The Haunted Pittock Mansion

Introduction

This Pittock Mansion located in Portland is actually haunted. Henry Pittock loves Portland so much that he still lives there and pays attention to who the visitors are. He will not scare anybody unless he or she is a famous local chef for the buzzing Portland restaurants. His goal is to be able to eat the current famous Portland menus by trapping the chefs in the mansion and forcing them to cook their famous dishes. One of the items that Mr. Pittock has not had is the Pok Pok chicken wings, a famous local item that is a must-try for everybody in Portland.

In this game, the user is the Pok Pok chef who happens to visit Pittock Mansion. The chef/user is trapped by Mr. Pittock ghost inside the mansion until **he successfully cooks and serves the chicken wings for Mr. Pittock on the dining room table.** Obviously, the user does not bring any ingredients when he visits. So, he must gather the necessary ingredients by calling the Pok Pok restaurants to deliver the necessary ingredients. Unfortunately, the chef's cellphone is out of battery and he is the only visitor in a slow Monday afternoon.

(1) Hence, he first must find a way to make a call using a wireless phone that is located in the living room underneath a couch. He then needs to charge this wireless phone. The charger is located inside one of the cabinets in the living room.
(2) Once he is able to get hold of somebody to deliver it, he should realize that the delivery cannot be made through the main entrance but rather through the dumbwaiter located in the laundry room. The entrance is always locked to prevent the chef from fleeing. The dumbwaiter, though, is protected by a fence from outside. In order for the delivery person to be able to access it, the chef has to throw the key through the window in the master bathroom. This key is located in the master bedroom underneath Mr. Pittock's bed.
(3) He will then need to go to the kitchen to cook it and collect the necessary utensils. He needs at least a knife, a cutting board, a big bowl, and a deep fryer. He first needs to cut the ginger, then mix it with chicken, salt, pepper, and soy sauce before finally cooking and serving it in the dining room, after which, he can get the **main gate key**.
(4) Of course, he has to complete all of these tasks before Mr. Pittock gets too hungry by 7 pm. So, the user has about 4 hours to do this.

Strategy to implement the program

- There is a Room base class that will have pure virtual functions (detailed pseudocode in the sections below) and derived classes for the different types of rooms.
- There will be an Item class that has names, weight in the constructors. These items have to be located in a container in a certain room. Item cannot have another item.
- There will be a Container class that serves its purpose to hold items. The container can be a bag for the player, things in the rooms that can hold smaller items like cabinets or couch. Container cannot be collected inside another container, i.e. container does not have a has-a relation. But container has a has-a relation to items.
- The game class has all those 3 types of classes: room (and its derived classes), container, and items. The game class will have a time that counts down until Mr.Pittock is too hungry. It will keep asking user the menu to move to a different room, collect items, or use items as long as the player is not outside yet by obtaining the main gate key. If time is 0, the program also stops and the player is trapped forever.

Pseudocode

Class Room:

Protected:

```
Room *front
Room *back
Room *right
Room *left
```

Public:

```
Room()
        Set all Room* to NULL
Virtual int menu();
Virtual Room* moveRight()
Virtual Room* moveLeft()
Virtual Room* moveBack()
Virtual Room* moveFront()
Virtual void setRight(Room*) // may be not needed
Virtual void setLeft(Room*)
Virtual void setFront(Rom*)
Virtual void setback(Room*)
```

- Class Entrance : public Room

Public:
Entrance()
    this->front =
    this->back =
    this->right =
    this->left =
int menu()
    1. unlock door

void unlockDoor()
    if key is obtained, door can be unlocked
    else “need to cook the wings first”

- CommonArea (always need to go to here in between rooms)

Protected:
    Room *up

Public:
    CommonArea()
        this->front =
        this->back =
        this->right =
        this->left =
        this->up = laundryRoom
    int menu()
        1. Look behind hanging pictures
        2. Go to living room
        3. Go to dining room
        4. Go to master bedroom
        5. Go up to laundry room
        6. Go outside // cannot go if key is not yet obtained (replacing entrance)

- Living Room (has a wireless telephone hidden underneath a couch)

Public:
    LivingRoom()
        this->front =
        this->back =
        this->right =
        this->left =

    int menu()
        1. Watch TV
        2. Play the piano

3. Sit on a rocking chair
4. Sit on the couch
5. Use the wireless phone charger
6. Make a call using wireless charger
7. Go back to common area
8. Go to dining room
9. Go to kitchen
10. Go to master bedroom

- Dining Room (serving place)

Public:

DiningRoom()

this->front =
this->back =
this->right =
this->left =

int menu()

1. Light the candle
2. Open the China hutch door
3. Open the China hutch drawer
4. Place items on dining table
5. Serve food on dining table
6. Go back to common area
7. Go to living room
8. Go to kitchen

- Kitchen

Public:

DiningRoom()

this->front =
this->back =
this->right =
this->left =

int menu()

1. Open drawer 1
2. Open drawer 2
3. Open drawer 3
4. Open cabinet 1
5. Open cabinet 2

6. Turn on the stove
7. Use cutting board
8. Place items on stove
9. Start cooking
10. Go back to common area
11. Go to dining room

Class Game

Private:
Container (a vector containing string)
//Has all of the rooms

Public:
Void play()

Class Item

Protected:
Std::string itemName
Int itemWeight // in lbs
Int itemSize // in

Public:
Item(std::string itemName, int itemWeight, int itemSize)
Std::string getItemName()
Int getitemWeight()
Int getItemSie()

Class Container

Include item.h

Protected:
Std::string containerName
Int maxWeight // in lbs
Int maxSize // in
std::vector<item*> container; // initialize vector containing pointer to items

Public:
Container(std::string containerName, int maxWeight, int maxSize)
Std::string containerName ()
Int getMaxWeight()
Int getItemSie()

Test Case

Test Case	Room	Item collected	Action/interaction with things	Expected Outcome	Observation
Test the room changing move function	Common Area change to living room	NA	NA	Chef should move to the living room	Chef moves to the living room
Test the item collection function	Living Room	Collect item underneath couch	NA	Bag should be loaded with the wireless phone	Bag is added with the phone
Test to let the game run out of time	Living Room	NA	NA	Screen should say Mr. Pittock is too hungry and he has trapped you forever	Screen says Mr. Pittock is too hungry and he has trapped you forever
Objective 1: collect phone, charge, and order items	Living Room	Phone (under couch) and charger (from cabinet)	Use the charger and call	Chicken should be ordered	Chicken is ordered
Objective 2: Collect fence and dumbwaiter key	Master Bedroom	Fence (in dresser) and dumbwaiter (under bed) key	NA	Fence and dumbwaiter keys are collected into bag	Fence and dumbwaiter keys are collected into bag
Objective 3: Throw the fence key through bath room window	Bathroom	NA	Open window and then throw the fence key	Fence can be opened/is opened when the delivery man arrives	Fence is open
Objective 4: operate dumbwaiter and collect items	Laundry Room	Chicken wings, soy sauce, salt, pepper, oil, ginger	Operate dumbwaiter with key, bring it down, and shout to the delivery man to load items to dumbwaiter, bring dumbwaiter up, and collect all the items.	Chicken wings, soy sauce, salt, pepper, oil, ginger should be loaded into bag	Chicken wings, soy sauce, salt, pepper, oil, ginger are loaded into bag
Objective 5: Go to kitchen to cut, mix, and cook chicken	Kitchen	Knife, cutting board, and deep fryer	Cut the ginger, mix it with chicken, soy sauce, salt, and pepper. Then cook it in deep fryer	Cooked chicken should be loaded to bag	Cooked chicken is loaded to bag
Objective 6: Serve the	Dining Room	Collect main gate after food	Serve the food	Chef should be free	Chef is free

cooked chicken and collect the main gate key and then exit		is served, and then go outside from the common area			

Reflection

- My initial design addresses the initialization of pointers to adjacent rooms from each derived room using another set of constructor. However, upon implementation, this caused an issue resulting in the creation of a setAdjacent room function that takes as parameters the rooms associated to the front, back, right, and left of the current room. For a room that has more than 4 pointers to rooms, another setRoomUp() method is created.

- Initial design to handle room change is done separately for each move to the front, back, left, or right with individual set method. This was later reconfigured by having one function changeRoom() that will return Room* type and the return to the front, back, left, or right is done inside the function by returning this->front.

- The base class Room was designed to not contain anything other than pure virtual functions and the derived Room classes have a function menu that will have options for all types of actions: collecting items as well as using them. This has proved to be cumbersome as collecting and using items are different in implementation. In collection items, the function needs to return an item that is then added to the bag, while using items should not return anything but uses the item or bag as the function argument.

- Initial design has an Item class for the things that the user needs to collect throughout the game and a container class for the user's bag. However, upon implementation, the container is also applied to things in the room that cn hold items.

- Hence, the program needs to create 2 additional functions to handle the collecting item and using item separately. A function explore() returns a pointer to item that the bag can store in the Game.cpp. A function interact(container*) takes as an argument a pointer to a container (or in this case the player's bag). It will then ask the user what item to use and remove from the bag to interact. Sometimes, the interaction does not need an item (such as opening a window).