Carnegie Mellon University
The Robotics Institute, School of Computer Science

Programming Familiarization Part 1

Team G
Bruce KwangKyun Kim
Sponsor: Dr. Oliver Kroemer
Mentor: Dr. John Dolan

February 21, 2022

Table of Contents

1 Part 1. Practice Problems

1.1 Given a pointer to struct D called "d", what expression will access "data", the member of struct A?

```
(((d->c).b)->a).data
```

1.2 Given the following structs, predict the memory layout of a struct A named "stack". You may assume that it is a 32-bit machine.

In a 32 bit computer, both the pointer size and float size can be 4 bytes. The following illustrates data type and its size in a 32-bit machine.

```
char 1
short 2
int 4
long 4
long long 8
float 4
double 8
long double 16
```

Struct B has two floating points, each with the size of 4 byte. Then, adding the two, the size of struct B is 8 bytes. It is also noticeable that no memory padding is needed. Looking into struct A, first there are three floating points, implying 12 bytes of memory. Adding on, B* ias a pointer to struct B, which is an address of 4 byte memory to the struct. Finally, B is struct B that has 8 bytes of memory. To sum up, the size of struct A is 24 bytes.

1.3 Given the functions below, what is the value stored in the points **a** and **b** after calling foo?

```
a.x = 1; a.y = 1; b.x = 0; b.y = 1;
p3.x = a.x + b.x = 1;
p3.y = a.y + b.y = 1;
p1.x = p1.x + p3.x = 1 - 1 = 0;
p1.y = p1.y - p3.y = 0 - 1 = -1;

Thus,
a.x = 0; a.y = -1; b.x = 0; b.y = 1
```

2 Part 2. Coding Problem

2.1 An Overview of AI

The AI predicts by considering two factors: 1) where the projectile will fall on the ground, and 2) where the current explosions are. To predict the projectile, I used equations of motion in general physics: $y = y_0 + vy \times time + \frac{1}{2} \times gravity \times times^2$ Using a quadractic equation, you can calculate the time when y = 0.

```
temp_t = (-b + sqrt(pow(b,2)-4*a*c))/(2*a);
```

Finally, using the equation of motion again, I can find where my predicted x position of the projectile will be located at. $x = x_0 + vx \times predicted_time$ This position is considered as a reference point for the predicted dangerous zone. Iterating all the projectile to find the predicted x location, and then considering the explosion range let me know which x points will be dangerous. With a similar logic, iterating all the explosions to find the current explosion x location. Again, explosion range to such explosion x point is considered.

The AI player will choose the spot, where the projectile will not land in the future, and where there are no current explosions.

2.2 Challenges I faced in implementing the AI

I faced some challenges throughout implementing the AI. First, I was confused with the game time and actual time calculated in the equations of motion. Understanding how each time defined in the program is different took me some effort.

Then, I had trouble creating an array of $dangerousSafeZone[]$ by dynamically allocating the memory. Because I used a constructor, I had to free this memory later to prevent memory leaks. Initially, I tried using heap in $determinSafeSpots()$ function, then freeing the memory at the end of $control()$function. This seemed weird because constructor and deleter are in different functions. In a way, this could increase the possibility of forgetting where the deleter is in the future.

Also, my AI showed the tendency of either moving right or left without stopping. In the $pickSafeSpot()$function, the AI was either designed to move left or right toward where safe spots would be. The problem was that the AI was continuously moving even when its current position is safe. Therefore, if a conditional statement to move the player to the left is written before the conditional statement to move the player to the right, the AI ended up being positioned at the left part of the game.

```
//An example of the AI player skewing to the left
int l = p->x;
int r = p->x;

while (l > 0 || r < w) {
    std::cout <<r << " " << l << std::endl;
    if (l > 0 && dangerZonePred[l] == 0){
        std::cout << "left " << l << " " << r << std::endl;
        return -1;
```

```
        }
        if (r < w && dangerZonePred[r] == 0){
            std::cout << "right "<<r << " " << l << std::endl;
            return 1;
        }
        r++;
        l--;
    }
```

2.3 AI performance on a Hard scenario

The AI program, however, does not consider how much time will it take for the projectile to land on the predicted ground. This is probably the reason why the program does not work well on the Hard scenario, which has more enemies and more projectiles shoot. The AI player wouldn't last more than 2 seconds with the current program on the Hard scenario.

To overcome this, I may have to consider changing $dangerSafeZone[]$ as an array of the time left until the explosion. Then for each projectile, I can calculate the time remaining until the projectile reaches to the land. By doing so, I will be able to know which safe spots will last longer and which will not.

2.4 My opinion on the assignment and my goals

The goal of the assignment states that it intends "to give you practical experience in writing and debugging actual C++ applications and give you more practice at memory in C++."

I think this was a great assignment to get familiarized with C++ programming and learn about the memory structure in C++. From this assignment, I learned about std::iterator, pointer, class in general, and the difference between public and private classes. I hope the assignment gives some questions about dynamic memory allocation and freeing the memory after its use to prevent memory leakage.