Final Project Design
Vijay Kumar
3/15/16
Design Document

Space
- a space is an abstract class with a pure virtual “special” function
- space will have multiple derived classes with special function that takes a int. If the correct argument is input into the parameters, good things start to happen, else, catastrophe awaits.
- each space will have at least four pointers to other spaces.

- when the PLAYER leaves one space and goes to the another, it’s space pointer points to a new space. Some spaces can only be gotten to via a specific space.

Team
- the team class is an abstract base class. It will allow for derived classes of other team mates
- this team class will hold strength values
- set strength and get strength functions
- functions that can manipulate data in a specific space

Protagonist
- it has a linked list (struct) within the class that contains a pointer to ints (or Team class). When a player builds his team, the new team mates are linked in the stack.
- The Player will be able to build a team of 5, including himself.
- The player will have a pointer to a space.
- Has a vector of strings that represent ‘things’ collected
- has an int for strength (get and set)
- has an int for money in pocket (get and set)

Different Team Mates
- each of the different possible team mates will be represented by ints.
- each team mate will perform well in a certain space, i.e. use a certain team mate at the airport and you can launch in the helicopter.
- some team mates will have positive effects when their functions are used in a certain room, (ie the engineer will be able to crack the safe without issue)
- others will have negative effects.(ie the “idiot” will try to crack the safe only to make it lock for good

Space 1:

The Streets: Middle of the night. Deserted town. A bum lying about. An old lady taking a dog on a walk. We have pointers to a **creepy Bar**, a **Diner**, the **Deserted Library**, and a **small town airport**.

Space function:

- if you give the bum food, the special function will return his last name that you jot down in your journal (array of strings)

Team actions
- we can add bum to our team. add 5 points to strength
- if we try and talk to the old lady, we get bit by the dog and our strength is lowered.
- the bum has a very low strength level, but is the only person that can fly the helicopter

Thing actions
- collect the bums name. sounds familiar? store in array of strings.

Space 2:
Creepy Bar: a few drunkards. One mousy looking man at the bar. A bartender. A young cocktail waitress. From here we can go back to the **street**, Into the **Diner**, the **safe** or to the **Airport**.
Team Actions
- we can add the man at bar; he is a laid off accountant — worked at the ivory tower ; has very low strength.
- bartender - he's a brut with a very high strength level. He can fight. had large strength value
- try to add the cocktail waitress and she say's f' off and throws drink in your face. deduct strength.

special function
- if you add bartender, room function returns p1023, a note from the remains of a busted safe. store it in the array

Space 3:
Diner: It's deserted aside from an employee, 20-something asthmatic kid playing video games on his cell phone. From here we can go back to the **street**, to the **airport**, up on the **Bar**, or on the kids **cell phone.**

special function:
returns string for code snippet.

Things we need: Asthmatic Kid need to pay him 500 bucks

Space 4:
Ivory Tower Roof: Doesn't have street level doors. Only access is through a helipad on the roof. There are **three doors** on the roof, each leading to somewhere unknown, or we can get back in the **helicopter**.

special function returns pointer to rooms behind doors.

Space 5:
Small airport:

a small shed is the terminal for this airport. the airport is completely deserted aside from a beat up helicopter. From here we can go to the **street**, the **diner**, the **bar** or the ivory tower

Things we need:
bum
need to find the key: key is under something heavy. bartender can grab but reduces strength

special function:
if we have key, the special function returns a pointer to the ivory tower rooftop (use templates?)

Things we need:

- **bum to fly helicopter**
- **key to start helicopter**

Space 6:
Ivory Tower door 1: through this door we find ourselves in a lobby with armed guards. From here we can go to a room marked **"main frame"**, back to the **helicopter**, or door **2 or 3.**

special function returns pointer to mainframe….. if strength is above zero.

things we need:

- **accountant to access locked doors.**
- **bartender to fight armed guards.**

space 7:
Ivory Tower door 2: through this door we find ourselves in a circular room filled with hooded people chanting. In the middle of the room is a table where a girl is tied down. She is about to be sacrificed. This room reduces strength dramatically, but if we save the girl before attempting to go to the mainframe, we get a strength boost reward. If we have a "bad" person on our team, the attempt to save the girl is thwarted and we don't get a strength boost. To get into the main frame we need a certain character. From here we can go to the **helicopter**, **door 1 or 3**, or to a **door** in the back of the room marked "**Main frame**".
things we need:

- can of gasoline
- lighter

special function returns pointer to mainframe….. if strength is above zero.

space 8:
Ivory Tower door 3: Through this door is an oxygen chamber. Pure, concentrated oxygen. This room causes our strength to diminish automatically due to the abundance of oxygen. we get very sleep in this room. on the other side of the room, a door marked "**Main frame**. We can go here or back to the **heli** or to doors **1 or 2**.

things we need:

- asthmatic character that gets stronger in the room

special function returns pointer to mainframe….. if strength is above zero.

space 9:
Main Frame room: This room needs a certain character to be able to break through the firewall and redirect all oxygen flow back to the streets.
special function takes string as parameter. if string is correct we a return an int that determines whether we won the game or not.

things we need:

- player to finish bit of code
- code hint is Kipp, P1022, Par4, L17, W8

```
class Space
{
private:
        Space * spaceName1;
        Space * spaceName2;
        Space * spaceName3;
        Space * spaceName4;
        string placeName;
        vector<string> people;
public:
template <class T1, class T2>
T1 virtual special(param T2) = 0

Space(Space* 1,Space* 2, Space* 3, Space* 4, string placename)

}


class Bar : public Space

{
public:
template<class T1, class T2>
T1 virtual special(param T2);
Bar(Space* 1,Space* 2, Space* 3, Space* 4, string placename):  Space(1, 2, 3, 4, placeName)
{ add strings to vector }
}


class Protag
{

private:
        int team strength;
        int money;
        vector<string> team;
        vector<string> notes;
        Space* place;
        int noteLimit = 0;

public:
Protag(); // set strength, money, and place to Street Space
set get functions for variables.
}
```

In Main:

- Create Spaces
- Create protag
- cout << what we see and what we can do.
- cout << menu of options
  - talk to lady with dog
  - talk to bum
  - go into bar
  - go into diner
  - go into airport
  - go into deserted library

```
if talk to lady
        - dog bites; strength reduced
        - back to menu
if talk to bum
        - find out he's a drunk retired, helicopter pilot
        - cout << menu of options
                - ask him to join team
                - go into bar
                - go into diner
                - go into airport
                - go into deserted library
if go into bar
        - cout << what we see and what we can do
        - cout << menu of options
                - talk to bartender
                - talk to cocktail waitress
                - talk to man at bar
                - go out to street
                        if talk to bartender
                                - ask to be on team
                                -  talk to cocktail waitress
                                - talk to man at bar
```

```
                    if ask to be on team
                        - gain access to safe behind bar
            if talk to cocktail waitress
                - throws drink in face - -reduce strength
                - back to bar menu of options
            if talk to man at bar
                - find out he’s an old accountant at ivory coast
                - gives you string of keys for notebook— add to vector
                - back to bar menu items

if go in to diner
    cout << what we see and what we can do
    cout << menu options
        - order some food - lose money gain strength
        - talk diner boy
            - pay him to be on team
            if boy is on team
                - special function returns pointer to cell phone space
```

Thoughts:

Every space has a menu in the form of a vector. When an option is no longer possible, that vector option is deleted (ie. no more bartender to chat with)

example of accessing a room:

player1.getPlace()->talkToLady() // if all spaces are friend classes of Protag, then each function can access member variables of player

Every room could have these fuctions:

- friend();
- badDecision()
- getKey();
- moveOn();

Final Changes:

The major changes I implemented for this project were found in the Menu Options. I used a vector of pairs (a string and an int). The string was the menu option and the int was a tag that associated the certain menu option to a room behavior. By tagging the menu options this way, I was able to randomize the menu options so the user wouldn’t learn the games behaviors right away, i.e. “if i enter option 1 something bad happens.”

Testing:

Test 1: Enter characters for input rather than integers.
Expected Result: If we don't use a proper input validation we'll get an infinite loop.
Actual Result: As expected, an infinite loop occurred. Therefore, I implemented a do while loop for the input. the loop continued until the user entered a valid integer.

Test 2: Add Keys and Friends
Expected Result: The vectors should populate properly if the correct calls are made
Actual Result: Found a segmentation dump caused by the FlushKeys(), which removed any blank strings from the vector. This happened because of a misused loop.

Test 3: Exiting
Expected Result: By selecting the appropriate menu option user will exit.
Actual Result: exiting works well

Test 4: Run the Game
Expected Result: Game should run without error in all cases
Actual Result: Was not able to find a path that broke the program. Success!!