Ngan Tran
trannga@oregonstate.edu
CS 162-400

Final Project: Design Reflection

Requirements

For the final project, we must create a single-player game utilizing OOP, inheritance, polymorphism and pointers. In this text-based adventure game, the player will move through a series of spaces in order to gather items to achieve some purpose. Although a lot of the design decisions are up to us, there are some specific requirements that must be met. There must be at least 6 spaces of at least 3 different types, with a space abstract class that will have special pure virtual functions. Each space class must all have at least four pointer variables to link all the spaces together. There should also be a way to keep track of the player's status and location. The player will have a container to store the items they collect as they travel through the spaces. This container must have a limit, or carrying capacity. In addition, the player will not simply be collecting items as they travel through the spaces, they must interact with parts of the space structure. Lastly, the game must have some sort of time limit to prevent the game from going on indefinitely.

Design

Game Title: Alien On The Farm

In this game, the player is a rebellious teenage Alien from planet Zenosyne who is taking its parent's new ship out for a spin. Being curious about what's on Earth, it decides to travel here to take a quick look. Unfortunately, while entering the Earth's atmosphere, the ship hits a large storm cloud and the Alien begins to lose control. After a crash landing on a grassy plain, the Alien is unharmed but it's a different story for the ship. There are two important parts of the sauce missing. The Alien notices something in the distance, figures it must be the ship parts, and starts heading that way with its backpack. Since it is nighttime on Earth, no humans saw what has happened.

The game begins with the player standing in the middle of a gravel road. In front of them is a house and behind them is a barn. They notice the two parts of the ship in different places: on top of a tree house and floating in a lake. The goal of the game is to collect these two pieces and head back to the crash site to fix the ship before the sun comes up. The player must collect the necessary items as they travel through the spaces (House, Barn, Treehouse, Boat, Lake, Garden, Road) in order to complete the task. Since everything is interesting to the Alien and it wants to take everything home, every item it finds will be automatically added to their backpack. Since the backpack only has a capacity of 5 items, the player will have to decide which items will not be helpful for the task and remove them to make room for the necessary items. If the sun rises before the player is able to get back to the crash site with the ship parts or if the humans spot them, they lose the game.

Ngan Tran
trannga@oregonstate.edu
CS 162-400

Below is the layout of the spaces and their connecting pointers:

Boat northPtr = NULL eastPtr = lake southPtr = garden westPtr = NULL	Lake northPtr = NULL eastPtr = NULL southPtr = house westPtr = boat	
Garden northPtr = boat eastPtr = house southPtr = NULL westPtr = NULL	House northPtr = lake eastPtr = treehouse southPtr = road westPtr = garden	Tree House northPtr = NULL eastPtr = NULL southPtr = NULL westPtr = house
	Road* northPtr = house eastPtr = NULL southPtr = barn westPtr = NULL	
	Barn northPtr = road eastPtr = NULL southPtr = NULL westPtr = NULL	

*The Player's starting position

How I will fulfill the assignment requirements:

- "Space" will be the abstract base class that the above locations on the farm will be derived from. The Space class will contain several pure virtual functions that will be implemented differently by each derived class.
- Each Space will have four pointers (north, east, south, west) to link them to the other locations
- There will be a Space pointer *current to track the Player's current location
- "Alien" class will contain the vector <string> backpack, used as the Player's container to store collected items from each space. This class will also contain member variables and functions to manage the items (add, remove, search for)
- "Game" class will contain the Player's current location (Space *current), search and special functions for each space, the play game loop, as well as a results function. This class will also contain Boolean variables to keep track of whether or not the Player has obtained each item, if the game is over, or if the player has won the game.
- To implement a time limit, there will be a number of tries variable in Game. This value is initialized to a constant int value in the Game constructor and will get decremented each time the play function makes a loop. The player will receive warnings when the number of tries variable is equal certain numbers.
- Class Hierarchy:
    - Game has-a Space and has-a Alien
    - Road, Barn, House, Garden, Treehouse, Lake, Boat is-a Space

Ngan Tran
trannga@oregonstate.edu
CS 162-400

GAME CLASS

Constants:

- **NUM_TRIES = 40:** number of times the game will loop until Player loses the game

Data Members:

- **Alien* player:** pointer to Alien
- **Space* (road, barn, house, garden, treehouse, lake, boat):** pointers to each location
- **numTries:** tracks the number of times the Game play() function has looped.
- **Space* currentSp:** Pointer to track Player's current location:
- Boolean variables (**part1Obt, part2Obt, boneObt, boatKeyObt, ropeObt, hookObt, flightObt, floatiesObt, blanketObt, blanketObt, dogToyObt, snacksObt**): set to equal true when each item has been obtained. These are necessary so the Player cannot pick up and add multiples of the same item to their backpack
- Boolean **dogDistracted:** true when dog bone or dog toy is in the Player's backpack when they try to search the House
- Boolean **won:** true when player wins the game. When true, play() loop ends
- Boolean **gameOver:** true when player gets caught or runs out of tries. When true, play() loop ends

Member Functions:

- **intro():** displays game intro message
- **linkSpaces():** links all Spaces together by calling each location's setPtrs function
- **setCurrent(Space*):** sets the Player's current Space location
- **play():** Takes the returned character value from each Space's move() function and performs the appropriate game actions and moves the Player through the spaces accordingly. This will loop until the won=true, gameOver=true, or when the player choses to give up
- **results():** displays the results of the game depending on if won or gameOver
- **search functions:** Spaces that can be searched will have their own search functions inside Game. Once the Player finds an item, it will be automatically added to their backpack, the Boolean variable for that particular item well be set to true, and the player will not be able to add the same item again (exception to this is the chicken eggs (in chicken coop – in barn) – once deleted from backpack, the player can pick up more if they search again). Some items are necessary to win the game, others are not.
- **special functions:** Some locations will have special functions for the Player to interact with the space

ALIEN CLASS

Data members:

- **vector <string> backpack:** player's container to keep items collected throughout game
- **position:** int variable to track position of each item in the vector
- **count:** int variable to keep track of the number of items in the backpack (capacity is 5

Member Functions:

- **addItem(string):** adds new item to the vector (string = item name)
- **searchItem(string):** search the vector for a item name
- **print():** displays the contents of the backpack

- **removeItem():** prompts user for item number and deletes item at that position in the vector
- **removeStr(string):** deletes the given item name from the vector

Space Class (abstract base class)

Data Members:

- **Space* north:** pointer to space north of the current space
- **Space* east:** pointer to space east of the current space
- **Space* south:** pointer to space south of the current space
- **Space* west:** pointer to space west of the current space
- **numVisits:** tracks the number of visits the player makes to the specific space location

Member Functions:

- **entryMsg():** Pure virtual function. Displays a message when the Player enters a space
- **move():** Pure virtual function that will be implemented differently by each space. Displays the menu options of different moves the Player can make in each space. Returns a char that will get used by the Game play() function to perform the appropriate game action.
- **setPtrs(Space*,Space*,Space*,Space*):** used to set the location pointers (north, east, south, west) for each space
- **Space* northPtr():** returns current Space's north pointer
- **Space* eastPtr():** returns current Space's east pointer
- **Space* southPtr():** returns current Space's south pointer
- **Space* westPtr():** returns current Space's west pointer
- **getNumVisits():** returns the number of the Player has made to the specific space location

Testing

I will start by checking to see if input validation works correctly for each of the menus, main as well as all the move() menus for each Space. Once I have determined that this is functioning, I will move on to test whether or not the spaces are linked together correctly by the location pointers. This will be done by going through the space menus and selecting different locations to go to next. If the place I end up at is not what is expected, I know I have to go back to check either the move() function of the previous space or the play() loop where the returned character is being used.

To test the Alien container vector, I will check the backpack to see that an item is added every time it is found during a search. I will also test to see if the remove function works properly by either manually choosing the item number to remove or by using the item and checking to see if was removed from the backpack (ex. dog bone and toy are removed after they are used).

Since there are so many different parts to this game, I will play the game multiple times, each time making sure that each choice results in the expected outcome as per my original design. I will also be testing to make sure that the time limit is implemented correctly and that the correct game results is displayed once the play() loop ends.

Ngan Tran
trannga@oregonstate.edu
CS 162-400

Test Case	Input	Expected Results	Actual Outcome
Main Menu			
Input not int	hello	Reprompt	As expected
Input too low	0	Reprompt	As expected
Input too high	4	Reprompt	As expected
Play Game	1	Begin the game	As expected
Game Hints	2	Display game hints	As expected
Quit Program	3	Program ends Exit code: 0	As expected
PLAY GAME	1	Display game intro Display game goals Starting location is in the Road Display Road menu	As expected
Road Menu			
Go to house	1	House entry message House menu	As expected
Enter barn	2	Barn entry message Barn menu	As expected
Check backpack	3	"Backpack is empty!" (unless player found items in another space) Return to Road menu	As expected
Give up	4	Return to Main Menu	As expected
House Menu			
Search the house	1	Cannot search house without dog bone or dog toy in backpack to distract dog Return to Road menu If have dog bone or toy Prompt location to search 1. Garage item: floaties 2. Main Room item: boat keys 3. Kitchen item: snacks	As expected
Go somewhere else	2	Prompt location selection 1. Back to Road (south) 2. To Tree House (east) 3. To Lake (north) 4. To Garden (west)	As expected

Ngan Tran
trannga@oregonstate.edu
CS 162-400

		5. Stay at House	
Check backpack	3	Items found in house will be in the backpack Return to House Menu	As expected
Give up	4	Return to Main Menu	As expected
Barn Menu			
Search barn	1	Prompt for are to search 1. Chicken coop item: chicken eggs 2. Horse stall item: rope 3. Pile of hay item: dog bone 4. Tool box item: flash light Add items found to backpack	As expected
Go back to road	2	Road menu	As expected
Take the wheelbarrow	3	Cannot take wheelbarrow since haven't found ship parts	As expected
Check backpack	4	Items found during search added to backpack	As expected
Give up	5	Return to Main Menu	As expected
Treehouse Menu			
Try to get ship part	1	If no flash light in backpack, cannot see anything If have flashlight but hook and rope not in backpack, cannot get the ship part down If have flashlight, hook and rope, can get the ship part down	As expected
Search tree house	2	Will find blanket Blanket added to backpack	As expected
Go back to house	3	House menu	As expected
Check backpack	4	Any found items during search will be in backpack	As expected
Give up	5	Return to Main Menu	As expected
Garden Menu			
Search garden	1	Without flashlight cannot search the Garden If have flashlight, will find dog	As expected

		toy – add to backpack	
Go to house	2	House menu	As expected
Go to boat	3	Boat menu	As expected
Check backpack	4	Items found so far in game will be in backpack If more than 5 items, must delete before adding more	As expected
Give up	5	Return to Main Menu	As expected
Boat Menu			
Start boat	1	If the boat keys are not in the backpack, cannot start boat If keys are in backpack, start boat, go to lake and get the ship part	As expected
Search boat	2	Find hook in the boat Hook added to the backpack	As expected
Walk to lake	3	Lake menu	As expected
Go to garden	4	Garden menu	As expected
Check backpack	5	Items found so far in game will be in backpack If more than 5 items, must delete before adding more	As expected
Give up	6	Main Menu	As expected
Lake Menu			
Go back to house	1	House menu	As expected
Go for a swim	2	If player has floaties in backpack, can go for a swim If ship part from lake still hasn't been obtained, swim to get it	As expected
Walk to the boat	3	Boat menu	As expected
Check backpack	4	Items found so far in game will be in backpack If more than 5 items, must delete before adding more	As expected
Give up	5	Main menu	As expected
Backpack			
Test the addItem, and removeItem functions of class Alien	Move through spaces, searching for items	Each item that is found during a search is automatically added to the backpack Barn	As expected

Ngan Tran
trannga@oregonstate.edu
CS 162-400

		-Chicken Eggs -Rope -Dog Bone -Flash Light House -Dog bone removed after use -Boat keys -Floaties ::find snacks:: message: "Backpack full!" ::remove Chicken Eggs Backpack now contains: 1. Rope 2. Flash Light 3. Boat Keys 3. Floaties 4. Snacks	
Time limit			
Test that a warning is displayed after 20, 30, 35, 39 moves	Play game, making 40 moves	Appropriate warning message is displayed after 20, 30, 35 and 39 moves	As Expected
Memory Leak			
Run program through valgrind to check all memory has been successfully deallocated			First time ran through – there was a leak because I forgot to delete the currentSp pointer in the game destructor

Ngan Tran
trannga@oregonstate.edu
CS 162-400

Reflection

Overall, this project went quite smoothly for me. Throughout the implementation process, I had to make a couple minor changes and added many extra features to the game. One of these changes was how I implemented the backpack. Originally, I only had a remove function that deleted an item by taking in an int position as its argument. I realized I needed another function that would allow me to delete an item by taking in a string item name. This is necessary when deleting a specific item after it has been used during the game (for example the dog bone, toy and floaties).

One feature I had to go back and add in my program was a way to prevent an item from being added to the backpack multiple times each time the player goes back to the same area in a space to search. After considering a couple different possibilities, I figured the best way to implement this was to use Boolean variables. I added a bool for each item (besides the chicken eggs). After an item has been found and successfully added to the backpack, the item's Boolean variable is set to true. When the player enters a space that has a search function, they will only be able to look for items in that space if the bool for the particular item is false.

In order to make the game more interesting, I added many different possibilities for the player to win the game. With my original design, if a player were to accidentally delete a necessary item such as the dog bone, the boat keys, the hook and or rope, they would not be able to win the game. To give the player more chances of winning, I added several ways to obtain the necessary items.

Two ways to distract the dog in order to search the house
1. Search the barn for the dog bone (in chicken coop)
2. Search the barn for flash light (in wooden tool box)
    a. Go to garden, search for dog toy

Two ways to get the ship part in the lake
1. Search house for boat keys (kitchen). Go to boat, use it to get to the ship piece
2. Search house for floaties (in garage). Go to lake, swim to get the ship piece

Two ways to get the ship part on the tree
1. Search barn for flash light (in wooden tool box)
    a. Search barn for rope (in horse stall)
    b. Search boat for hook
    c. Use flashlight, rope and hook to get ship part from the tree house
2. Visit the tree house three times
    a. Tree house will become unstable and fall – ship part easy to get now

This assignment ended up being my favorite project in this course because we had so much freedom with it. It was very satisfying to put everything we learned in this course to practice to design and implement a complex (and functioning!) program.