Tan, Jaim-kia T.

Programming Exercise 01 - Vector Math Review

---

1. (5pts) You are tasked to make a top-down shooter game where a worm-like boss is lined with two turrets that shoot a player. Turrets will only shoot whenever a player is within its line of sight, but the two turrets are faced perpendicular to the boss, one to the left and one to the right. The turrets' line of sight covers the entire sides of the boss.

Given that the player starts the boss fight facing the boss head-on, how would you program the turrets to shoot the moment the player steps into their line of sight?

Give/Define your own vectors that are useful to the given problem. Show the equation/s that you will use and how would you use these equation/s.

Solution:

For each turret, two vectors must be involved. The normal vector of the side of the boss where the turret is: $\vec{n}$, and the vector from the turret to the player $\vec{v}$. the normal vector $\vec{n}$ is simply a vector formed perpendicular to the worm-like boss' side depending on which turret it is (left or right turret). Meanwhile the vector from the turret to the player $\vec{v}$ is just the calculated vector resulting from the subtraction of their positions via Destination - Source.

With these two vectors, we would first compute their normalized vectors by using the formula of the vector divided by its magnitude. This isolates the direction from the magnitude, which is needed for the next step.

$$\hat{\vec{a}} = \vec{a}/|\vec{a}|,\ |\vec{a}| > 0,\ |\vec{a}| = \sqrt{a_1^{\ 2} + a_2^{\ 2} + \ldots + a_n^{\ 2}}$$

After normalizing, we take the unit vectors and get their dot product, with the following formula: $\vec{n} \bullet \vec{v} = n_1 v_1 + n_2 v_2 + \ \ldots\ + n_n v_n$. In a practical application of this, the vectors would be updated in real time depending on the position of both the boss and the player. Since the line of sight of the turrets cover the **entire** side, if the dot product is in between 0 and 1 (inclusive; $0 \le \vec{n} \bullet \vec{v} \le 1$), then the player is in the turret's line of sight. This is because the player would be directly to the side (but is still considered to be in the turret's line of sight at 90 degrees from the middle which is when the dot product is 0 and when it is 1, the player is directly in front of the turret.

---

2. (5pts) Your team is tasked to create a horror game, which features stone and marble statues that move when the player is NOT looking at them (think the weeping angels from Doctor Who). The only time that a player is considered to be looking at a statue is if it faces the statue within 90 degrees of the player's line of sight. How would you go around and code the behavior of the statues and how would you check if a player would be looking at a statue or not?

Give/Define your own vectors that are useful to the given problem. Show the equation/s that you will use and how would you use these equation/s.

Solution:

Considering that the player is considered to be looking at a statue within 90 degrees of a player's line of sight, the boundary is 45 degrees left and right of the center. In solving this problem, we could do a similar approach to the first problem but changing the boundaries in terms of the maximum number the dot product can be. Reusing the variables from the first problem, we can let the middle of the line of sight of the player be represented by a vector $\vec{n}$, and the vector formed from the player's position (Source) to the statue (Destination) which can be represented by the vector $\vec{v}$. From these vectors, we normalize them so that we can use this formula: $\vec{n} \bullet \vec{v} = cos(\theta)$. The dot product is a good representation of the angle between two vectors. When it is 1 $(\theta = 0^o)$ they are overlapping each other and when it is 0 $(\theta = 90^o)$, they are perpendicular to each other. Thus we simply compute for $cos(45^o)$ which is $1/\sqrt{2} \approx 0.7071$ to be our new threshold instead of 0.

In conclusion we can program the statue to move when the dot product between the two vectors (that update in real time based on their positions and the player's line of sight) is **outside the range of** $0.7071 \leq \vec{n} \bullet \vec{v} \leq 1$.