Tyler Forrester Final Project

1 Design Document

The goal of this project was to use dynamic memory linked lists and other features of cpp to make a text based video game. I chose to write a small game themed around Boffy the Vampire Slayer where a player would collect items for different areas and then use those items to fight a Boss Character. If Boffy played too long per the design document, she would die from the hostile environment around her. To accomplish these goals, I used abstract classes and inheritance along with previously built code. I describe this process below.

The first need in the Boffy game was for a family of classes that would represent game play. The base class would be called space and contain information that all the different spaces would need. These methods and fields included four pointers to other spaces, a Boffy pointer to give access to the Boffy character in each square. This potentially isn't necessary but seems to simplify the coding by allowing all squares access to Boffy without having to pass pointers between them. It does have the side effect however of deallocating memory to this object dangerious. Since this particular program only has one main character and Boffy will always exist in every action, the design works. However if I was to make a different program with more change of control characters, I would need to choose a different design. The spaces also include a name element to hold different game play locations. The Space class also includes an pure virtual function called special. This pure virtual function holds the controls for the game play and provides the roadmap for different sorts of game play of which I created three derived classes, a Start Space, an Item Space and a Fight Space. Each space has its own unique introduction text that gives the player context on the situation along with a multiple choice menu. I wrote the menu and the text functions into the derived classes, but did not include a pure virtual functions for them in the Space class. My thought was that I may want situations in the future where a Space's special function handles the text and gives the no menu options to the users. Think a game ending sequence with art work. The Space class also includes a getChoice() function. Each derived classes menu was the same length so using a

single function to get choices made coding the project easier and avoided duplication.

Start Space is where the player first comes into the game. It initializes all of the other space objects along with their respective pointers and names. I choose to have all four pointers of each space object to point to the four other spaces in the game. This seemed to give me the most flexibility in designing future movements. However some of the pointers were not used in game play, but could be easily added on as additional menu options in future versions or in some cases were left as Easter Eggs for easy testing. I decided to add as the first menu item a practice zone which links to a newly created fight space. The case creates a new Boffy character along with a new fight square and then allows Boffy to die a horrible death to the villain. In the death, the game drops hints about what items need to be found to beat the villain. It acts as a mini tutorial as well as fulfilling the requirement of in game memory allocation and deallocation. The other four menu options lead to other rooms. The first three rooms are Itemspaces which gives the hero the chance to pick up an item before his fight with the villain. The last menu option is not noted in the menu but is an easter egg which allows the player to skip directly to the boss fight. I'm going to use it for testing, but think it's a fun feature to leave after game play is done. At the start of the special function there is a call to boffy's timer. The timer increments until the game is over. After the 10^{th} move, only one more action in a room is allowed. Movement to other rooms however will not kill Boffy. This lets a player move to the Boss Level and get his chance at the boss before he dies. When the time expires, Creature "dead" variable is sent to true. This breaks the do while loop and ends the game.

The derived Class Item Space has menu options to move to the other two Item Spaces in the game along with the Fight Space. The allows Boffy to move freely around the board. It wasn't reasonable to expect Boffy to move back to the Home square once he started the quest. However I left the pointers to home in place in case in a future version. I needed Boffy to teleport home to restart the game. The Item spaces give the options to search the environment for items. I chose to have the items stay with the Creature class, since Boffy would be carrying them rather than with the Item Space. This provides more variations of game play in the future. The search function have a random 2 -sided dice roll. If the function is true, then an item is picked randomly from the list in the creature class. I then

check if the item has already been added to the pack if it has the item is discarded. The other half of the time, the player gets a message about finding nothing. This makes finding the items slightly harder. Once an item is found another creature class function is called to add the item to your pack. Only two out of the three items are allowed to be put in your pack. There is no mechanism to delete items, so at this point should be time to fight the boss. Against the special function takes a similar form to the Start Space with a timer, dowhile loop, and cases. When the timer expires and one more action in a room (rather than movement between rooms) then the game ends in defeat.

The Fight Space is the most complicated space. I borrowed heavily from Assignment 3's fighting game for this element. In fact, Boffy is a subtype of a modified version of that Assignment's Creature class. Fight Space starts with the same special/menu/text format as the first two derived space types. It again has five menu options. The options 2-4 let the player continue their search for items before attempting to attack the villain. The fifth option appears to the be the same, letting the player bolt across the room, but instead the villain kills the hero, ending the game. The first option is to fight the villain, which sends us into a battle function that was heavily borrowed from Assignment Three. However, when Boffy's attack function is called it now prompts the player to use items in Boffy's bag. These items will have different effects on the villain. However if no items are used, Boffy will always die. The goal of the game is to find the item that kills the villain and to use it. When this happens Boffy will be set to dead. I intend to initialize another pointer to Boffy in the battle function. This pointer will be of Creature type which allows the fight to happen. A new derived class of creature called Dreamwalker will be the villain. It has impressive strength and armor. Once the battle is over, the game ends and the player is prompted to play again.

I am taking all of the functions present in the Creature class in Assignment 3 and then adding more methods for the Final Project. These extra methods are setSpace(), getSpace(), checkItems(), addItem(), timer(), getDead(), setDead(), setWon(), getWon(), timer() and clearItems(). The space methods return and set pointers to the Space. This allows me to tell where a creature is at on the game board. The checkItems function looks for items in the linked queue. I need to write new function in queue, which only iterates through the list to find an Item. This way I can find out whether not certain items are present in Boffy's bag. The

addItem calls the addBack function for LabFa to add items to the linked list. I need a way to deallocate memory. That's what clearItems does. I'm going to call in Main after the game ends. The dead and won are self explanatory. The Boolean dead breaks the loop and ends the game. The won returns whether or not Boffy was victorious in the game. This will allow for two separate text statements at game end. Time provides a clear ending to the game and sets Dead to true after 10 menu selections. However actual death is only executed after the next action in a room (as opposed to movement. By moving all the functions for game play to creature. I've made it extendable to having different character types.

Boffy and Dreamwalker keep very close to their assignment Three roots. Assigned from the added game play in the Boffy's attack both are very similar. Well except for the fact that Dreamwalker always kill Boffy if he gets a chance to attack. This is an easy way to keep the coding time down on an irrelevant part of the assignment.

2 TEST CASES

My test cases were larger in this assignment in the previous do to the nature of the assignment. I found it easier to test multiple routes in the game play at once. Only stopping when one failed a bug test. In general valgrind ended up being very important. The number of allocations and deallocations made memory management more challenging than in the past. Previous assignment's work was not retested, including Creature Attack, InputValid, Dice, and Unmodified Queue Functions.

Test Cases	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Move Between Rooms	2 to Item Space	Main() FirstSpace()ItemSpace()	Successfully move to next file and display information	Successfully move to next file and display information
Move to FightSpace	5 to fight Space Creatures Fight!	Main() FirstSpace()FightSpace()	Moves Creatures Fight	Moves Creatures Fight
Creatures Fight	1 in FightSpace	Main() FirstSpace()FightSpace()Battle()	Battle Successfully executes returning winner	Battle Successfully executes returning winner
Program knows who won	Return value from Fight	Main() FirstSpace()FightSpace()Battle()Main()	Returns correct outcome	Returns correct outcome
Boffy wins	Boffy uses Light Orb to win	Main() FirstSpace()FightSpace()Battle()Main()	Returns correct outcome	Returns correct outcome
Boffy dies from Timer Death	10 moves	Main() FirstSpace()ItemSpaceMain()	Boffy dies after first action after 10 moves	Boffy dies after first actionsafter 10 moves
Memory Leaks in Queue	Add item to queue see if it properly deletes	Boffy->clearItems()	No memory leaks	No memory leaks
Practice Mode Works without SegFaults	Input 1 in FirstSpace()	FirstSpace()FightSpace()	No Memory Leaks	
Check Bag for Different Items. Use Items on Eyghor	Try Light Orb	FightSpace()	Eighor dies	Eighor dies
Check Bag for Different	Try Sword of Gabriel	FightSpace()	We die with right text	We die with right text

Items. Use Items on Eyghor				
Check Bag for Different Items. Use Items on Eyghor	Try Sleeping Powder	FightSpace()	We die with right text	We die with right text
Check Memory allocation	Run Valgrind on full game	Main()StartSpace()ItemSpace()FightSpace()Boffy()Dreamwalker()Queue()	No memory leaks	No memory leaks
Pack holds two items	Try to put 3 items in pack	Creature()Queue()	"Too Bad, Pack is Full. No more Items allowed.\n	"Too Bad, Pack is Full. No more Items allowed.\n
Move through all 5 options in FightSpace	1 - 5	FightSpace()	One Fight 3 moves to Item Spaces One Death	One Fight 3 moves to Item Spaces One Death
Move through all 5 options in ItemSpace	1-5	ItemSpace()	Two Item Pickups One Fight Space Two moves to other Item	Two Item Pickups One Fight Space Two moves to other Item
Move through all 5 options in StartSpace	1-5	StartSpace()	One Practice Fight 3 Item Spaces One Easter Egg	One Practice Fight 3 Item Spaces One Easter Egg
Practice Battle	StartSpace()= 1	StartSpace() FightSpace()	Displays one option to fight. Fight has outcome Correctly allocates and deallocates memroy	Displays one option to fight. Fight has outcome Correctly allocates and deallocates memroy

Pickup Items	ItemSpaces = 1 and 2	ItemSpace() Creature() Queue()	Items are added up to 2. Sometimes nothing is found.	Items are added up to 2. Sometimes nothing is found.
Grammar	Gameplay	My Eyes	Hopefully not too many spelling errors	Hopefully not too many spelling errors

3 REFLECTIONS

The game was hard to brainstorm, but once the design was in place it went fairly quickly. The biggest challenges were handling the memory allocations and the text. Formatting and the UI still leaves much to be desired. I'm excited for the time when my games will have real user interfaces. Writing paragraphs in *cout* statements is both tedious and inefficient. Also I'd like to make Boffy's battling function more robust, add more characters, and item options. From a brief google search the key to do this well is to add some sort of database functionality for both items and creatures, where they can be loaded in via JSON or even txt files. Alas both are out of the scope of this program. In the final product, I added more break statements than I had expect to make the clarity and game flow better. I also needed to add a practice mode to Fight Space so that I wasn't allowing full game play, by allowing the practice accounts to run through the maze and go to other areas. I accomplished this with a Boolean that hid menu clauses and then by setting all integer values between 1-5 to 1 so that the getChoice() function in the Space class did not send invalid input to the invalid memory locations.