Christopher Lanham

Implementing ZIL using FLEX/BISON

For my term project I chose to implement a subset of the Zork Implementation Language (ZIL) which can be used to build interactive fiction games. My goal for this project was to read in a file containing ZIL code and then build the game so I can play it. I used the paper “Learning ZIL - or - Everything You Always Wanted to Know about Writing Interactive Fiction but Couldn’t Find Anyone Still Working Here to Ask” as my manual on how to create ZIL code.

My project reads in a file containing ZIL instructions, parsers it into tokens, and then store those tokens into many different C structures. Those structures are then stored in arrays so they can be easily accessed when playing the game. After reading in the ZIL code my program automatically runs the game.

Token data are stored into the following structures:

Struct object;

This stores all the data related to an object such as name, description, and its location

Struct room;

This stores all the data related to a room such as name, description, and exits

Struct operation;

This contains variables to store information for different type of operations. Also contains flags which are set so I can tell which operation it is. In my program I have verb, tell, equal, remove, set, and setg operations. Verb compares the current verb with another word. Tell prints out a string. Remove removes an object. Setg sets a counter or flag to a specified number.

Struct cond;

This holds a conditional statement. It has an operation for its predicate and an array of operations as its clauses.

Struct statement;

A statement contains either an operation or a conditional statement.

Struct routine;

This stores the routine name and an array of all its statements

Struct command;

This stores a user defined command and its statements which is produced with the Understand instruction.

I used the following grammar was used for this program:

Input: | input instruction | input command;

This allows the grammar to match any number of inputs, whether it is ZIL code instructions or user commands, by using recursion.

Instruction: LB SET WORD NUM RB | LB TITLE STRING RB | LB AUTHOR STRING RB | LB RELEASE STRING RB | LB STORYDESC STRING RB | LB ROOM WORD rProperties RB | LB OBJECT WORD oProperties RB |

LB ROUTINE WORD LP RP operations RB | LB UNDERSTAND WORD operations RB;

ZIL code instructions have to be between < and >. LP and RP stand for left parenthesis and right parenthesis. A WORD is just a single word not in quotes. A STRING matches any number of words or chars as long as they are between quotes.

Regular expression used for String: "\""([^"\""]+)"\""

Room properties:

rProperties: | rProperties rProperty;

rProperty: LP LOC WORD RP | LP DESC STRING RP | LP LDESC STRING RP | exit

exit: LP NORTH TO WORD RP | LP EAST TO WORD RP |LP SOUTH TO WORD RP | LP WEST TO WORD RP | LP UP TO WORD RP |LP DOWN TO WORD RP

Object properties:

oProperties: |oProperties oProperty;

oProperty: LP LOC WORD RP | LP DESC STRING RP | LP LDESC STRING RP | LP ADJECTIVE WORD RP | LP ACTION WORD RP;

Instead of using one big properties grammar that can be used for both the rooms and objects, I found using separate oProperties and rProperties made it easier when adding all the data to my structures.

Operations: | operations operation;

Operation: LB COND LP operations RP RB | LB VERB COMMA WORD RB | LB EQUAL COMMA WORD NUM RB | LB EQUAL COMMA WORD NUM RB | LB REMOVE COMMA WORD RB | LB TELL STRING RB | LB SETG WORD NUM RB;

Command: NORTH | SOUTH | EAST | WEST | UP | DOWN |DROP WORD | TAKE WORD | LOOK | INVENTORY | EAT WORD ;

Routines are called by when interacting with objects or by other routines. In my program I have the variables called poisoned, poison\_counter, wounded, and wound\_counter. I did not implement a way to create variables so any SETG or EQUAL operation must reference these variables. I could not get routines or the understand instruction to fully work.

References:

Learning ZIL

<http://www.xlisp.org/zil.pdf>

Demonstration of the program running the game from Project 2

