Dungen

Overview

Dungen is a suite of two programs which create and manipulate a common database.

The database consists of several sections which, when loaded by the database loader program DBASE, describes a "dugeons and dragons" environment. The database also contains all the vocabulary necessary to manipulate the database properly.

The program which allows the manipulation of the database is MUD (Multiple User Dungeon). The program administers up to 36 players at any one time, allowing controlled interaction between the players and the database; and the players and themselves.

Communication with MUD is in the form of imperative statements of between 2 and 4 words (sometimes more). The statements may be connected with words such as and, then to give the impression of english.

The main difference between this game and others like it; notably ADVENT, ZORK and HAUNT, is the interaction between the players which is allowed. Other differences are the open-endedness of MUD in that there is no top score only advancement to a higher level (every 1000 points or so)

Database

The database is described in great detail in the sections following this one. The idea is to make the environment of the dungeon as distinct as possible from the database manipulation program. This allows the dungeon creator to concentrate on the dungeon itself and not need to rewrite the manipulation program as well. (As needs to be done in ADVENT).

There are 6 sections to the database brief descriptions of each are given below.

Directory	-	declares room names to be used in the rest of the database definition.
Rooms	-	defines the rooms declared above.
Vocabulary	-	defines the manipulation language available.
Travel	-	defines the linkages between the rooms defined above.
Objects	-	defines the objects to be found in the database.

Data - cont.

Messages - defines random text to be printed out, i.e. Help Hints etc.

Each section is introduced by its name prefixed by a star (*). The overall design of the database owes much to the ADVENT database however the Vocabulary section is much extended, and the overall appearance is much improved by the avoidance of numbers wherever possible!

Directory Section.

The section is introduced with

*Directory<tab>number.

the number indicates how much space should be reserved for the room directories.

The format within the section is simply the names of each room seperated with tabs or end of line characters.

room1<tab>room2 etc...

the first six characters of every name must be unique.

Rooms Section

The section is introduced with the line

*Rooms.

The section defines the type of room which the room is. It also gives the short and long form description of the room if any.

There are 3 pieces of information associated with a room in this section.

a) The attributes. (LIGHT, ROOM, CORRIDOR)
b) The room title
c) The room description

```
name<tab> attributes
[illegible]<tab> short form description
[illegible]<tab> long form description
    <tab>    which may be longer
    <tab>    than one line.
```

~~a null description or entry is signified with the character / (slash)~~

duplicate descriptions may be accommodated for by the construction

%Roomname.

Rooms - cont.

at which point the relevant description from that room is substituted.

(Actually a pointer to the correct description is substituted)

This practise is to be encouraged since it allows more descriptions to be used.

Vocabulary Section

VERB { MOTION
ACTION
SPECIAL

OBJECTS	-	that which is not a room or player
CLASSES	-	define collections of objs.
ADJECTIVES	-	object modifier.
ADVERBS	-	verb modifier.
PRONOUN	-	last 'object' referenced.

The vocab. section defines the means by which ~~other~~ players interact with each other and the database.

Commands are in the form of imperative clauses strung together to make sentences. Each clause is about 3 or 4 words long and normally cause some change in the status of the database, the player or some other player.

In the semi-formal description which follows, words in capital letters are 'pseudo-reserved' words, which have pre-defined meanings for the database loader. However these words may be defined for use within the dungeon produced, as long as there is no ambiguity caused by such a definition.

Vocab. Section.

<u>Motion Verbs</u>

<word> MOTION ~~{synonyms}~~

Indicates to the loader that this word is a valid motion verb. Synonymous verbs are indicated by suffixing MOTION with an already defined motion verb from the required group. (Possibly)

for example

```
w         MOTION
west      ~~MOTION~~ SYN     w
westerly  ~~MOTION~~ SYN     w          (or west)
```

all describe the same motion.

Vocab. Section

SPECIAL verbs

```
                 [ <function>  ]
<word> SPECIAL   [ ~~<synonym>~~ ]
```

This makes available to the player 'predefined functions' of the database manager or interpreter. Currently implemented functions are.

HELP	-	type out message 1
EXPLAIN	-	type out message 2
TELL	-	communicate with other players
SAY	-	re type the line after say in quotes.
GET	-	pick up object
DROP	-	drop object.
KILL	-	enter fight routine between players
QUIT	-	finish game
SCORE	-	type out experience points.
DUMP*	-	dump database
LOAD*	-	load database
LOOK	-	describe room and contents.
DEBUG*	-	place interpreter/manager in debugging mode.

* Require MASTER WIZARD privs

Vocab. section.

Action verbs

Action verbs cater for database dependent functions. For example . . . feed bear with food . . .

The interpreter has no way of knowing beforehand what is contained within a database, so the ACTION construct allows the dungeon creator to associate various primitive functions with a word. The format of an action verb entry is.

```
word<t>ACTION<t>objclass<t>reqclass<t>action<t>m,n.
```

where.

objclass	is the class of objects which the verb expects, to be able to perform the specified action.
reqclass	is the class of objects which the verb expects to be present so the action can take place.
action	the action performed by the action verb if all is successful.
m	is the message number typed if the verb is successful.
n	is the message type otherwise.

Vocab. section.
<u>Action verbs</u> cont.

the following actions are allowed.

INC	-	increment object's property value by one. (up to maxprop, print failure message if attempts to increment above maxprop)
DEC	-	decrement object's property value by one. (down to zero, print failure message if attempts to decrement below zero)
TYPE	-	simply type out m or n depending on whether succesful or not.
MOVE	-	move object to location m (i.e. the m'th room declared) if succesful otherwise type out message n.
TRANSPORT	-	shift the player (only) to location m if succesful to location n otherwise if -1 it kills him/her if zero nothing happens.
LSAB	-	shift player + treasures to location m if succesful (lock stock and barrel) to n otherwise (-1 - kills ϕ - nothing).

Vocab. section

Objects

Defines an object name and indicates the class to which the object belongs. The format is.

word<tab>OBJECT<tab> class name.

where classname is a predefined class word. The word any indicates this object will satisfy and class requirements and none indicates that the object will satisfy none.

Vocab. section
Class words

Defines the collective word for a group of objects. They are for use with action verb definitions. The format is

word<tab>CLASS~~{synonymous classword}~~

common classes are

valuables<tab>treasure etc

the database loader generates a unique class number for the word ~~or synonymises it with a pre-declared class word.~~

Vocab. section.

Adjectives

These words are used to differentiate between two similar objects, for example if the player is carrying two types of food and (s)he says

feed bear with food

this might elicit the response

which sort of food

the player might then respond

the tasty food

and the game would continue and we shall leave unanswered the question "does the bear eat the player or not?"

The format is.

```
word<tab>ADJECTIVE{<tab> synonym}
```

there is a maximum of 36 non synonymous adjectives.

Vocab. section

Adverbs

These modify the verbs present in the clause in which they are mentioned. Most verbs ignore them, they are included for complete--ness. The format is

word<tab> ADVERB~~{<tab> synonym}~~

Vocab. section.

Pronouns

These are all automatically synonymous and pick up the last object referenced by either the player or the interpreter. Similary for players. The format is.

```
word<tab>PRONOUN
```

Synonyms

The format is

```
word<tab>SYN<tab>synonymous word.
```

the database loader simply copies the dictionary node already created, except of course for the word itself.

Travel Section

This section defines the connective properties of the rooms within the database. Each room has one or more entries in the table. Each entry is of the form.

name<tab>conditions<tab>newname<tab> motion verbs.

name	is the room name that is current
newname	is the room to which you are transported if the motion is successful.
motionverbs	is a list of motion verbs which get you to newname if...
conditions	the conditions which must be satisfied before the motion can be successful.
	currently defined conditions are.

N, NONE	-	unconditional.
E, EMPTY	-	must not be carrying anything.
any class name	-	must be carrying an object of that class, before motion is successful.

Objects section

This section defines the objects present within the database and their properties, adjectives and current location. The format is.

```
loc<tab>name<tab>prop<tab> adj, adj
$ø <tab> description for prop = ø
$1    "           "      "    "   1
 .
 :
 .
$n    "           "      "  prop = n
```

~~proper~~ prop is the current prop value

adj   are any adjectives which distinguish between it and other, similar objects.

Message Section

Defines the database dependent messages, which are typed out by action verbs etc. The format is:

```
n<tab> message
 <tab> any continuation.
```

The number of messages (in fact the maximum index number) must follow the section header like so

```
*Messages<tab>number.
```

this reserves a table number/k + remainder long.