Grand Canyon University

Project 6 - Take a Calculated Chance

Justin Dietrich and Ryan Scott

CST-415: AI in Games and Simulations Lecture & Lab

Dr. Ricardo Citro

March 29, 2022

A brief description of the game or simulation (one paragraph)

1. Our game will be from a top-down view, set in an environment that has paths, walls, and other types of cover or obstacles. The player can move around the environment, trying to eliminate enemies while trying not to get eliminated themselves. Project 6 features a new enemy that fires a missile where it thinks the player is going to be.

How are the concepts listed above relevant and its purpose? (one paragraph)

2. We use the concept of probability with this new long-ranged tower enemy. The enemy has to pre-determine where the player is going to be in order to hit them with a long-ranged missle. It uses probability to determine the chance that the player ends up on each tile and chooses the tile with the highest chance of the player ending up on.

Which search method will be used? (one paragraph and bullet points outline)

3. The enemy calculates chance using a **Kalman filter**. There is a chance that the player continues moving in its current direction, turns, or goes backwards. The chances descend in value respectively. The algorithm calculates 5 moves, or however many should be needed (you just have to change 1 integer to adjust it).

   Reference: https://en.wikipedia.org/wiki/Kalman_filter

Github: https://github.com/AsePlayer/CST-415

Example Scripts

```
float[] chances = new float[] { 0.5f, 0.2f, 0.1f, 0.2f };
Vector2[] dirs = new Vector2[4];
//creates all 4 possible directions
//player's current direction is considered 'up'
float increment = Mathf.PI / 2;
dirs[0] = move; //up
dirs[1] = new Vector2(dirs[0].x * Mathf.Cos(increment) - dirs[0].y * Mathf.Sin(increment), dirs[0].x * Mathf.Sin(increment) + dirs[0].y * Mathf.Cos(increment)); //right
dirs[2] = new Vector2(dirs[1].x * Mathf.Cos(increment) - dirs[1].y * Mathf.Sin(increment), dirs[1].x * Mathf.Sin(increment) + dirs[1].y * Mathf.Cos(increment)); //down
dirs[3] = new Vector2(dirs[2].x * Mathf.Cos(increment) - dirs[2].y * Mathf.Sin(increment), dirs[2].x * Mathf.Sin(increment) + dirs[2].y * Mathf.Cos(increment)); //left
```

All of the possible directions are lined up with their respective chance of occurring.

```
//generates another layer, using the calculated chances
for (int i = 0; i < 4; i++)
{
    if (chances[i] > 0)
    {
        tile n = new tile();
        n.x = t.x + Mathf.RoundToInt(dirs[i].x);
        n.y = t.y + Mathf.RoundToInt(dirs[i].y);
        n.chance = t.chance * chances[i];

        calcTiles(dirs[i], n, depth + 1);
    }
}
```

The function is recursive. Each call splits the current chance of the tile into all the other tiles that are to be investigated next.

```
//adds possible tiles to list with percent chance
2 references
void calcTiles(Vector2 move, tile t, int depth)
{
    //adds tile probability to matching spot on map
    if (depth >= 5)
    {
        chanceTable[t.x, t.y] += t.chance;
        return;
    }
```

Once the desired depth is reached, it adds the chance of the current tile to a table containing the chance of each tile.

How will you overcome unforeseen obstacles during implementation? What is your 'plan B'?

4. Just getting the calculations to show up will show that the ai is successful. The enemy doing something about the calculated data (like attacking) comes secondary in this assignment. If worst comes to worst, we can repurpose our calculations for a different mechanic that is a little more feasible.

How is the project aligned with the current topic objectives?

5. The project shows that we can create a program that can calculate probabilities based on current states.

Chance	The chance that the player ends up on each tile is calculated. The chances all add up to 1.0.
Kalman filter	The chance that the player ends up on the next tile depends on the previous choice of the player.
Product rule	Chances are split into the next calculation by multiplying the current chance by the next chance.

What will appear on the screen: animation, user interactions, information dashboards, UI elements, etc.

6. Project 6 shows the calculations of the program with a decimal value on each tile that displays the chance. There are also small upgrades to the UI to improve clarity.

List the platform and software tools you plan on using

7. Stuff we will use:

   Unity (with C# scripts)

   Adobe Photoshop

   Adobe Illustrator

   MS Paint

   Audacity

Table to calculate chance:

No Turn	0.5
Turn Right	0.2
Turn Around	0.1
Turn Left	0.2

Direction of movement is updated after each chance.

Screenshots below: