CMSC425 Midterm 1 Prep

The midterm on April 1st will be in class, closed book. There will be 5 to 6 questions, up to seven pages, with the first question short answer and the rest applications of the concepts. Questions from the homeworks are fair game, as are questions from lectures before spring break and the practice midterm exams from spring and fall 2018. Question on Unity will be limited, and based on what you should have learned in Project 1.

This class is not for you to learn all the algorithms and methods used in game programming, but to learn enough so you can pick up more on your own. Key is "vector think" – using vectors and vector operations for geometric computations and algorithms in efficient, compact and robust ways. Rather than the standard equation y=mx+b for a line, use the more robust p(t)=p+t*v. Rather than compute the angle between two vectors to determine if it's less than or greater than 90 degrees, compute the dot product and check its sign - much faster than inverse cosine. Vector think includes working with polygonal shapes. And it includes the following, and more:

• Using parametric curves as a function of t rather functional curves as function of x
• Using 3D Affine transformations on points and vectors
• Affine and convex combinations of points and vectors
• Homogenous coordinates and matrices to represent data and operations
• Avoiding computing cos and sin directly when you can use dot and cross products
• Using normal a lot
• Knowing when you need to normalize vectors (when you need accurate lengths)

Possible concepts and questions include:

1. Basics of Unity game loop (Day2)
2. Difference between input by polling and events (Day2)
3. Concept of game object list (Day2)
4. Unity Entity-Component structure (Day2)
5. Model View concept of objects (Day2) (but not full MVC)
6. Difference between a point and a vector (Day3)
7. Basic affine vector operations of scaling, addition, difference, point-vector addition (Day3)
8. Point-vector representation of a line, line segment and ray as function of t
9. What it means for a vector equation to be coordinate free (Day3)
10. Midpoint of a line or shape and other convex combinations of a set of points (Day3)
11. Magnitude of a vector, and vector normalization (Day3)
12. Dot product of two vectors (Day3)
13. The cosine law formula (Day3)
14. Using the cosine law formula to compute the angle between two vectors; to compute the cosine; to do a forwards/backwards test on the sign of cosine. (Multiple days)
15. Relative to (10), using lerp (Day4) and tweening (Day5)
16. Computing the perpendicular bisector of a line (Day5)
17. Midpoint displacement algorithm (Day5)

18. Compute orthogonal projection of a vector onto another. (Day5)
19. Compute orthonormal projection (Day6) and know the difference.
20. Know what it means for two (or three) vectors to be orthonormal (Day6)
21. Compute intersection of circle and ray (Day6)
22. Compute distance from point to line or plane (Day6)
23. Compute cross product (Day6)
24. Using the sin rule for cross products to get area and sin (Day7)
25. Represent 2D and 3D points and vectors in homogenous coordinates (Day7)
26. Computing orthonormal frames of reference from two vectors (even if not orthogonal themselves) (Day7)
27. Define, use and give homogeneous matrices for the six affine transformations (Day7,8)
28. Define and use angle-axis rotations (but not quaternions – see below) (Day7,8)
29. Solve problems with intersections of cones, lines, spheres, planes, cylinders (Day9)
30. Solve problems with projectile equations (Day9)
31. Solve problems that have to do with angle and sign of angle between vectors, frames of reference, and planes (Day9)
32. Define and illustrate the five main collider types given in lecture (Day9)
33. Give equations/algorithms for finding collisions between common shapes. (Day10)
34. Describe how to use data structures to efficiently find collisions (Day10).
35. Work with the (forward) kinematics of skeletal figures, translating between their coordinate systems, and computing the position of a point on the skeleton. (Day10)
36. Know what a metajoint is and how it would be used in this context (Day10)
37. Describe what skinning/rigging is. (Day10)
38. Describe problems with simply skinning (Day10)
39. Use weighted linear blending to solve skin cracking (Day10)
40. Describe navigation problems, including for articulated shapes (Day11)
41. Describe how a game designer might add waypoints to solve navigation (Day11)
41. Describe at a high level the use of a NavMesh in navigation (Day11)
42. Describe and apply at a more concrete level the steps of NavMesh construction (Day11) including polygon smoothing and triangulation, and adding waypoints to the resulting triangulated NavMesh.

Compute
Questions will *not* include:
1. Quaternions and rotation interpolation.
2. Fractal shapes
3. Writing Unity code (maybe reading clear examples)
4. Too much about how motion animations are stored as joint interpolations (Day10)
5. Problems in the practice exams that don't relate to the items above.