CS 410, Fall 2012, Review Topics/Questions for the First Midterm
September 20, 2012

1. You should be able to describe the high level and key distinctions between the 2 approaches to rendering images in graphics: specifically the projective pipeline versus ray tracing.

2. You should be able to describe in plain English what is meant by the term “polygonal mesh” and why it is important in computer graphics.

3. You can recognize whether a polygon is or is not convex. Moreover, you should be able to provide a formal statement of the condition that must hold for convex polygons, and which is violated by non-convex polygons.

4. Much of the modeling done in computer graphics uses triangles. Four sided polygons are arguably the next most common. When asked about the relative merits of these two, you should have a quick and ready answer with respect to a problem which may arise with 4 sided polygons that simply cannot arise for 3 sided polygons.

5. You are now familiar with the matrix representation of a triangle strip, and in particular, you should have a ready answer for the question of how many triangles are specified by a 3 x 10 matrix.

6. The mathematics of 3-D modeling, both of objects and cameras, rests squarely upon a solid working understanding of the following 3 terms: scalar, vector and point. Your understanding of computer graphics at this point means you're able to succinctly define each of these 3 mathematical concepts.

7. The concept of a vector space is straightforward, for the sake of this class you may define it simply as the space of all vectors which may be defined using a fixed number of scalars: typically 3. The concept of an affine space is somewhat more difficult to hang onto, but you now know the one thing that affine spaces have that the vector spaces do not, and you can use it to remind yourself about the difference between these 2 concepts.

8. You now appreciate that the dot product has significance for geometric modeling at many levels. At the most pragmatic level, you certainly know how to compute the dot product between 2 vectors. At a much deeper level, the dot product represents the transition from affine spaces to Euclidean spaces. You're now in a position to explain why.

9. You have now seen how basis vectors play a truly fundamental role in how we express the geometry of object models and cameras. Here's a thought question you should be ready to tackle. In order to span a space, is it absolutely essential that basis vectors be orthogonal? Another way to think about this question is as follows, if it is not a requirement that basis vectors be orthogonal, why does it appear that we place such a high premium on selecting orthogonal basis vectors in practice.

10. The simple English phrase “the same point” is more ambiguous and perhaps fraught with peril than you would have initially imagined before taking this course. At a minimum, there are 2 fundamentally distinct and separate interpretations of the phrase. You should now be prepared to describe each of these and contrast them with each other.

11. The words “normalize” and “normal” get used a lot in computer graphics. Their meaning is somewhat context dependent, and in this regard, you fortunately will understand what it means to normalize a normal vector in order to produce a unit normal. If you are looking for a concrete example in which to clarify the sentence above, consider how one might define all the points which lie on a plane perpendicular to the vector (2, 2, 1) and exactly 3 units from the origin as measured in the direction (2, 2, 1).

12. You are now aware that there are 2 rather distinct ways of introducing the concept of rotation in the 2-D plane. Most references (e.g. Wikipedia) simply state the rotation matrix in terms of the cosine and sine of an angle theta. There is nothing incorrect about this approach, but it does not really help one understand what is actually taking place when a 2-D point is rotated. Fortunately, you can now take your knowledge of the dot product and derive a 2-D rotation matrix without ever having to touch a cosine or a sine.

13. There are many ways to motivate homogeneous coordinates, fortunately you are now in a position to quickly provide a pragmatic motivation in terms of chaining together, or perhaps one should say composing, sequences of transformations. Further, if making this argument to another it is best to illustrate using concrete examples of commonly named types of 2-D transformations.

14. It goes without saying that at this point in the course you can write down examples of 2-D rotation, translation, scaling, non-uniform scaling, and sheer matrices - all in homogeneous coordinates. Just as important, if you are presented with a set of 2-D points show before and after the application of one or several of these transformations, you can now work out by looking at the relative positions of the points pre- and post-transformation which operation or combinations of operations were carried out.

15. Perhaps one odder aspect of learning computer graphics is the need first to learn about Euler angles and subsequently to learn why in practice they are virtually never the best way of capturing the rotation of an object or a camera. Fortunately, you can both explain what Euler angles are and why they are not heavily used.

16. In general, one technique for determining a 2-D transformation is to view the terms in the transformation matrix as unknowns and the coordinates of points before and after the transformation as constraints. For arbitrary combinations of rotation, translation, and scaling, this is not typically the sort of thing that would be asked in in an exam. However, for simpler types of transformations, perhaps a combination of translation and scaling, it is entirely reasonable exercise.

17. If you are asked over lunch in a job interview to compute the cross product between the vectors (1, 2) and (2, 3), be certain that you would know why your reaction should be one of puzzled amusement, aware of the fact that you are being baited into possibly revealing a basic misunderstanding of geometry.

18. There are 2 ways you should have the cross product committed to memory. The 1st is to be able to explain in English the geometric interpretation of the resultant vector. The 2nd is the algebraic approach in which you can actually compute the cross product of 2 vectors.

19. If one were to say there's something special about the cross product between 2 unit length and perpendicular vectors, you fortunately could quickly say what that special property is.

20. In 3 dimensions as opposed to 2 dimensions, it is a bit harder to make up matrices that represent valid 3-D rotations. That said, is not all that hard, and you now know how. You should be able to explain by example the rules governing the construction of a valid rotation matrix.

21. Just as computer graphics folks don't typically think much of Euler angles, many of us have a fondness for the axis angle approach to specifying 3-D rotations. Fortunately, you can now illustrate this by showing how to rotate 45° about the vector (2, 2, 1).

22. Given its relative simplicity when compared to perspective projection, orthographic projection can be easily overlooked. However, you do understand its uses, and you could certainly write down an example of an orthographic projection matrix.

23. Our lectures on modeling cameras have used phrases such as “bear in the box” and “pinhole room”. What underlying critical concept rests beneath the illustrations that have in turn given rise to these phrases?

24. A predator with orthographic sight (orthographic projection) would not be fooled by an elephant hiding behind a rabbit. Not so for standard issue predators whose eyesight follow the laws of perspective projection. You should have no trouble sketching an illustration of this admittedly somewhat off-the-wall approach to contrasting orthographic and perspective projection.

25. In the lecture introducing perspective projection, the 1st perspective projection matrix was derived for the case where the image plane is moved an amount d in front of the focal point (perspective reference point). This was then contrasted with an alternative formulation involving an image plane at the origin, the perspective reference point behind the image plane by an amount d, and finally points in the world being imaged in front of the image plane an amount z. Superficially, the 2 matrices look extremely similar, one can even arrive at 1 starting from the other by simply swapping ones and zeros between two positions within the matrix. That said, they behave very differently, and fortunately you're in a position now to explain the difference.

26. Some computer graphics folks choose to think of perspective projection as a linear operator, while others do not. It matters less who is right then that you can clearly state the crux of each side in this argument.

27. Pointing a camera in the context of computer graphics involves both moving the camera and orienting the camera. Each of these operations is manifested in a 4 x 4 homogeneous transformation, the 1st involving translation, and the 2nd involving rotation. Each of these steps you understand and can illustrate using concrete numerical examples.

28. Of the 2 aspects associated with placing a camera relative to a scene, orienting the camera is the more involved. Typically, orientation is specified using a view plane normal and an up vector. Much of the work earlier on in the semester developing the understanding of basis vectors was in preparation for understanding geometrically the process of constructing the rotation matrix that orients a camera model. Therefore, you need not memorize the placement of elements in this matrix. You instead are comfortable derive it from the geometry associated with the view plane normal and up vector.

29. Clipping object geometry to the view plane or more generally in 3-D to the view volume is the last step before finally beginning the process of coloring pixels - typically called rasterization. For the 2-D case, the Cohen-Sutherland algorithm was presented. You should now be in a position to illustrate/explain this algorithm.

30. The 2 terms “frustum" and “canonical view volume” are used nearly interchangeably when describing 3-D clipping. Beth terms now make sense to you, and you should be able to explain them in your own words and with your own simple drawings. Doing so includes the critical concepts of near and far clipping planes.

31. One benefit of mastering Bresenham's algorithm is that it reviews and accentuates one's understanding of how to represent and manipulate 2-D lines. So, for example, after understanding this algorithm you can easily convert between representations such as slope-intercept and implicit-form. In addition, either form is easily derived given the endpoints of a line segment.

32. The essence of Bresenham's algorithm lies in transforming scan conversion of lines from the problem of finding nearest pixels to the problem of incrementally selecting between 2 choices using nothing more than integer arithmetic. This is a process you can illustrate by hand for perhaps 2 or 3 steps of the algorithm.