CMSC425 Midterm 1 Prep

The midterm on October 24th 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, the midterm from spring 2019, and the practice midterm exams from spring and fall 2018. (Any questions specficially excluded will be listed at the end here.) Questions 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
2. Difference between input by polling and events
3. Concept of game object list
4. Unity Entity-Component structure
5. Model View concept of objects (but not full MVC)
6. Difference between a point and a vector
7. Basic affine vector operations of scaling, addition, difference, point-vector addition
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
10. Midpoint of a line or shape and other convex combinations of a set of points
11. Magnitude of a vector, and vector normalization
12. Dot product of two vectors
13. The cosine law formula
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.
15. Using parametric equations to define a line, circle or other 2D or 3D curve.
16. Computing the perpendicular bisector of a line

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

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

Links to previous practice and actual exams are on these pages:

Spring 2018:
https://www.cs.umd.edu/class/spring2018/cmsc425/handouts.shtml

Fall 2018:
https://www.cs.umd.edu/class/fall2018/cmsc425/handouts.shtml

Spring 2019:
https://www.cs.umd.edu/class/spring2019/cmsc425/handouts.shtml