Nick Marozick
12/3/17

Final Project CS-162 Design & Reflection

Design:

Game Class Contains:

- Space Pointers for initializing linked Structure and keeping track of structure areas
- Space pointer to keep track of playerLocation
- Player pointer to create a player object
- Game class does not deal directly with items (Player collects items)
- Game constructor takes 8 strings to label the names of the rooms when initializing the LinkedStructure
- Turn Incrementer to add up each round
- gameOn function which uses:
    - bool conditionals to win by either reaching the RiverEscape or GateEscape named rooms
    - bool conditional loss if 25 moves elapses without escaping

Player Class Contains:

- Player default constructor
- Vector of Item Pointers with 6 Item Pointer Limit for items throughout the game
- getItem(Item* &x) Function that takes pointer to an item and the address. These items are taken from room and it matches its address to those items

Item Class Contains:

- Item Constructors taking a string to label each item- Item names later used in binary search functions by the other classes
- std::string getName() function that returns names

Virtual Abstract Space Class Contains:

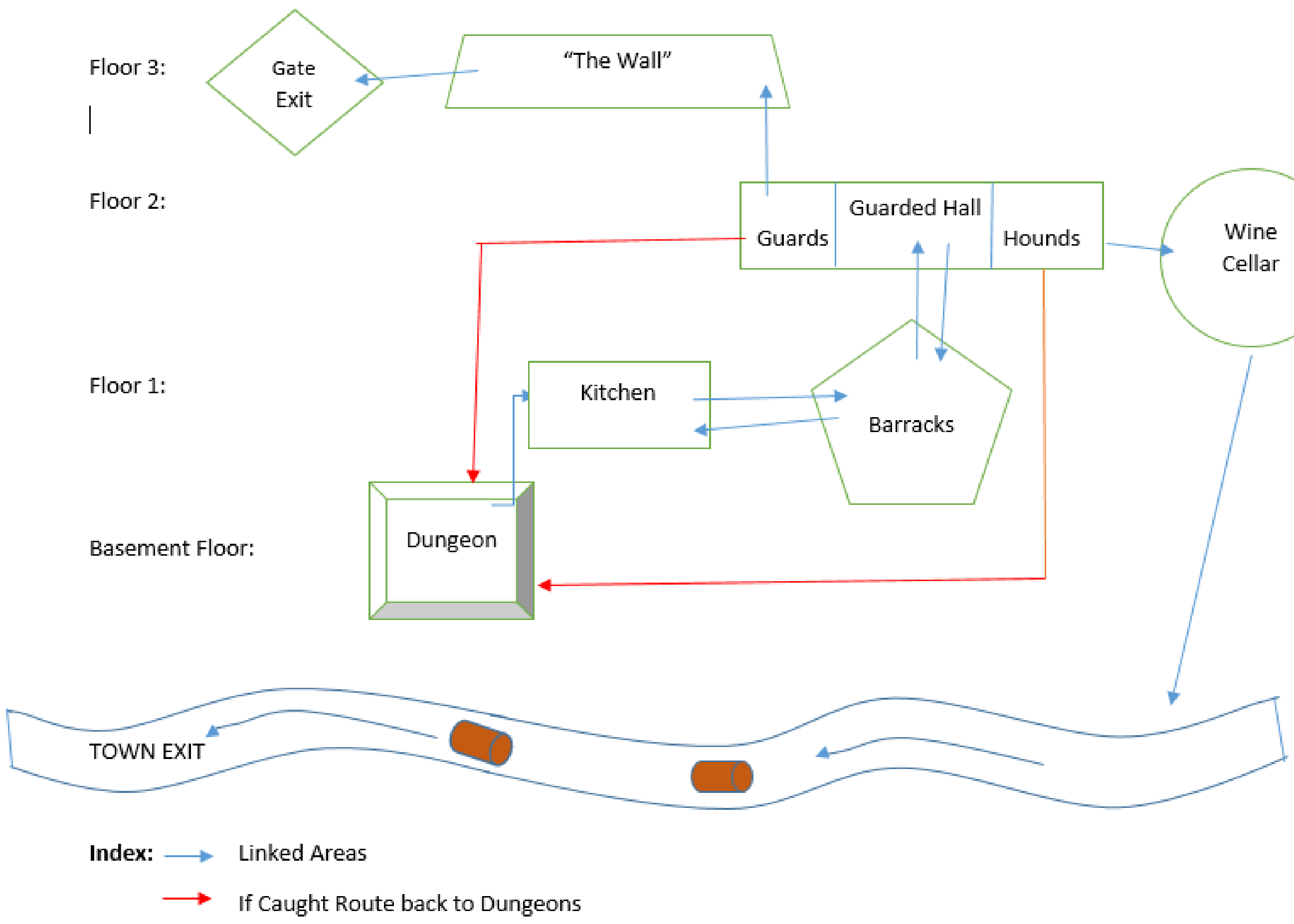
- 4 Space Pointers- top, bottom, right, left
- Virtual Functions: void menu(Player*, Space* &s), void print (Player*), std::string getName();

Child Classes of Space/ Types:

Each of the rooms below (8) are of all different types and require interaction with people, the room, or items to keep moving forward toward the goal. GuardedHall is a cool area, because based on your items and actions, you can make your way toward your escape or instantly get thrown back in the dungeons and lose all your items.

1) Dungeon (Start Location)
2) Kitchen
3) Barracks
4) GuardedHall
5)WineCellar
6) TheWall
7) TownExit (if reached game is won)
8) GateExit (if reached game is won)

LinkedStructure Map:



TEST TABLE:

Test Case	Input Values	Function	Expected Outcome	Observed Outcome
Input 8 strings	8 strings	void Game::(std::string *8)	Properly builds linked list	Properly builds linked list
Input values out of range	0, -3, 7	virtual void Space::Menu() (derived)	Rejects input and waits for a valid int	Rejected input and ask for new input
Input string value	asbdgipfjkhf	virtual void Space::Menu()	Rejects input and waits for a valid int	Rejects random string and waits for valid int
prints map	Player*	virtual void Space::print(Player*)	Prints map if player object map is present	Prints map if player object map is present

Input double value	10.7	void Space::Menu()	Truncates value and rejects if over range	Truncates value and rejects if over range
Terminates when gameWon bools return as true	gameWon=true;	void Game::gameOn()	Ends properly	Ends properly
Continues while turn is not at 25 yet and game is not won	gameWon=false; turnIncrementer<25 ________________ gameWon=true; turnIncrementer=25	void Game::gameOn()	Ends Properly	Ends Properly
Gives option to play or quit	bad input- strings, values out of range ________________ good input	void Game::gameMenuStart()	Properly gives option to play or quit. Rejects bad input	Properly gives option to play or quit. Rejects bad input
Game prints correctly	Map item	virtual void print(Player* p) (Space Derived Classes)	Prints properly	Prints properly
Adds stores items properly	Proper Items	void getMap(Player* p); void getFood(Player* p); (Dungeon)	Adds stores items properly	Adds stores items properly
Interface works properly in Dungeon Menu	N/A	void interactWithPrisoner1(); void interactWithPrisoner2() void feedPrisoner1(); (Dungeon)	Interface works and menu options adjusted accordingly	Interface works and menu options adjusted accordingly
Adds stores items properly	Proper Items	void readGuardManual(Player* p); void getGuardArmor(Player* p); void getGuardID(Player* p); void getTurkeyLeg(Player*); (Barracks & Kitchen)	Adds stores items properly	Adds stores items properly

Bools and guarded hall reactions/ options updated accordingly	Player container items, action with guards, action with hound.	void wait();
		void approachGuards(Player* p, Space* &s);
		void feedHound(Player* p);
		void caughtByGuards(Player* p, Space* &s);
		void caughtByHound(Player* p, Space* &s);
		(GuardedHall)
			GuardedHall room works properly, with adjustments based on player interaction with the room including the hounds & guards and Items in container	GuardedHall room works properly, with adjustments based on player interaction with the room including the hounds & guards and Items in container

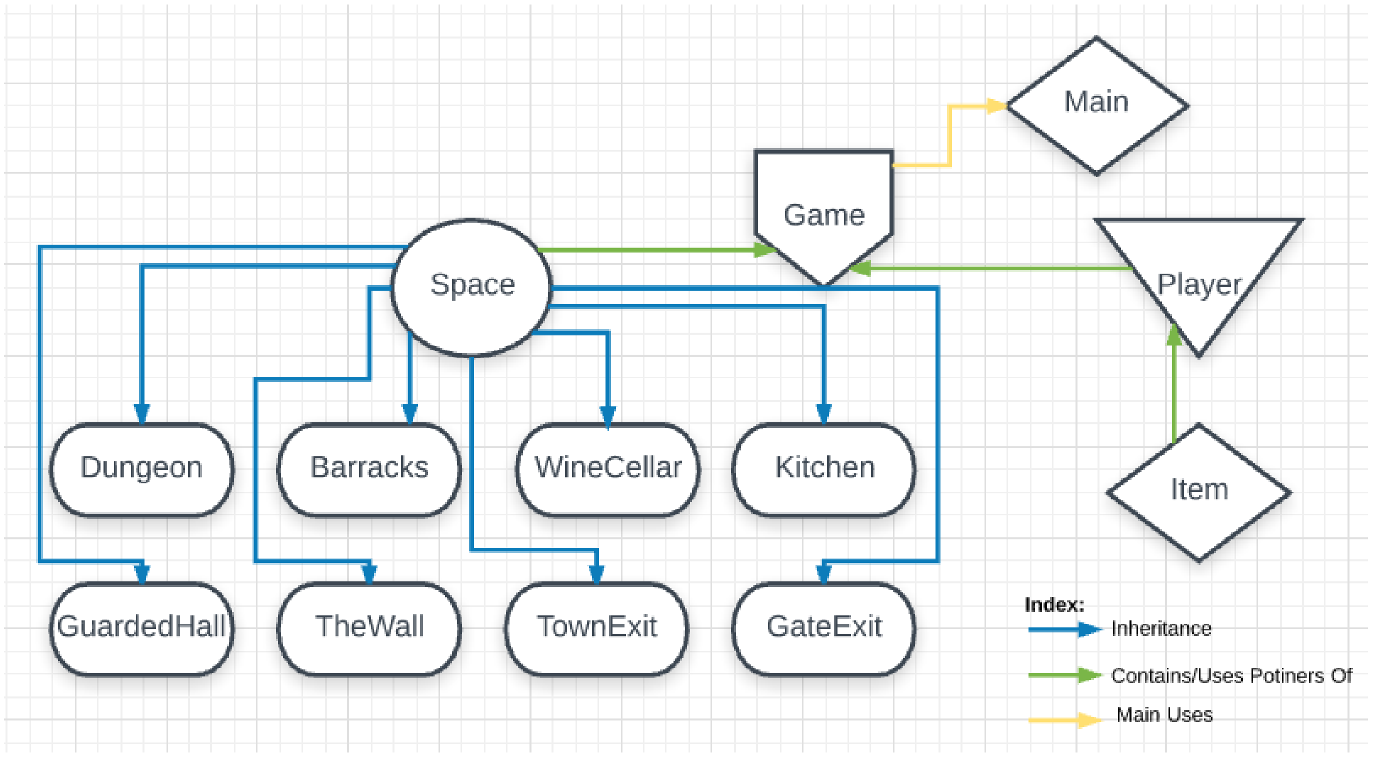
Properly checks for items and sets bools properly	Player Item Container	void checkItems(Player* p) void checkMap(Player* p) (GuardedHall & Dungeon) (All Space Derived Classes)	Properly checks for items and sets bools properly	Properly checks for items and sets bools properly
Properly updates game Boolean and win condition when reached	Game Object Space* playerLocation reached GateExit or TownExit addresses	virtual std::string getName() (Gate & TownExit)	Properly updates game Boolean and win condition when reached	Properly updates game Boolean win condition when reached

Print proper info and increment turns	function calls	void serveFriarMead();
		void eatFood();
		void gamble();
		void drinkWine();
		(Kitchen, TheWall, WineCellar)
			Print proper info and increment turns	Print proper info and increment turns

Navigate game object player location to correct area	Space* playerLocation	void escapeToHatch(Space* &s)
		void enterGuardedHall(Space* &s);
		void headBackToKitchen(Space* &s)
		void headBacktoBarracks(Space* &s);
		void headToTheWall(Space* &s);
		void headToWineCellar(Space* &s)
		void enterBarracks(Space* &s);
		void takeShiftOutsidePalace(Space* &s);
		void headToGateExit(Space* &s)
		void hopInBarrel(Space* &s)
		void headToTownExit(Space* &s);
		(Space Derived Class Movements)
			Seamlessly navigate playLocation pointer to the correct spaces	Seamlessly navigate playLocation pointer to the correct spaces

Items added to player container correctly	Item Pointers	void playerGetItem(Item* &x) (Player Item Inventory)	Items added to player container properly	Items added to player container properly

Class Heirarchy Diagram:



Reflection:

This was definitely an interesting development assignment. It was unique to have something so broad and open ended, which made me circulate a multitude of ideas in my head initially. There were really so many ways to go with this. I guess I have always enjoyed medieval type stories and settings, so thinking of having so many rooms and an objective, had me thinking an "Escape Room" type sequence the entire time. I had a wide array of initial themes, but eventually chose the castle dungeon escape game and proceeded with my design.

It took me a while to visualize the Linked Structure- especially since some on piazza equated it to a 2D Array of Linked Nodes- however that possibility is really only there if you have a linear setup of 2 stories. I always had a 3 level and asymmetrical design in my head, so it helped me to set up my Linked Space Structure using the 4 pointers in whichever fashion I preferred. Once I had the Linked Structure in mind, I began to think of how I would handle items and also how I would store them. I eventually decided on Item objects, so that they can set up in rooms and ultimately contained within my player class as well. I used a vector as our container needed a finite limit, of which I chose (6) items. As I have enjoyed using some of the STL containers, if design choice was given, I likely would have used a list or deque for this as it seemed more intuitive.

Some of the initial design ideas that I had in my first build out had to be changed when it came time to compile. One issue that I had was that my Space class and Player class were both including each other. I basically wanted my player to have a Space* pointer which would keep track of its location, and could be changed as it interacted with each inherited Space classes menu to the given Space derived object's address. In tandem, I wanted Space class to be able to work with a Player* pointer to manipulate its address when it changed rooms and also for the Player to be able to add items from the spaces. Ultimately, having both of these classes reliant and depending on one another- it created a strange compiling loop and the compiler noted that the #include statements were too far nested within one another. After further researching the topic, I better understood the feedback loop. I decided to instead have my game object house the necessary Space Pointers it needed to setup the LinkedStructure, also have a Space Pointer for playerLocation, and lastly a Player Pointer- and having the Player class generally only collect and use items. Once everything compiled, I ran into one more issue as my design wasn't working quite as planned with passing the playerLocation space pointer from game to each Space's menu; the derived classes from Space menu function was supposed to quarterback all actions and also, when the player decided to change rooms, including adjusting the playerLocation address to match the new room. Most of the logic was in place to work, the only gap was that I was only passing a Space* playerLocation pointer and not the address as well. Once I passed the addresses, the navigation worked like a dream. One other minor switch that I made on a couple of functions was where I had initially had static ints within a function reflecting the amount of dialogues with said individual, which would change what they said next- I changed those to bool conditions, where the bool was housed within the derived Space class itself. The reason that I had to make the change is the static ints weren't registering/ incrementing properly as after each space specific menu call, the game would go back to game gameOn() function and clear the static ints within the room. The print function was also very interesting- it showed up looking one way on my editor, notepad++, then another within vim editor, and a third way upon printing. It took some fine tuning to get it to show what I wanted it too, and had to add extra characters to get it to print cohesively. I look forward to getting stronger on ASCII or printing command/ console display end in the future. Overall, I think the final project reinforced pointer activity even more and brought up some good reminders, as well as introduced me to Linked Structures.