Loomine

By: Ryan & Matthew

Loomine as a language

```
program : stmt_list END
;

stmt_list : stmt | stmt_list stmt;

stmt : ID ASSIGNOP mapstmt      { assignMapSymbol($1); }
     | ID ASSIGNOP plrstmt      { assignPlrSymbol($1); }
     | ID ASSIGNOP gmstmt       { assignGameSymbol($1); }
     | ID ASSIGNOP dstmt        { assignDoorSymbol($1); }
     | ID ASSIGNOP estmt        { assignEnemySymbol($1); }
     | RUN gmstmt               { runGame(); }
;
```

Loomine as a language

```
mapstmt : MAP LPAREN DIGIT COMMA DIGIT COMMA CHARACTER COMMA CHARACTER RPAREN { genMap($3, $5, $7, $9); }
    | ID                    { genIdentifierMap($1); }
    | mapstmt ADDD dstmt    { addDoor(); }
    | mapstmt ADDE estmt    { addEnemy(); }
;

plrstmt : PLAYER LPAREN DIGIT COMMA DIGIT COMMA CHARACTER COMMA CHARACTER COMMA CHARACTER COMMA CHARACTER COMMA CHARACTER RPAREN { genPlr($3, $5, $7, $9, $11, $13, $15); }
    | ID                    { genIdentifierPlr($1); }
;

estmt : ENEMY LPAREN DIGIT COMMA DIGIT COMMA CHARACTER COMMA DIGIT RPAREN {genEnemy($3, $5, $7, $9); }
    | ID                    { genIdentifierEnemy($1); }
;

dstmt : DOOR LPAREN DIGIT COMMA DIGIT COMMA CHARACTER RPAREN {genDoor($3, $5, $7); }
    | ID                    { genIdentifierDoor($1); }
;

gmstmt : GAME LPAREN DIGIT RPAREN { genGame($3); }
    | ID                    { genIdentifierGame($1); }
    | gmstmt ADDM mapstmt    { addMap(); }
    | gmstmt ADDP plrstmt    { addPlr(); }
;
%%
```

Elements of the language

- {ID}: [a-z][a-z0-9]*
- {CHARACTER}: \".\"
- End character: $$
- Assignment operator: :=
- Parentesis: [( ] and [ )]

Data Types

Doors

Allow players to progress from one map to another map in a game

Requires an (x,y) position and a token char

- Syntax:

```
ID := door(DIGIT, DIGIT, “CHARACTER”)
```

- Follows the form:

```
ID :=  door(x , y , “token”)
```

- EX:

```
d := door(5,5,”@”)
```

Players

Allows a user to play a game

Requires an (x,y) position, token char and 4 directional controls

- Syntax:

```
ID := player(DIGIT, DIGIT, “CHARACTER”, “CHARACTER”, ”CHARACTER”, ”CHARACTER”, “CHARACTER”)
```

- Follows the form:

```
ID :=  player(x , y , “token”, “up”, ”right”, “down”, “left”)
```

- EX:

```
p := player(19,5,”P”,”W”,”D”,”S”,”A”)
```

Enemies

Automated objects that will chase players around a map

Requires an (x,y) position, token char and speed in milliseconds

- Syntax:

```
ID := enemy(DIGIT, DIGIT, “CHARACTER”, DIGIT)
```

- Follows the form:

```
ID :=  enemy(x , y , “token”, speed)
```

- EX:

```
e := enemy(8,3,”M”, 20)
```

Maps

Allow players to move around a game space graphically

Requires nxm dimension set, a border char, and an inner map char

- Syntax:

```
ID := map(DIGIT, DIGIT, “CHARACTER”, “CHARACTER”)
```

- Follows the form:

```
ID :=  map(rows , columns , “inner character”, “border character”)
```

- EX:

```
m := map(20,25,”-”,”|”)
```

Games

- These incorporate maps, doors, players and enemies to make a console-based game
- Requires a speed in ms that will operate as the refresh rate

·Syntax:

```
ID := game(DIGIT)
```

·Follows the form:

```
ID :=  game(speed)
```

·EX:

```
g := game(50)
```

Error Handling

- adding player initial position out of bounds
- adding enemy out of bounds
- initializing a map with 0 dimensions
- adding a door to a map when the next map does not have that location as a valid location
- spawning an enemy on a door
- trying to run a game with no player or map

Our Struggles

- Creating a grammar
- Indexing through our command list
- Developing the game class
- Making edits as we added more data types and features

DEMO

Questions?