CS 162
Pamela Yin

Final Project Reflection

Test Tables: Plan/Results

Menu input test cases for Final Project

Test Case	Input Values (Assuming values from 1-7)	Driver Functions	Expected Outcomes	Observed Outcomes
Non-integer	Asdf 2.5 - =123	numInputVal()	Reprompt for another input	Reprompt for another input
Input too low (<1)	0	numInputVal()	Reprompt for another input	Reprompt for another input
Input in correct range	2	numInputVal()	Program accepts and move to next line of code	Program accepts and move to next line of code
Input extremely low	1	numInputVal()	Program accepts and move to next line of code	Program accepts and move to next line of code
Input extremely high	5	numInputVal()	Program accepts and move to next line of code	Program accepts and move to next line of code
Input too high	10	numInputVal()	Reprompt for another input	Reprompt for another input

Test Cases

Test Case	Predicted Outcome	Actual Outcome
Patience < = 0, Game Won = false	Game Over	Game Over
Step count < = 0, Game Won = false	Game Over	Game Over
Patience > 0, Step Count > 0, Game Won = false	Continue game	Continue Game
Patience > 0, Step Count > 0, Game Won = true	Win	Win

CS 162
Pamela Yin

Final Project Reflection Document

Design

Game: Kitten’s journey to find treat

Spaces: kitchen, bathroom, bedroom, living room, patio, outside

Types:

Kitchen (use collar put on cabinet door, pull to open – mission accomplished)

Bedroom (drink water – get refreshed – move faster + 2 steps)

Find a nail sticking out from the table, use to pull off collar, put in bag

Living room – find and pick up mouse toy

Patio – jump onto chair, can see through the window, get hint: cabinet door can’t be opened, must need a round object to put on knob to pull – collar)

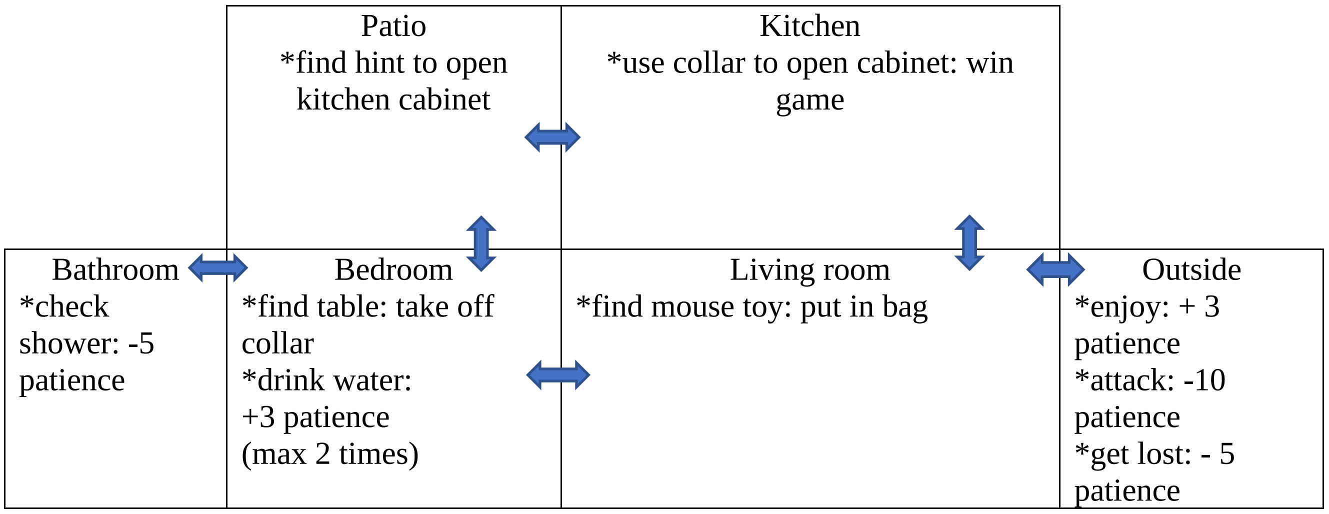
Bathroom – get wet, body is soaked, needs to groom – lose 5 steps

Outside – get outside, freak out, get lost, come back – lose 10 steps

Container: bag/ mouth – items: mouse toy, collar

Time limit: until human comes back home (100 steps)

Map Layout


Patio
*find hint to open kitchen cabinet
Kitchen
*use collar to open cabinet: win game
Bathroom
*check shower: -5 patience
Bedroom
*find table: take off collar
*drink water: +3 patience (max 2 times)
Living room
*find mouse toy: put in bag
Outside
*enjoy: + 3 patience
*attack: -10 patience
*get lost: - 5 patience

Main

1. Menu function – call menu

CS 162
Pamela Yin

Menu – integrate in main?

1. Display main menu
    a. Play
    b. Exit
2. Play again menu

Input Validation

1. Check for number inputs
2. Check for integer inputs
3. Check for number within range

Space class – abstract base class - room

1. Pointers (unused = NULL)
    a. Top, right, left, bottom
2. Functions
    a. Normal functions
        i. Map (text/graphic)
    b. Pure virtual functions
        i. Descriptions
        ii. Menu
            1. Move to different room
            2. Pick up item
                a. Mouse – living room
                b. Collar (after take off) - bedroom
            3. Use item
                a. Table – bed room – take off collar
                b. Use collar – kitchen
                c. Water – bedroom – can use 3 times
                    i. +3 steps
                    ii. Max 2 times
            4. Explore
                a. Chair – patio – get hint (thoughts)
                b. Bathroom – check shower
                    i. -5 steps
                c. Outside – explore
                    i. Get lost by chasing squirrels and butterflies: -5 steps
                    ii. Get in fight with stray cat: -10 steps
        iii. Pickup item
        iv. Use item
        v. Lose time
        vi. Gain time

CS 162
Pamela Yin

Living room class

1. Pure virtual functions override
    a. Descriptions
    b. Menu
        i. Move left to bedroom
        ii. Move right to outside
        iii. Move up to kitchen
        iv. Explore
            1. Find a mouse – pick up?
                a. Yes
                    i. Pickup item
                b. No

Patio class

1. Pure virtual functions override
    a. Descriptions
    b. Menu
        i. Move down
        ii. Explore
            1. Go on the chair
                a. Get hint message

Bedroom class

1. Pure virtual functions override
    a. Descriptions
    b. Menu
        i. Move right to bedroom
        ii. Move up to kitchen
        iii. Explore
            1. Find a table
                a. Use item – mouse: nothing happens
                b. Use item – collar: collar is off
                    i. Store in bag

Outside class

1. Pure virtual functions override
    a. Descriptions
    b. Menu
        i. Move left to living room
        ii. Explore – 1/3 chance
            1. Enjoy the sunshine, feel refreshed and dry +3 steps

2. Wander, get lost – 5 steps
3. Get attacked – 10 steps

Kitchen class

1. Pure virtual functions override
    a. Descriptions
    b. Menu
        i. Move left to balcony
        ii. Move down to living room
        iii. Explore
            1. Find cabinet
                a. use collar – open – get treats- win

Bathroom class

1. Pure virtual functions override
    a. Descriptions
    b. Menu
        i. Move right to bedroom
        ii. Explore
            1. Find a toy ball
                a. Put in bag
            2. Go into shower
                a. Get wet -5 steps

Containers

1. House container
    a. Spaces (rooms) pointers as nodes
    b. Linked list – link the rooms using space pointers
2. Bag container
    a. Vector object to hold items in bag

Bag class

1. Item object
2. addToBag(item x)
3. displayBag()
4. useItem(item x)

Score class

1. variables: int points, steps
2. functions
    a. losePatience(int points)
    b. gainPatience(int points)

c. loseSteps(int steps)
d. gainSteps(int steps)
e. displayStatus() – show steps, patience count

CS 162
Pamela Yin

Reflection

The most difficult part about this project was that there are no set criteria. Planning the program was very hard. After I set a theme, I think about parts that can be implemented using my knowledge from this class. For some themes I thought of, either it did not fit the criteria well or it was overly complicated. Planning for the program without having a lot of extra unnecessary or complicated implementation was hard part. It was very easy to go overboard to make a perfect and appealing game, which I noticed has a lot of drawbacks. For example, I tried to create game where I can make the cat lose patience and also lose time. However, given that they are in 2 different classes, it was hard to keep track and check every round, so I have taken that out.

I also spent a lot of time trying to find out fancy features like clear screen, enter ASCII text graphics, and use different color/highlight rooms when it is in that particular room. I spent some time on getting some of those done but I figured I did not have a lot of time to implement that, so I went back to get the codes running without bugs first.

There are many things I've changed from planning to finished program. Initially I wanted to make a container for house and put all the rooms in there like how I have been using the struct nodes, but given the space pointers in the space class, I decided to delete the house container. Same as the Bag container, I thought about having it as separate class where items are kept, but while I was implementing the game, I found out I actually need to move the items from one place to another, which would require 2 separate containers. In the end, I have created all the items in the game class, set them in corresponding spaces, and have inventory in Cat class that picks up all the moved items.

One of the hardest part and the one I learned the most was using friend class. I was implementing my explore functions in each spaces, but I had trouble linking it to action of the cat. Since Cat class is separate, I was not sure how to make the cat do something when something in space triggered it. After extensive researching and googling, I found that I could use a friend class to do that. From reading the material, it seemed like I can use all private and public members of the class, but when I tried to get the size of the inventory vector or access the inventory element, it would return 0. Later I realized I need to pass in my cat pointer and have that point to the function with inventory to access it correctly. After that, it gave me ideas to create extra helper functions that retrieved private vector inventory and use my cat object and cat functions as if I'm working with cat class. Initially I tried to make inventory public, to be able to access it from any classes, but people suggest by all means to avoid it for the secureness of the data in the program. Friend class was something we haven't really explored through the course, which I now am very comfortable using.

This program was not too difficult that whenever I'm stuck, I am able to find out what I can do to work around it, unlike beginning of the class. It was matter of planning well and distributing my time to not focus on unnecessary details. Even though my program can be improved a lot both aesthetically and functionality, I took my time to make what I have resourceful and work properly and I'm pretty satisfied with my work.