The following shows a table we can consider using for the “POC” (Proof of Concept) for Zork: Phase II.

I would highly recommend simply giving each object a single number value, and it will keep that same value whether it is a “Direct Object” or an “Indirect Object”. While we can start the enumeration a 0, if we search for a <key,value> pair in a map, any value *not* in the map will resolve to a 0, which would falsely indicate “N”, if that were the value associated with the 0.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **ACTION** |  | **OBJECT** | *Synonyms* | | | ***DO?*** | ***IO?*** |
| 1 | N | 1 | MAILBOX |  |  |  | Y |  |
| 2 | NE | 2 | LEAFLET | BROCHURE | PAMPHLET |  | Y |  |
| 3 | E | 3 | SWORD |  |  |  | Y | Y |
| 4 | SE | 4 | ROPE |  |  |  | Y |  |
| 5 | SE | 5 | SACK | BAG |  |  | Y |  |
| 6 | SW | 6 | TABLE |  |  |  | Y | Y |
| 7 | W | 7 | WINDOW |  |  |  | Y |  |
| 8 | NW | 8 | LANTERN | LAMP |  |  | Y | Y |
| 9 | N | 9 | AXE |  |  |  | Y | Y |
| 10 | U | 10 | KNIFE |  |  |  | Y | Y |
| 11 | D | 11 | EGG |  |  |  | Y | Y |
| 12 | I | 12 | NEST |  |  |  | Y | Y |
| 13 | O | 13 | THIEF |  |  |  | Y | Y |
| 14 |  | 14 | TROLL |  |  |  | Y | Y |
| 15 | 15 |  |  |  |  |  |  |
| 16 | 16 |  |  |  |  |  |  |
| 17 | 17 |  |  |  |  |  |  |
| 18 | 18 |  |  |  |  |  |  |
| 19 | QUIT | 19 |  |  |  |  |  |  |
| 20 | LOOK | 20 |  |  |  |  |  |  |
| 21 | OPEN | 21 |  |  |  |  |  |  |
| 22 | CLOSE | 22 |  |  |  |  |  |  |
| 23 | TAKE | 23 |  |  |  |  |  |  |
| 24 | DROP | 24 |  | PUT | PLACE |  |  |  |
| 25 | THROW | 25 |  | TOSS |  |  |  |  |
| 26 | MOVE | 26 |  |  |  |  |  |  |
| 27 | KILL | 27 |  |  |  |  |  |  |
| 28 | SWING | 28 |  |  |  |  |  |  |
| 29 | CLIMB | 29 |  |  |  |  |  |  |
| 30 | TURN | 30 |  |  |  |  |  |  |

Your task for the next stage of Zork is to continue the “Accept Command” and “Process Command” part of the game loop for Zork.

Right now, we’re going to work on the Accept Command part. We’ll get a string and parse it into a string vector of tokens, then examine the tokens to see if we can further reduce it to only 3 tokens at most: an Action word, a Direct Object and an Indirect Object. These three tokens are positional in the vector after eliminating “garbage words” (words which are not recognized as A, DO or IO). Note that a word might be in the DO listing and is acceptable in that position, and might being a not processed as an IO item, even though it’s not a ‘garbage word’ (in the recognized list of objects).

The table shows a listing of Action words with associated numerical values and a listing of Objects with numerical values. I would recommend using the same value for objects whether or not they are DO or IO items.

How do we know if an object is a DO or an IO (or both)? We simply have a list of DO and IO items (numerical values only..?). We can check the DO listing to see if the item number is there (“searching”), or… if it’s in the IO token location in the token vector of strings, we check the IO listing for the item number. If the item is not listed, the word is a “garbage” word.

How to proceed initially:

Just develop the program to accept a command from the user at the keyboard. Parse the command into tokens. My suggestion would be to start with the first token (the “Action” token). If the token is not an Action word, then collapse the token vector by removing that element, and examine the 1st token again for an Action word (assuming that there is a token left). Do this until either you have no tokens or you have found a legitimate “A” word. Next, if you have more than a single token, proceed to the D.O. token and do the same. Keep erasing that token while it’s not a legitimate D.O. word or until you have exhausted all the tokens. Finally, if you have more than two tokens, do the same with the I.O. token.

Display the following:

*Command (repeat the command)*

*A #*

*DO #*

*IO #*

Examples are listed below.

***KILL THIEF WITH KNIFE***

***KILL 27***

***THIEF 12***

***KNIFE 9***

***THROW AXE AT TROLL***

***THROW 25***

***AXE 8***

***TROLL 13***

***OPEN MAILBOX***

***OPEN 21***

***MAILBOX 0***

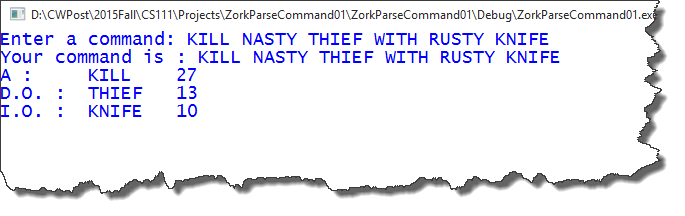
***NORTH (or N)***

***NORTH 0***

***GO NORTH***

***NORTH 0***

**Sample output:**

******

At this point in time you are pretty much done with Zork: “2A”, and can go on with “2B”. Consider this however: what do we now use to handle the command: the string in the vector, or the numeric value with which it is associated?

**Handling the command: Zork: 2B**

As discussed during the lecture, we can start backward with examining the Indirect Object and (assuming we have one) and seeing if there is a ‘handler’ for it. If so, search for a handler for the Direct Object (we have to have one, if we parsed correctly). If not, simply state that you don’t understand the command. If there is a handler routine, then search to see if there is an Action routine for that DO within the IO. If so, handle it, otherwise indicate that you don’t understand the command.

If no Indirect Object exists, the see if a handler routine exists for the Direct Object. If so, see if there is a handler for the Action word otherwise indicate that you don’t understand the command.

Finally, in the event that you only have an Action token, see if there is a handler for the Action word and process it, otherwise indicate that the command is not recognized, or not a valid command.

List of valid commands (reduced to their **A**, **DO**, and **IO** tokens) and their responses:

**NORTH**

*(go in that direction or indicate that the player cannot go as appropriate)*

**KILL TROLL**

“WITH WHAT?”

**KILL TROLL SWORD**

“TAKING A MIGHTY SWING WITH THE SWORD, YOU HAVE KNOCKED THE TROLL SENSELESS AND DAZED.”

**KILL THIEF SWORD**

“YOU NEED TO BE QUICKER. THE THIEF SAW YOU START TO SWING THE SWORD AND NIBMLY JUMPED ASIDE, STRIKING YOU UNCONCIOUS IN THE PROCESS. HE THEN KILLED YOU WITH YOUR OWN SWORD. YOU HAVE DIED.”

*(REGENERATE THE PLAYER IN A LOCATION SURROUNDING THE HOUSE OR IN THE FOREST USING A RANDOM LOCATION)*

**KILL THIEF KNIFE**

“YOU ALMOST TOOK HIM BY SURPRISE, BUT HE DODGED YOUR CLUMSY ATTEMPT AND LEFT, BUT NOT BEFORE TAKING EVERYTHING FROM YOU.”

**THROW KNIFE THIEF**

“NEXT TIME, PRACTICE A LITTLE BEFORE THROWING KNIVES. THE THIEF CAUGHT IT AND THREW IT RIGHT BACK AT YOU, PIERCING YOUR HEART. YOU HAVE DIED IN VAIN.”

*(RESPAWN THE PLAYER IN A RANDOM LOCATION SURROUNDING THE HOUSE OR IN THE FOREST)*

**TAKE ROPE**

“TAKEN.”

*(MAKE SURE THAT IT IS ‘TAKEABLE’… PRESENT AND VISIBLE, AND THE PLAYER DOESN’T ALREADY HAVE IT)*

*(ADD IT TO THE PLAYER’S INVENTORY.)*

**TAKE LEAFLET**

“TAKEN”

*(MAKE SURE THAT IT IS ‘TAKEABLE’… PRESENT AND VISIBLE, AND THE PLAYER DOESN’T ALREADY HAVE IT)*

*(ADD IT TO THE PLAYER’S INVENTORY.)*

**DROP LEAFLET**

“DONE.”

(YOU CAN RANDOMLY ALTERNATE A RESPONSE WITH “OK” OR “DROPPED”)

(MAKE SURE TO REMOVE IT FROM THE PLAYER’S INVENTORY AND TO ADD IT TO THE LOCATION’S INVENTORY)

**PUT LEAFLET MAILBOX**

“DONE.”

(MAKE SURE THE MAILBOX IS AT THE LOCATION, AND THAT IT IS OPEN.)

(YOU CAN RANDOMLY ALTERNATE A RESPONSE WITH “OK” OR “DROPPED”)

(MAKE SURE TO REMOVE IT FROM THE PLAYER’S INVENTORY AND TO ADD IT TO THE MAILBOX’S INVENTORY)