2} \left( 1 \cdot 2 - (u \cdot w), 1 \cdot w + 2 \cdot u + (u \times w) \right)$$

$$= \frac{1}{2} \left( 2 + (u \cdot w), w - 2u + (u \times w) \right).$$

Using the facts that $(u \cdot w) = 2$ and $(u \times w) = (1, 0, -1)$ we have

$$q(pq^{-1}) = \frac{1}{2} \langle 2 - 2, (1, 2, 1) + (0, 2, 0) + (1, 0, -1) \rangle$$

$$= \frac{1}{2} \langle 0, (2, 4, 0) \rangle = \langle 0, 1, 2, 0 \rangle,$$

as desired.

(e) To obtain the final result, we extract the vector part of $qpq^{-1}$, which is $(1, 2, 0)$. It is easy to see that this is the result of rotating the vector $(0, 2, 1)$ by an angle of $\pi/2$ about the $y$-axis.

**Solution 5:**

(a) Implicit model: We can determine whether a point lies inside or outside the object by simply evaluating the model’s function $f(x, y, z)$ and check its sign.
(b) Parametric model: It is possible to generate a grid of vertices on the surface by enumerating a series of \( s \) values, a series of \( t \) values, and then computing the coordinates for each pair \((s_i, t_j)\). Given this grid, it is easy to connect them together into a grid.

(c) This is a bit trickier than the other two. Basically, we want to determine the largest and smallest possible \( x \), \( y \), and \( z \) coordinates of any point on the object’s surface. For the parametric surface, we can do this approximately by invoking and taking the maximum and minimum coordinates among the vertices of the grid. (This is not perfect, but is pretty close if we generate sufficiently many vertices.) I do not see an easy way to do this for general implicit models. It should be possible for relatively simple special cases (e.g. convex bodies), through an analysis of the derivatives. However, for complex bodies, the derivative structure might be quite complex.

Solution 6:

(a) Consider the collapse of the edge \( e_i = (u_i, v_i) \). Let \( w_i \) denote the new vertex created at the midpoint of \( u_i \) and \( v_i \). There are two triangular faces incident to \( e_i \), lying to its left and right. After the collapse, these triangles are mapped to two edges that are incident to the new vertex \( w_i \). Call these edges \( e'_i \) and \( e''_i \). All the other edges that were incident to \( u_i \) and \( v_i \) will be incident to \( w_i \) after the collapse.

In order to “undo” the collapse, we first need to know where the original vertices were located. We could do this by storing the coordinates of both \( u_i \) and \( v_i \). This is actually redundant. We already know their midpoint, and so it would suffice to store only one of them, or to store the vector between them.

The above information will tell us where the original vertices are, but it does not determine which edges are connected to them. We need to know which of the edges attached to \( w_i \) are incident to \( u_i \) and which are incident to \( v_i \). To do this, we store the two post-collapse edges \( e'_i \) and \( e''_i \). All the edges counterclockwise from \( e'_i \) to \( e''_i \) will be assigned to one expanded vertex (say \( u_i \)) and all the edges counterclockwise from \( e''_i \) to \( e'_i \) will be assigned to the other (say \( v_i \)).

(b) In the case of a half-edge collapse, all the same information is maintained, but because \( w_i \) is the same as one of the collapsed pair (say \( u_i \)), we only need to store the other vertex \( v_i \). We store the same edge information as in part (a).

Solution 7: (These answers are not absolute.)

(a) Free throw: Motion capture would be easy. Just find a basketball player and wire him up.

(b) Pool break: As spheres pool balls are easy to model in physics, so procedural animation would be the best.

(c) Lion/Gazelle: Motion capture might be an option, but I wouldn’t want to be the poor technician who has to attach sensors to your lion. For insurance reasons, this would be better left to the keyframe artists in the company’s art department. Procedural animation might be useable, e.g. to compute a pursuit path for the lion or an evading path for the gazelle.

(d) Tank explosion: Explosions can be simulated pretty well in physics, so procedural animation would be best.

(e) Dragon: Unless you can genetically engineer a dragon, motion capture is not really an option. Keyframe animation would be good, since a dragon is an imaginary creature, and your company’s artists probably could imagine its flight. This might be assisted by some procedural animation, e.g. to deal with complex 3-dimensional rotation and the effects of gravity and wind resistance.