Project Zombie

A Short Zombie Adventure

By Kyle Deppe, Tsiania Hughes, and Joel Abshier

Abstract

Project Zombie is a short zombie adventure game based on the old game Oregon Trail. Using objects to control different characters and encounters, the goal of the game was to get players to play 10 turns against each other - the first one to die loses. In each turn they are to receive a story along with a set of options, of which they could choose one. Choosing the right option would lead to increased odds of passing the encounter. The game is designed to brutally punish the players for bad play, making it possible to have a bad outcome even in a correct encounter.

Introduction

The three creators of the project are all lovers of video games. We desired to create a game that would have the funny, and often quirky, storylines of games in the arcade style (like Oregon Trail) that also gave players a unique experience. When laying out a plan for this project, some of our main goals were as follows:

- Have multiple playable characters, each with unique storylines
- Have a large pool of random encounters, allowing the game to feel different with each playthrough.
- Make the game competitive by having decreasing stats for each player, with the goal being to outlast the other player.
- Make it possible to exit during the game and resume playing with save/load functions to files.

Background

As with other games in this genre, a lot of the gameplay would occur in random encounters. This core system has a lot to do with rolling and random number and weighing different stats accordingly. For our system, each encounter has an odds of being passed, the players luck, and whether or not they made the best choice for the encounter. To handle this logic, a number between 1-100 was generated. In order to pass the test, they needed to have a roll higher than the minimum for passing the encounter.

As for the Characters, they are a very unique part of the game. Within each player their stats and their specific stories are held. And there are methods for setting, adding, removing, and displaying each of these. This was a fairly simple use of objects with different data members.

The game loop is all contained within one method in the main class. This keeps track of what state that the game is currently running in and will only abort whenever to state reaches a certain value. Each state corresponds to different parts of the game (Menu, playing, etc).

Proposed Method / System Description / Implementation

Note: For this section, the UML diagram will be broken into blocks, in the appendices a full diagram can be seen.

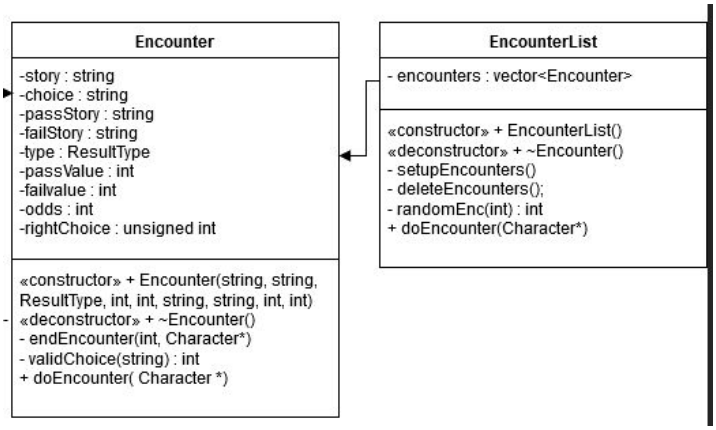


Figure 1 - Encounter and EncounterList UML

The bulk of the gameplay for Project Zombie would be captured in each of the random encounters that the player would face, so naturally, there was a lot of thought put into how these would play out. To store these, encounters are initialized as objects and then stored in a map within another object. Developing it this way gave us the ability to easily add and remove encounters from the list. To ensure that the players don't receive the same encounters during one game, after an encounter occurs, it will be removed from the list. All of the input and output for the players decisions, along with the associated stories would be handled within methods of the Encounter class. This data was allocated at the start of the game and then freed at the end of the runGame method.

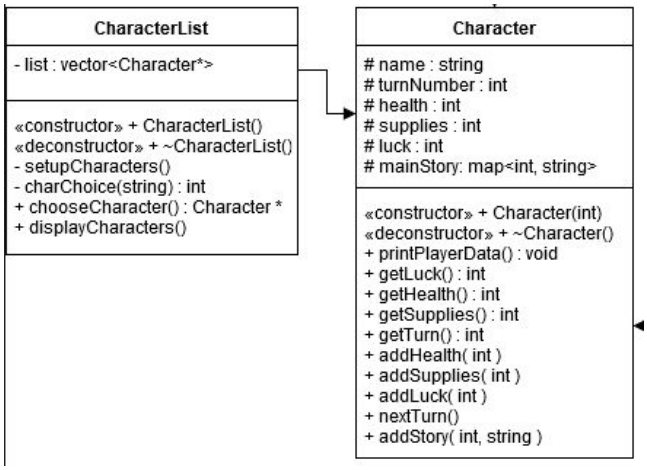


Figure 2 - Character and CharacterList UML

The Character class was designed to keep track of a selected player's stories and stats throughout the game. The players stories are held within a map, which we decided to use instead of a vector because we wanted to have an associated turn number with it and reference it by that. At the start of the game all six available players are generated and the players can select which they wish to use. At the initialization of each player, they are given a set of random stats (within a range) that will affect how their game will play. Characters have a health, luck, and supplies stat. The health and supplies stat essentially function the same, where if either ever reaches 0, the player will lose the game and the other player will win. The luck stat however is fixed for the lifetime of the Character. Luck factors in as part of the odds for each encounter, so players with a higher luck stat have a higher chance of passing encounters. Within this class we used operator overloading in order to change the functionality of this for the << and >> operators. This was so that whenever we wanted to write the player data to a file, the appropriate data members would be saved and we only have to write it once. The data for the character class was allocated at the start of the game within the player selection, and after the game ends, it is freed.

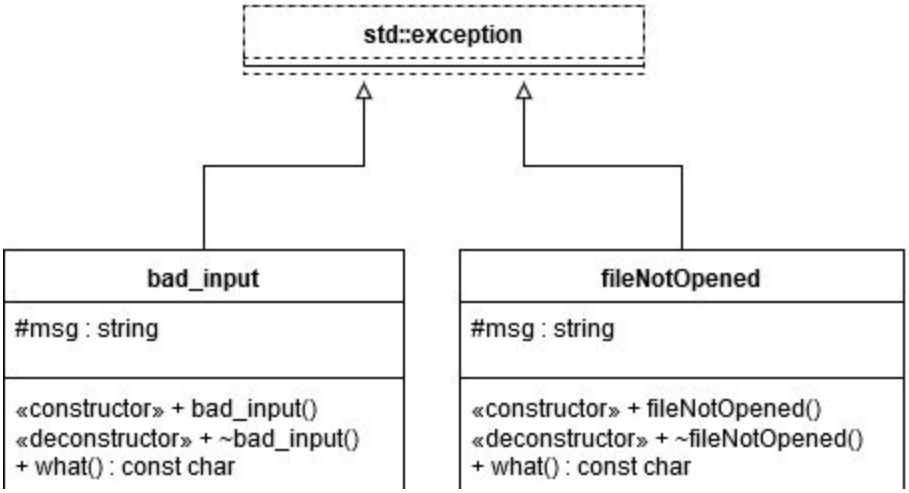


Figure 3 - Custom Exception Classes for input and file reading

At almost every step of the game, the players needed to be inputting data, whether an option for an encounter or on a menu. To handle this different functions and methods were developed for selecting. To ensure safe execution, all data was read in as strings and then tested before being converted into the value needed. If anything was wrong along the process, exception handling was usually used to throw errors that could be caught on a higher level. In addition to this error checking, files had to be saved, loaded, and erased throughout the application. To do this, files were always checked to be open and exceptions were thrown if they weren't. The game was automatically saved from within the runGame() method at the end of each turn, if the players did manage to reach turn 10, the game file would instead be erased, so that the players cannot relive the happy moment of winning.

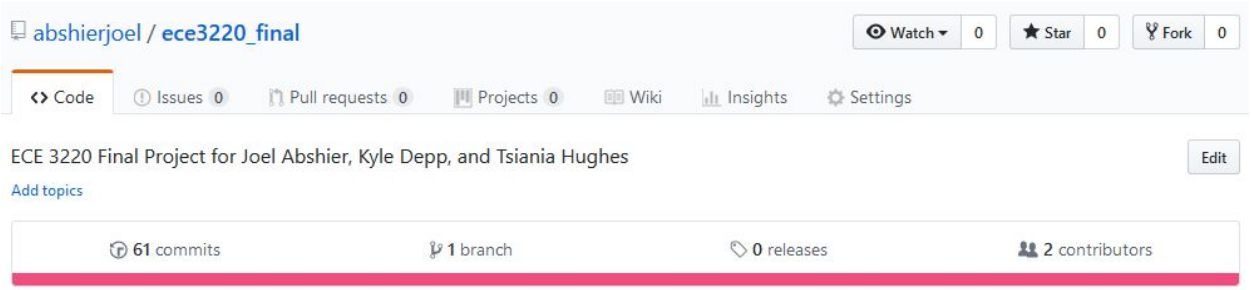


Figure 4 - GitHub Header

To manage the whole project we used a GitHub repository. This helped us be able to work while we were all in different locations without getting ever mixing up code and erasing newer versions. This also served as a great tool for communicating the changes that other team members were making.

Experiments and Results

A lot of the testing was done after each method was created. Once we had a solid skelton we started adding more entities to the project. After the first iteration where there were two different versions, we decided to be careful from then on out. The system we have is extremely hard to break, after adding a lot of error checking after each user input to avoid the game crashing. Using GitHub we each had the most updated version between the 3 of us and was able to test it on our personal laptops, a lot of the testing was done by ourselves. Tsiania was able to have a few of her friends without any background knowledge of coding try to break it. Then get any feedback they had on the actual game itself, which is an important step in corporations to make sure the product does what's intended and user experience.

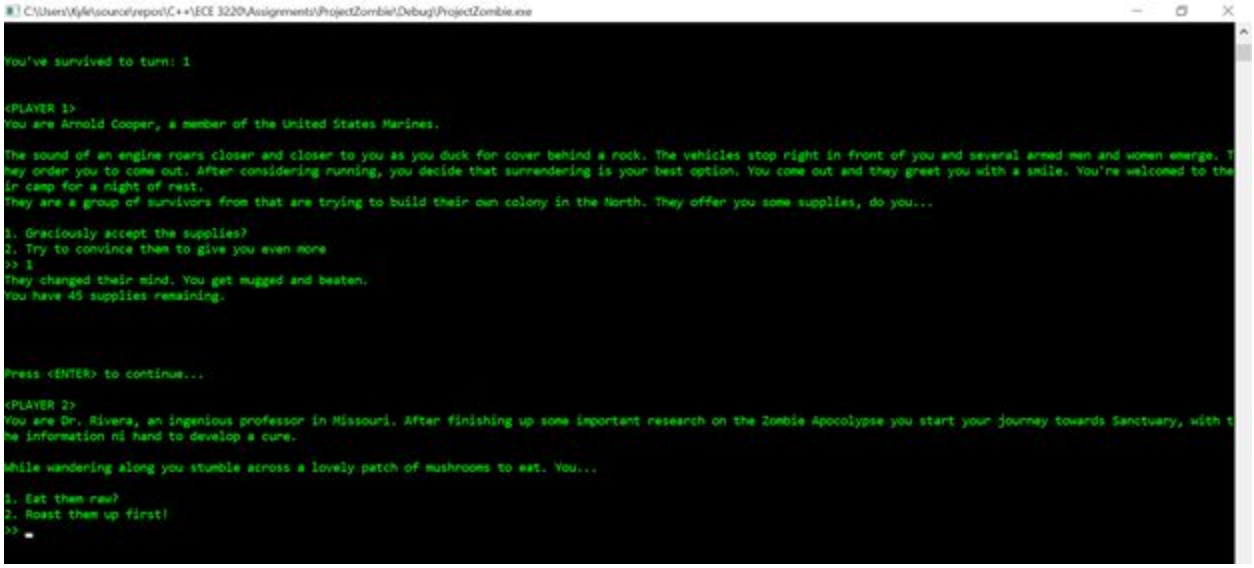
After getting the assigned partners we made it a weekly thing to meetup friday during the normal class time and discuss how the game was to be designed. We discussed how many people should be playing the game in one sitting (was one player and changed to two players), discuss possible classes (Character, Story, etc) and their functions within the game, and overall goals to be accomplished each week. The work was divided based on the availability of each person, depending on if the person had other projects or testing. We met our goals at each friday checkmark for the most part until we had to crunch down for the 2 weeks. Then after the presentation we added a few more things to the code and checked it over.

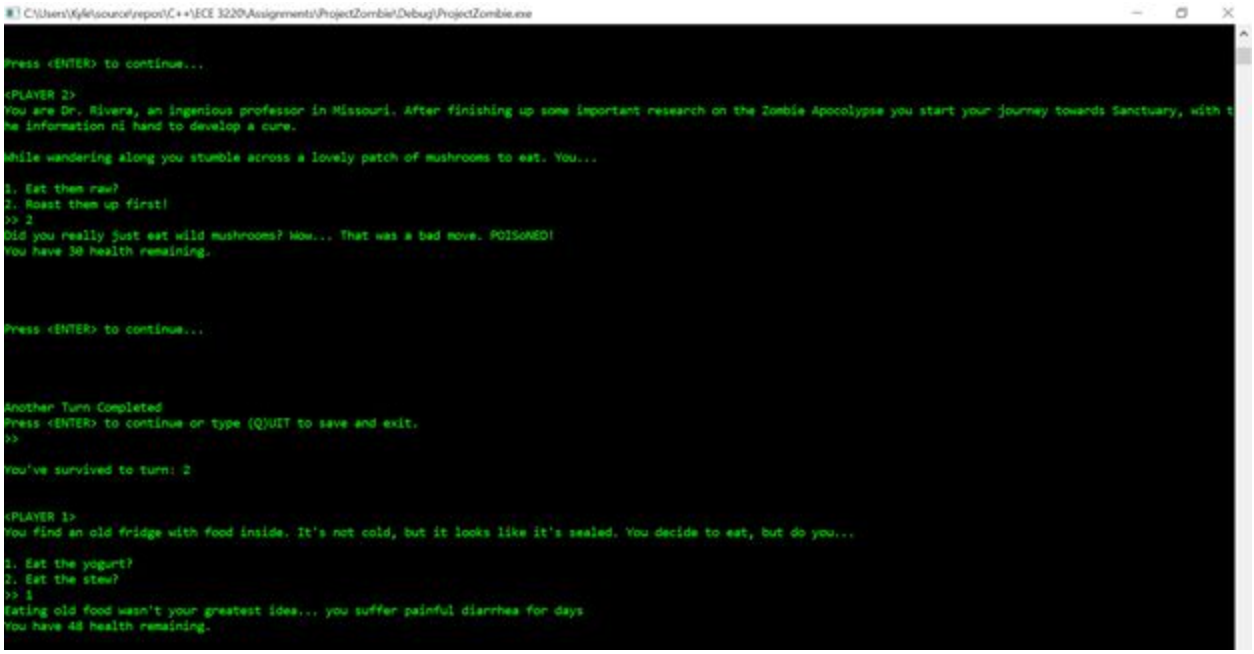

Press <ENTER> to initiate Project Zombie

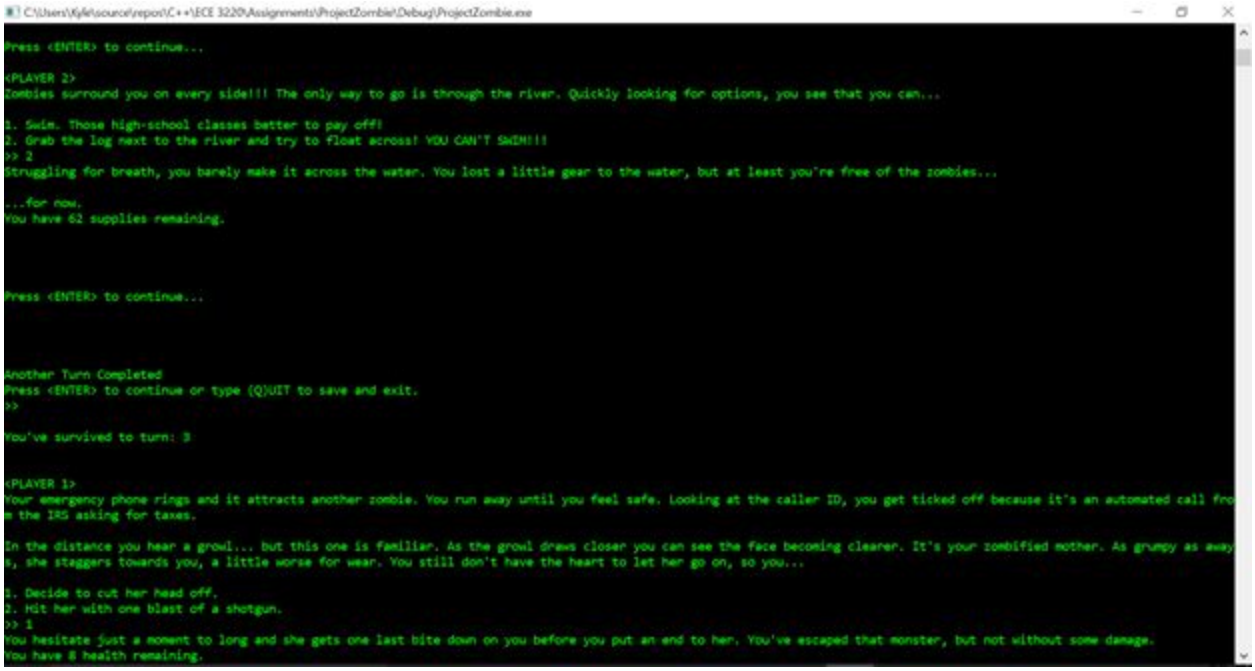

C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
PROJECT ZOMBIE
Welcome to Project Zombie
Please select an option:
1. Play Game
2. Load Game
3. See Instructions
4. Exit
>> 1
Creating a new game...
Which file would you like to save to?
1. Slot 1 - EMPTY
2. Slot 2 - EMPTY
3. Slot 3 - EMPTY
>> 1
Slot 1 selected!
<PLAYER 1>
Please select a character:
1. Arnold
2. Clark
3. Luis Rivera
4. Lil' Pupper
5. Abigail
6. Darwin
>> 1
<PLAYER 2>
Please select a character:
1. Clark
2. Luis Rivera
3. Lil' Pupper
4. Abigail
5. Darwin
>> 2

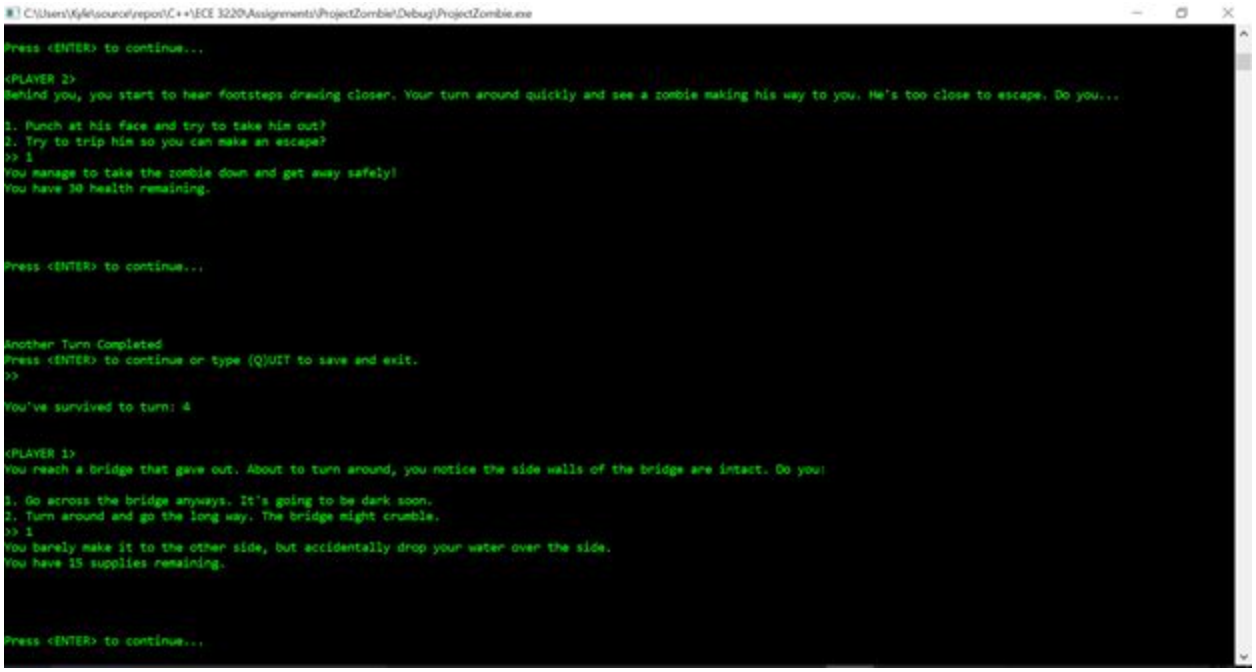

C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
<PLAYER 1>
Arnold's Stats
Health: 78
Supplies: 80
Luck: 74
<PLAYER 2>
Luis Rivera's Stats
Health: 65
Supplies: 67
Luck: 72
Press <ENTER> to initiate Project Zombie

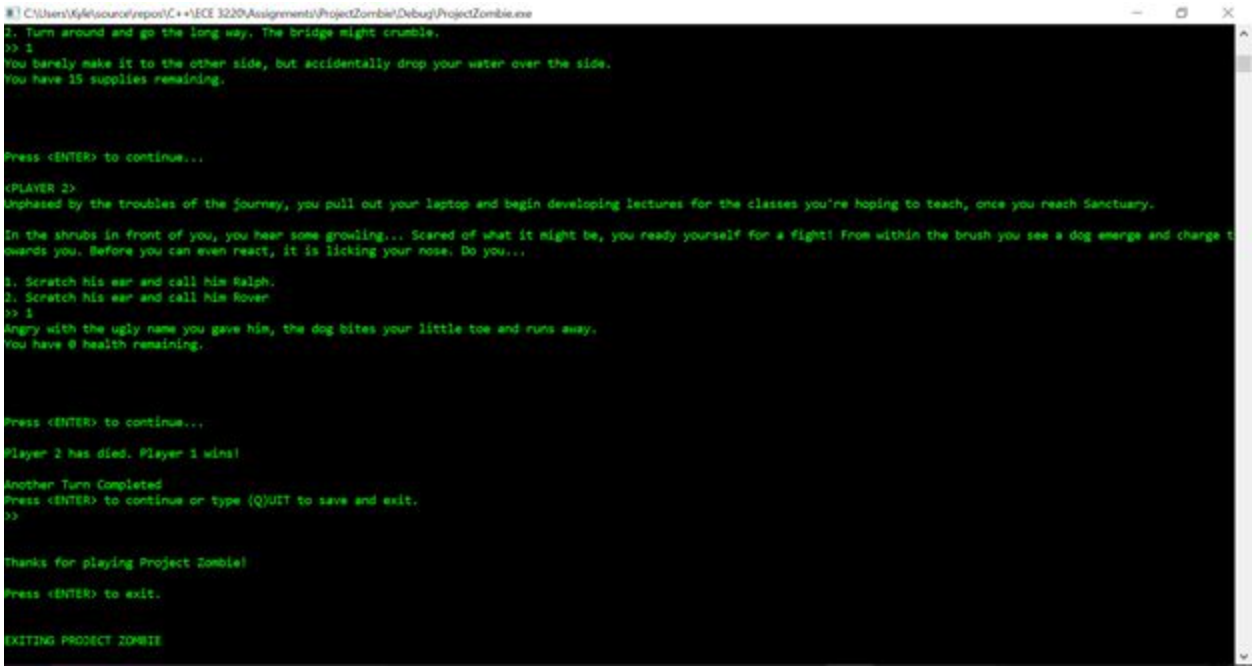

C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
//
//// Press <ENTER> to initiate Project Zombie ////
//
You've survived to turn: 1
<PLAYER 1>
You are Arnold Cooper, a member of the United States Marines.
The sound of an engine roars closer and closer to you as you duck for cover behind a rock. The vehicles stop right in front of you and several armed men and women emerge. T
hey order you to come out. After considering running, you decide that surrendering is your best option. You come out and they greet you with a smile. You're welcomed to the
ir camp for a night of rest.
They are a group of survivors from that are trying to build their own colony in the North. They offer you some supplies, do you...


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
You've survived to turn: 1
<PLAYER 1>
You are Arnold Cooper, a member of the United States Marines.
The sound of an engine roars closer and closer to you as you duck for cover behind a rock. The vehicles stop right in front of you and several armed men and women emerge. T
hey order you to come out. After considering running, you decide that surrendering is your best option. You come out and they greet you with a smile. You're welcomed to the
ir camp for a night of rest.
They are a group of survivors from that are trying to build their own colony in the North. They offer you some supplies, do you...
1. Graciously accept the supplies?
2. Try to convince them to give you even more
>> 1
They changed their mind. You get mugged and beaten.
You have 45 supplies remaining.
Press <ENTER> to continue...
<PLAYER 2>
You are Dr. Rivera, an ingenious professor in Missouri. After finishing up some important research on the Zombie Apocolypse you start your journey towards Sanctuary, with t
he information ni hand to develop a cure.
While wandering along you stumble across a lovely patch of mushrooms to eat. You...
1. Eat them raw?
2. Roast them up first!
>>


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
Press <ENTER> to continue...
<PLAYER 2>
You are Dr. Rivera, an ingenious professor in Missouri. After finishing up some important research on the Zombie Apocolypse you start your journey towards Sanctuary, with t
he information ni hand to develop a cure.
While wandering along you stumble across a lovely patch of mushrooms to eat. You...
1. Eat them raw?
2. Roast them up first!
>> 2
Did you really just eat wild mushrooms? Wow... That was a bad move. POISONED!
You have 30 health remaining.
Press <ENTER> to continue...
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
You've survived to turn: 2
<PLAYER 1>
You find an old fridge with food inside. It's not cold, but it looks like it's sealed. You decide to eat, but do you...
1. Eat the yogurt?
2. Eat the stew?
>> 1
Eating old food wasn't your greatest idea... you suffer painful diarrhea for days
You have 48 health remaining.


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
Press <ENTER> to continue...
<PLAYER 2>
Zombies surround you on every side!!! The only way to go is through the river. Quickly looking for options, you see that you can...
1. Swim. Those high-school classes better to pay off!
2. Grab the log next to the river and try to float across! YOU CAN'T SWIM!!!
>> 2
Struggling for breath, you barely make it across the water. You lost a little gear to the water, but at least you're free of the zombies...
...for now.
You have 62 supplies remaining.
Press <ENTER> to continue...
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
You've survived to turn: 3
<PLAYER 1>
Your emergency phone rings and it attracts another zombie. You run away until you feel safe. Looking at the caller ID, you get ticked off because it's an automated call fro
m the IRS asking for taxes.
In the distance you hear a growl... but this one is familiar. As the growl draws closer you can see the face becoming clearer. It's your zombified mother. As grumpy as away
s, she staggers towards you, a little worse for wear. You still don't have the heart to let her go on, so you...
1. Decide to cut her head off.
2. Hit her with one blast of a shotgun.
>> 1
You hesitate just a moment to long and she gets one last bite down on you before you put an end to her. You've escaped that monster, but not without some damage.
You have 8 health remaining.


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
Press <ENTER> to continue...
<PLAYER 2>
Behind you, you start to hear footsteps drawing closer. Your turn around quickly and see a zombie making his way to you. He's too close to escape. Do you...
1. Punch at his face and try to take him out?
2. Try to trip him so you can make an escape?
>> 1
You manage to take the zombie down and get away safely!
You have 30 health remaining.
Press <ENTER> to continue...
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
You've survived to turn: 4
<PLAYER 1>
You reach a bridge that gave out. About to turn around, you notice the side walls of the bridge are intact. Do you:
1. Go across the bridge anyways. It's going to be dark soon.
2. Turn around and go the long way. The bridge might crumble.
>> 1
You barely make it to the other side, but accidentally drop your water over the side.
You have 15 supplies remaining.
Press <ENTER> to continue...

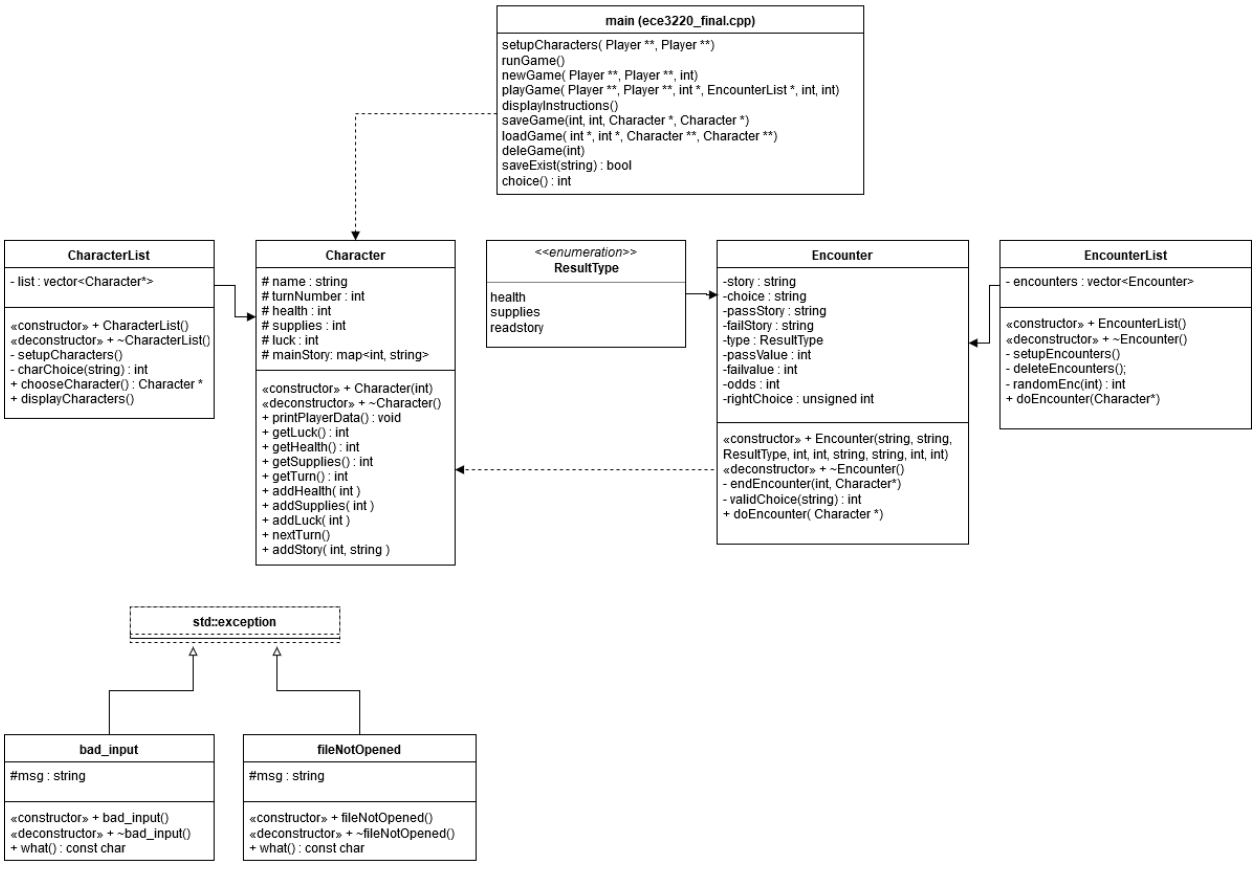

C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
2. Turn around and go the long way. The bridge might crumble.
>> 1
You barely make it to the other side, but accidentally drop your water over the side.
You have 15 supplies remaining.
Press <ENTER> to continue...
<PLAYER 2>
Unphased by the troubles of the journey, you pull out your laptop and begin developing lectures for the classes you're hoping to teach, once you reach Sanctuary.
In the shrubs in front of you, you hear some growling... Scared of what it might be, you ready yourself for a fight! From within the brush you see a dog emerge and charge t
owards you. Before you can even react, it is licking your nose. Do you...
1. Scratch his ear and call him Ralph.
2. Scratch his ear and call him Rover
>> 1
Angry with the ugly name you gave him, the dog bites your little toe and runs away.
You have 0 health remaining.
Press <ENTER> to continue...
Player 2 has died. Player 1 wins!
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
Thanks for playing Project Zombie!
Press <ENTER> to exit.
EXITING PROJECT ZOMBIE

Discussion and Conclusions

The hardest part of the program to write was probably the save/load feature. Once we figured out how to get the save.dat files to load up into the game, everything went a lot more smoothly. Our original system was supposed to be on Microsoft's Visual Studio, however after the first iteration we decided that it needed to be more cross compatible so that Eclipse can run it without adding any additional libraries. While the rest of the code itself wasn't as hard to write, the story took a long time to create so that it felt like a "whole game" instead of just something that you play once and then have seen everything.

Appendices

Full UML Diagram



Appendices: Project Files

Below is all of the source code files for the project.

///********** ece3220_final.cpp | Main File **********///

///********** Character.h **********///

///********** Character.cpp **********///

///********** Encounter.h **********///

///********** Encounter.cpp **********///

Project Zombie

A Short Zombie Adventure

By Kyle Deppe, Tsiania Hughes, and Joel Abshier

Abstract

Project Zombie is a short zombie adventure game based on the old game Oregon Trail. Using objects to control different characters and encounters, the goal of the game was to get players to play 10 turns against each other - the first one to die loses. In each turn they are to receive a story along with a set of options, of which they could choose one. Choosing the right option would lead to increased odds of passing the encounter. The game is designed to brutally punish the players for bad play, making it possible to have a bad outcome even in a correct encounter.

Introduction

The three creators of the project are all lovers of video games. We desired to create a game that would have the funny, and often quirky, storylines of games in the arcade style (like Oregon Trail) that also gave players a unique experience. When laying out a plan for this project, some of our main goals were as follows:

- Have multiple playable characters, each with unique storylines
- Have a large pool of random encounters, allowing the game to feel different with each playthrough.
- Make the game competitive by having decreasing stats for each player, with the goal being to outlast the other player.
- Make it possible to exit during the game and resume playing with save/load functions to files.

Background

As with other games in this genre, a lot of the gameplay would occur in random encounters. This core system has a lot to do with rolling and random number and weighing different stats accordingly. For our system, each encounter has an odds of being passed, the players luck, and whether or not they made the best choice for the encounter. To handle this

logic, a number between 1-100 was generated. In order to pass the test, they needed to have a roll higher than the minimum for passing the encounter.

As for the Characters, they are a very unique part of the game. Within each player their stats and their specific stories are held. And there are methods for setting, adding, removing, and displaying each of these. This was a fairly simple use of objects with different data members.

The game loop is all contained within one method in the main class. This keeps track of what state that the game is currently running in and will only abort whenever to state reaches a certain value. Each state corresponds to different parts of the game (Menu, playing, etc).

Proposed Method / System Description / Implementation

Note: For this section, the UML diagram will be broken into blocks, in the appendices a full diagram can be seen.

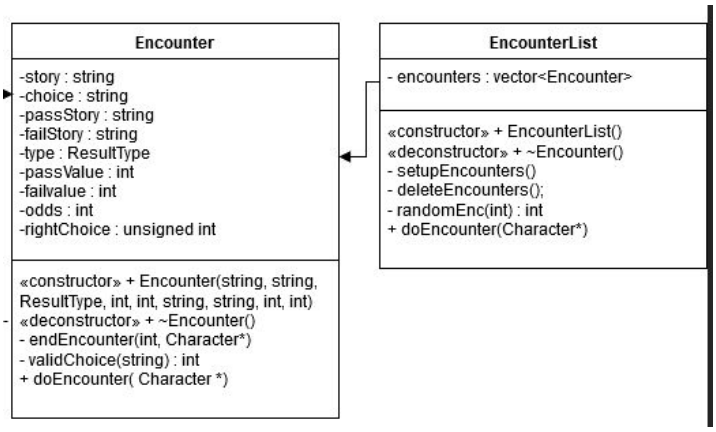


Figure 1 - Encounter and EncounterList UML

The bulk of the gameplay for Project Zombie would be captured in each of the random encounters that the player would face, so naturally, there was a lot of thought put into how these would play out. To store these, encounters are initialized as objects and then stored in a map within another object. Developing it this way gave us the ability to easily add and remove encounters from the list. To ensure that the players don't receive the same encounters during one game, after an encounter occurs, it will be removed from the list. All of the input and output for the players decisions, along with the associated stories would be handled within methods of the Encounter class. This data was allocated at the start of the game and then freed at the end of the runGame method.

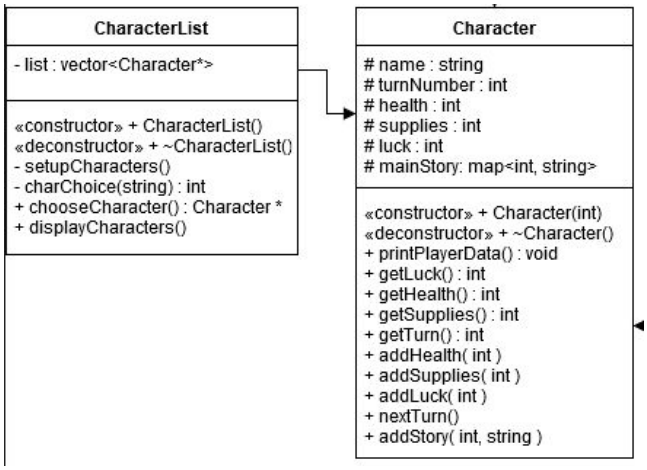


Figure 2 - Character and CharacterList UML

The Character class was designed to keep track of a selected player's stories and stats throughout the game. The players stories are held within a map, which we decided to use instead of a vector because we wanted to have an associated turn number with it and reference it by that. At the start of the game all six available players are generated and the players can select which they wish to use. At the initialization of each player, they are given a set of random stats (within a range) that will affect how their game will play. Characters have a health, luck, and supplies stat. The health and supplies stat essentially function the same, where if either ever reaches 0, the player will lose the game and the other player will win. The luck stat however is fixed for the lifetime of the Character. Luck factors in as part of the odds for each encounter, so players with a higher luck stat have a higher chance of passing encounters. Within this class we used operator overloading in order to change the functionality of this for the << and >> operators. This was so that whenever we wanted to write the player data to a file, the appropriate data members would be saved and we only have to write it once. The data for the character class was allocated at the start of the game within the player selection, and after the game ends, it is freed.

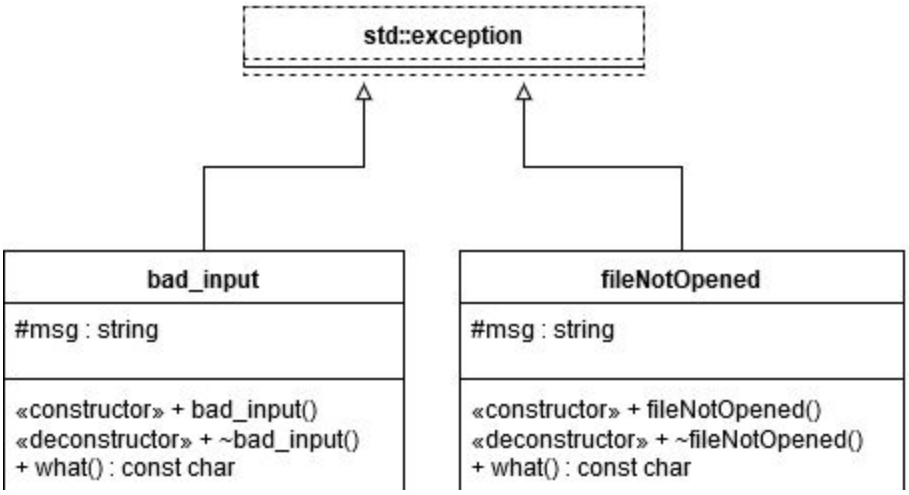


Figure 3 - Custom Exception Classes for input and file reading

At almost every step of the game, the players needed to be inputting data, whether an option for an encounter or on a menu. To handle this different functions and methods were developed for selecting. To ensure safe execution, all data was read in as strings and then tested before being converted into the value needed. If anything was wrong along the process, exception handling was usually used to throw errors that could be caught on a higher level. In addition to this error checking, files had to be saved, loaded, and erased throughout the application. To do this, files were always checked to be open and exceptions were thrown if they weren't. The game was automatically saved from within the runGame() method at the end of each turn, if the players did manage to reach turn 10, the game file would instead be erased, so that the players cannot relive the happy moment of winning.

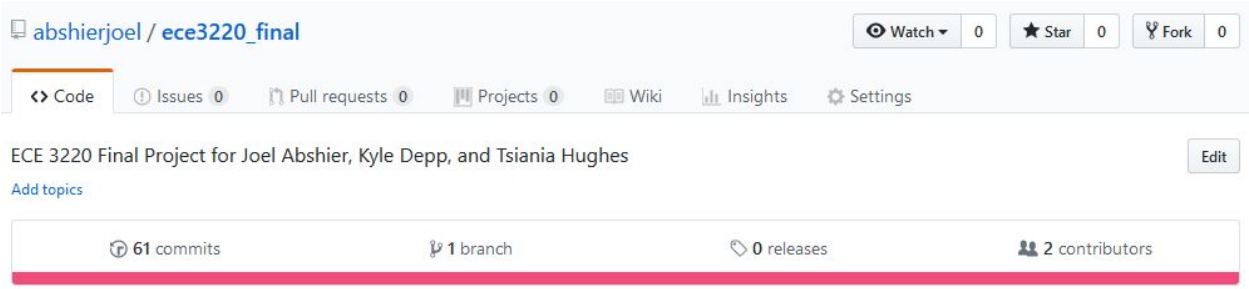


Figure 4 - GitHub Header

To manage the whole project we used a GitHub repository. This helped us be able to work while we were all in different locations without getting ever mixing up code and erasing newer versions. This also served as a great tool for communicating the changes that other team members were making.

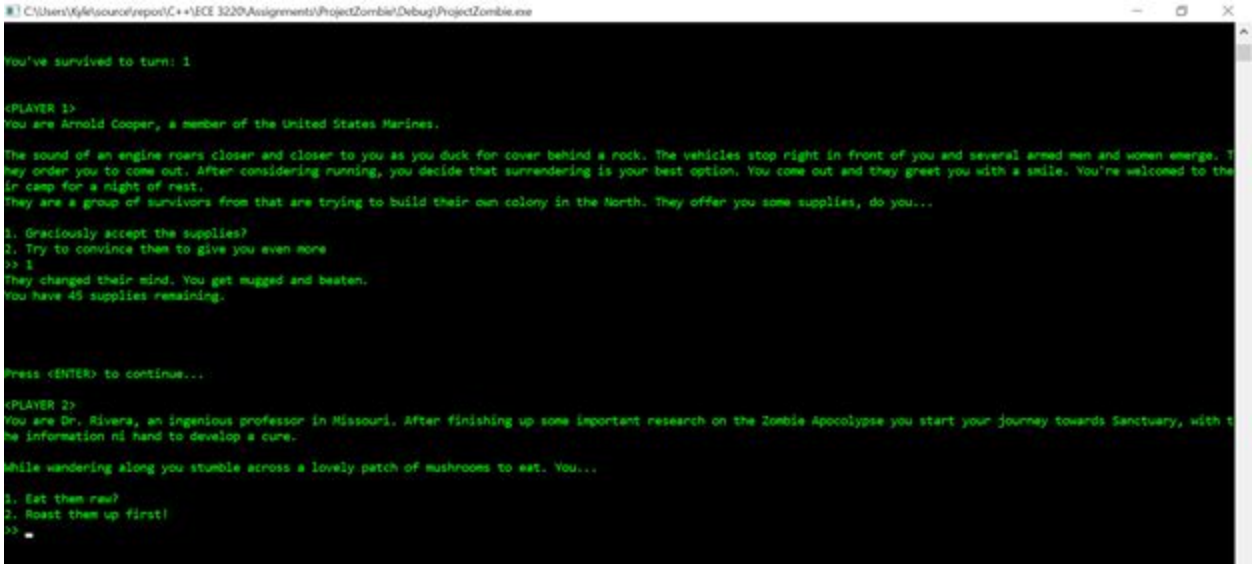
Experiments and Results

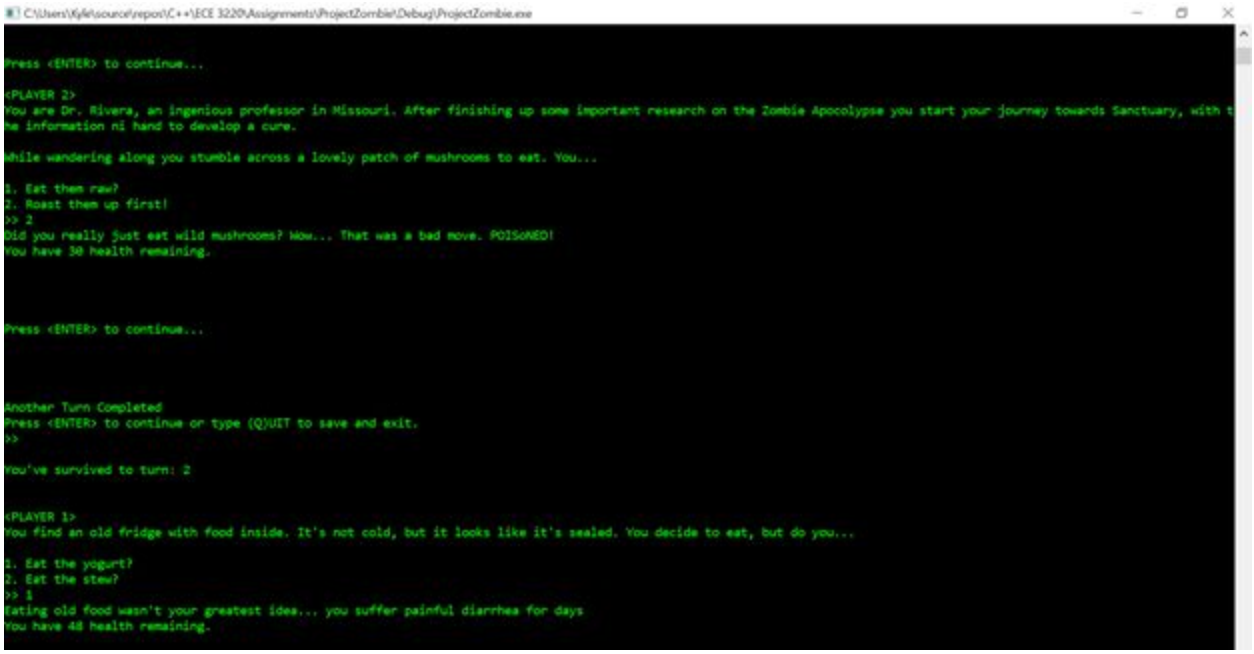
A lot of the testing was done after each method was created. Once we had a solid skelton we started adding more entities to the project. After the first iteration where there were two different versions, we decided to be careful from then on out. The system we have is extremely hard to break, after adding a lot of error checking after each user input to avoid the game crashing. Using GitHub we each had the most updated version between the 3 of us and was able to test it on our personal laptops, a lot of the testing was done by ourselves. Tsiania was able to have a few of her friends without any background knowledge of coding try to break it. Then get any feedback they had on the actual game itself, which is an important step in corporations to make sure the product does what's intended and user experience.

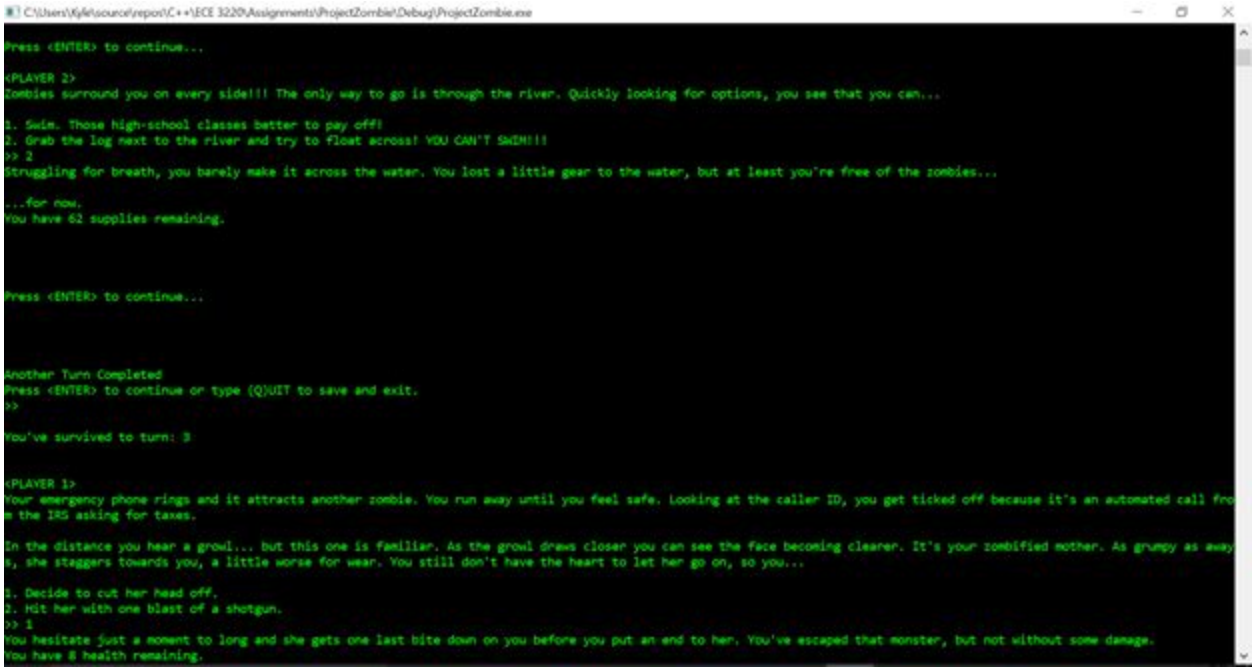
After getting the assigned partners we made it a weekly thing to meetup friday during the normal class time and discuss how the game was to be designed. We discussed how many people should be playing the game in one sitting (was one player and changed to two players), discuss possible classes (Character, Story, etc) and their functions within the game, and overall goals to be accomplished each week. The work was divided based on the availability of each

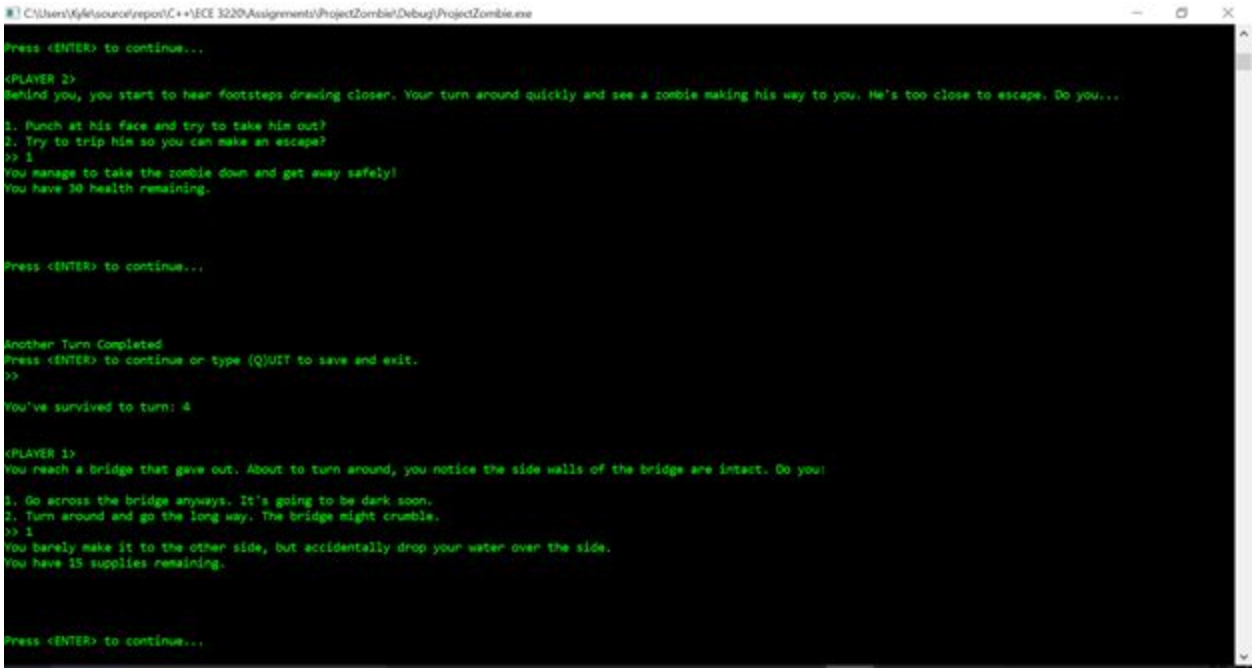
person, depending on if the person had other projects or testing. We met our goals at each friday checkmark for the most part until we had to crunch down for the 2 weeks. Then after the presentation we added a few more things to the code and checked it over.

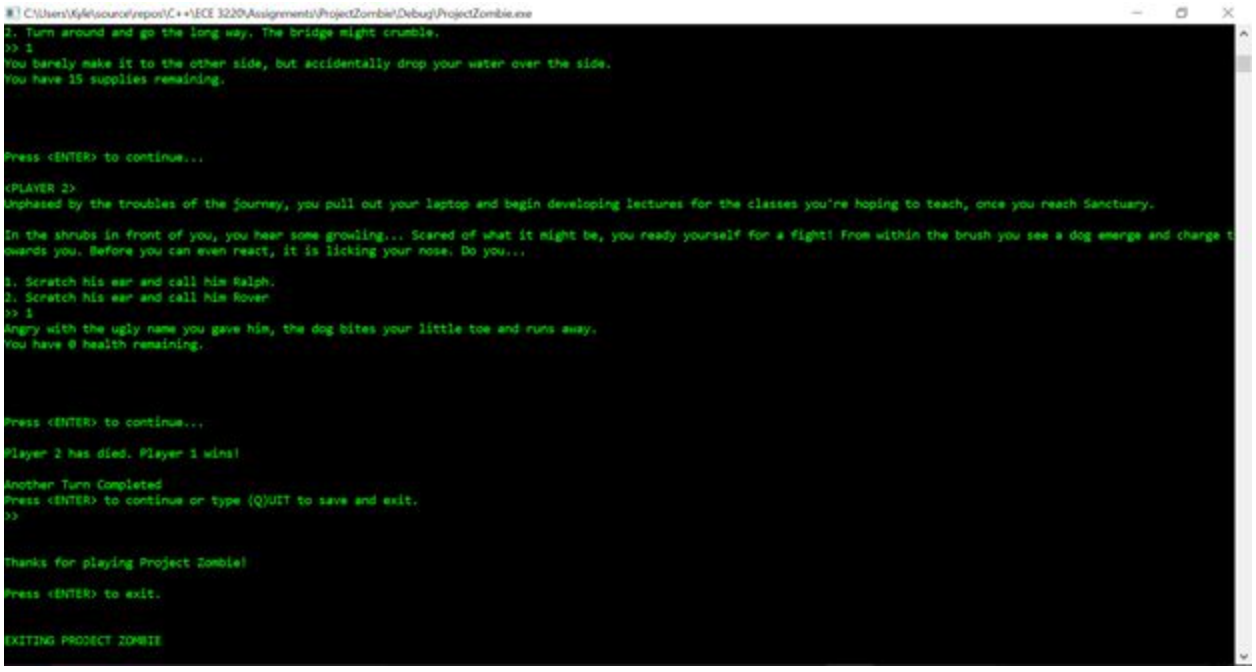

C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
//
//// Press <ENTER> to initiate Project Zombie ////
//
You've survived to turn: 1
<PLAYER 1>
You are Arnold Cooper, a member of the United States Marines.
The sound of an engine roars closer and closer to you as you duck for cover behind a rock. The vehicles stop right in front of you and several armed men and women emerge. T
hey order you to come out. After considering running, you decide that surrendering is your best option. You come out and they greet you with a smile. You're welcomed to the
ir camp for a night of rest.
They are a group of survivors from that are trying to build their own colony in the North. They offer you some supplies, do you...


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
You've survived to turn: 1
<PLAYER 1>
You are Arnold Cooper, a member of the United States Marines.
The sound of an engine roars closer and closer to you as you duck for cover behind a rock. The vehicles stop right in front of you and several armed men and women emerge. T
hey order you to come out. After considering running, you decide that surrendering is your best option. You come out and they greet you with a smile. You're welcomed to the
ir camp for a night of rest.
They are a group of survivors from that are trying to build their own colony in the North. They offer you some supplies, do you...
1. Graciously accept the supplies?
2. Try to convince them to give you even more
>> 1
They changed their mind. You get mugged and beaten.
You have 45 supplies remaining.
Press <ENTER> to continue...
<PLAYER 2>
You are Dr. Rivera, an ingenious professor in Missouri. After finishing up some important research on the Zombie Apocolypse you start your journey towards Sanctuary, with t
he information ni hand to develop a cure.
While wandering along you stumble across a lovely patch of mushrooms to eat. You...
1. Eat them raw?
2. Roast them up first!
>>


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
Press <ENTER> to continue...
<PLAYER 2>
You are Dr. Rivera, an ingenious professor in Missouri. After finishing up some important research on the Zombie Apocolypse you start your journey towards Sanctuary, with t
he information ni hand to develop a cure.
While wandering along you stumble across a lovely patch of mushrooms to eat. You...
1. Eat them raw?
2. Roast them up first!
>> 2
Did you really just eat wild mushrooms? Wow... That was a bad move. POISONED!
You have 30 health remaining.
Press <ENTER> to continue...
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
You've survived to turn: 2
<PLAYER 1>
You find an old fridge with food inside. It's not cold, but it looks like it's sealed. You decide to eat, but do you...
1. Eat the yogurt?
2. Eat the stew?
>> 1
Eating old food wasn't your greatest idea... you suffer painful diarrhea for days
You have 48 health remaining.


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
Press <ENTER> to continue...
<PLAYER 2>
Zombies surround you on every side!!! The only way to go is through the river. Quickly looking for options, you see that you can...
1. Swim. Those high-school classes better to pay off!
2. Grab the log next to the river and try to float across! YOU CAN'T SWIM!!!
>> 2
Struggling for breath, you barely make it across the water. You lost a little gear to the water, but at least you're free of the zombies...
...for now.
You have 62 supplies remaining.
Press <ENTER> to continue...
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
You've survived to turn: 3
<PLAYER 1>
Your emergency phone rings and it attracts another zombie. You run away until you feel safe. Looking at the caller ID, you get ticked off because it's an automated call fro
m the IRS asking for taxes.
In the distance you hear a growl... but this one is familiar. As the growl draws closer you can see the face becoming clearer. It's your zombified mother. As grumpy as away
s, she staggers towards you, a little worse for wear. You still don't have the heart to let her go on, so you...
1. Decide to cut her head off.
2. Hit her with one blast of a shotgun.
>> 1
You hesitate just a moment to long and she gets one last bite down on you before you put an end to her. You've escaped that monster, but not without some damage.
You have 8 health remaining.


C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
Press <ENTER> to continue...
<PLAYER 2>
Behind you, you start to hear footsteps drawing closer. Your turn around quickly and see a zombie making his way to you. He's too close to escape. Do you...
1. Punch at his face and try to take him out?
2. Try to trip him so you can make an escape?
>> 1
You manage to take the zombie down and get away safely!
You have 30 health remaining.
Press <ENTER> to continue...
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
You've survived to turn: 4
<PLAYER 1>
You reach a bridge that gave out. About to turn around, you notice the side walls of the bridge are intact. Do you:
1. Go across the bridge anyways. It's going to be dark soon.
2. Turn around and go the long way. The bridge might crumble.
>> 1
You barely make it to the other side, but accidentally drop your water over the side.
You have 15 supplies remaining.
Press <ENTER> to continue...

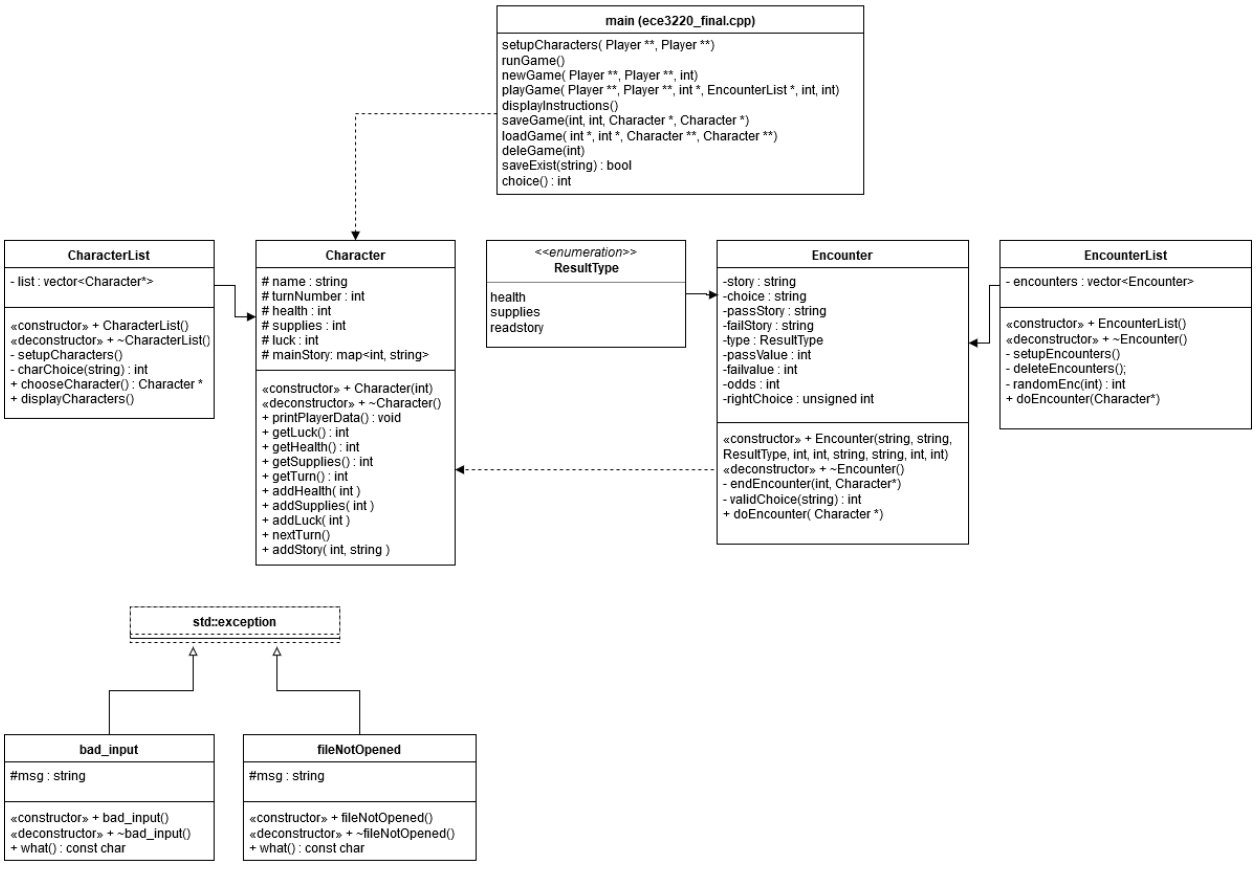

C:\Users\Kyle\source\repos\C++\ECE 3220\Assignments\ProjectZombie\Debug\ProjectZombie.exe
2. Turn around and go the long way. The bridge might crumble.
>> 1
You barely make it to the other side, but accidentally drop your water over the side.
You have 15 supplies remaining.
Press <ENTER> to continue...
<PLAYER 2>
Unphased by the troubles of the journey, you pull out your laptop and begin developing lectures for the classes you're hoping to teach, once you reach Sanctuary.
In the shrubs in front of you, you hear some growling... Scared of what it might be, you ready yourself for a fight! From within the brush you see a dog emerge and charge t
owards you. Before you can even react, it is licking your nose. Do you...
1. Scratch his ear and call him Ralph.
2. Scratch his ear and call him Rover
>> 1
Angry with the ugly name you gave him, the dog bites your little toe and runs away.
You have 0 health remaining.
Press <ENTER> to continue...
Player 2 has died. Player 1 wins!
Another Turn Completed
Press <ENTER> to continue or type (Q)UIT to save and exit.
>>
Thanks for playing Project Zombie!
Press <ENTER> to exit.
EXITING PROJECT ZOMBIE

Discussion and Conclusions

The hardest part of the program to write was probably the save/load feature. Once we figured out how to get the save.dat files to load up into the game, everything went a lot more smoothly. Our original system was supposed to be on Microsoft's Visual Studio, however after the first iteration we decided that it needed to be more cross compatible so that Eclipse can run it without adding any additional libraries. While the rest of the code itself wasn't as hard to write, the story took a long time to create so that it felt like a "whole game" instead of just something that you play once and then have seen everything.

Appendices

Full UML Diagram



Appendices: Project Files

Below is all of the source code files for the project.

```
///************** ece3220_final.cpp | Main File ************///

//=========================================================================
=====
// Assignment            : ECE 3220: Final Project
// Authors                        : Joel Abshier, Kyle Deppe, Tsiania Hughes
//=========================================================================
=====

#include <iostream>
#include <fstream>
#include <sstream>
#include <cstdlib>
#include <ctime>
#include "Character.h"
#include "Encounter.h"
#include "Exceptions.h"

using namespace std;

void setupCharacters(Character **, Character **);
void runGame();
void newGame(Character **, Character **, int * );
void playGame(Character *, Character *, int * , EncounterList *, int, unsigned int = 0);
void displayInstructions();
void saveGame( int turn, int save, Character * player1, Character *player2 ) throw (
fileNotOpened );
void loadGame( unsigned int * turn, int * save, Character ** player1, Character ** player2 ) throw
( fileNotOpened );
void deleGame( int save );

string Choice(string &choiceString) throw ( bad_input );


/*
 * setupCharacters
 *                ARGS: Character **, Character **
 *                RTRN: void
 *                DESC: This function is designed simply to take in the two double-pointers to
Character objects. It will creae an instance of the character
 *                list, which will generate a all 6 Character objects. Then the chooseCharacter
method will be called, returning the point to each character
 *                selected
```

```
 */
void setupCharacters(Character ** player1, Character ** player2)
{
	CharacterList * charList = new CharacterList();                                    // This
needs a delete() still

	cout << endl << "<PLAYER 1>";
	Character * NewPlayer1 =  charList->chooseCharacter();

	cout << endl << "<PLAYER 2>";
	Character * NewPlayer2 =  charList->chooseCharacter();

	*player1 = NewPlayer1;
	*player2 = NewPlayer2;

	delete charList;
}

/*
 * runGame
 *		ARGS: none
 *		RTRN: void
 *		DESC: This is the function that is essentially designed to run the main menu and
call the other functions/methods. It's the controller.
 *		It has the following structure. The outer section is checking the gamestate, there
are 4, a gamestate for the menu, a gamestate for playing the game,
 *		a gamestate for the game ending, and a gamestate for winning/losing the game.
 *		Within the gamestate there are other menus that will operate and other methods
that will be called
 */
void runGame()
{
	string choiceString = "0";
	int gameState = 0;
	int menuChoice = 0;
	int saveSel = 0;
	unsigned int resTurn = 0;

	//Create pointers for the two character objects
	Character * player1 = NULL, *player2 = NULL;

	//At the start of every game a fresh list of encounters will be generated
	EncounterList * encounters = new EncounterList();
```

```
	//If the gamestate is not in ending mode, run the application
	while ( gameState != -1 )
	{
		//Main menu state
		if( gameState == 0 )
		{
			//Validate that a proper choice has been made
			try
			{
				choiceString = Choice(choiceString);
			}
			catch( bad_input & e )
			{
				cout << e.what() << endl;
				choiceString = "0";
			}
			menuChoice = stoi(choiceString);

			/*
			 * The menu options:
			 * 1. Start a new game
			 * 2. Load a game (this also runs the first turn with playGame
			 * 3. Display instructions/backstory
			 * 4. Exit the game
			 */
			switch( menuChoice )
			{
				case 1:
					gameState = 1;
					newGame( &player1, &player2, &saveSel );
					break;
				case 2:
					try
					{
						//Load the save file, this method will put data in each pointer's value
						loadGame( &resTurn, &saveSel, &player1, &player2 );

						//Run the first turn of the game from here, as the game will not begin until there is another loop.
```

```
                                        playGame( player1, player2, &gameState,
encounters, saveSel, resTurn );

                                        //Set the gamestate to the playing game state
                                        gameState = 1;
                                }
                                catch( ... )
                                {
                                        gameState = 3;
                                }
                                break;
                        case 3:
                                //Call the method to show instructions
                                displayInstructions();
                                break;
                        case 4:
                                //End the game
                                gameState = 3;
                                break;
                }

        }
        //This is the playing game state of the game
        else if ( gameState == 1 )
        {
                //This is where all the gameplay actually takes place
                playGame( player1, player2, &gameState, encounters, saveSel );

                //End of turn
                cout << endl << "Another Turn Completed" << endl <<
                                "Press <ENTER> to continue or type (Q)UIT to save and
exit." << endl;

                //Get String for Next Turn. IF it's quit, then save and quit, otherwise go to
next turn.
                string nextTurn;
                cout << ">> ";
                getline(cin, nextTurn);
                cout << endl;

                //If they choose to quit, set the gamemode ot shutdown
                if( (nextTurn == "QUIT") || (nextTurn == "Q") || (nextTurn == "quit") ||
(nextTurn == "q") )
```

```
                        {
                                gameState = 2;
                        }

                }
                //Player Died Gamestate
                else if ( gameState == 2 )
                {
                        gameState = -1;
                }
                //WIN GAME state
                else if ( gameState == 3 )
                {

                        //Delete the save
                        deleGame( saveSel );

                        //Say Goodbye!
                        cout << endl << "Thanks for playing Project Zombie!" << endl;
                        //TODO: Add cake. There will be cake.
                        gameState = -1;

                }
        }

        //When all is lost...
        delete( player1 );
        delete( player2 );
        delete( encounters );

}

/*
 * saveGame
 *              ARGS: int, int, Character *, Character *
 *              RTRN: void
 *              DESC: This function takes in most of the runtime data specific to each game
(except for encounters) and writes them to a file
 *              that is associated with the slot selected when playing the game.
 */
void saveGame( int turn, int save, Character * player1, Character *player2 ) throw (
fileNotOpened )
{
```

```
	//Build the file name
	string fileName = "slot" + to_string(save) + ".zom";
	//cout << "Saving " << fileName << endl;

	//Create new file input stream
	ofstream write;
	write.exceptions ( ofstream::failbit | ofstream::badbit );

	try
	{
		write.open( fileName );

		//Write the turn number to the file
		write << turn << endl;

		//Create references os that we can use an operator
		Character &p1 = *player1;
		Character &p2 = *player2;

		//Write the data for each player to the file
		write << p1;
		write << p2;

		write.close();
	}
	catch ( ifstream::failure & e ) {
		//The file could not be read
		std::cerr << "Exception opening/reading/closing file\n";
		cout << e.what() << endl;
	}
	catch(...)
	{
		std::cerr << "File Could Not Be Opened!\n";
	}

}

/*
 * deleGame
 *		ARGS: int
 *		RTRN: void
 *		DESC: This function simply deletes the file with the given number associated.
This is called when the players finish the game
```

```
 */
void deleGame( int save )
{

	string fileName = "slot" + to_string( save ) + ".zom";
	remove( fileName.c_str() );

}

/*
 * saveExist
 *		ARGS: string
 *		RTRN: bool
 *		DESC: This is a simply yes/no, utility function to check whether a file with the
given name exists. It's used when checking save slots
 */
bool saveExist( string name )
{
	if ( ifstream( name ) )
	{
	    return true;
	}
	return false;
}

/*
 * loadGame
 *		ARGS: unsigned int *, int *, Character **, Charcater **
 *		RTRN: void
 *		DESC: Very similar to the save function, this function takes in pointers to most of
the game data, and instead of saving them to a file
 *		with the associated slot number, it loads the data from that, if it exists.
 */
void loadGame( unsigned int * turn, int * save, Character ** player1, Character ** player2 ) throw
( fileNotOpened )
{

	//This is nice to know :)
	cout << endl << "Loading Game..." << endl;

	//Choose a save slot
	cout << "Which game would you like to load?\n" << endl;
```

```
	//For each save slot, check if there is a save that exists, and let the user know
	for( int i = 1; i <= 3; i++ )
	{
		string file = "slot" + to_string(i) + ".zom";
		if( saveExist( file ) )
		{
			cout <<  to_string(i) + ". Slot " + to_string(i) + " - SAVE EXISTS\n";
		}
		else
		{
			cout << to_string(i) +  ". Slot " + to_string(i) + " - EMPTY\n";
		}
	}

	//This next block of code is just recieving input from the user and putting that value in the saveSlot variable.
	string choiceString = "0";
	cout << ">> ";
	getline(cin, choiceString);

	//There are only three slots allowed. Make sure they picked one of those
	while( (choiceString != "1") && (choiceString != "2") && (choiceString != "3") )
	{
		cout << "Not a valid save slot. Try again: " << endl;
		getline(cin, choiceString);
	}

	//Change the saveslot for the current game
	*save = stoi( choiceString );
	string fileName = "slot" + to_string( *save ) + ".zom";


	//Create new file input stream
	ifstream read;
	read.exceptions ( ifstream::failbit | ifstream::badbit );

	//Try opening the file for reading
	try
	{
		read.open( fileName );

		read >> *turn;
```

```
                //Create a new Player 1 and populate data
                Character * save1 = new Character();
                Character * save2 = new Character();

                read >> *save1;
                read >> *save2;

                read.close();

                cout << "Loaded Players" << endl;
                //save1->printPlayerData();
                //save2->printPlayerData();

                *player1 = save1;
                *player2 = save2;

        }
        catch ( ifstream::failure & e ) {
                //The file could not be read
                std::cerr << "Exception opening/reading/closing file\n";
                cout << e.what() << endl;
                throw fileNotOpened();
        }
        catch(...)
        {
                std::cerr << "File Could Not Be Read!\n";
                throw fileNotOpened();
        }

        cout << "Game Loaded!\n" << endl;

}

/*
 * newGame
 *              ARGS: Character **, Character **, int *
 *              RTRN: void
 *              DESC: As the name would suggest, this function is called whenever the player
starts a new game. At first it asks the player which save
 *              slot they will be using. Then it continues to call the setupCharcters function so
that each player can select their desired characters.
 *              Lastly, and most importantly, it prints out one heck of a zombie!!!!
 */
```

```
void newGame(Character ** player1, Character ** player2, int * save )
{
	string buffer = "";
	cout << endl << "Creating a new game..." << endl;

	//Choose a save slot
	cout << "Which file would you like to save to?\n";

	//For each save slot, check if there is a save that exists, and let the user know
	for( int i = 1; i <= 3; i++ )
	{
		string file = "slot" + to_string(i) + ".zom";
		if( saveExist( file ) )
		{
			cout <<  to_string(i) + ". Slot " + to_string(i) + " - SAVE EXISTS\n";
		}
		else
		{
			cout << to_string(i) +  ". Slot " + to_string(i) + " - EMPTY\n";
		}
	}

	string choiceString = "0";
	cout << ">> ";
	getline(cin, choiceString);
	while( (choiceString != "1") && (choiceString != "2") && (choiceString != "3") )
	{
		cout << "Not a valid save slot. Try again: " << endl;
		getline(cin, choiceString);
	}

	*save = stoi( choiceString );

	cout << "Slot " << *save << " selected!" << endl;


	setupCharacters(player1, player2);

	cout << endl << endl << endl << "<PLAYER 1>";
	(**player1).printPlayerData();
	cout << endl << "<PLAYER 2>";
	(**player2).printPlayerData();
```

```
        cout << endl << endl << endl << "                ////////////////////////////////////////////////////////";
        cout << endl << "               /////   Press <ENTER> to initiate Project Zombie   /////";
        cout << endl << "               ////////////////////////////////////////////////////////";
        getline(cin, buffer);

        cout << endl << endl << "                              .~#########%%;~.";
        cout << endl << "                             /############%%;`\\";
        cout << endl << "                             /############%%;,;,\\";
        cout << endl << "                            |##############;;;;.,.|";
        cout << endl << "                            |############%;;;;;.,.|";
        cout << endl << "                      XX       |##/~~\\####%;;;/~~\\;,|       XX";
        cout << endl << "                     XX..X      |#|  o  \\##%;/  o  |.|      X..XX";
        cout << endl << "                    XX.....X     |##\\\\____/##%;\\\\____/.,|     X.....XX";
        cout << endl << "                  XXXXX.....XX      \\\\#########/\\;;;;;;,, /      XX.....XXXXX";
        cout << endl << "                 X |......XX%,.@      \\\\######/%;\\\\;;;;, /      @#%,XX......| X";
        cout << endl << "                 X |.....X  @#%,.@     |######%%;;;;,.|     @#%,.@     X.....| X";
        cout << endl << "                 X  \\\\...X     @#%,.@    |# # # % ; ; ;,|   @#%,.@       X.../  X";
        cout << endl << "                  X# \\\\.X       @#%,.@                @#%,.@          X./  #";
        cout << endl << "                   ##  X         @#%,.@             @#%,.@            X   #";
        cout << endl << "                  , \"# #X          @#%, .@          @#%, .@           X ##";
        cout << endl << "                    `###X           @#%,.@      @#%,.@            ####'";
        cout << endl << "                   . ' ###          @#%.,@  @#%,.@             ###`'";
        cout << endl << "                    . \"; \"           @#%.@#%,.@              ;\"` ' ." ;
        cout << endl << "                     '               @#%,.@                 ,.";
        cout << endl << "                        ` ,          @#%,.@  @@              `'";
        cout << endl << "                                     @@@  @@@ ";

        cout << endl  << endl << endl << endl;;
}

/*
 * playGame
 *              ARGS: Charcter *, Character *, int *, EncounterList *, int, unsigned int
 *              RTRN: void
 *              DESC: Once again, as the naem might suggest, this function is where the main
game happens. All of the action of encounters and turns
 *              and making decisions is called from right here. At the start of each turn it will
attempt to display the story for each player on this turn
 *              and then try to give each player their encounter for this turn, from the list of
encounters.
 *              If, after the encounters, either players are dead (or both), the endgame
gamestate will be triggered and the game will end
 *              If the players reach turn 10, the game will output their last stories, and then end.
```

```
 *                  At the end of each turn, the game will save.
 */
void playGame( Character * player1, Character * player2, int * gameState, EncounterList *
encounters, int save, unsigned int resTurn )
{

        static unsigned int gameTurn = resTurn;

        //Start Turn Message
        cout << "You've survived to turn: " << gameTurn + 1 << "\n\n" << endl;

        //Player 1 Turn
        cout << "<PLAYER 1>" << endl;
        player1->showStory( gameTurn );
        //Player 1's Random Encounter
        encounters->doEncounter( player1 );

        //Player 2 Turn
        cout << "<PLAYER 2>" << endl;
        player2->showStory( gameTurn );
        encounters->doEncounter( player2 );

        //IF PLAYERS ARE DEAD
        if( ( player1->getHealth() == 0 ) || ( player1->getSupplies() == 0 ) )
        {
                cout << "Player 1 has died. Player 2 wins!" << endl;
                *gameState = 3;
                return;
        }
        else if( ( player2->getHealth() == 0 ) || ( player2->getSupplies() == 0 ) )
        {
                cout << "Player 2 has died. Player 1 wins!" << endl;
                *gameState = 3;
                return;
        }

        ++gameTurn;

        if( gameTurn == 10 )
        {
                //Player 1 Final Story
                cout << "<PLAYER 1>" << endl;
                player1->showStory( gameTurn );
```

```
                //Player 2 Final Story
                cout << "<PLAYER 2>" << endl;
                player2->showStory( gameTurn );

                //Set gamestate to the ending state!
                *gameState = 3;
        }
        else
        {

                saveGame( gameTurn, save, player1, player2 );

        }

        cout << endl << endl;

}

/*
 * displayInstructions
 *              ARGS: none
 *              RTRN: void
 *              DESC: This is a simple function to display some instructions for the game and
some backstory.
 */
void displayInstructions()
{
        cout << endl << "Instructions:"
                << endl << "The world has been overrun by zombies. Last night a radio
broadcast announced that"
                << endl << "there's a safe haven in San Diego, California. Now, all of the
survivors in America"
                << endl << "are heading west. In the beginning the players will choose a
character to be"
                << endl << "assigned health, supplies, and luck. During the game the players'
decisions"
                << endl << "impact their stats, where decisions that match the character's
personality"
                << endl << "tend to favorably increase stats the most. The game ends when a
player's health has"
                << endl << "reached 0 or both players get to San Diego."
                << endl;
```

```
}

/*
 * Choice
 *		ARGS: string
 *		RTRN string
 *		DESC: This is the function that runs the main menu of the game. It outputs the
options and then does error checking until the user
 *		selects a proper option.
 */
string Choice(string &choiceString) throw (bad_input)
{
	cout << endl << "Please select an option: "
		<< endl << "1. Play Game"
		<< endl << "2. Load Game"
		<< endl << "3. See Instructions"
		<< endl << "4. Exit"
		<< endl << ">> ";

	getline(cin, choiceString);

	if( (choiceString != "1") && (choiceString != "2") && (choiceString != "3") && (choiceString
!= "4") )
	{
		throw bad_input();
	}

	return choiceString;

}

/*
 * main
 *		DESC: The all important main function!! This function calls runGame() and then
asks the user to hit enter once the game ends...
 */
int main(void)
{
	string buffer = "";

	srand(time(NULL));
	cout << "PROJECT ZOMBIE" << endl << endl << "Welcome to Project Zombie";
```

```
	//LAUNCH PROJECT ZOMBIE!!!!!
	runGame();

	//Pressing enter says goodbye forever!!!
	cout << endl << "Press <ENTER> to exit.";
	getline(cin, buffer);

	cout << endl << endl << "EXITING PROJECT ZOMBIE" << endl;

	return EXIT_SUCCESS;
}



///********** Character.h **********///

#pragma once

#ifndef CHARACTERS_CHARACTER_H_
#define CHARACTERS_CHARACTER_H_

#include <string>
#include <map>
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

class ABCCharacter {
protected:
	string name;
	int health;		//A number between 0-100
	int supplies;	//A number between 0-100
	int luck;		//A number between 0-100 that increases the likelyhood of
passing checks
	std::map<int, string> mainStory;		// I didn't use this. It might be useful but it
seems like we can just hard code our stories.
											// (See
Character.cpp)
public:
	ABCCharacter() {}
	~ABCCharacter() {}
```

```
	virtual int getLuck() = 0;
	virtual int getHealth() = 0;
	virtual int getSupplies() = 0;
	// virtual int getTurn() = 0;
	virtual string getName() = 0;

	virtual void addHealth(int number) = 0;
	virtual void addSupplies(int number) = 0;
	virtual void addLuck(int number) = 0;
	virtual void showStory(int turnNumber) = 0;

	virtual void addStory(int turn, string story) = 0;
};

class Character : protected ABCCharacter {
protected:
public:
	Character() {}
	Character( string _name, int _health, int _supplies, int _luck );
	~Character();

	void printPlayerData();

	int getLuck();
	int getHealth();
	int getSupplies();
	// int getTurn();
	string getName();

	void addHealth(int number);				// Input a negative number for subtraction. Auto sets to 0 or 100 for extreme values.
	void addSupplies(int number);
	void addLuck(int number);
	void showStory( int turnNumber );

	void addStory(int turn, string story);

	//These are operator overloads for when this object is written to a file. This will make it so that the data for the character will be written
	friend std::ostream& operator<<(std::ostream & os, const Character & player )
	{
		os << player.name << '\n';
		os << player.health << '\n';
```

```
		os << player.luck << '\n';
		os << player.supplies << '\n';
		return os;
	}

	//These are operator overloads for when this object is loaded from a file. This willl load each line of the file into this object.
	friend std::istream& operator>>(std::istream & is, Character & player)
	{
		string readName;
		getline( is, readName );
		getline( is, player.name, '\n' );
		is >> player.health;
		is >> player.luck;
		is >> player.supplies;
		return is;
	}

};

class CharacterList
{
	private:
		vector <Character*> list;
		void setupCharacters();
		int charChoice( string choice ) throw( int );
	protected:
	public:
		CharacterList();
		~CharacterList();
		Character* chooseCharacter();
		void displayCharacters();
};

#endif /* CHARACTERS_CHARACTER_H_ */



///********** Character.cpp ***********///

#include "Character.h"
```

```
/*
 * Character constructor!
 * To make a character you have to put in a name, and the other values are optional
 */
Character::Character( string _name, int _health = 100, int _supplies = 10, int _luck = 50 )
{
	name = _name;
	health = _health;
	supplies = _supplies;
	luck = _luck;
}

Character::~Character()
{
}

/*
 * printPlayerData
 *		This is just a nifty method used at the start of the game to display the random
stats
 */
void Character::printPlayerData()
{
	cout << endl << name << "'s Stats" << endl
			<< "Health: " << health << endl
			<< "Supplies: " << supplies << endl
			<< "Luck: " << luck << endl;
}

//Return the luck
int Character::getLuck()
{
	return luck;
}

//Return the health
int Character::getHealth()
{
	return health;
}

//Return the supplies
int Character::getSupplies()
```

```
{
	return supplies;
}

//Return the name
string Character::getName()
{
	return name;
}

/*
 * showStory
 *		This is a simple method that takes in the turn number and checks to see if there is an existing record in the mainStory map. If there is, it gets printed
 */
void Character::showStory( int turnNumber )
{
	std::map<int, string>::const_iterator iter;

	//Check through the entire map
	for ( iter = mainStory.begin(); iter != mainStory.end(); iter++ )
	{
		//If there is a record with a key equal to the turn number
		if( iter->first == turnNumber )
		{
			//Show that story!
			cout << iter->second << "\n" << endl;
			return;
		}
	}

}

/*
 * addHealth
 *		A setter used for adding and subtracting health from the player. This will not let the health exceed 100 or go below 0
 */
void Character::addHealth(int number)
{
	//If health would be below 0 with addition
	if ((health + number) < 0)
	{
```

```
		health = 0;
	}
	//If health would be above 100 with addition
	else if ((health + number) > 100)
	{
		health = 100;
	}
	else
	{
		health = health + number;
	}
}

/*
 * addSupplies
 *		A setter used for adding and subtracting supplies from the player. This will no let the supplies exceed 100 or go below 0
 */
void Character::addSupplies(int number)
{
	//If supplies would be below 0 with addition
	if ((supplies + number) < 0)
	{
		supplies = 0;
	}
	//If supplies would be greater than 100
	else if ((supplies + number) > 100)
	{
		supplies = 100;
	}
	else
	{
		supplies = supplies + number;
	}
}

/*
 * addLuck
 *		A setter used for adding and subtracking luck from the player. This will not let the luck exceed 100 or go below 0
 */
void Character::addLuck(int number)
{
```

```
	//If the luck would be less than 0
	if ((luck + number) < 0)
	{
		luck = 0;
	}
	//If the luck would be greater than 100
	else if ((luck + number) > 100)
	{
		luck = 100;
	}
	else
	{
		luck = luck + number;
	}
}

/*
 * addStory
 *		A simple method used for adding a story element associated with a key (turn) to the map
 */
void Character::addStory( int turn, string story )
{
	//Add story to map
	mainStory.emplace( turn, story );
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


CharacterList::CharacterList()
{
	setupCharacters();
}

CharacterList::~CharacterList()
{
	int size = list.size();
	int i = 0;

	for (i = 0; i < size; i++) {
		delete list[i];
	}
```

```
}

void CharacterList::setupCharacters()
{
	//Here is where we can write all of the charcter information, and we'll only have to do it
once

	// Make sure to write a delete() for all of these.
	Character * arnoldCooper = new Character( "Arnold", 50 + (rand() % 41), 50 + (rand() %
41), 35 + (rand() % 41) );
	list.push_back( arnoldCooper );

	Character * clarkKent = new Character( "Clark", 50 + (rand() % 41), 50 + (rand() % 41),
35 + (rand() % 41) );
	list.push_back( clarkKent );

	Character * drRivera = new Character( "Luis Rivera", 50 + (rand() % 41), 50 + (rand() %
41), 35 + (rand() % 41) );
	list.push_back( drRivera );

	Character * lilPupper = new Character( "Lil' Pupper", 50 + (rand() % 41), 50 + (rand() %
41), 35 + (rand() % 41) );
	list.push_back( lilPupper );

	Character * abigailWillow = new Character( "Abigail", 50 + (rand() % 41), 50 + (rand() %
41), 35 + (rand() % 41) );
	list.push_back( abigailWillow );

	Character * darwinArnold = new Character( "Darwin", 50 + (rand() % 41), 50 + (rand() %
41), 35 + (rand() % 41) );
	list.push_back( darwinArnold );



	//Basically, right here we can add story elements into the game.
	/*
	 * FORMAT:
	 * (name).addStory( TURN_NUMBER, STORY );
	 *
	 * HERE'S THE IMPORTANT THIS:
	 *	Not every turn needs a story. These are the unique story elements for each
character, but each encounter also includes some story specific
	 *	to that encounter.
	 */
```

```
//ARNOLD'S STORIES!
arnoldCooper->addStory(0, "You are Arnold Cooper, a member of the United States Marines.");
arnoldCooper->addStory(2, "Your emergency phone rings and it attracts another zombie. You run away until you feel safe. Looking at the caller ID, you get ticked off because it's an automated call from the IRS asking for taxes.");
arnoldCooper->addStory(4, "You've reached a door and it's locked. Unfolding your handy dandy swiss army knife, you kneel down to unlock it. Crap, you don't remember how to do this. So you kick the door the heck down and walk on in.");
arnoldCooper->addStory(7, "Reaching a rural area, there's a rustle in the woods. Ignoring it, you keep going. But the rustling doesn't stop. Panicking, you break into a sprint. "
" Whatever is after you, it's right on your heels. You keep running until you trip and fall on your face. The creature trips over you and breaks its own neck on the fall. Well that was close.");
arnoldCooper->addStory(10, "You've reached salvation. Walking right up to the base, you get greeted at the front gate and walk on in. Welcome to San Diego.");

//CLARK KENT'S STORIES
clarkKent->addStory(0, "You are Clark Kent, a journalist from New York City.");
clarkKent->addStory(2, "You look up to the sky, and see a bright light. It crashes into the ground next to you. That's weird. You ignore it and keep on going.");
clarkKent->addStory(4, "Nighttime arrives. Reaching an urban area, you walk slowly, searching for a place to find food. When suddenly, BAM, some weird guy"
" in a bat suit comes out of no where and slams his fist into your face. How rude. You snap his neck and keep looking for food."
" It has never occurred to you before, but you realize you're kind of ridiculously strong.");
clarkKent->addStory(7, "A zombie steps into your path. You blink a little more firmly than usual and he disintegrates at your feet. And everything behind him"
" for about ten feet is a path of destruction. Woops. Now, back to whatever I was doing...");
clarkKent->addStory(10, "You look up ahead and see your destination: San Diego. Taking a step forward, you remember something. You're freaking Superman!"
" This realization elates you with energy. Taking a deep breath, you prepare for what you must do. Grabbing the ground firmly with"
" two hands, you pull and stretch it apart. Muscles aching, you pull harder and harder with the force of a billion men until, finally, the Earth"
" breaks in half. That oughta do it. Content, you fly out into space to go live with some other alien race that's better than humanity.");
```

```
//LUIS RIVERA'S STORIES
drRivera->addStory(0, "You are Dr. Rivera, an ingenious professor in Missouri. After finishing up some important research on the Zombie Apocolypse you start your journey towards Sanctuary, with the information ni hand to develop a cure.");
drRivera->addStory(3, "Unphased by the troubles of the journey, you pull out your laptop and begin developing lectures for the classes you're hoping to teach, once you reach Sanctuary. " );
drRivera->addStory(5, "Lying down for another night of rest you hear a noise creeping towards you. Turning switftly and arming yourself, you look up to see a figure moving towards you. As the moonlight reaches its "
                      "face you finally recognize the face of none other than Tushar! He comes forward and greets you with a hug and the two of you sit down for a night of debating whether or not Eclipse is a good IDE."  );
drRivera->addStory(8, "Nearing your journey's end you start to feel very excited! You're going to save the world. The cure to this disease is in the palm of your hand!" );
drRivera->addStory(10, "In the distance you can see the faint outline of a wall! There it is! You move forward as fast as you can until you reach the wall. You're welcomed inside and you begin your new life developing computers in this world without electricity. "
                       "The new R-Box 9001 features state of the art features like binary. Running of treadmill power, your new machines push the colony into the best tech of the 20th century! "
                       "You feel like a hero as you finally finish the vaccination and begin distributing it to all of Sanctuary...\n\nTwo months later everyone has been vaccinated. Unfortunately, the proper sanitization wasn't used and most of the colony died from infection. Leaving only a humans left to be the end of humanity.");


//LIL' PUPPER'S STORIES
lilPupper->addStory( 0, "You are lil' pupper! You're the cutest 'lil guy in town <3, but underneath that fluffy coat of fur, you have vicious, survival skills.");
lilPupper->addStory( 2, "The zombie apocolypse seems completely wonderful to you! As the jolly 'lil doggy you are, you trot along over the hills towards Sanctuary." );
lilPupper->addStory( 4, "You've made it so far! You yelp and bark with so much joy and reward yourself by having a good 'ol time chasing your 'lil tail!" );
lilPupper->addStory( 7, "Although the zombie growls and delicious smells are alluring to you, you point your 'lil nose forward and push ahead, determined to wag your 'lil tail in safety once again!" );
lilPupper->addStory( 10, "The smell of good 'ol Purina Puppy Chow (not a sponsor) fills your nostrils as you approach the gate to Sanctuary. You're welcomed in and the humans there love you."
```

"\nFor weeks they can't seem to keep their hands off of you, and you love it! You wag your tail and relax day and night. Then one day "
"as you're reclining by the fire butcher Doug comes over to play with you. He throws your toy over into his shop and you chase it down "
"to bring back and make him proud. But as you go into the shop and start sniffing around, you hear the door close and Butcher Doug picks you up "
"to cuddle. You're loving it as he scratches your ears and puts your chubby, puppy body on his table."
);

//ABIGAIL WILLOW'S STORIES
//ABIGAIL WILLOW'S STORIES
abigailWillow->addStory(0, "You are Abigail Willow, a student at Westhall Middle School. You're 13 years old carrying only whats in your backpack and your phone."
"The Zombies don't wait for you to get home before they attack. You escape the building and its chaos. From a crashed car next to the building you hear from the radio about San Dieg0 "
"trying to save the human race. Without a doubt start your adventure to San Diego.");
abigailWillow->addStory(2, "As you take a breather you see a boy running towards you shouting. 'Run!!!' he says and soon you see why, a horde of zombies make there way around the corner he ran from."
"You follow close behind him until you both take shelter in a house that was left open. 'I'm Alex' he says once we caught our breath. Afterward you both head you seperate ways. You only ddepend on yourself");
abigailWillow->addStory(4, "You are halfway there, it's only been a few weeks but you can feel how little there is left of humanity. You are seeing less stragglers and the ones who you do see you avoid."
"People are desperate");
abigailWillow->addStory(7, "You soon realize how bleak and dangerous the world has become. You never really stopped to think on your parents or friends (or fake friends) who could all be dead now. "
"You refuse to travel at night time, wild animals are becoming braver as they realize humans aren't the top predators anymore, and humans have fallen back to the old justice ways.");
abigailWillow->addStory(10, "At the rising dawn you see the San Diego with people guarding the perimeter. With a sigh of relief you come out of hiding and walk slowly towards the gates trying to hold back tears."
"As the gates open to you and a military personel walks out to you saying 'Welcome' and you can't help but cry. 'I'm actually alive, I made it here to safety...'");

```
	//DARNWIN ARNOLD'S STORIES
	darwinArnold->addStory(0, "You are Darwin Arnold, you are a  survivalist contradictory to your name. No one believed in your learning and now they will all regret it hahaha. This is the time to prove yourself"
		"This is your time to shine. You aren't at your safe place where your wife is since you were delivering some cargo as a truck driver. But you will be there in the end, surviving.");
	darwinArnold->addStory(2,"'Oh my gosh this world sucks' you think to yourself. You thought you were prepared but now that it's happening, you are questioning everything you know. You start writing the field guid to zombies "
		"to make time go by faster whenever you're stopping for the night when the light is allowed");

	darwinArnold->addStory(4, "You witness a large community of humans get torn to shreds by zombies and your sanity starts to decrease. You thought you could escape this reality at night in your sleep but you only recieve nightmares. "
	"You continue your journey home to your wife...");
	if (darwinArnold->getHealth() <= 30)
		darwinArnold->addStory(7, "You can no longer sleep and you are constantly hearing voices whether they are screams of pain, groans of zombies, or your inner demon. You are contemplating giving up now...");
	else
		darwinArnold->addStory(7, "You do basic work outs to get you tired so you can gain rest and remind yourself of your wife. This works and you gain some of your sanity back. The journey has been long but you are holding on");

	darwinArnold->addStory(10, "You see your secluded home a few hundred meters away. Sprinting the remainder of the way you feel your hope rise seeing majority of the animals still looking healthy. You run to the fence and shake the bell"
		" longing to see your wife. You see her face appear from inside the lookout tower and you can help but yell her name 'Susan! I'm Here!'");
}

/*
 * chooseCharacter
 *		This is the method that gets called during the game setup for each character to select a character from the list. Once a character is
 *		selected, they have to be removed from the list
 */
Character* CharacterList::chooseCharacter()
{
	//Display instruction
```

```
	string choiceString = "0";
	cout << endl << "Please select a character:" << endl;

	//Output a list of all remaining Characters
	displayCharacters();

	cout << ">> ";
	getline(cin, choiceString);
	unsigned int charSelect = 0;

	//This simple validates whether they have chosen a charcacter in the list
	while( charSelect == 0 )
	{
		try
		{
			//If a valid character has been selected
			charSelect = charChoice( choiceString );
		}
		catch( ... )
		{
			cout << endl << "Invalid choice. Try again." << endl;
			displayCharacters();
			cout << ">> ";
			getline(cin, choiceString);
		}
	}

	//Subtract one from the value, because the map goes from 0-5 and the user has optiosn
1-6
	--charSelect;

	//Save a pointer to the selected character to be returned
	Character * rtn = list[charSelect];

	//Remove the pointer from the list of characters that can be selected
	list.erase( list.begin() + ( charSelect ) );

	return rtn;

}

/*
 * charChoice
```

```
 *                  Simply input validation for when the player is selecting a character.
 */
int CharacterList::charChoice( string choice ) throw( int )
{
        //If nothing was input
        if( choice.empty() )
        {
                throw 1;
        }

        //If they did not input an integer
        if( !isdigit(choice[0]) )
        {
                throw 2;
        }

        //Convert string to integer
        unsigned int charNum = stoi( choice );

        //If the value is less than 1
        if( charNum < 1 )
        {
                throw 3;
        }

        //If they input a number greater than the length of the list (must be invalid)
        if( charNum > list.size() )
        {
                throw 4;
        }

        return charNum;
}

/*
 * displayCharacters
 *                  This is a simple method for displaying the name of every character in the list, with
a number associated. This is sued for selecting characaters
 */
void CharacterList::displayCharacters()
{
        for( unsigned int i = 0; i < list.size(); i++ )
        {
```

```
            std::cout << (i + 1) << ". " << list[i]->getName() << endl;
        }
}
```

///********** Encounter.h ***********///

```
/*
 * Encounter.h
 *
 *  Created on: Nov 13, 2017
 *      Author: abshi
 */
#include <map>
#include "Character.h"
using namespace std;

#ifndef ENCOUNTER_H_
#define ENCOUNTER_H_

enum ResultType { health, supplies, readstory };

class Encounter
{
        private:
                string story;
                string choice;
                string passStory;
                string failStory;
                ResultType type;
                int passValue;
                int failValue;
                int odds;
                unsigned int rightChoice;

                void endEncounter( int value, Character * player );
                int validChoice( string choiceStr ) throw ( int );

        public:
                Encounter( string _story, string _choice, ResultType _type, int _passValue, int
_failValue, string _passStory, string _failStory, int _odds, unsigned int rightChoice );
                void doEncounter( Character * player );
```

```
		virtual ~Encounter();
};

class EncounterList
{
private:
	vector <Encounter> encounters;
	void setupEncounters();
	void deleteEncounter();
	int randomEnc( unsigned int );
public:
	void doEncounter( Character * player );
	EncounterList();
};

#endif /* ENCOUNTER_H_ */



///************** Encounter.cpp ************///

/*
 * Encounter.cpp
 *
 *  Created on: Nov 13, 2017
 *      Author: abshi
 */

#include "Encounter.h"

Encounter::~Encounter() {
	// TODO Auto-generated destructor stub
}

//Basic encounter constructor. Almost everything is required because encounters have many
unique parameters
Encounter::Encounter( string _story, string _choice, ResultType _type, int _passValue, int
_failValue, string _passStory, string _failStory, int _odds, unsigned int _rightChoice = 0 )
{
	story = _story;
	choice = _choice;
	type = _type;
	odds = _odds;
```

```
	rightChoice = _rightChoice;
	passValue = _passValue;
	failValue = _failValue;
	passStory = _passStory;
	failStory = _failStory;
}

/*
 * doEncounter
 *	This method is where the player gets the story for an encounter and makes their decision on what to do. Then they will get an outcome depending
 *	on their choice and a luck factor
 */
void Encounter::doEncounter( Character * player )
{

	//Show the story about this encounter
	cout << story << endl;

	//Show the options for the encounter
	cout << choice << endl << ">> ";

	//Ask player for choice
	string choiceString;
	getline(cin, choiceString);
	unsigned int choiceNum = 0;

	//Simply validate whether that number was valid
	while( choiceNum == 0 )
	{
		try
		{
			choiceNum = validChoice( choiceString );
		}
		catch( ... )
		{
			cout << "Invalid choice. Try again: " << endl;
			//Show the options for the encounter
			cout << choice << endl << ">> ";
			getline(cin, choiceString);
		}
	}
```

```
	--choiceNum;

	/*
	 * This is the part of the application where a "die will be rolled" to determine whether or not the player will pass the check.
	 * To pass the check the player needs to have a die roll greater greater than the odds variable for each encounter
	 * This is the process:
	 * 1. Roll a die between 1-100
	 * 2. Add +10 if the player chose the best choice on the encounter
	 * 3. Add a modifier 1-15 if based on the players luck.
	 */

	//1. Roll a die between 1-100
	int dieRoll = ( rand() % 100 );

	//2. Add +10 if the player chose the best choice on the encounter
	if( choiceNum == rightChoice )
	{
		dieRoll += 10;
	}

	//3. Add a modifier 1-15 if based on the players luck.
	dieRoll += ( player->getLuck() / 100 ) * 15;

	//Since the odds of passing are increased (between 1-100) for each encounter and we're using a minimum number for passing, we need to subtract the odds from 100
	int minRoll = 100 - odds;

	//FOR TESTING: Print the roll and minmum roll
	//cout << "You rolled  " << dieRoll << endl;
	//cout << "You needed  " << minRoll << endl;

	//Finally, check if the player has a high enough roll to pass the encounter
	if( dieRoll >= minRoll )
	{
		//If they pass the encounter
		cout << passStory << endl;
		endEncounter( passValue, player );
	}
	else
	{
		//If they fail the encounter
```

```
                cout << failStory << endl;
                endEncounter( failValue, player );
        }

        cout << "\n\n\n\nPress <ENTER> to continue...";
        string tempString;
        getline(cin, tempString);
        cout << endl;
}

/*
 * validChoice
 *              A method for error checking the player's choice during an encounter
 */
int Encounter::validChoice( string choiceStr ) throw ( int )
{
        //If they didn't input anything
        if( choiceStr.empty() )
        {
                throw 1;
        }

        //If they didn't input an integer
        if( !isdigit(choiceStr[0]) )
        {
                throw 2;
        }

        //Get integer value from string
        unsigned int choice = stoi( choiceStr );

        //If they put in a number less than 1
        if( choice < 1 )
        {
                throw 3;
        }

        //If they input a number greater than 2
        if( choice > 2 )
        {
                throw 4;
        }
```

```
	return choice;
}

/*
 * endEncounter
 *		This function handles the modifcation of stats and telling the player what they have remaining.
 */
void Encounter::endEncounter( int value, Character * player )
{
	switch( type )
	{
		case health:
			player->addHealth( value );
			cout << "You have " << player->getHealth() << " health remaining." << endl;
			break;
		case supplies:
			player->addSupplies( value );
			cout << "You have " << player->getSupplies() << " supplies remaining." << endl;
			break;
		case readstory:
			break;
	}
}

EncounterList::EncounterList()
{
	setupEncounters();
}

void EncounterList::setupEncounters()
{

	/*
	 * THESE ARE THE ENCOUNTERS:
	 * This whole section is where is big list of all the game's random encounters are constructured. There's a really specific format to be followe,
	 * but adding encounters will be extremely easy for us this way.
	 *
	 * Here's an example of how to define an encounter:
	 * Encounter <varname> = Encounter(
```

```
        *               "This is the introduction story",
        *               "1. Choice 1\n2.Choice 2.",
        *               <supplies, health, or story>,
        *               PASS_VALUE (int),
        *               FAIL_VALUE (int),
        *               "This is the story they read if they pass the encounter",
        *               "This is the story they read if they fail the encounter",
        *               ODDS OF PASSING THE ENCOUNTER (int between 0-100),
        *               CHOICE that gives them the bonus chance of passing (int either 0 or 1)
        *       );
        *
        */

        Encounter bridge = Encounter(
                "You reach a bridge that gave out. About to turn around, you notice the side walls of the bridge are intact. Do you: ",
                "\n1. Go across the bridge anyways. It's going to be dark soon.\n2. Turn around and go the long way. The bridge might crumble.",
                supplies,
                0,
                -30,
                "Luckily, you make it to the other side of bridge safely.",
                "You barely make it to the other side, but accidentally drop your water over the side.",
                75,
                1
        );

        Encounter woods = Encounter(
                        "Looking back to where you came from, you see movement in the woods. You leave quickly! Once far enough away you're far away enough, you stop and take a rest. Do you eat a snack?",
                        "\n1. Yes, I just biked for hours and need more energy!\n2. No. I need to conserve my food in case there's none in the next town over.",
                        health,
                        10,
                        -30,
                        "You're feeling better already.",
                        "You feel sick.",
                        80,
                        1
        );
```

```
	Encounter randZombie = Encounter(
			"Behind you, you start to hear footsteps drawing closer. Your turn around quickly and see a zombie making his way to you. He's too close to escape. Do you...",
			"\n1. Punch at his face and try to take him out? \n2. Try to trip him so you can make an escape?",
			health,
			0,
			-35,
			"You manage to take the zombie down and get away safely!",
			"You have taken a bad bite, but the zombies appetite seems satisfied. You make your escape as he feasts.",
			40,
			2
	);

	Encounter ladder = Encounter(
			"You come across a lonely building that looks like it was held up by survivors at one point. On the roof you see what could be supplies. Do you...",
			"\n1. Climb the rusty ladder on the side of the building? \n2. Attempt to scale the wall?",
			supplies,
			10,
			-40,
			"You reach the top of the roof and are lucky enought to find some leftover supplies from the previous survivors.",
			"You try to climb up but the building begins to crumble and you fall, breaking some of your own supplies. Zombies are fast approaching, so you move on.",
			50,
			1
	);

	Encounter sleep = Encounter(
			"You're feeling too tired to move on and you need to rest. To your right you see a small crevice to hide in and to your left you see some trees you could climb in to sleep. Do you...",
			"\n1. Hide in the cave? Staying out of sight would be best! \n2. Climb the tree? Nothing can reach you there.",
			supplies,
			0,
			-25,
			"You awaken, feeling refreshed, and gather your belongings.",
			"During the night some of your food was dragged away by animals. Nevertheless, you get up and move on.",
```

```
                30,
                1
        );

        Encounter falling = Encounter(
                "While crossing a bridge on the way to Santuary, it gives way and you slip. Hanging on by only a thread, do you...",
                "\n1. Reach up quickly before the wire snaps? \n2. Try to throw your gear up to lessen the weight?",
                health,
                -5,
                -30,
                "You manage to pull yourself up and continue on with only a few cut and bruises.",
                "Despite your best efforts, the wire snaps and you plunge into the water. You hit it hard and swallow a lot of water. Luckily, you manage to make it to the shore, but you're hurt badly.",
                40,
                2
        );

        Encounter drown = Encounter(
                "Zombies surround you on every side!!! The only way to go is through the river. Quickly looking for options, you see that you can...",
                "\n1. Swim. Those high-school classes better to pay off! \n2. Grab the log next to the river and try to float across! YOU CAN'T SWIM!!!",
                supplies,
                -5,
                -45,
                "Struggling for breath, you barely make it across the water. You lost a little gear to the water, but at least you're free of the zombies... \n\n...for now.",
                "You see the log sink down into the water from it's weight. You are pulled under by the current and find yourself down the river, free from the zombies, but missing a lot of your supplies!",
                70,
                1
        );

        Encounter car = Encounter(
                "You find an old car that seems broken down, but it has gas! Do you...",
                "\n1. Use your deus ex machina mechanic skills to fix that bad boy up for a faster ride? \n2. Kick it until it turns on",
                supplies,
```

```
                50,
                -25,
                "Somehow, you manage to get the best rolling and you continue on!",
                "As expected, you fail. With a heavy heart, you continue on...",
                10,
                2
        );

        Encounter cake = Encounter(
                "You find cake. You must eat the cake. How will you eat the cake?",
                "\n1. Eat cake with fingers. \n2. Eat cake with face",
                health,
                0,
                10,
                "The cake is a lie.",
                "The cake is a lie.",
                50,
                1
        );

        Encounter fridge = Encounter(
                "You find an old fridge with food inside. It's not cold, but it looks like it's
sealed. You decide to eat, but do you...",
                "\n1. Eat the yogurt? \n2. Eat the stew?",
                health,
                10,
                -30,
                "Luckily, you're able to digest the food, and except for some bad gas, it
really makes you feel better!",
                "Eating old food wasn't your greatest idea... you suffer painful diarrhea for
days",
                25,
                2
        );

        Encounter shrooms = Encounter(
                "While wandering along you stumble across a lovely patch of mushrooms
to eat. You...",
                "\n1. Eat them raw? \n2. Roast them up first!",
                health,
                5,
                -35,
                "You eat the mushrooms and it reminds you of days past... yum :)",
```

```
                "Did you really just eat wild mushrooms? Wow... That was a bad move.
POISoNED!",
                40,
                2
        );

        Encounter mizzou = Encounter(
                "On your way to Sanctuary, you stumble across the ruins of one of the
greatest instituions that ever was..."
                "The University of Missouri. Looking at Naka Hall (or EBW because
everyone is still calling it that) you decide to...",
                "\n1. Search the basement, that's where they're hiding the good stuff. \n2.
Go upstairs! I bet Doctor Rivera has good tastes!",
                supplies,
                20,
                -35,
                "You're lucky enough to find some delicious snacks in the offices!",
                "You leave the building without a single useful bit of gear or food, and on
your way out the door falls on you, because this university hates you.",
                60,
                2
        );

        Encounter puppy = Encounter(
                "In the shrubs in front of you, you hear some growling... Scared of what it
might be, you ready yourself for a fight! From within the brush you see a dog emerge and
charge towards you. Before you can even react, it is licking your nose. Do you...",
                "\n1. Scratch his ear and call him Ralph. \n2. Scratch his ear and call him
Rover",
                health,
                5,
                -35,
                "This puppy is so thrilled at his new name! He makes you feel so happy
and rejuvenated!",
                "Angry with the ugly name you gave him, the dog bites your little toe and
runs away.",
                50,
                1
        );

        Encounter mother = Encounter(
                "In the distance you hear a growl... but this one is familiar. As the growl
draws closer you can see the face becoming clearer. It's your zombified mother. As grumpy as
```

```
aways, she staggers towards you, a little worse for wear. You still don't have the heart to let her go on, so you...",
                    "\n1. Decide to cut her head off. \n2. Hit her with one blast of a shotgun.",
                    health,
                    0,
                    -40,
                    "Her head is gone. The nagging is gone. You're finally free.",
                    "You hesitate just a moment to long and she gets one last bite down on you before you put an end to her. You've escaped that monster, but not without some damage.",
                    25,
                    2
        );

        Encounter gang = Encounter(
                    "Before you can react, you are surrounded by a gang of survivor armed to the teeth, your only option is to talk your way out. You...",
                    "\n1. Pretend that you don't speak English and hope for mercy. \n2. Offer them some of your best supplies to spare your life.",
                    supplies,
                    -10,
                    -35,
                    "The gang decides to have mercy on you and let's you go and only takes a few of your supplies",
                    "The gang laughs in your face before stripping you of most of your possessions.",
                    30,
                    1
        );

        Encounter cpp = Encounter(
                    "You find a computer that has information to help you get to sanctuary, but you have to write some C++ because. You...",
                    "\n1. Write some classes because you're a boss? \n2. Write a rotate ellipse function very poorly?",
                    health,
                    0,
                    -30,
                    "You write the code and get the information you need!",
                    "All this coding has been so taxing on your weak eyes...",
                    75,
                    2
        );
```

```
        Encounter survivors = Encounter(
                        "The sound of an engine roars closer and closer to you as you duck for
cover behind a rock. The vehicles stop "
                        "right in front of you and several armed men and women emerge. They
order you to come out. After considering running, you decide that "
                        "surrendering is your best option. You come out and they greet you with a
smile. You're welcomed to their camp for a night of rest.\n"
                        "They are a group of survivors from that are trying to build their own
colony in the North. They offer you some supplies, do you...",
                        "\n1. Graciously accept the supplies? \n2. Try to convince them to give
you even more",
                        supplies,
                        25,
                        -35,
                        "They decide to give you what they were offering and a little extra
because you're a little extra ;)",
                        "They changed their mind. You get mugged and beaten.",
                        25,
                        1
        );

        Encounter water = Encounter(
                        "The thirst has been setting in and you know that you need to find some
water soon. You find a small puddle and decide to drink the water. Do you...",
                        "\n1. Try to filter it through a piece of cloth \n2. Drink the water as is",
                        health,
                        20,
                        -30,
                        "Thank goodness you got some water that was clean! You're feeling a
little better",
                        "The stale water wreaks havoc on your system and you fall dangerously
ill.",
                        20,
                        1
        );

        Encounter child = Encounter(
                        "As you make your way through the city, a little girl starts making her way
towards you. You don't have the heart to kill the child. So you...",
                        "\n1. Run away quickly and hope the girl doesn't move fast. \n2. Hide in a
nearby building until she passes",
                        supplies,
                        0,
```

```
                -25,
                "You watch as the little creeps along slowly, luckily, not noticing you.",
                "The little girl makes eye contact with you, and to your surprise she
moves like lightning. She attacks you and gets a bite in before you can put her down",
                55,
                2
        );

        Encounter store = Encounter(
                "In a small town you come across a convenient store. The windows are
barred and the door is locked, but there are a lot of supplies inside. You're going to try to break
in, do you...",
                "\n1. Go through the window. \n2. Try to crawl under the building",
                health,
                10,
                -40,
                "You manage to get inside and find some delicious Hostess (not a
sponsor) snacks!",
                "While trying to break in, you cut yourself on some rust and develop
tetanus!",
                50,
                1
        );


        encounters.push_back( bridge );
        encounters.push_back( woods );
        encounters.push_back( randZombie );
        encounters.push_back( ladder );
        encounters.push_back( sleep );
        encounters.push_back( falling );
        encounters.push_back( drown );
        encounters.push_back( car );
        encounters.push_back( cake );
        encounters.push_back( fridge );
        encounters.push_back( shrooms );
        encounters.push_back( mizzou );
        encounters.push_back( puppy );
        encounters.push_back( mother );
        encounters.push_back( gang );
        encounters.push_back( cpp );
        encounters.push_back( survivors );
        encounters.push_back( water );
```

```
	encounters.push_back( child );
	encounters.push_back( store );

}

/*
 * doEncounter
 *		This is the method that gets called from the playGame method to give each player an encounter. It will take in a player object, whose stats may
 *		be modified and check. Then it selects an encounter from the list of encounters generatred on initialization.
 */
void EncounterList::doEncounter( Character * player )
{

	try
	{
		//Pick a random number from 0 to the size of the encounter list
		int encNum = randomEnc( encounters.size() );

		//Create a pointer to that encounter
		Encounter encounter = encounters[encNum];

		//Call the doEncounter method on that encounter
		encounter.doEncounter( player );

		//Erase that encounter from the list of encounters
		encounters.erase( encounters.begin() + encNum );
	}
	catch( int & e )
	{
		//No Encounter FOUND!
		cout << "Somehow the list of encounters became empty. FATAL ERROR" << endl;
	}

}

/*
 * randomEnc
 *		This si a simple method for selecting an encounter from the list.It's passed an unsigned int that should be > 0. Then it randomly generates
 *		and integer between 0 and the value passed. Then returns that value.
```

```
 */
int EncounterList::randomEnc( unsigned int n )
{

	if( n < 1 )
	{
		throw -1;
	}

	//Generate random int between 0-n
	unsigned int f = ( rand() % n );

	//Return int
	return f;

}



///************** Exceptions.h ************///

class bad_input: public std::exception
{

   protected:
	string msg = "\nInput is invalid! Try again.";

   public:
	explicit bad_input()
	  {}

	virtual ~bad_input() throw (){}

	virtual const char* what() const throw ()
	{
		return msg.c_str();
	}

};

class fileNotOpened : public std::exception
{
protected:
```

```
    string msg = "File could not be opened!";

public:
    explicit fileNotOpened() {}
    virtual ~fileNotOpened() throw() {}
    virtual const char * what() const throw()
    {
            return msg.c_str();
    }
};



///************** Exceptions.h ************///

class bad_input: public std::exception
{

    protected:
            string msg = "\nInput is invalid! Try again.";

    public:
            explicit bad_input()
              {}

            virtual ~bad_input() throw (){}

            virtual const char* what() const throw ()
            {
                    return msg.c_str();
            }

};

class fileNotOpened : public std::exception
{
protected:
    string msg = "File could not be opened!";

public:
    explicit fileNotOpened() {}
    virtual ~fileNotOpened() throw() {}
    virtual const char * what() const throw()
```

```
    {
        return msg.c_str();
    }
};
```