CS 410, Fall 2012, Comprehensive Review
December 6, 2012

First third of the Semester.

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.

Second third of the Semester.

33. You now have, if you did not already, a basic familiarity with the characterization of visible light in terms of a spectrum where the wavelengths vary from the shortest around 380 nm to the longest around 780 nm.

34. There are several interrelated ideas that you now can explain to a newcomer all related to the term "trichromaticity".

35. Consider the following statement: "The color of light is something very different to a physicist or a hyperspectral sensor than it is to a human eye." You should be prepared to give a concrete example of how this statement is true.

36. One of the most fundamental practical concepts regarding color goes by the name "RGB Cube". Fortunately, you now completely understand what this term is describing.

37. As useful as the RGB cube is, it is not always the best way to think about color. An alternative color space is HSV. Having learned not only what HSV stands for but how it relates to the RGB cube you, should be prepared to answer questions that relate to mappings from one space to the other.

38. To exercise your knowledge of RGB color space, you would certainly be able to match up color codes with the 16 named colors established in the HTML 4 specification (Google “named Web safe colors”). In particular, the colors would be: White, Silver, Gray, Black, Red, Maroon, yellow, Olive, Lime, Green, Teal, Blue, Navy, Fuchsia and Purple.

39. At the heart of computer graphics lies 3 kinds of reflectance: ambient, diffuse (Lambertian), and specular. Your knowledge of all 3 is thorough and relates both to mathematical definitions and intuition with respect to identifying which of these are present in a given rendering of a scene.

40. Ambient illumination may be thought of in several ways. First, is an extraordinarily convenient

and simple hack. Second, a key element in realizing a believable rendering of many scenes. Third, a very coarse approximation to a physical phenomena common in some parts of the world.

41. Two of the three kinds of reflectance require explicit light sources, and one does not. Be clear about which is which.

42. One and only one of the 3 types are reflections require knowledge about the relationship between the viewer (camera) and the position on the surface whose illumination is being computed. Be clear about which requires this knowledge.

43. Consider if you would agree with this statement: "The essence of Lambertian reflectance is the way photons are reflected back in the direction of the light source with greater intensity/frequency than they are reflected off to the sides."

44. The computation of diffuse reflection from a point on a surface depends upon the intensity of the light source, the normal to the surface, and the vector indicating the direction from the point on the surface back to the light source. The exact equation and motivating diagram is something you can reproduce from memory.

45. There is one more thing that enters into computing diffuse reflectance, and is typically expressed using either 3 or 9 constants (K). What are these constants expressing in terms of physical modeling of surfaces, and why sometimes only 3 and other times 9?

46. For a light source located at a position in the world (x, y, z) and a point on a surface (sx, sy, sz), what is the vector L pointing back to the light source?

47. The sun is a light source typically modeled as being infinitely far away. Under such a circumstance, you should readily be able to work out the appropriate procedure to compute the direction from a surface point to the light source.

48. It is sometimes said of diffuse versus specular reflection, that one is deep while the other is superficial. While this might be a somewhat odd statement, there is a point to it. What is that point?

49. Of the 3 forms of reflection, specular reflection involves the greatest amount of required information. In particular, it requires a vector representing the direction to the light source (also required for diffuse), a surface normal (again same as diffuse), a reflection Ray R that was not required for diffuse illumination, and finally a vector representing the direction to the camera/viewer. Of these, the reflection Ray R is arguably the most involved to compute. Since it is a fundamental element in computing specular reflection, you of course understand the construction of how it is expressed relative to the other known elements.

50. It seems the parameter alpha crops up in many places in computer graphics. It plays a critical role in Phong specular reflectance. It is commonly assigned values such as 5 or 50, or even higher. It is very useful that you can explain precisely what alpha is doing.

51. There is an essential and very interesting, , aspect about specular reflection and its associated constant k_s. Namely, it is a scaler rather than being either 3 or 9 values. Given your understanding of specular reflection, you can certainly explain why only a single scalar value is required.

52. As is often the case in technical disciplines, the 2 terms "illumination" and "shading" are related but should not be confused. This becomes particularly true when you avoid confusing Phong illumination and Phong shading.

53. In lecture, it was suggested that in some circumstances it's best to tell a lie when it comes to the normal vector to a surface patch. Calling this a lie is a stretch, but the underlying issue is important. When is it better to lie about the surface normal, and why?

54. Interpolation is a fundamental operation that takes on great deal of practical relevance in the context of surface shading and texture mapping. Your understanding of linear interpolation should be rock solid and you should have no difficulty working out simple numerical problems involving interpolation of color values across a surface (typically along the scan line).

55. Bilinear interpolation in a texture map takes on a slightly different flavor. Given real valued coordinates in a texture map representing the precise placement of a pixel to be colored, you should have no difficulty working out the appropriate color value based upon bilinear interpolation between the 4 nearest pixels in the pixel map.

56. Interpolation of vectors is a somewhat distinct operation relative to interpolation of scalars. Demonstrating understanding of what it means to interpolate between vectors, be clear in your own mind about how to compute a vector C representing 25% of a vector A and 75% of a vector B.

57. In discussing texture maps it was convenient to emphasize that often transformations between coordinate systems are most easily solved for given matched points expressed before and after the transformation. An example using 3 point correspondences and a 2 x 3 transformation matrix was worked out in class. Similar problems involving 3 point correspondences should be no trouble to work out having now seen it done once.

58. Texture maps are most typically nothing more than images. Some are meant to look natural when they are tiled across a surface, and others are not. What does it mean for a texture map to look good when it is tiled?

59. In the context of texture maps, issues associated with representations appropriate for different levels of resolution come to the fore. In a MipMap (image pyramid), if the highest resolution is 64 x 64 pixels, you should be totally comfortable defining resolutions above this level.

60. It would be a lot to ask anyone to commit to memory the precise equations used to solve for the 8 DOF of freedom projective transform introduced in the context of texture mapping. It would not however be at all unreasonable to provide worked examples and inquire as to such things as the meaning associated with a four-point correspondence between 2 views of a rectangular planar face.

61. How polygon pipeline rendering architectures remove hidden surfaces through the use of a Z-buffer is arguably one of the most fundamental concepts anyone knowledgeable about computer graphics should have committed to memory.

62. Some of the perspective projection matrices used in this class are utterly unsuitable for Z-buffering. Others are not unsuitable and are instead completely appropriate. Be sure you're able to recognize which is which.

63. Not all polygons deserve to be rendered. One of the simples possible tricks to reduce the amount of work by half (in general) is called back face culling. It all boils down to the dot product, and how it boils down to a dot product is something you can readily draw and explain.

64. Sometimes in 3-D modeling for computer graphics being "two-faced" is not such a bad thing. Be ready to explain how come.

65. When you get down to it, the use of an alpha channel to create the appearance of semitransparent objects is not that hard to understand. In the lecture on hidden surface removal this topic came up and it is a good thing because now you will be able to explain it to others. Note, It is also explained in the textbook on page 66 and most likely in a myriad of different places on the web.

66. Many of the Components of the world to camera transformation developed for the perspective projection pipeline are readily recycled in the development of a ray tracing algorithm, and more particularly the process of the enumerating rays. What is lovely about this process, and which you should be readily able to explain to someone else, is the way in which pixels and their associated rays are enumerated in world coordinates without ever having to shift any of the scene model into camera coordinates.

67. Often, before asking if a ray intersects a bounded planar polygon in the 3-D world, one first asks where the ray intersects the infinite plane. Now that you are completely comfortable with the parametric form of a 3-D ray as well as the equation for an infinite plane, you can workout from first principles explicitly the solution for the intersection.

68. The general problem of determining whether a point lies within a 2-D possibly non-convex polygon first arises in our course in the context of ray tracing. The approach, often called the odd even parity rule, is elegant and should be something you can easily draw, explain, and illustrate by example.

69. For some simple 3-D polygons, in particular triangles, it is possible to simultaneously solve for the point of 3-D intersection and answer the question is the point of intersection inside or outside the triangle. The trick relies upon a very clever characterization of all points on the infinite 3-D playing as expressed by motion along 2 edges of the triangle. The mathematics for this computation is something you can now derive given your understanding of the parametric form of a ray as well as your understanding of non-orthogonal basis vectors.

70. There is a minor but important difference between 2 alternative ways of representing a ray. In one, less work is required by the code when moving from pixel to pixel to pixel. In the other, the t-values associated with the first intersection may be interpreted directly as distances into the world. Within this context, you should be able to precisely describe the 2 representations and highlight why the 2 behave differently.

Last third of the Semester.

71. You should be comfortable writing down the equation that defines the x coordinate of a cubic curve parameterized by t. Of course, likewise for y and z.

72. You should now be able to explain in broad terms what it means to relate a geometric specification

of a cubic curve to the actual coefficients defining the 3rd order polynomials.

73. We have studied in particular 2 types of cubic curves, each with their own particul way of specifying the curve's geometry. The 2 types are: Hermite and Bezier. You should be able to explain succinctly how the geometry for each is specified and how the 2 are related.

74. If shown a set of alternative specific geometric specifications for either Hermite or Bezier curves you should be able to match up those specifications with examples.

75. For this course you are not expected to commit to memory the precise form of the blending functions associated with Hermite, Bezier or B-Spline curves. That said, a basic understanding of the 3 types leads naturally to an ability to quickly recognize and distinguish between plots of the blending functions for each.

76. When joining together curve segments, in particular Hermite and Bezier curves, there are different orders of connectivity. There are 3 in particular that matter and for which each builds upon the previous. You should be able to explain in your own words each of these and how they successively require (impose) an additional constraint.

77. You are now intimately familiar with many of the details associated with the creation of a ray tracing system. For the most part, the final exam will not concentrate heavily on details associated with ray tracing. That said, since you have this detailed knowledge it should not come as a surprise that there may be questions testing this knowledge on the final exam. Here is one example of such a question: What is the difference between a ray casting system and a ray tracing system?

78. Here's another example of a general question regarding rRay tracing: What, if any, obstacles are there to massively parallel implementations of ray tracing?

79. In Ray tracing, how is the direction of the ray determined? When explaining this to someone it is necessary to be precise.

80. Proper creation of shadows using a perspective projection pipeline style system is actually rather tricky. In contrast, you're in a strong position to explain how the addition of shadows to a ray tracing system involves a minor retooling of capabilities already present in the system.

81. Two concepts are easily confused. The first is the field of view of a pinhole camera. The second is the resolution, or pixel density, of an image. In a well-built rendering system controls used to modify the field of view are clearly separate from the controls used to change the pixel density. Having now built your own rendering system, you're familiar with this distinction, and therefore should have an easy time explaining it to others and fielding questions relating these two concepts.

82. The distinction between a rational versus a nonrational B spline was only touched upon in this course and the mathematical details were not fully developed. However, the punchline associated with why so many within the computer graphics community find rational B-Splines so useful is a general idea that you can relate back to our earlier work with perspective projection matrices and it is something you should be able to explain to others succinctly.

83. There is a clever way of determining the intersection of a sphere and a ray. Since you have already implemented this as part of your programming assignments, you're aware that solving a general problem without the aid of a calculator would be annoying. That said, the construction of the algorithm is elegant and important, and you certainly could work out a concrete numerical

example if the constituent components of the example were well enough chosen to make ensuing arithmetic straightforward.

84. Being familiar now with the distinction between reflection and refraction, you should be able to demonstrate this understanding by precisely definining the distinction between the two. In addition, illustrating how different combinations of indices of refraction alter the direction of a refraction ray should be straight forward.

85. In closing, here's a high-level issue that you are now prepared to speak about in an informed manner. Some might argue that the success of 3-D graphics based on the perspective projection pipeline and polygon rendering is so firmly established that not only is it the dominant paradigm today, but it will remain the dominant paradigm for the indefinite future. Others might argue that the sun is setting on this paradigm and it will be superseded by the ray tracing paradigm. Think about how you would argue for both of these positions, since each argument has its merits.