Spring ’16 CIS 410/510 Midterm 2 Review

You may bring two pages of notes, front and back.

Questions will be in short-answer format with partial credit for partial answers.

You will be asked to write pseudocode resembling Mono (the C# variant used by Unity).

You may assume that you have access to a vector math library that supports vector addition, vector subtraction, vector-scalar multiplication, and basic scalar functions (e.g., sqrt, pow, sin, cos, arccos).

You may also assume that have access the following functions:

```
float dot(Vector3 u, Vector3 v);
Vector3 cross(Vector3 u, Vector3 v);
```

Topics:

- Vector-vector addition and subtraction, vector-scalar multiplication
- Dot and cross products, scalar projection
- Linear interpolation
- 3D sound - listeners, emitters, Doppler effect, occlusion/obstruction
- Digital Signal Processing (DSP) - uses for reverb, pitch shifting, compression, low-pass filtering
- Rays and line segments, uses for ray casting in games
- Sphere-sphere, AABB-AABB collision implementations, AABB from points
- Euler integration equations from linear accelerations to velocities to positions
- Path nodes and navigation meshes, Euclidian, Manhattan distances
- Greedy best first and A* path finding algorithms (not implementations)

1. [10] Describe three features in games that could be implemented using ray casting:

2. [15] Consider the following function:
   ```
   Vector3 midpoint(Vector3 r, Vector3 d);
   ```
   Implement the above function so that it returns the midpoint of the line segment starting at \( r \) and traveling in the direction of \( d \):

3. [20] Consider the following function:
   ```
   void aabb(Vector3[] points, out Vector3 min, out Vector3 max);
   ```
   Implement the above function so that it configures the min and max vertices defining an Axis Aligned Bounding Box (AABB) exactly encompassing the specified points.

4. [10] A* search relies on maintaining a collection of “open” nodes and being able to efficiently add nodes to this collection and find the node with the least heuristic distance to the goal. What data structure should be used to store the collection of “open” nodes? Why?