Tucker Lavell
CS162 Fall 2017
Project 4: Fantasy Combat Game Part 2
Documentation

Problem

Take the Fantasy Combat Game created for Project 3, and turn it into a tournament. When the game starts, the user will be prompted to play or exit. There are two teams, and the user selects how many people they want each team to have (same number for both teams). The game will then ask the user to start selecting characters and naming them. Each character will fight their positional counterpart on the other team. For each round, the names of the combatants, and which won, should be displayed to the user. The winner is added to the back of their team's lineup and has some amount of strength points restored. The loser will be moved to the front of the bench, all losers will sit on the same bench. For each win a team receives +2 points and for each loss a team receives -1 point. The tournament ends when one team has no fighters left, the team with the higher score wins. After presenting the score, the program should ask the user if they would like to see the loser pile. If they choose to see it, they will see the losers from last to first. After that the user will be given the option to play again or exit.

Design

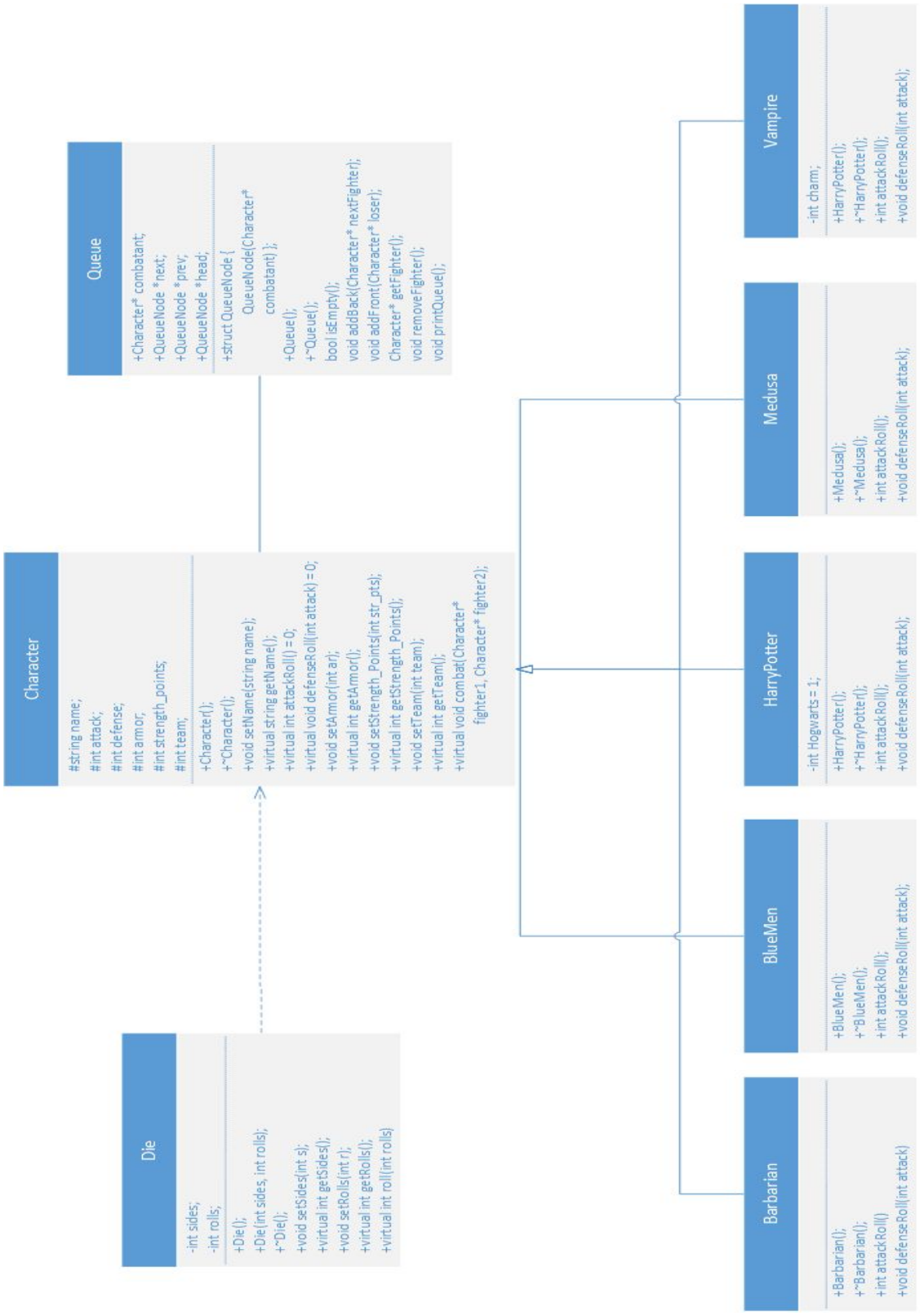
```
class Die {
	protected:
		int sides;
		int rolls;
	public:
		Die();
		Die(int sides, int rolls);
		~Die();
		void setSides(int s);
		virtual int getSides();
		void setRolls(int r);
		virtual int getRolls();
		virtual int roll(int rolls) {
			if (rolls < 1) {
				return 0;
			}
			return (rand() % sides + 1) + roll(rolls - 1);
		}
};
```

```
class Queue {
	struct QueueNode {
		Character* combatant;
		QueueNode *next;
		QueueNode *prev;
		QueueNode(Character* fighter) { combatant = fighter; }
	};
	QueueNode *head;
	public:
		Queue();
		~Queue();
		bool isEmpty();
		void addBack(Character* nextFighter);
		void addFront(Character* loser);
		Character* getFighter();
		void removeFighter();
		void printQueue();
};
class Character {
	protected:
		string name;
		int attack;
		int defense;
		int armor;
		int strength_points;
		int team;
	public:
		Character();
		~Character();
		void setName(string name);
		virtual string getName();
		virtual int attackRoll() = 0;
		virtual void defenseRoll(int attack) = 0;
		void setArmor(int ar);
		virtual int getArmor();
		void setStrength_Points(int str_pts);
		virtual int getStrength_Points();
		void setTeam(int team);
		virtual int getTeam();
```

```
        virtual void combat(Character* fighter1, Character* fighter2);
        // will be similar to previous part, but less wordy
};
class Barbarian : public Character {
    public:
        Barbarian();
        ~Barbarian();
        int attackRoll() { return attack = new Die(6, 2).roll(); }
        void defenseRoll(int attack) { defense = new Die(6, 2).roll(); }
};
class BlueMen : public Character {
    public:
        BlueMen();
        ~BlueMen();
        int attackRoll() {  return attack = new Die(10, 2).roll(); }
        void defenseRoll(int attack) {
        if (currentStr_Pts > 0 &&  currentStr_Pts <= 4) { defense = new Die(6 ,1).roll(); }
        if (currentStr_Pts > 5 &&  currentStr_Pts <= 8) { defense = new Die(6 ,2).roll(); }
        if (currentStr_Pts > 0 &&  currentStr_Pts <= 4) { defense = new Die(6 ,3).roll(); }
        }
};
class HarryPotter : public Character {
    private:
        int Hogwarts = 1;
    public:
        HarryPotter();
        ~HarryPotter();
        int attackRoll() { return attack = new Die(6, 2).roll(); }
        void defenseRoll(int attack) {
        defense = new Die(6, 2).roll();
        if (str_pts <= 0 && Hogwarts > 0) {
            setStrength_Points(20); Hogwarts -= 1; }
        }
};
```

```
class Medusa : public Character {
	public:
		Medusa();
		~Medusa();
		int attackRoll() { attack = new Die(6, 2).roll();
		if (attack == 12) { return attack = 1000; }
		else { return attack; }
		}
		void defenseRoll(int attack) { defense = new Die(6, 1).roll(); }
};
class Vampire : public Character {
	private:
		int charm;
	public:
		Vampire();
		~Vampire();
		int attackRoll() { return attack = new Die(12, 1).roll(); }
		void defenseRoll() { charm = RNG(0, 10);
		if charm < 6 { attack = 0; }
		else { defense = new Die(6, 1); }
		}
};
```

Class Hierarchy Diagram



Test Table

Test Case	Inputs	Expected Outcome	Observed Outcome
User gives more than 1 character the same name	3 Default Blue Men on T1 3 Default Medusa on T2	T1_Blue Men T1_Blue Men #2 T1_Blue Men #3 T2_Medusa T2_Medusa #2 T2_Medusa #3	T1_Blue Men T1_BlueMen T1_Blue Men #2 T2_Medusa T2_Medusa T2_Medusa #2
User Gives all characters the same name	3 Blue Men on T1 named George 3 Barbarians on T2 named George	T1_George T1_George #2 T1_George #3 T2_George T2_George #2 T2_George #3	T1_George T1_George #2 T1_George #3 T2_George T2_George #2 T2_George #3
User makes all characters the same type and name	3 Default Harry Potters on T1 3 Default Harry Potters on T2	T1_Harry Potter T1_Harry Potter #2 T1_Harry Potter #3 T2_Harry Potter T2_Harry Potter #2 T2_Harry Potter #3	T1_Harry Potter T1_Harry Potter #2 T1_Harry Potter #3 T2_Harry Potter #4 T2_Harry Potter #5 T2_Harry Potter #6
T1 Character wins a fight	T1_Blue Men T2_Harry Potter #2	Blue Men goes to the back of T1, and next in line goes to the front Harry Potter goes to the top of the Bench and is deleted from T2	Blue Men requeues, and next goes to the front Harry Potter gets removed from T2, but is not properly added to the Bench(Linked List insertion messed up)
A team has different character types, with the same name	(Blue Men) T1_Sam (Medusa) T1_Sam (Barbarian) T1_Sam	(Blue Men) T1_Sam #1 (Medusa) T1_Sam #2 (Barbarian) T1_Sam #3	(Blue Men) T1_Sam #1 (Medusa) T1_Sam #2 (Barbarian) T1_Sam #3

Reflection

I think I did a great job on this project. My biggest thing to work on from Project 2 was having neater code in larger projects. Even though I wasn't completely satisfied with the structure for Project 3, I submitted it because it met requirements and no memory leaks. This time I was determined to make the code look better, not needing all the couts for each round really helped.

I thought this project would be super easy considering it was basically putting together Project 3 and Lab 6/7. I was wrong, but it did come together quickly...at first. My first step was to combine L7 and P3, I basically just copied and pasted them together. While thinking it through, I realized I need an int to pass around for team 1 and 2. I almost got this working first try, but I forgot to declare the t1, t2, and losers queues, I had only done them in the header. After declaring them, it worked how I intended, with all my old couts still there showing each step. I wanted it to take it 2 teams of multiple characters and battle them into infinity.

Next I added an addFront and benching function. It took the round loser and added them to the front of the losers queue and deleted them from their respective team. At this time, I was only using Queues built in addBack, to "requeue" the winner on their respective team. I did this by adding an "else if" to addBack which would recognize that the fighter passed in was already the head. Then it would move the queue forward one and add the winner to the back. I was doing this poorly, by creating a new node but never deleting the old one. Thankfully that was an easy realization and fixed it before I even ran it. When I did run it, I noticed a small problem with losers, I was benching the temp that I was deleting, instead of benching the loser. After fixing the temp problem, I noticed another, larger problem, my addFront was not working correctly. I did lots of playing around and got it to a usable state, and it worked with all the old stuff.

Then I deleted all the couts and double checked it was still working. When I ran it, I noticed I hadn't accounted for displaying who vs who only once. So it was still showing that for every round of combat the fighters did, then printing who won. That was an easy fix, I added a static int to count the rounds, then when a player is declared the winner the counter is set back to 0, for the next set of fighters. The next problem was with printing losers. At first I was only getting a memory address, because I was printing fighter, not fighter->getName.

Printing the losers was still wrong after the fighter->getName, my addFront linked list logic was totally messed up. I SPENT DAYS trying to fix this, in the meantime I came up with a way to account for identical names. First I inserted their team to the front of their name, then I tried to account for multiple characters having the same name. The way I had the name comparison at first I was only checking if their name was the same as the head. This sort of worked for all the same type, but it wouldn't append the # to the end of the second character. I solved this with a funky loop and a break, but I knew it was a hacky fix. I also noticed that sometimes I would get a read/write exception in my character combat, but I thought it was related to the hackyness.

While procrastinating addFront, and adding the other stuff, I came upon another problem that stumped me. It was making me press enter to continue to the question about about printing. Thankfully I had worked on this early, then worked on Lab 8. I had a similar problem in Lab 8 this week, but since the scope of that was much smaller than the project, I was able to fix it. It was issues with leaving \n in the cin buffer, and the cin.clear and cin.ignore I had, did not work for it. On the project it was one of my "trusted" support functions that was causing this, getValidInterger, which is one of my favorites to use, because I like switches. Because of Lab 8 though, I was able to much more easily find where the \n was getting stuck.

After finishing L8, I had to come back to P4. After not looking at the project, or a linked list, for a couple days, I had the clarity to fix what was remaining. First I fixed how like names are handled. I added a function to my Queue class called checkFighters which takes in the fighterAdding pointer. Then I used a while loop to compare fighterAdding's name to the names of the people already on the team. FINALLY, after a few days break, I got getFront to work. In turn, printing also became functional, because it was relying on list constructed by getFront.

Once I got getFront working, I thought I was done, but then I remembered I forgot a recovery function. I added another pure virtual function to Character called recoveryRoll. It has a variable called maxStr_Pts and it rolls a single dice, the size of the difference between max and current Str_Pts. Then that is added to the character's current health.

After playing through my program a few times, I found out I still had one issue, the absolute worst one. The read/write exception in combat was not caused the hacky renaming, but was completely unknown. Sometimes it would happen, sometimes it wouldn't, it was hard to reproduce. I stepped through almost every step of each function, for hours and could not figure out what was wrong. I had a friend step through every step with me, numerous times, we could not figure it out. After a long while I was at least able to reproduce it. It was easiest to do with teams of 3, with one team of Vampire, Barbarian, and Blue Men and the other team was the same but reversed. What seemed like what was happening was, if one team was winning by a lot, I would get the exception. Stepping through it for a while again, I noticed that once in awhile, head->prev->prev was empty. I had mostly been checking the ones like head->next, head->next->next, head->next->prev, head->prev, head->prev->next for every node, when I was stepping through. When I saw that, I started second guessing my addFront, because I had struggled so much with it. My friend suggested it might have something to do with my teams instead, since it was happening where characters fought. It was so confusing, because all of the code worked perfectly on the other assignments. I came to the realization that it was something that seemed correct, but wasn't. The way I was requeuing the winners was leaving head->prev->prev empty, but it worked perfectly with teams of 2, and occurred very deep inside the team of 3. Once again I had messed up a linked list. After a little testing I finally got it, and the whole program worked.

My luck didn't end there though, I had another problem when I was trying to compile it on flip. I knew it was something with the linker, but it wasn't clear to me what it was. Trying to fix it allowed me to clean up my includes. I had forgotten to add create Queue in the makefile.

The linked list issues I had on this were extremely frustrating, they definitely made it the hardest assignment for me yet. Somehow, I think, I still managed to come out with some clean code. I think I learned A LOT from this project, and it also changed my opinion from the last project. I think I could do game design now, it seems like it would be a good challenge. Sorry for the length, I had a lot to say about this project. It's been so stressful for me, even though I gave myself enough time.

First compile forgot to declare t1, t1, and losers as a queue
First made it work with old stuff (the logic including couts in the rolls for each character) plus teams(into infinity), had to add team tracker
Then i added benching loser and readding winner and incrementing score
At first was requequing improperly (creating a new node before trying to shuffle) now I have an else if that handles the shuffling if the fighter taken in is the same as the head fighter
Small Problem was benching temp at first
Linked list (addFront) for losers not correct
Got it working with old stuff plus benching and requeuing
Then i deleted all the couts and double check that it was still working
Small problem printing who vs who too much. Added static int round counter, that increments only for each set of fighters, then resets back to 0 when a winner is declared
Problem with printing losers mem addresses
Forgot to do fighter->getName()
addFront() wasn't working I just didn't notice, after playing with it for a while I finally got it
Inserted team number in front of name
Making me press enter to continue to print
Trouble appending # to like names
Problem going over 7 fights (actually problem continuing if one team beats the other by a lot)
Forgot recovery roll
The way i was shuffling was the problem