Comp 396 Final Report
Quantifying Threat in First Person Video Games

Thursday, April 11, 2015

Supervised by Clark Verbrugge

George Macrae 260401458

Contents

Introduction

Many modern video games feature different modes of difficulty such that the game is challenging and interesting to a variety of players; however, since difficulty is an abstract and often subjective concept, quantifying difficulty is non-trivial. Often, game factors such as difficulty are defined with by the intuition of the game developers and then tested extensively through human trial. This practice of beta-testing is an expensive, time consuming and non-standardized way of measuring important game mechanics.

In hopes of developing standardized measures of game factors like difficulty, this project aims to provide a tool that can analyze game environments such that player's experiences can be quantified. In order to do this, a playable game was first designed and built. The game was modeled to be a simple three dimensional stealth game, where the player plays in a first person camera view. The objective of the game is to navigate the maze while avoiding a set of guards. After a level is completed, player and guard path traces are saved such that they can be analyzed by the program during a replay.

The replay analysis program focuses on measuring the danger that the player is aware of and predicting the player's path as they complete the level. Comparing perceived danger with player's deviation from the predicted path allows the study of how players react to different situations. This project proposes two metrics to quantify threat that a player perceives; a distance danger metric, and an angle danger metric. The danger metric calculates threat with an emphasis on the distance between the guards and the player. The angle danger metric calculates threat with an emphasis on the angle between the front of the guard and the player. Path prediction is done using a dead reckoning algorithm. Dead reckoning bases the future position of the player based on the current and past positions of the player.

The limited experimentation supports that the measures of danger quantified by the danger metrics are accurate representations of threat that a player feels in the implemented game environment, but there needs to be further human trials for meaningful conclusions. Developing tools that such as this one can help developers measure a player's perceptions at different points in time of their games and thus can build more interesting and versatile games with greater ease.

Related Work

This project, in many respects, is a continuation of Jonathan Tremblay, Pedro Andrade Torres and Clark Verbrugge's Measuring Risks In Stealth Games [1]. Measuring Risks In Stealth Games studied the path choices a player may take in a two-dimensional top-down stealth game based on very similar metrics, one based on distance, another based on line of sight, and a third on "near misses" of being detected. The results found indicate that the metrics were accurate representations of human perception of risk in the respective game environment. The main difference between this project and the work of Jonathan Tremblay, Pedro Andrade Torres and Clark Verbrugge is the different game environment. The game developed here is three-dimensional whereas the game in Measuring Risks In Stealth Games is two-dimensional point of view. In addition to changing the player's view of the game, graphical representation of the enemy's field of view is also removed. By limiting the players view and removing graphical representation of the enemy's field of view, the player's knowledge of the game environment is decreased significantly and now must rely more heavily on their perception of threat to complete a level. Hence, the danger becomes a more significant factor when quantifying level design and therefore this environment further validates the danger metrics.

Methodology

Overview

First, several implementations of a simple game had to be designed and built. Each game implementation, or level, consists of maze with a start and end point. Each level contains a number of guards with predefined patrol routes, behaviour, and field of view. The goal of the game is for a player to navigate the level without

Methodology continued on next page...

entering a guard's line of sight. In other words, the environment models a simple three dimensional first person stealth game, in which you navigate to a goal while avoiding guards' detection. The player's path is saved so that player's behaviour can be analyzed after the level is completed.

In this game, a difficult level would be one in which a player must be constantly fleeing and avoiding the guards' detection. Obviously, the amount of guards directly relates to the level of difficulty, but more importantly is the guards' placement and their behaviour. For example, four guards placed in the corner opposite the player and facing the wall do not provide much of a threat, but two guards that actively move throughout and patrol a level ensure more of the playable ground is within the guards' sight, making the game more dangerous. To most realistically measure the danger of the level, two danger metrics are proposed - a distance danger metric, and an angle danger metric. Both measures take into account not only the number of visible guards, but also the distance between the player and the guards and which direction they are facing with respect to the player.

Once we have the game traces and the metrics to measure the associated danger, we study how danger affects player behavior and ultimately the difficulty of the game. The game traces record information including the position and rotations of the players and guards, as well as the guards' field of view, and the distances and angles in relation to the player. When replaying the level, this information is loaded in such a way that a previously played game can be rewatched at different frame rates. As the game is replayed, the danger at points of their path are measured and superimposed as danger meters onto those positions.

The replay mode also uses a dead reckoning path prediction method to predict where a player's next positions will be based on the player's current and previous positions. The danger at the player's predicted positions is calculated as well. This is so that a player's actual path can be compared with their predicted path and its associated danger. Through experimentation in using correlations between a player's path and their perceived danger, we can study how much a player deviates from predicted positions.

Danger Metrics

The player has a first person line of sight, meaning they have a limited viewing angle of what is in front of them. In other words, a player does not have complete knowledge of their surroundings, and so the danger that a player detects may not be completely accurate to what their actual danger may be. However, since the player behaviour based on their perception is of interest, the danger they perceive is what is measured. The measurement of the danger is therefore based on the amount of enemies within a players' line of sight. There are three factors that determine perceived danger: first, the number of guards, second, the distance between each guard and the player, and thirdly, the angle of player relative to the front face of the guards.

The two calculated danger metrics are the distance danger metric and the angle danger metric. The distance danger metric assigns a greater weight to the distance between the guards and the player, and the angle danger metric assigns a greater weight on the angle between the front of the guard and the player. For both metrics, a total danger value is calculated by summing the danger values calculated for each guard within sight. Danger is not linearly related to the number of visible guards. Instead, the danger value for each enemy is sorted by decreasing danger. The total danger is the sum of each danger value divided by its index in the list (the most dangerous guard's value is divided by 1, the 2nd most dangerous guard's value is divided by two, and so forth). This allows the most dangerous guard to contribute most to the danger value, with each successive guard contributing less. The reasoning behind this is simply that if there are two or more guards, you may not necessarily be in twice or more danger. For example, if there is guard just behind another guard, much of their line of light overlaps, so the total area that needs to be avoided is not twice as large as a single guard's line of sight. To see how this implemented see Listing 1 in the Appendix.

The danger metrics consist of an angle danger term and a distance danger term. Angle danger is based on formula 2. Angle danger is between 0 and 1 such that it will equal 1 if the guard's forward vector is within 45 degrees of the player and then decreases to 0 as the guard rotates away from the player in either direction. Notice that the angle danger term is not a linear relationship. This simultaneously makes the

Methodology continued on next page...

value half the maximum value once outside the threshold, and make the rate of decay slower as the angle increases. The reason for this is once a player is out of the guard's field of view (which they do not know but 45 degrees is a reasonable guess), then the danger drops by half the maximum value. After that the rate of decay is exponential, decreasing more significantly at smaller angles. This is because when the player is behind the player, the angle matters less than the fact they are behind the player and so the value should change less dramatically. The distance danger term is calculated by using the safe distance constant and dist, the actual distance between the player and the guard. Safe distance is a predetermined constant such that if the player is a safe distance distance away from the guard, the guard is of little to no concern to them and the term equals 0. Distance danger is calculated according to formula 2. Notice that danger associated with distance is not modeled as a linear relationship. This allows the distance danger to decrease slower when the player is close to the guard. This is because if the player is with a guard's sight at two different distances from a guard, the threat should not be significantly different as they are both points of imminent danger. Distance danger is also normalized between 0 and 1. It is equal to 0 if the player is safe distance away and increases as the dist decreases. (See Listing 2 and 3 in the Appendix).

$$Angle\ Danger = \begin{cases} 1 & \text{if } angle < 45 \\ -\frac{\sqrt{angle}}{\sqrt{180}} + 1 & \text{otherwise} \end{cases} \tag{1}$$

$$Distance\ Danger = -\frac{MIN(dist^2, safedistance^2)}{safedistance^2} + 1 \tag{2}$$

The difference between the two metrics is how the two terms are added together. For the distance danger metric, the distance danger term is multiplied by two before addition. For the angle danger metric, the angle danger term is multiplied by two before addition. The sum in both cases is then normalized between 0 and 1 (by being divided by 3) and then multiplied by a predetermined max danger value. This ensures that both metrics are normalized between 0 and the max danger value.

$$Angle\ Danger\ Metric = \frac{((2 * Angle\ Danger) + Distance\ Danger)}{3} * maxDangerValue \tag{3}$$

$$Distance\ Danger\ Metric = \frac{(Angle\ Danger + (2 * Distance\ Danger))}{3} * maxDangerValue \tag{4}$$

Replay and Analysis

The level analysis happens during the replay of a game trace. During a replay, the program calculates the danger metrics, draws the player's path onto the level, and draws projected positions, as well as the danger values at the projected points.

As the program replays a game trace, the danger a player perceives is calculated every frames per meter (a predetermined variable that is set to 10 by default) frame. A green and a cyan bar are created on the player's position at that frame, representing the distance danger metric and the angle danger metric, respectively. The heights of the bars represent the danger values calculated at that frame. See Figures 1 and 2.

Methodology continued on next page...

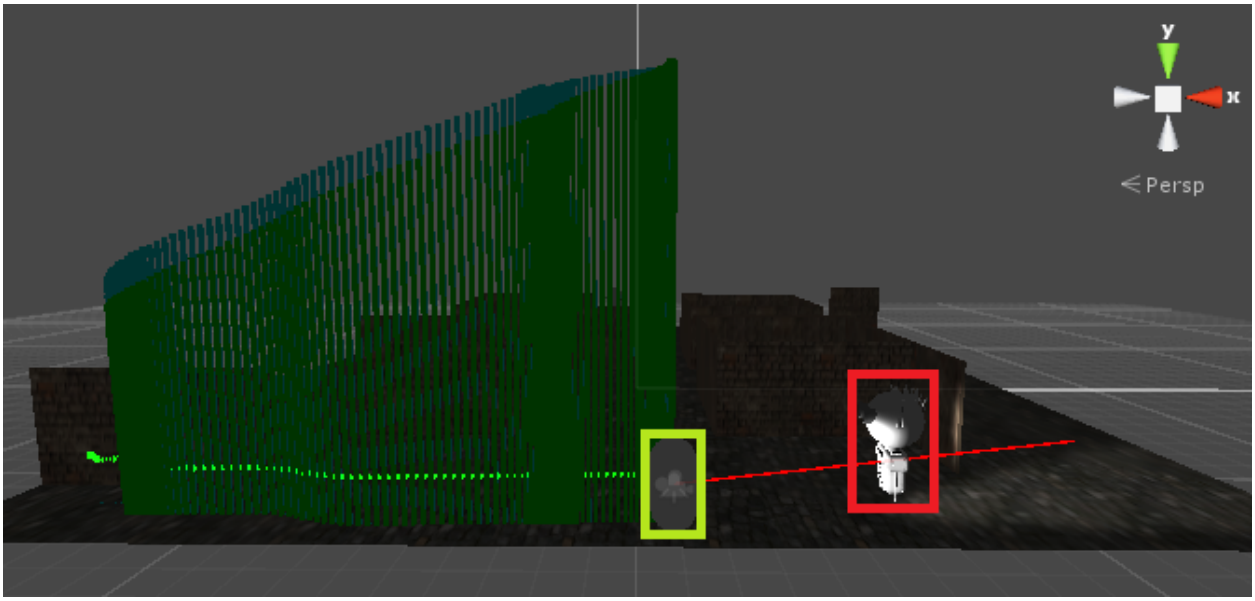

Figure 1: The player in a losing situation. The green box is the player, the red box is the enemy, the red line is the player's forward vector, the green bars represent the angle danger metric and the cyan bars represent the distance danger metric.

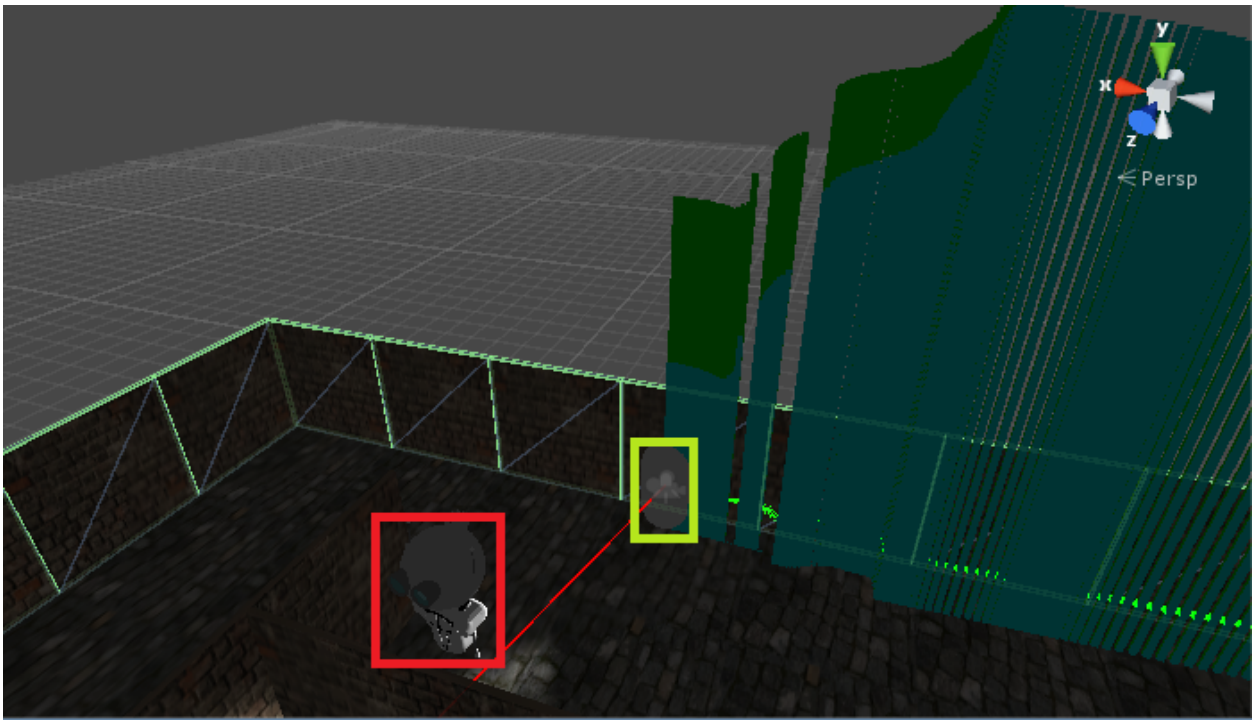

Figure 2: The enemy turns and moves away from a player in danger and the metrics decrease accordingly. The green box is the player, the red box is the enemy, the red line is the player's forward vector, the green bars represent the angle danger metric and the cyan bars represent the distance danger metric

A parameter path predictor is used to change the "Dead Reckoning (DR) algorithm that estimates future movements based on past behavior..."[2] (A. F. Sellem, H.A. Hussein, I. Talkhan). The Dead Reckoning path projection algorithm is used in this project. Every path predictor frames the player's position path predictor frames into the future is predicted. The algorithm creates a new vector with direction and

magnitude from the player's position path predictor frames from current frame to the player's position at the current frame. That vector is added to the player's position at the current frame as the projected position path predictor frames into the future. The predicted paths are represented by blue lines drawn on to the game trace.

To see if the player is avoiding danger, danger was measured at the predicted positions. Since enemy paths are fixed, the positions and rotations of the guards at the projected time are simulated so a danger at the projected point can be determined. However, when calculating the projected danger, not only the guards who are in sight of the projected player were considered, but also the guards who were within the player's sight at their current position. The reasoning for this is because we are predicting future actions of the player, we should consider factors that might affect those decisions. Guards a player can see at the current position may not be not be visible at the projected position and therefore not considered when calculating the danger metrics. This would not be an accurate measure of projected danger as the player knows there are guards since they are visible at the current position of the player, and presumably, they want to avoid such guards. The future danger values are also represented in bars on the map; red is for the distance danger metric and magenta is for the angle danger metric.

Results

To show that the proposed metrics were are accurate representations of perceived threat within the game environment a simple experiment was set up. A player was asked to play through 6 different set scenarios. The player was asked to give a rating from 0 to 10 of how much danger they felt they were in at certain points of interest throughout the levels. The player's ratings were then compared with the recorded metric values. The observed results support the claim that the proposed metrics are accurate (to varying degrees for different situations) representations of perceived threat.

The set scenarios were: the player was near a guard with different rotations, near two guards, far from 1 guard, and far from two guard. Each scenario was measured at 3 different rotations of the guards: 180 degrees, 90 degrees, and 0 degrees. It was observed that the distance danger metric was on average 14.56

The metrics were further validated by comparing the discrepancies between the projected path in the trace and the actual path with respect to the danger metric values, since this would suggest players are changing their path to avoid danger.

It is shown that there may be correlation between projected danger and deviation from the projected path. First, the path prediction by dead reckoning is shown to be an accurate projection of the player's path. As can be seen in Figure 3, when path predictor is set to around 25 frames, the dead reckoning path is nearly identical to the actual path. When the frame rate is increased, the discrepancy between the actual path and projected path becomes greater, especially when the player makes quick or sudden turns.

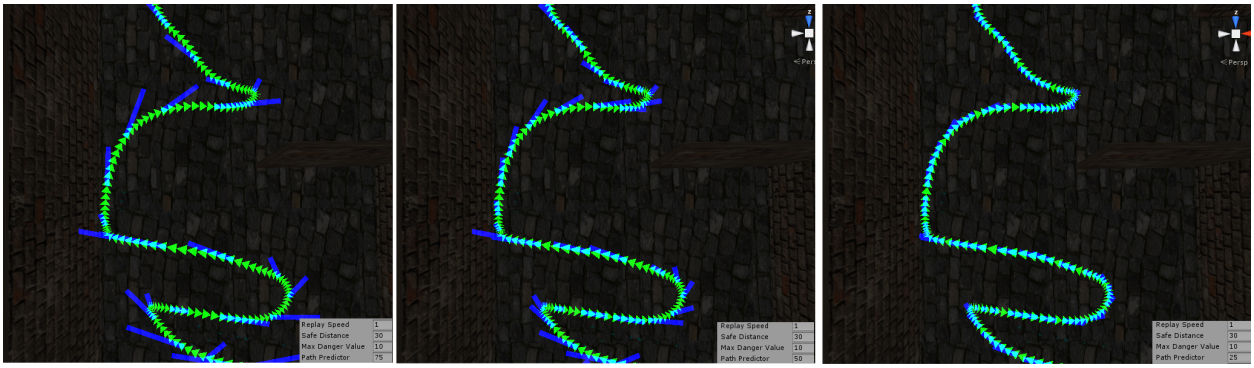

Figure 3: The same path with the variable path predictor set to 75, 50, and 25, respectively. The green arrows represent the player's path. The blue lines represent the dead reckoning projections of future

Results continued on next page...

positions (the bottom of the line is where the prediction was made, the top of the line is the predicted position).

Now that it is established that dead reckoning and our metrics our accurate, we can see a relationship between danger and deviation of the projected path. The player then played through a set level. In Figure 4 we see a different trace where a player experiences steadily increasing danger. Finally, it is observed that the player moves forward until turning sharply and deviating from the path as the danger reaches a threshold. The player decided that if they continued on their path, they would be in too great a risk of becoming detected thus showing the relation between the projected danger and the player's deviation from the path.

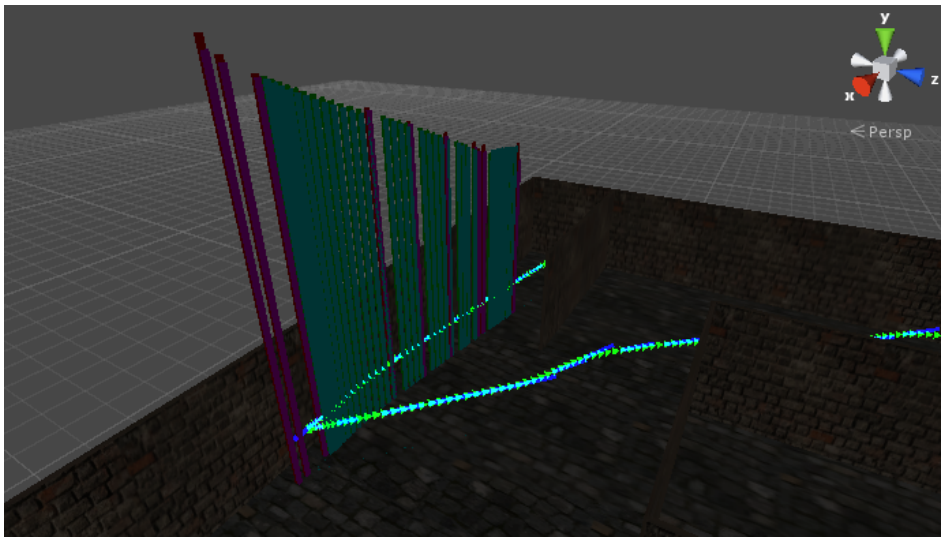

Figure 4: A player's path deviates from the projected path as their danger rises. Green and cyan bars represent players perceived danger metrics at their respective path points, red and magenta bars represent danger metrics at projected points,

There was only one test subject which was the projects creator and level designer. Therefore, these results are biased and inconclusive. Parameters safe distance, max danger value, and path predictor were set to 30, 10 and 50, respectively. Safe distance set at 30 is equal to the level's width dimension, while max danger value and path predictor were set arbitrarily.

Conclusion and Future Work

The goal of this project was to demonstrate that the metrics discussed were accurate representations of danger that a player perceives throughout game play and to correlate the measure of perceived threat behavior of a player in hopes of developing better analytical tools for game development. Results suggest that the metrics were indeed accurate, however, since the experimentation subjects were limited to the level designer and creator, the results are biased and therefore inconclusive.

The next step in this project would be further experimentation to further validate the claim that metrics. The metrics could be extended or more metrics could be introduced. For example, research of the guards' positions relative to other guards was beyond the scope of this project and therefore was not considered despite being potentially important areas of research. Currently, the metrics are just an addition of danger values calculated for the visible guards. Therefore, a player would be in same danger if there were three guards in front of them as if there were three guards surrounding him (assuming the guards' distances and angles were the same). Clearly, this is not accurate as a player would be in greater danger if they were surrounded.

Conclusion and Future Work continued on next page...

Path prediction could also be optimized as well. Rather than predicting the path based on two points, it would be more accurate to use curve fitting algorithm to predict the path. After these metrics are optimized, new game features could be introduced. Examples of these would be more intelligent guards, more interactive levels, or stealth elements such as shadow, light, and noise considerations.

In conclusion, with further research and development, the metrics tool developed and discussed here shows potential to be a useful analytical tool for future quantification and standardization of perceived threat in player experiences of video games.

References

[1] Jonathan Tremblay, Pedro Andrade Torres Clark Verbrugge "Measuring Risk in Stealth Games" [2] A. F. Seleem, H. A. Hussein, I. Talkhan " Dead Reckoning in a Distrbuted Interactive Tank Gunner Simulator", *13th International Conference on AEROSPACE SCIENCES & AVIATION TECHNOLOGY, ASAT- 13*, May 26-28, 2009

Appendix and Source Code

All code is open source and available at https://github.com/egroeg92/Stealth-Research

	Players Rated Danger	Angle Danger Metric	Distance Danger Metric	% Difference Angle Danger	% Difference Distance Danger
One Guard Far at ~180 Degrees	2	1.15	1.63	42.5	18.5
One Guard Far at ~90 Degrees	3	2.49	2.3	17	23.33
One Guard Far at ~0 Degrees	4.5	7.56	5.13	68	14
One Guard Near at ~180 Degrees	7	3.63	6.54	48.14	6.57
One Guard Near at ~90 Degrees	8	5.44	7.48	32	6.5
One Guard Near at ~0 Degrees	9.5	9.83	9.67	3.47	1.79
Two Guards Far at ~180 Degrees	3	2.1	3.33	30	11
Two Guards Far at ~90 Degrees	4	4.98	4.75	24.5	18.75
Two Guards Far at ~0 Degrees	6	10	8.01	66.67	33.5
Two Guards Near at ~180 Degrees	8	5.44	9.21	32	15.13
Two Guards Near at ~90 Degrees	9	8.27	10	8.11	11.11
Two Guards Near at ~0 Degrees	10	10	10	0	0

Table 1: Collected Danger Measurements Over Six Senarios

Listing 1: Danger Value using Metrics Code

```
dangerValueAngle = 0;
dangerValueDist = 0;

ArrayList dangersAngle = new ArrayList ();
ArrayList dangersDist = new ArrayList ();
for (int i = 0; i < enemies.Count ; i++) {
     if(player.canSee(enemies[i])){
     dangersAngle.Add
               (calculateAngleThreatMetric(enemies[i].angle, enemies[i].distance));
```

```
10      dangersDist.Add
            (calculateDistanceThreatMetric(enemies[i].angle, enemies[i].distance));
        }
    }
    dangersAngle.Sort();
15  dangersDist.Sort();
    float j = 1;
    for(int i = dangersAngle.Count - 1; i >= 0 ; i-- ){
        dangerValueAngle += (float)dangersAngle[i]/j;
        dangerValueDist += (float)dangersDist[i]/j;
20      j++;
    }

    dangerValueAngle = (dangerValueAngle > maxDangerValue)
                    ? maxDangerValue : dangerValueAngle;
25  dangerValueDist = (dangerValueDist > maxDangerValue)
                    ? maxDangerValue : dangerValueDist;
```

Listing 2: Distance Metric Code

```
    //distance danger metric
    float calculateDistanceThreatMetric(float angle, float dist){
        float danger = 0;
        float angleDanger;
5       if (angle <= 45)
            angleDanger = 1;
        else
            angleDanger = -(Mathf.Sqrt(angle)/Mathf.Sqrt (180f)) + 1;
        float distance = dist * dist;
10      distance = (distance < (safeDistance*safeDistance))
                    ? distance : (safeDistance*safeDistance);
        float distDanger = -(distance/(safeDistance*safeDistance)) + 1;
        danger = (angleDanger + (2* distDanger))*maxDangerValue/3;
        return danger;
15  }
```

Listing 3: Angle Metric Code

```
    //distance metric
    float calculateDistanceThreatMetric(float angle, float dist){
        float danger = 0;
        float angleDanger;
5       if (angle <= 45)
            angleDanger = 1;
        else
            angleDanger = -(Mathf.Sqrt(angle)/Mathf.Sqrt (180f)) + 1;
        float distance = dist * dist;
10      distance = (distance < (safeDistance*safeDistance))
                    ? distance : (safeDistance*safeDistance);
        float distDanger = -(distance/(safeDistance*safeDistance)) + 1;
        danger = (angleDanger + (2* distDanger))*maxDangerValue/3;
        return danger;
15  }
```