Shih-Chao Hung, Yue Chen, Qinyuan Song

Escape Room

1. What is the domain?

This is a contextual room escape decryption game. The user's goal is to escape the room through interaction with objects in the rooms.

2. Who are the users?

The rooms in this game are relatively complicated. Thus, it requires the players to be careful enough to find details hidden in the room. Besides, we give a lot of help when users type in commands. So, this game is friendly to any users no matter if they are good at programming languages. This complicated game can attract attention for some players who like to decrypt or like to play escape games. They would like to play this kind of game at home.

3. What kinds of things will those users do with your project?

This is a game which has a theme about decryption and escape from sealed rooms. Users can use typing in commands to interact with the program. The goal of this game is to get needed items from the sealed rooms and then open the locked exit door.

The basic commands used in this game include “search something”, “move to someplace” and “check what I have”. Players can use these commands to interact with some objects in the sealed room to find the method to escape.

4. What similar applications, libraries, or languages already exist and how will yours be different?

Using Haskell to implement this game is very innovative. All the interfaces are based on text. This program will print the result of each command which is typed in by players. This is a completely new game based on haskell. So, there are no similar tools nor languages that exist already.

5. The important type data and design decisions.

Type and data:

Room: this is the data to save the room information.
Objects: This is the data to save the object information. The object means something the player can interact with, such as table and door.
Item: This is the data to save the item information. Items means something can be put into player’s package
Cmd: This data is the commands that the player can use to change or show some data.

Function:
cmd: get the command and deal with commands. 'cmd' can solve searching doors and objects, check the bag and move function.
Getfunctor: This is a functor of most 'get' functions.
o_door: This function shortens the object's door type.
o_obj: This function shortens the object's door type.
run_code: Getting the input from the user and transforming the input to cmd function to implement the command.
Check_lock: It is the function which can check the object lock situation and try to unlock it by the key in the bag or password.

6.Design decisions

1. We used record syntaxes as a state for four kinds of elements, including room, object, item and player. We think a structure with record syntax can make all statements of one element as an entirety. In the beginning, the data structure is relatively simple. All elements have some same subelements, including id and name. We design a functor named Getfunctor. This design can form the get function form, which will make adding new get function code process easier for developers and set the get function form to prevent some error caused by different kinds of get function. That can make this project extensible.
2. During designing, the first focus is about how to deal with players' input. It is important because this game is based on text. Players may not be a master hand in programming. It is unrealistic for them to type in the correct functions. Our plan is to check what players type in. If it can be a command, turn it into a cmd(data of command).
3. We use functional ways to build the static variables.
   Like o_door :: Int -> String -> String -> String -> Bool -> Lock -> Objects
   It builds an 'Objects' variable by a function. This way can make the data . That can make the developers to add the