CS 410, Fall 2020, First Third of Semester Review

The following is not an exhaustive list of what has been covered, and to be very explicit, the Midterms and Final may cover material discussed in class not been mentioned below. That said, hopefully the following will be a helpful as you review the broad topics and techniques we have covered so in CS410.

1. 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 time in the course means you're able to succinctly define each of these 3 mathematical concepts.

2. 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.

3. 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.

4. 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, when making this argument to another it is best to illustrate using concrete examples of commonly named types of 2-D transformations.

5. Homogenous coordinates introduce an ambiguity. There are an infinite number of ways to designate a 2D point (x,y) using homogeneous coordinates and only one of these is canonical. Given any sample from this infinite range of possibilities you may easily now re-express a 2D point in canonical form.

6. 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 drawn 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.

7. Perhaps one the odder aspects 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.

8. 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.

9. 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.

10. If one were to say there's something special about the cross product between 2 unit-length and orthgonal vectors, you fortunately could quickly explain what is special about this case.

11. 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, it 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.

12. Just as computer graphics folks don't typically like 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).

13. The geometric concept of a frustum is important to camera modeling. You are now able to explain this geometric construct in your own words and with your own simple drawings. Doing so includes the critical concepts of near and far clipping planes.

You are also aware that there is some redundancy in how cameras are specified. For example, an image plane doubled in size and moved twice as far from the focal point results in rendering the same image.

14. In ray tracing the first major formulation shoots one ray per pixel. As introduced in CS410, this means it is essential to be able to enumerate in 3D world coordinates the 3D position of individual pixels. Given a camera specification, you can derive the necessary formulas and associated code to accomplish this task.

15. Also using a camera specification, you understand and can explain the process of generating rays, one per pixel, given a camera specification.

16. You can inspect a camera specification and answer basic questions. For example, comparing two specifications, which will make an object appear larger.

17. In parameterizing a camera, the up vector plays a critical role in selecting the orientation of the camera. Be quick to understand the 'gravity' of the situation – and also this terrible joke.

18. There is a kind of word problem you can now easily solve: a space ship leaves earth in a direction U, how far has it traveled when it passes closest to a planet at position P. Assume the earth is the origin.

19. 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 between 380 nm to 780 nm.

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

21. 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.

22. 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.

23. 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 what HSV represents and how it relates to the RGB cube you, should be prepared to answer questions that relate to mappings from one space to the other.

24. Colors are often specified using hexadecimal numbers. For example, #800080 is purple. You should be comfortable answering questions about colors where the colors are specified in this notation.

25. You are comfortable with questions of the style: "Which of these 2 colors is more saturated?" Or alternatively, "Which of these 2 colors appears brighter?"

26. The use of an alpha channel to blend the color of one object with that behind it has now been carefully reviewed and you should be comfortable answering questions about alpha blending.

27. Three kinds of reflectance lie at the heart of computer graphics: ambient, diffuse (Lambertian), and specular. At this point in the semester you have a clear understanding of two: ambient and diffuse (Lambertian) illumination.

28. For diffuse, i.e. Lambertian, illumination you understand how to compute the illumination on a point on a surface given a light source as well as material properties for the surface normal.

29. Calculating where a ray and sphere intersect, and determining if they intersect, is an excellent time in computer graphics to introduce the general technique of mixing parametric and implicit forms in order to solve geometric problems. Consequently, you now are able to use the brute force approach to ray sphere intersection to illustrate in concrete terms this general approach.

30. 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 the ensuing arithmetic straightforward.

CS 410, Fall 2020, Second Third of Semester Review

1. It is hard to impossible to appreciate modern notions of high dynamic range (HDR) color without some grounding in the work on human perception of color as explored in the 1920s and 1930s and among other things our modern definition of Chromaticity and the associated Chromaticity Diagram. Fortunately, you are now literate enough in these matters to explain the original monochromatic matching experiments and the subsequent response curves and XYZ color space. Also, you can explain what this implies about possible limits in our most commonly used RGB color system.
2. Two of the three kinds of reflectance require explicit light sources, and one does not. Be clear about which is which.
3. 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 of the three types of illumination requires this extra information, i.e. direction to the camera.
4. For a light source located at a position in the world (x, y, z) and a point on a surface (sx, sy, sz), you can write down the precise values in a vector pointing back to the light source.
5. One might say of diffuse versus specular reflection that one is deep while the other is superficial. While this is admittedly a somewhat odd statement, it captures something of importance that you in turn can now explain.
6. 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.
7. It seems the parameter alpha crops up in many places in computer graphics. It plays a critical role in Phong specular reflectance and is commonly assigned values such as 5 or 50, or even higher. You can explain precisely what alpha is doing both in words and with an equation.
8. There is an essential and very interesting aspect of specular reflection and its associated constant(s) k. Namely, there are 3D modeling packages/languages where k is expressed as a scalar rather than a three-tuple (three scalar constants). Given your understanding of specular reflection, you can now explain why only a single scalar value is a good choice in many circumstances. Likewise, you can explain the flexibility introduced by using a three-tuple.
9. Sometimes in 3-D modeling for computer graphics being "two-faced" is not such a bad thing. Be ready to explain what such an off-handed remark actually means.
10. 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, the 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.
11. Given the direction of a ray defined by a vector arriving at a point, as well as the 3D coordinates of the point, and finally the surface normal, you are able to choose between a series of alternative possible reflection rays provided the choices are qualitatively distinct. In other words, incorrect choices may be ruled out based on some clear conflict such as pointing in the overall wrong direction.
12. Both diffuse and specular reflection have dot products as key to their computation. In both cases, a negative value for the dot product has important geometric implications and also requires explicit handling in code. As you implement a ray tracer, and when you go to explain the process to others, you are comfortable now with the meaning and proper handling of those situations where the dot product yields a negative number.
13. For recursive ray tracing, two distinct concepts have been introduced in general terms as well as precisely in SageMath code. These two concepts are attenuation and reflectivity. Having experimented with both in the SageMath notebook on recursive ray tracing discussed in lecture, you are comfortable with how each operates and how they are distinct from each other.
14. With all other camera and scene parameters left constant, you are now comfortable explaining what happens when recursion depth is increased. In particular, you can correctly argue the following proposition: with increased recursion the intensity of any given pixel must remain constant or increase.
15. You should be able to describe the high level and key distinctions between the projective pipeline versus ray tracing approaches to rendering 3D scenes.
16. Given its relative simplicity when compared to perspective projection, orthographic projection can be easily overlooked.

However, you understand some of the important contexts in which it is most useful and you can define orthographic projection by writing down an example of an orthographic projection matrix.

17. A predator endowed with orthographic vision (orthographic projection) is not 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.
18. In the lecture introducing perspective projection, the first perspective projection matrix was derived for the case where the image plane is moved an amount d in front of the focal point. The second formulation placed the image plane at the origin with the focal point behind the image plane by an amount d. The corresponding perspective projection matrices look similar; swapping 1s and 0s between two positions yields one from the other. However, they behave differently and understanding these differences is important.
19. Some people in computer graphics 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's argument. (It would be good to ask about this item in our review session since the issue has probably not received the attention it deserves.)
20. The terms "frustum" and "canonical view volume" are used nearly interchangeably when describing the perspective projection pipeline. Both terms now make sense to you, and you should be able to explain them in your own words and with your own simple illustrations.
21. A complete working model of the perspective projection pipeline has been presented using SageMath. Since you have experimented with different camera specifications using this notebook, you can associate how an object – in our case a simple house – appears in the canonical view volume based upon alternative camera specifications.
22. In the perspective projection pipeline SageMath notebook you have a complete symbolic presentation of the transformations used to transform 3D world coordinates into 2D vertices on an image plane. Therefore, you can easily recognize and reason about the significant components in this transformation.
23. In the traditional rendering pipeline, mapping polygon vertices into the canonical view volume is only the first step in a two-step process. That second step involves painting pixels based upon those polygons in such a way that hidden surfaces are not rendered. At the highest level, this means you can distinguish between the manner in which a ray tracing algorithm is pixel driven where a rendering pipeline his polygon driven.
24. Cohen and Sutherland introduced a very elegant algorithm for quickly determining if a line segment needed to be clipped to the image plane and also where it needed to be clipped, i.e. against which side of the four-sided image plane. While arguably not in common use today, this is an elegant algorithm that you should nonetheless understand. It stands in for a myriad of more modern techniques where cleverness leads to efficiency.
25. It may be very mechanical, but being able to determine the precise endpoint of a 2D line segment resulting from clipping against the rectangular image plane it is something you can do and do quickly. Keep in mind this is a special case of the problem that arises when intersecting two parametric line segments. Consequently, it has a simpler solution.
26. The odd-even parity rule often that defines what it means for a pixel to be interior to a polygon and also represents a key aspect of how polygon filling algorithms our encoded. As part of your basic literacy and computer graphics you understand this rule and can use it to answer questions about whether an example is or is not correctly filled.
27. You can be grateful that you are almost certainly never going to be called upon to write a scan conversion algorithm that fills pixels given the vertices of a polygon. That said, understanding scan conversion is important if for no other reason that it represents one of those places in computer graphics where an understanding of actual pixel geometry along with the interaction of multiple polygons becomes important. It is also important insomuch as processes like rendering with a Z buffer don't make sense unless you first understand what rendering looks like at the pixel level.
28. Given that we covered polygon filling the semester, you should be able to answer questions about which pixels are in which pixels are not filled following the basic algorithm introduced in lecture 18.
29. In modern realizations of the rendering pipeline actual code is written to shade individual vertices and even pixels. Unfortunately, the complexity in degrees of freedom open to the programmer preclude our studying these shaders in any detail. However, prior to the introduction of shaders, shading could be counted upon two arise from a hand full of well understood algorithms. Three that you should understand well our: flat shading, Gouraud Shading and Phong Shading.
30. Interpolating intermediate values while moving across a polygon comes up again and again as a fundamental operation in shading polygonal fragments using traditional perspective projection pipeline rendering engines. It also so happens that in the context of ray tracing surface normal interpolation is important. Fortunately, you understand how each vertex of a triangle may have a distinct surface normal and also how to use our existing ray- triangle intersection code to rapidly compute a smoothed normal at any point on the triangle.
31. You now understand the value of a class of approximation algorithms which employ random sampling to create better and

better solutions to a problem. A good example is to calculate PI, and you could sit down easily and write code (python, etc.) to implement an approximate solution given nothing more than a uniform random number generator and the recollection that the area of a circle is PI*r**2.

CS 410, Fall 2020, Last Third of the Semester Review

1. You are comfortable with models of illumination that include both reflection and refraction. Refraction in particular is interesting because of the manner in which light passing through a solid is bent as defined by Snell's Law. Perhaps fully memorizing Snell's Law is unnecessary. However, being able to describe how light paths change upon transition between one material to another based upon indices of refraction must now come naturally.
2. There is a notable visual aid in thinking about refraction through a glass sphere based upon a photograph taken by Kelsey Rose Weber (Lecture 20 Slide 9). You will want to hold onto that image in your mind and be able to call upon it when thinking about refraction and explaining it to others.
3. You were not responsible for coding refraction through a polygonal solid in CS410. However, you are responsible for understanding some of the key elements of polygonal solids and refraction. Specifically, the idea of interpolating surface normals was discussed in lecture to 'smooth' the surface and you can at least explain the surface normal interpolation in broad terms. You also understand the impossibility of refracting through a bunny with holes (and you can generalize the problem from this description).
4. Ray tracing through a sheet of plate glass using refraction is possible and also boring. Knowing why it is boring is one side benefit of having worked through the geometry of refraction.
5. In CS410 you have not been asked to cast 'partial' shadows through semi-transparent objects. While you are therefore freed from the responsibility of committing to memory an algorithm for doing this type of light-through-objects calculation, you can easily explain at least one reason that doing this properly in the context of ray tracing is difficult.
6. Refraction through a sphere is a specialized and simplified case of general refraction through objects. One hallmark of this simplification is a specialized algorithm for finding the exit point for a refraction ray given the entry point. You can derive this specialized computation from first principles given the point where the ray is entering and the direction of travel through the sphere.
7. In Lecture 21 and the accompanying SageMath notebook a series of examples using a semi-transparent sphere with materials of differing indices of refraction are rendered. Through study of these images you can make relative judgements about the magnitudes of indices of refractions for spheres of differing materials based solely on how they bend light and hence behave in multi-object rendered scenes.
8. As is common in graphics, we go to great length to avoid ever calling on trigonometric functions in rendering code. Consequently, you now are familiar with at trigonometry free means of computing the direction of a refraction ray for light entering a material. The details in the algebra are complex enough that committing them to memory is less then worthwhile. However, given your understanding of the derivation, you are now able to recognize if a derivation being presented to you is correct.
9. Ray tracing with reflection and refraction implies that in a scene full of semi-transparent objects recursive calls may be modeled using a binary tree. You can use this mental-model to explain aspects of how your refraction enabled recursive ray tracer behaves.
10. Yet another material property is introduced when refraction is introduced; the opacity constant. Understanding how this constant, often abbreviated 'ko', means you can match up alternate scene renderings with different possible 'ko' values – provided the change in 'ko' is sufficient to be visually distinct.
11. Shadows are a shadowy concept in the presentation materials for CS410; often being nearly hidden from view. There is a reason for this shady treatment: shadows are easy for ray tracers. Just because not much gets said does not mean shadows are uninteresting or unimportant, and you have no trouble recognizing when they are present or modifying code to enable shadows should someone else have forgotten to do so.
12. Bicubic Parametric Curves are a fist taste of a rich family of curve and surface representations. Thanks to the Lectures at the end of this semester and the accompanying SageMath Notebooks you have a complete and thorough algebraic and geometry understanding of Hermite Curves, Bezier Curves, Bezier Surface patches, B-spline Curves and B-splines. Just for example, you would have no problem explaining to another person the geometric and algebraic setup for these shapes.
13. Often the CS410 instructor goes off on tangents, but seldom are tangents more on-topic than when introducing Hermite Curves. Indeed, with Hermite Curves, there is an entire physics metaphor involving time, direction of travel and speed, that

can proof extremely helpful in developing an intuition for how these curves behave. Enjoy the fact that you can now work with this metaphor and even use it to think through some of the most common constraints one might want to impose when joining one curve to another.

14. Given the numerical setup of a curve or surface you could match it with a graphical illustration of the same curve provided the alternatives are reasonably distinct.
15. Given the Hermite, Bezier and B-spline blending functions, you could tell which is which. More important, you understand what is special about Bezier blending functions and why it matters. Likewise, you understand what is special about B-spline blending functions and why it matters.
16. The idea of de Casteljau Curves is extremely useful. While understanding the construction may in no way ever change how your write a single line of code, it will at some point prove invaluable for better understanding the nature of Bezier Curves. Because it is so important, be clear in your ability to explain these ideas to others.
17. In Lecture 24 a distinction was made between the dimensionality of a geometric object such as a curve or surface and the embedding dimension. Hang onto this distinction and be ready to use it when explaining curves and surfaces to others.
18. The notion of degrees of freedom plays an important role in parametric curves and surfaces. So, for example, you can now discuss the degrees of freedom intrinsic to a 3D Bezier curve.
19. Bezier curves and surfaces must pass through some control points and not others. You now understand which are which.
20. B-spline segments, while never required to pass through control points, are designed to work together when using more a larger sequence of control points. You are now able to work out how successive segments are blended together to form longer spline curves.
21. When drawing shapes in many common drawing packages, PowerPoint comes to mind, many of you have seen ‘wings’ attached to points in which growing, shrinking, and rotating these wings changes the curve. While before this was all magic, now you see this instantly as an instantiation of Bezier Curves.
22. While NURB surfaces received scant real treatment in CS410 this semester, there is one truly important property of NURBs that arises in the context of traditional pin-hole camera rendering. Make sure you at least are aware of this property and why it is of such practical importance.
23. Monte Carlo Ray Tracing has a huge computational disadvantage relative to what you are doing in CS410. However, it has a huge advantage in terms of capturing indirect lighting effects. You are now comfortable explaining why Monte Carlo Ray Tracing as presented in lecture 17 captures indirect lighting, both mathematically and by example in a rendered image.
24. In Monte Carlo Ray Tracing as presented, be aware that not all randomly selected unit length vectors used to simulate diffuse illumination are equally likely. Also, be able to explain why they are not, i.e. how are they generated.
25. In Monte Carlo Ray Tracing as presented, when 100 samples are used per pixel, there will be 100 distinct random directions chosen for approximating diffuse illumination on the first object hit. If mirror reflection is included for this object, how many distinct directions are used to initiate reflection? Presuming these hit another non-mirrored surface, be ready to answer whether the same red, green and blue values are always returned.