Holly Buteau

Final Project: Plan Z

March 13, 2016

Design:

Space:

Protected:

```
bool dark;
bool visited;
Space *ptr1;
Space *ptr2;
Space *ptr3;
Space *ptr4;
Space *ptr5;
```

Public:

```
Space();
virtual void getPointers(Space *p1, Space *p2, Space *p3, Space *p4, Space *p5) = 0;
virtual void Use() = 0;
```

```
    virtual Item* Take() = 0;
    virtual void Look() = 0;
    virtual void Start() = 0;
    virtual Space* Leave() = 0;
    virtual void Inventory(Item *item) = 0;
    virtual bool getVisited() = 0;
```

Bedroom:

```
private:
    vector<Item*> bedroomItems; //vector of bedroom items
    bool seeGun;
    bool chairMoved;
    bool gunTaken;
    bool seeChair;
    bool seeRoom;

public:
    Bedroom();
```

~Bedroom();

virtual void getPointers(Space *p1, Space *p2, Space *p3, Space *p4, Space *p5); //takes as an argument pointers to different rooms

virtual void Use();

virtual void Start();

virtual Item* Take();

virtual Space* Leave(); //returns the pointer to the space chosen to move to

virtual void Look();

virtual void Inventory(Item* item); //places an item pointer into inventory

virtual bool getVisited();

Attic:

private:

vector<Item*> atticItems; //vector of attic items

bool seeRadio;

bool useRadio;

bool lightsOn;

bool radioTaken;

```
    bool seeRoom;


public:
    Attic();
    ~Attic();
    virtual void getPointers(Space *p1, Space *p2, Space *p3, Space *p4, Space *p5); //takes as an argument pointers to different rooms
    virtual void Use();
    virtual void  Start();
    virtual Item* Take();
    virtual Space* Leave(); //returns the pointer to the space chosen to move to
    virtual void Look();
    virtual void Inventory(Item* item);//places an item pointer into inventory
    virtual bool getVisited();
```

Kitchen:

private:

vector<Item*> kitchenItems; //vector of kitchen items

bool seeNote;

bool seeList;

bool seeGoods;

bool useList;

bool foodTaken;

public:

Kitchen();

~Kitchen();

virtual void getPointers(Space *p1, Space *p2, Space *p3, Space *p4, Space *p5); //takes as an argument pointers to different rooms

virtual void Use();

virtual void Start();

virtual Item* Take();

virtual Space* Leave(); //returns the pointer to the space chosen to move to

virtual void Look();

virtual void Inventory(Item* item); //places an item pointer into inventory

virtual bool getVisited();

Bathroom:

private:

vector<Item*> bathroomItems; //vector of bedroom items

bool seeMedicine;

bool lightsOn;

bool medTaken;

bool seeRoom;

public:

Bathroom();

~Bathroom();

virtual void getPointers(Space *p1, Space *p2, Space *p3, Space *p4, Space *p5);//takes as an argument pointers to different rooms

virtual void Use();

virtual void Start();

```
        virtual void Inventory(Item* item);
        virtual void Look();
        virtual Item* Take(); //places an item pointer into inventory
        virtual Space* Leave();//returns the pointer to the space chosen to move to
        virtual bool getVisited();
```

LivingRoom:

```
private:
        vector<Item*> livingroomItems;//vector of kitchen items
        bool seeRemote;
        bool seePoker;
        bool pokerTaken;
        bool tvUsed;

public:
        LivingRoom();
        ~LivingRoom();
```

virtual void getPointers(Space *p1, Space *p2, Space *p3, Space *p4, Space *p5);//takes as an argument pointers to different rooms

virtual void Use();

virtual void Start();

virtual Item* Take();

virtual Space* Leave();//returns the pointer to the space chosen to move to

virtual void Look();

virtual void Inventory(Item* item);//places an item pointer into inventory

virtual bool getVisited();

Player:

private:

vector<Item*> inventory; //player's inventory

Space* currentLocation; //players current location

int steps; //steps to determine time

Public:

Player();

void checkInventory();

void Menu();

```
    void setLocation(Space *room);
```

Item:

```
private:
    string name;
public:
    Item();
    ~Item();
    void setName(string n);
    string getName();
```

Design Cont.:

I will begin with data members. Each room class inherited 5 pointers, one for each room including itself. Each room also inherited two booleans : dark – to check if the light was off in the room, and visited – to label whether the room had been visited. I chose the visited boolean so that the expository text for each room was only given the first time the player entered the room. Each room also had several data members no inherited but very similar. Each room had its own inventory with one special item that the player needed in order to win the game. Each room had multiple booleans to change what was available to the player based on what they had already done. All rooms had one boolean to prevent the player from leaving (chairMoved, useRadio, lightsOn, useList. tvUsed) until the player had completed the actions that switched these booleans, the player could not leave the room. Each room had a boolean that switched if the player picked up the special item. If the item was picked up, they would not see it if they looked again. This boolean was also used to determine if they could take the item, because if they could not see it, they could not take it. All of the Booleans in the room classes were to control what text was displayed to the user, what options the user had, and whether the user could leave.

The player class also had data members. The player had an inventory to store items that had been picked. The player had a pointer to a space object for its current location, the player also had steps to control the time limit. The item class had one data member, the item name.

My room classes all have the same methods. They all had constructors and destructors. Each class had an inventory method that took an Item pointer as an argument and placed that item into the room's inventory. GetVisited was a method that returned whether the room had been visited before. Start was a method that displayed expository text whenever the user entered the room for the first time, and displayed the name of the room. GetPointers took five space pointers as arguments. The method set the arguments equal to the inherited pointers. This way, I was able to create my rooms in main and pass their pointers to be used when the player left. Look was a method that allowed the player to explore the room. Each room had multiple options. Some of the options were merely flavor text, wasting the user's steps. Other options contained the necessary equipment the player needed. Take was the option to grab items in the room and add them to the player inventory. Use was the special method that allowed players to manipulate the environment and allow them to leave. Leave gave the user the choice of which room to go to next, and then returned the pointer to the player. The player then set the returned pointer to be equal to its current location and called start on that location.

The player class had several methods. CheckInventory displayed the inventory for the user to check for the items they already took from the rooms. SetLocation takes a space pointer as a argument and sets that pointer to be the current room the player is in. Menu merely gives the players the choices and calls those methods from the current location. Item only had methods for setting and getting its name. I chose to do this so that the name of the specific items could be displayed in the inventory.

Reflection:

I was really proud of myself for making this game. Most of my initial design made it to the final code. I only really struggled with movement between rooms and handling the menu. Initially, I was giving each room its own menu and controlling all the methods from there. Then I realized I needed a player class to be able to carry the items. The original way I was doing was the rooms controlled everything, but there was no way to pass an inventory or keep track of steps. I decided to have the player class control everything and when the user made menu choices, the player called its current location's methods. I also had a hard time figuring out how to allow the player to move between rooms. I attended several lab sessions and asked a TA but I kept hitting dead ends. Then, I realized that I could create all the rooms in main and pass those rooms to each room, then set their pointers to those rooms and when the player

moved, the pointer would be passed to the player class. I also made the decision to remind players of the step limit only after they had done several things. I chose this because initially when the player wakes up, they have no idea what's going on. I wanted the player to explore and figure out what was happening before I told them that zombies are converging on their location. I thought it fit with the theme of the project. All in all, I was very pleased with my game. I didn't struggle too greatly with it. I'm proud that I only had two hurdles to overcome.

Test Case	Input Values	Expected Outcome	Observed Outcome
Bedroom : Look	Choose look in menu	2 options: closet or window	2 options: closet or window
Bedroom : Take	Choose take in menu	Option to take gun only if user has checked the closet	Option to take gun only if user has checked the closet
Bedroom : Use	Choose use in menu	Option to use chair only is user has called “Look”	Option to use chair only is user has called “Look”
Bedroom: Leave	Choose leave in menu	Only if the player has moved the chair, user is given options to go to other rooms, player pops to chosen room	Only if the player has moved the chair, user is given options to go to other rooms, player pops to chosen room
Bathroom : Look	Choose look in menu	Can only use if light is turned on. 4 options: Medicine Cabinet, Bathtub, Closet, Drawers	Can only use if light is turned on. 4 options: Medicine Cabinet, Bathtub, Closet, Drawers
Bathroom : Take	Choose take in menu	Option to take first aid kit only if user has checked the medicine cabinet	Option to take first aid kit only if user has checked the medicine cabinet
Bathroom : Use	Choose use in menu	Option to use lightswitch. User can’t choose this after completing objective	Option to use lightswitch. User can’t choose this after completing objective
Bathroom: Leave	Choose leave in menu	Only if the player has turned on the light, user is given options to go to other rooms, player pops to chosen room	Only if the player has turned on the light, user is given options to go to other rooms, player pops to chosen room
Attic : Look	Choose look in menu	Can only use if light is turned on. 2 options: Radio, Boxes	Can only use if light is turned on. 2 options: Radio, Boxes
Attic : Take	Choose take in menu	Option to take radio only if user has checked the radio	Option to take radio only if user has checked the radio

Attic: Use	Choose use in menu	Option to use lightswitch. Option to use radio only is user has checked it	Option to use lightswitch. Option to use radio only is user has checked it
Attic: Leave	Choose leave in menu	Only if the player has used the radio, user is given options to go to other rooms, player pops to chosen room	Only if the player has used the radio, user is given options to go to other rooms, player pops to chosen room
Kitchen : Look	Choose look in menu	5 options: refrigerator, table, cabinets, pantry, stove	5 options: refrigerator, table, cabinets, pantry, stove
Kitchen : Take	Choose take in menu	Option to take nonperishable food only if user has checked the pantry	Option to take nonperishable food only if user has checked the pantry
Kitchen: Use	Choose use in menu	Option to use checklist or note	Option to use checklist or note
Kitchen: Leave	Choose leave in menu	Only if the player has seen the list on the refrigerator, user is given options to go to other rooms, player pops to chosen room	Only if the player has seen the list on the refrigerator, user is given options to go to other rooms, player pops to chosen room
Living Room : Look	Choose look in menu	4 options: posters, fireplace, coffee table, TV	4 options: posters, fireplace, coffee table, TV
Living Room: Take	Choose take in menu	Option to take fireplace poker only if user has checked the fireplace	Option to take fireplace poker only if user has checked the fireplace
Living Room: Use	Choose use in menu	Option to use fireplace results in automatic loss. Option to use remote only if user has checked the tv.	Option to use fireplace results in automatic loss. Option to use remote only if user has checked the tv.
Living Room: Leave	Choose leave in menu	Only if the player has used the remote, user is given options to go to other rooms, player pops to chosen room	Only if the player has used the remote, user is given options to go to other rooms, player pops to chosen room

Player Inventory	Choose inventory in menu	Correctly displays inventory in order of items taken. Ends the game when the player has all necessary items.	Correctly displays inventory in order of items taken. Ends the game when the player has all necessary items.
Item insertion	Send item pointers to classes in main	Items are available to take from rooms and displayed in player inventory	Items are available to take from rooms and displayed in player inventory