CSC241 Project 1

by Alex Pantaleev and Lynna Cekova

The project consists of building a small virtual world simulation that allows a human user to interact with it through a text-based, command-line interface.

Description of the world

The virtual world consists of rooms that are connected with doors, and creatures. Every room can have at most four doors, connecting it to its neighbors on the North, South, East, or West. Rooms and doors always stay in the same place. Creatures, on the other hand, can move between rooms (for example, enter a room or leave it). A creature leaving a room always means that the creature is entering another room—there are no corridors and such. There are three types of creatures: the player character (PC), non-player characters (NPCs), and animals. There is only one player character. There can be multiple NPCs or animals.

A room may contain at most 10 creatures. That includes the PC, if s/he is in the room. If another creature tries to enter the room, it is denied access with the message "Room full."

Rooms and creatures all have unique names, as well as text descriptions. Each creature also has access to a "look" command, which lets it see the name, state, and description of the room it is in and the names, types, and descriptions of the other creatures in this room. Rooms can be dirty, half-dirty, or clean. Creatures have preferences of whether the room that they are in should be dirty or clean: Animals want their rooms to be clean; NPCs want their rooms to be dirty; the PC doesn't care. Animals will only stay in rooms that are clean or half-dirty, while NPCs will only stay in rooms that are dirty or half-dirty. The PC can stay in any type of room. The PC also has a "respect" parameter, which has an integer value and represents how much all other creatures in the world respect the PC. The initial value of this parameter upon game startup is 40.

All creatures can perform certain actions that affect themselves, other creatures, or the rooms. The PC, animals, or NPCs can all perform a "clean" or "dirty" action over a room. Only the PC can choose when one of the "clean" or "dirty" actions is performed *in the room where the PC currently is*, either by performing the respective action him/herself, or by making one of the animals or NPCs do it. If the "clean" action is performed over a dirty room, the room becomes half-dirty; if it is performed over a half-dirty room, the room becomes clean. Similarly, if the "dirty" action is performed over a clean room, it becomes half-dirty; a half-dirty room becomes dirty. The "clean" action doesn't affect clean rooms, and the "dirty" action doesn't affect dirty rooms. When the state of a room changes, all creatures that cannot be in a room of the respective state leave the room and move to an adjacent one, each leaving creature picking the adjacent room randomly. For example, animals will leave if the room has been half-dirty but has now become dirty because someone performed a "dirty" action. If the adjacent room is not fit for the creature either (for example, it is also dirty), the creature will not leave again, but will perform the action that would make the room fit for it, once. For example, an animal will perform the "clean" action once to a dirty room and make the room half-dirty; then the animal will stay there. If a creature cannot go to an adjacent room because the respective adjacent room is full, the creature will try to go to another adjacent room. If all adjacent rooms are full, it will drill a hole in the ceiling and crawl out of the house, which means that this creature will disappear not only from the room but form the whole simulation. Because the PC has been as inconsiderate as to chase a creature out of the house, every creature in the room where the PC is will "grumble" or "growl" at the PC once, in addition to all other "grumble" and "growl" or

“smile” and “lickFace” actions the creatures might be performing because of the room's state having been affected. (See below about grumbling, etc.) Note: only the creatures who have not already left will do that additional “grumble” or “growl” because of the discontent creature leaving through the roof. E.g, if there are two unhappy animals and one leaves for an adjacent room, taking the last spot and making all adjacent rooms full so that the second animal cannot go anywhere, this first animal will not be in the original room to growl when the second animal sees it cannot change rooms and leaves through the roof.

The PC can also make a creature leave the room, in a direction of the PC's choice, even if the creature likes the current room. Again, if the creature doesn't like the adjacent room it moves to, it will perform the respective “clean” or “dirty” action on it. If it cannot execute the PC's order to move to an adjacent room because the target room is full, it will stay where it is and “grumble” or “growl” once.

NPCs can “smile” and “grumble.” Animals can “growl” and “lickFace.” They do it only in the room where the PC is, in reaction to the PC's actions. “smile” and “lickFace” are expressions of gladness; “grumble” and “growl” are expressions of discontent. Any time the PC causes “clean” or “dirty” to happen in the room where the PC is, all other creatures in the room react respectively and according to their preferences. (Those who leave react before they leave). For example, if “clean” is performed, animals lick the PC's face; if “dirty” is performed, they growl. Any time the PC receives a “smile” or “lickFace,” the PC's “respect” value increases by 1. Any time the PC receives “grumble” or “growl,” the “respect” value decreases by 1. (For example, if three animals growl, “respect” is lowered by 3 total). If the player makes another creature perform the “clean” or “dirty” actions (as opposed to performing them him/herself), this particular creature reacts three times to the room's state having changed, instead of just reacting once (smiles or licks the PC's face 3 times, for example). Every other creature in the room still reacts (in gladness or discontent) only once per creature.

If the “respect” parameter of the PC falls to 0 or below, the game ends with the PC in disgrace. If the “respect” rises above 80, the game ends with the PC in praise.

Input, output, and game interface

The game interface is a command line, where the user can input String commands, and where text messages are shown to the user.

Upon program startup, the user should be asked to provide an input file with information about the world (rooms and creatures). Your program should parse this file, and, based on the information in this file, create the rooms and the creatures that are in them (including the PC). A sample input file is provided to you, together with the description of the file format. You can use this file in writing and testing your project (or you can make your own input files if you so wish), but keep in mind that other input files will be used upon evaluation of the projects. (This means you do need to parse the file based on the file format; if you put the input values in the code themselves, I will catch you.)

Next, the user should be presented with a prompt to issue her/his String commands. When read by your program, these String commands should be parsed and used to make the PC perform actions or order other creatures to perform actions. Output will be returned to the user (telling what is happening as a result of the actions), followed by another prompt for a user command. The user will be presented with prompts for commands until the user types “Exit,” or until the game ends (in praise or disgrace). If the game ends, the user receives a message. (Sample messages are“Congratulations, you are praised!” or “Shame on you! You lose.” Feel free to make your own messages.)

Some examples of user-issued commands:

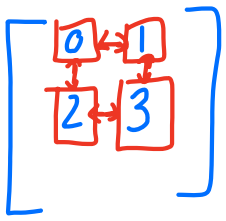
- The user typing “look” should list the name, state, and description of the room and the names, types, and descriptions of creatures in the room.

  For example: Violet Room, half-dirty, a large round room with violet draperies; Peter, animal, a brown dog; Mary, NPC, a tall woman; Lily, animal, a black-and-white cat.
- The user typing “clean room” will cause the PC to clean the room s/he is in (and all respective consequences to happen). Print out what is happening. Assuming the PC had a respect level of 100 before s/he cleaned the room, you can print, “Violet room cleaned. Mary the NPC grumbles and leaves the room North. Lily the animal licks your face. Peter the animal licks your face. Your respect is now 101.” or “Mary the NPC grumbles and leaves the house through the roof because all adjacent rooms full. Lily the animal growls. Peter the animal growls. Lily the animal licks your face. Peter the animal licks your face. Your respect is now 99.” [Note: In the second case Lily and Peter first growl because of Mary leaving the house, then lick your face because of the room becoming clean. It happens in this order because, in this case, you first examine Mary and all reactions connected to Mary happen before other creatures are examined. The order of actions is fine, as would be another order, depending on in what sequence the creatures are examined (which itself depends on how they are arranged in a data structure).
- The user typing “Mary clean room” will cause the PC to make Mary clean the room if she is still in the room. If she is not, there should be a message saying that there is no such creature in the room.

Wording of messages and commands: The wording of the messages themselves is of no importance, as long as all the information of what is happening is presented. I don't care if you say “Peter licks your face” or “Peter's tongue goes all over your face and he drools over you” as long as you give all the information. The wording of the commands is important so that the user knows what to type. E.g. “Mary leave North” and “Mary go N” would both fine for causing the Mary NPC to go North, but you need to decide what exact command will work for your program. Define the format of your own String commands and show them to the user, together with a description of what they do, any time the user types “help” (without the quotes) in the prompt.

Game

Names = H's aka John Doe = 2



n numRooms; (between 1 and 100)

For each room, you provide 5 integers (Give 0 for clean, 1 for half-dirty, 2 for dirty, or -1 for no neighbor for the state and a number (name) of the respective neighbors

↳ nam

↑ north south east west

state of current room

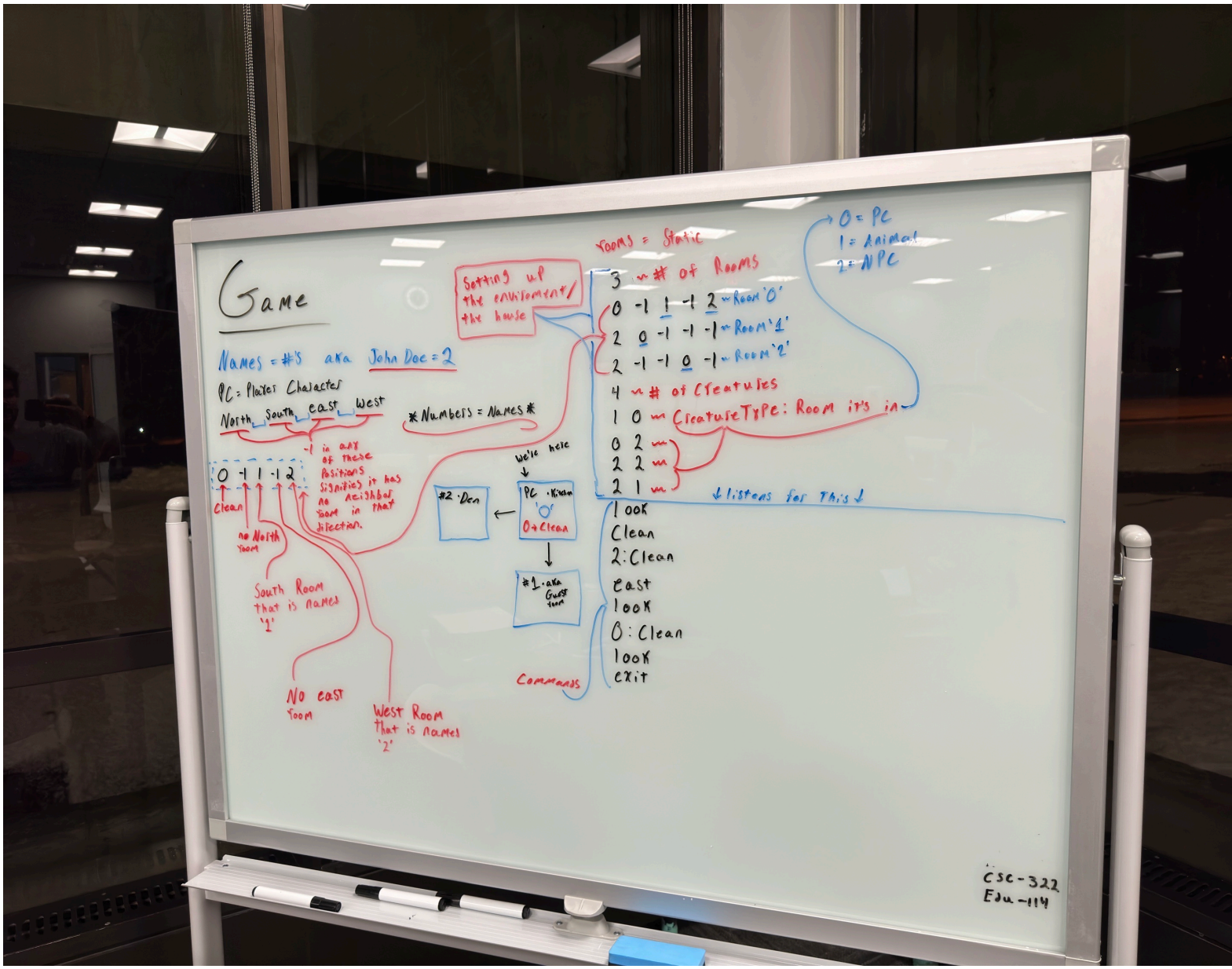
m numCreature; (Between 1 and 100)

For each creature, you provide 2 integers (For creatureType: 0 is PC, 1 is animal, and 2 is NPC)

(For location, it is the room # you want the creature, numbering starting at 0.

↑ ↑

Creature Type | Location


Game
Setting up the environment/ the house
Rooms = Static
3 ~ # of Rooms
0 -1 1 -1 2 ~ Room '0'
2 0 -1 -1 -1 ~ Room '1'
2 -1 -1 0 -1 ~ Room '2'
4 ~ # of Creatures
1 0 ~ CreatureType: Room it's in
0 2 ~
2 2 ~
2 1 ~
0 = PC
1 = Animal
2 = NPC
Names = #'s aka John Doe = 2
PC = Player Character
North South east west
* Numbers = Names *
-1 in any of these positions signifies it has no neighbor room in that direction.
0 -1 1 -1 2
Clean
no North room
South Room that is named '1'
No east room
West Room that is named '2'
We're here
#2 · Dan
PC · Kiran '0' 0 = Clean
#1 · aka Guest room
↓ listens for this ↓
look
Clean
2:Clean
east
look
0:Clean
look
exit
Commands
CSC-322
Edu-114