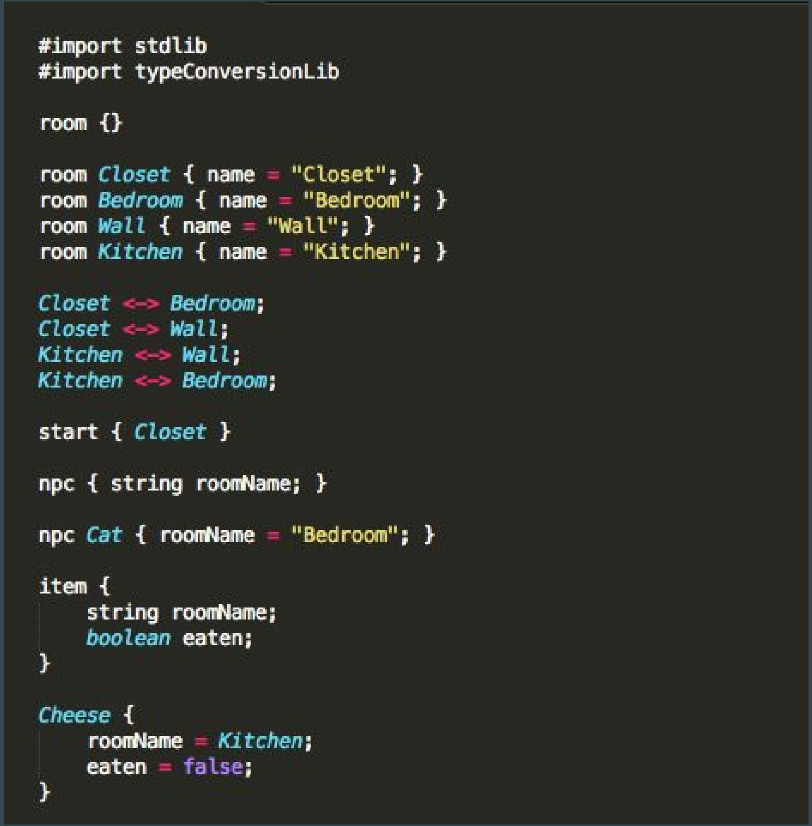
T.B.A.G.

a (t)ext (b)ased (a)dventure (g)ame language

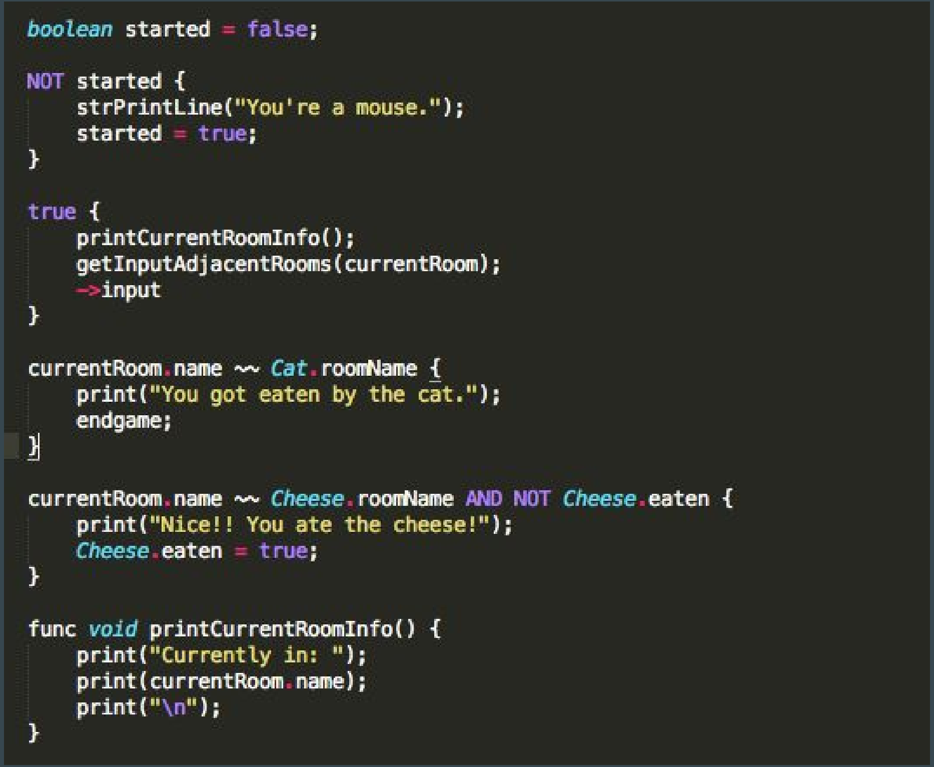
Intro

- Optimized for text based adventure games, can be used for others
- Easy to define rooms, NPCS, items
- Event-driven system
- Why Events?

mouse_cat.tbag

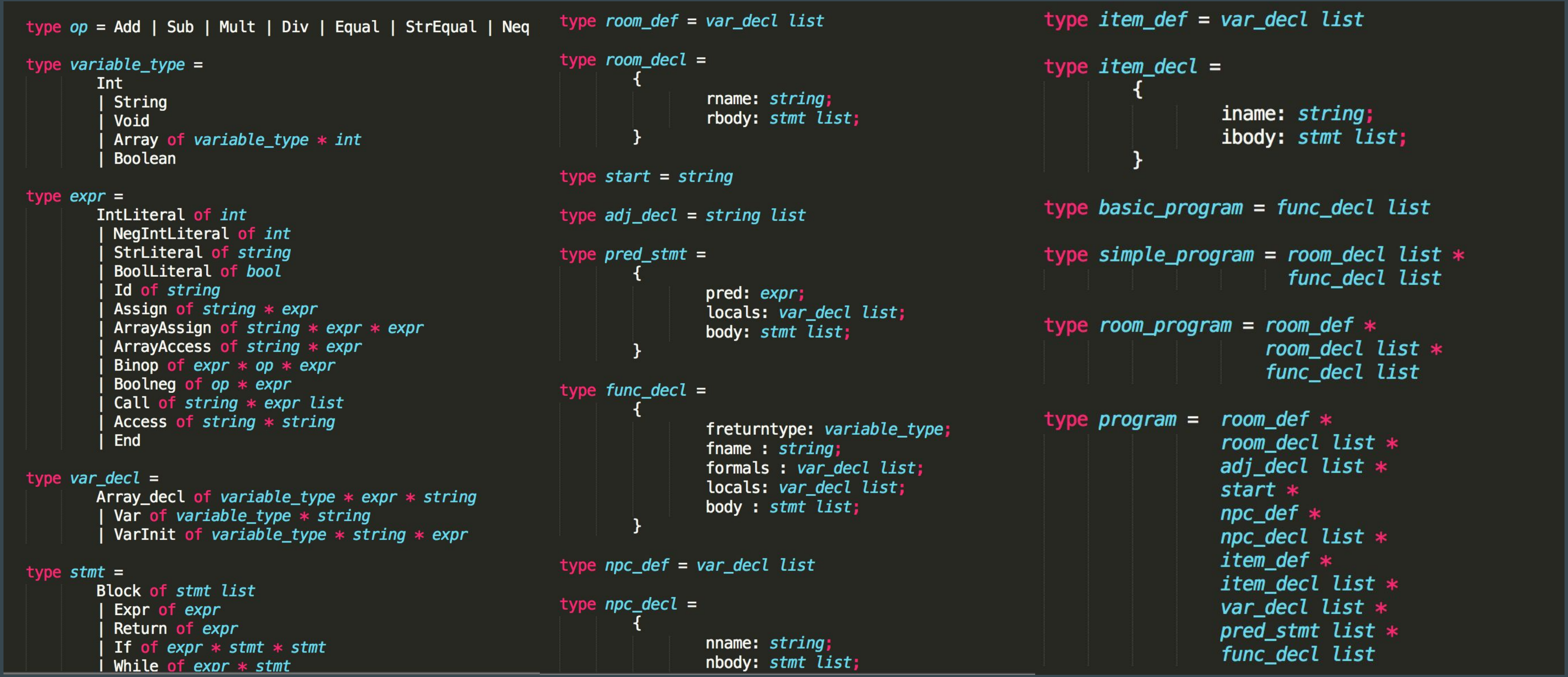


```
#import stdlib
#import typeConversionLib

room {}

room Closet { name = "Closet"; }
room Bedroom { name = "Bedroom"; }
room Wall { name = "Wall"; }
room Kitchen { name = "Kitchen"; }

Closet <-> Bedroom;
Closet <-> Wall;
Kitchen <-> Wall;
Kitchen <-> Bedroom;

start { Closet }

npc { string roomName; }

npc Cat { roomName = "Bedroom"; }

item {
    string roomName;
    boolean eaten;
}

Cheese {
    roomName = Kitchen;
    eaten = false;
}
```



```
boolean started = false;

NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}

true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    ->input
}

currentRoom.name ~~ Cat.roomName {
    print("You got eaten by the cat.");
    endgame;
}

currentRoom.name ~~ Cheese.roomName AND NOT Cheese.eaten {
    print("Nice!! You ate the cheese!");
    Cheese.eaten = true;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}
```

AST, Program Structure

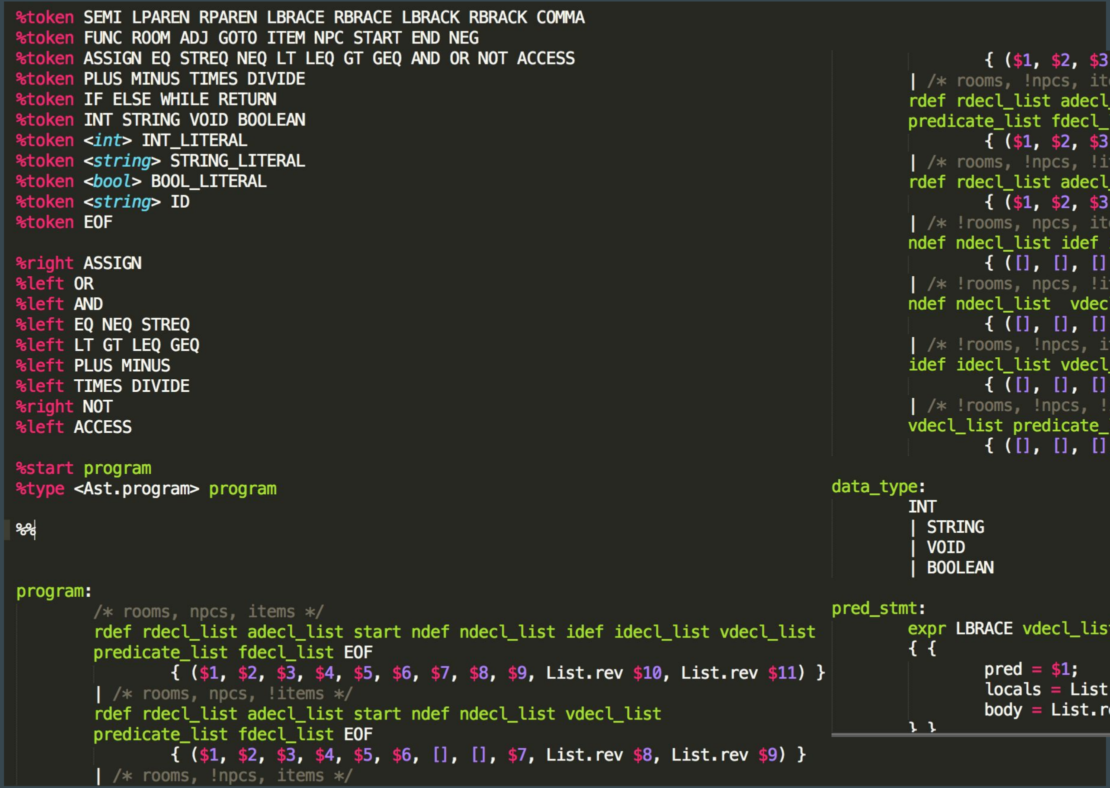


```
type op = Add | Sub | Mult | Div | Equal | StrEqual | Neq

type variable_type =
        Int
        | String
        | Void
        | Array of variable_type * int
        | Boolean

type expr =
        IntLiteral of int
        | NegIntLiteral of int
        | StrLiteral of string
        | BoolLiteral of bool
        | Id of string
        | Assign of string * expr
        | ArrayAssign of string * expr * expr
        | ArrayAccess of string * expr
        | Binop of expr * op * expr
        | Boolneg of op * expr
        | Call of string * expr list
        | Access of string * string
        | End

type var_decl =
        Array_decl of variable_type * expr * string
        | Var of variable_type * string
        | VarInit of variable_type * string * expr

type stmt =
        Block of stmt list
        | Expr of expr
        | Return of expr
        | If of expr * stmt * stmt
        | While of expr * stmt
```

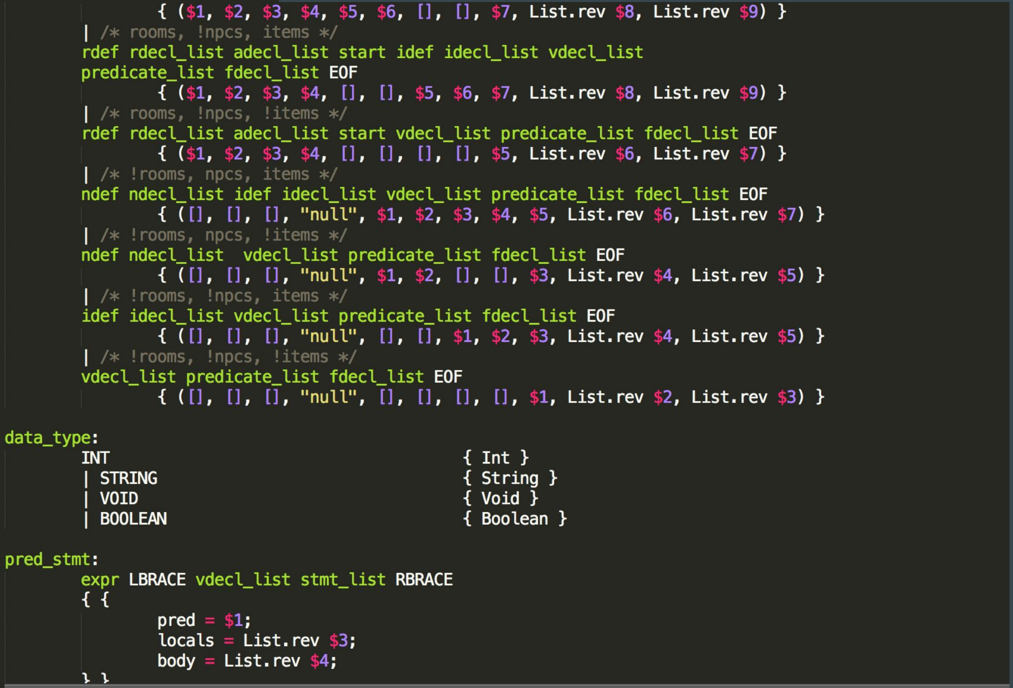
```
type room_def = var_decl list

type room_decl =
        {
                rname: string;
                rbody: stmt list;
        }

type start = string

type adj_decl = string list

type pred_stmt =
        {
                pred: expr;
                locals: var_decl list;
                body: stmt list;
        }

type func_decl =
        {
                freturntype: variable_type;
                fname : string;
                formals : var_decl list;
                locals: var_decl list;
                body : stmt list;
        }

type npc_def = var_decl list

type npc_decl =
        {
                nname: string;
                nbody: stmt list;
```

```
type item_def = var_decl list

type item_decl =
        {
                iname: string;
                ibody: stmt list;
        }

type basic_program = func_decl list

type simple_program = room_decl list *
                      func_decl list

type room_program = room_def *
                    room_decl list *
                    func_decl list

type program =  room_def *
                room_decl list *
                adj_decl list *
                start *
                npc_def *
                npc_decl list *
                item_def *
                item_decl list *
                var_decl list *
                pred_stmt list *
                func_decl list
```

Parser



```
%token SEMI LPAREN RPAREN LBRACE RBRACE LBRACK RBRACK COMMA
%token FUNC ROOM ADJ GOTO ITEM NPC START END NEG
%token ASSIGN EQ STREQ NEQ LT LEQ GT GEQ AND OR NOT ACCESS
%token PLUS MINUS TIMES DIVIDE
%token IF ELSE WHILE RETURN
%token INT STRING VOID BOOLEAN
%token <int> INT_LITERAL
%token <string> STRING_LITERAL
%token <bool> BOOL_LITERAL
%token <string> ID
%token EOF

%right ASSIGN
%left OR
%left AND
%left EQ NEQ STREQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%right NOT
%left ACCESS

%start program
%type <Ast.program> program

%%


program:
        /* rooms, npcs, items */
        rdef rdecl_list adecl_list start ndef ndecl_list idef idecl_list vdecl_list
        predicate_list fdecl_list EOF
                { ($1, $2, $3, $4, $5, $6, $7, $8, $9, List.rev $10, List.rev $11) }
        | /* rooms, npcs, !items */
        rdef rdecl_list adecl_list start ndef ndecl_list vdecl_list
        predicate_list fdecl_list EOF
                { ($1, $2, $3, $4, $5, $6, [], [], $7, List.rev $8, List.rev $9) }
        | /* rooms, !npcs, items */
```



```
                { ($1, $2, $3, $4, $5, $6, [], [], $7, List.rev $8, List.rev $9) }
        | /* rooms, !npcs, items */
        rdef rdecl_list adecl_list start idef idecl_list vdecl_list
        predicate_list fdecl_list EOF
                { ($1, $2, $3, $4, [], [], $5, $6, $7, List.rev $8, List.rev $9) }
        | /* rooms, !npcs, !items */
        rdef rdecl_list adecl_list start vdecl_list predicate_list fdecl_list EOF
                { ($1, $2, $3, $4, [], [], [], [], $5, List.rev $6, List.rev $7) }
        | /* !rooms, npcs, items */
        ndef ndecl_list idef idecl_list vdecl_list predicate_list fdecl_list EOF
                { ([], [], [], "null", $1, $2, $3, $4, $5, List.rev $6, List.rev $7) }
        | /* !rooms, npcs, !items */
        ndef ndecl_list  vdecl_list predicate_list fdecl_list EOF
                { ([], [], [], "null", $1, $2, [], [], $3, List.rev $4, List.rev $5) }
        | /* !rooms, !npcs, items */
        idef idecl_list vdecl_list predicate_list fdecl_list EOF
                { ([], [], [], "null", [], [], $1, $2, $3, List.rev $4, List.rev $5) }
        | /* !rooms, !npcs, !items */
        vdecl_list predicate_list fdecl_list EOF
                { ([], [], [], "null", [], [], [], [], $1, List.rev $2, List.rev $3) }

data_type:
        INT                                             { Int }
        | STRING                                        { String }
        | VOID                                          { Void }
        | BOOLEAN                                       { Boolean }

pred_stmt:
        expr LBRACE vdecl_list stmt_list RBRACE
        { {
                pred = $1;
                locals = List.rev $3;
                body = List.rev $4;
        } }
```

Semantic Checker

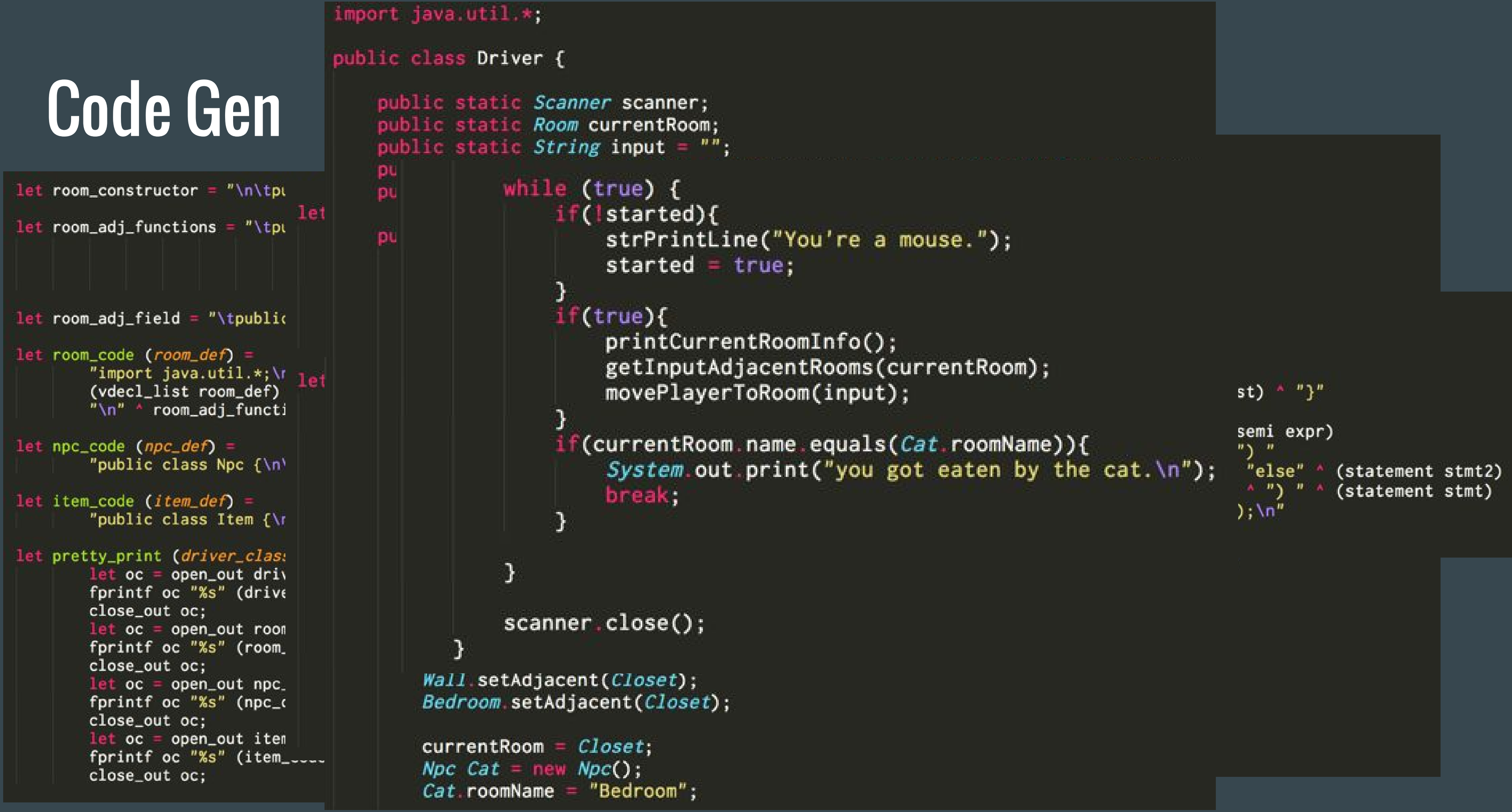
- Abandoned a typed SAST
- Semantically correct AST
- Symbol Table - scope
- Environment
    - Symbol Table
    - Return type
    - Current function
    - Global variables
    - Global functions
    - Room, Item, NPC defs
    - Room, Item, NPC decl’s
    - Predicate statements/handlers
- Built-in functions (challenge)

Java Builder

TBAG World

Room Definition
Room Declarations
Adjacency Declarations
Start Declaration
NPC Definition
NPC Declarations
Item Definition
Item Declarations
Variable Declarations
Predicates / Handlers
Function Declarations

Code Gen



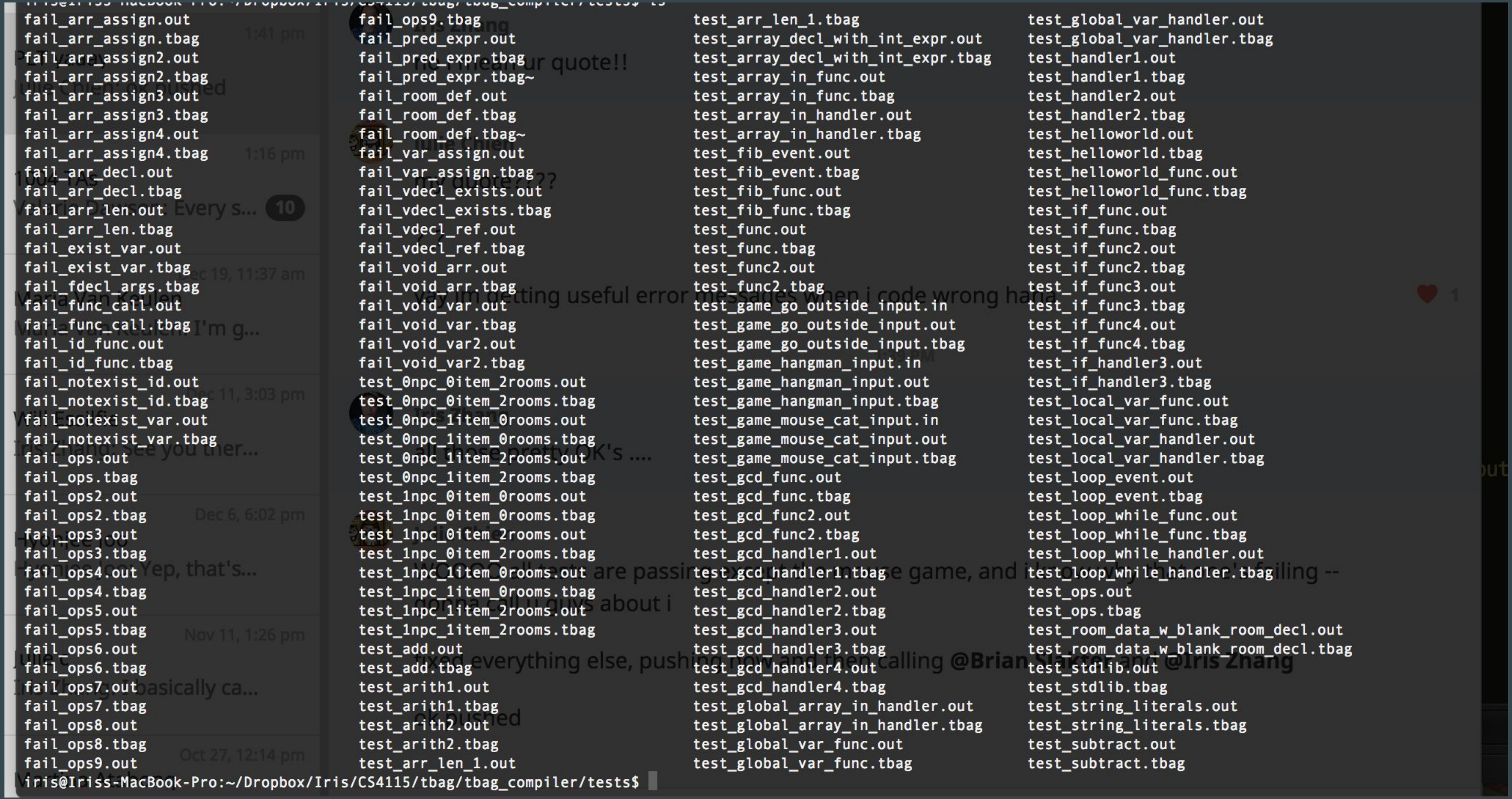
```
let room_constructor = "\n\tpu

let room_adj_functions = "\tpu


let room_adj_field = "\tpublic

let room_code (room_def) =
        "import java.util.*;\n
        (vdecl_list room_def)
        "\n" ^ room_adj_functi

let npc_code (npc_def) =
        "public class Npc {\n\

let item_code (item_def) =
        "public class Item {\n

let pretty_print (driver_clas
        let oc = open_out driv
        fprintf oc "%s" (drive
        close_out oc;
        let oc = open_out room
        fprintf oc "%s" (room_
        close_out oc;
        let oc = open_out npc_
        fprintf oc "%s" (npc_c
        close_out oc;
        let oc = open_out iten
        fprintf oc "%s" (item_
        close_out oc;
```

```
let

let
```

```
import java.util.*;

public class Driver {

    public static Scanner scanner;
    public static Room currentRoom;
    public static String input = "";
    pu
    pu              while (true) {
                        if(!started){
    pu                      strPrintLine("You're a mouse.");
                            started = true;
                        }
                        if(true){
                            printCurrentRoomInfo();
                            getInputAdjacentRooms(currentRoom);
                            movePlayerToRoom(input);
                        }
                        if(currentRoom.name.equals(Cat.roomName)){
                            System.out.print("you got eaten by the cat.\n");
                            break;
                        }

                    }

                    scanner.close();
                }
        Wall.setAdjacent(Closet);
        Bedroom.setAdjacent(Closet);

        currentRoom = Closet;
        Npc Cat = new Npc();
        Cat.roomName = "Bedroom";
```

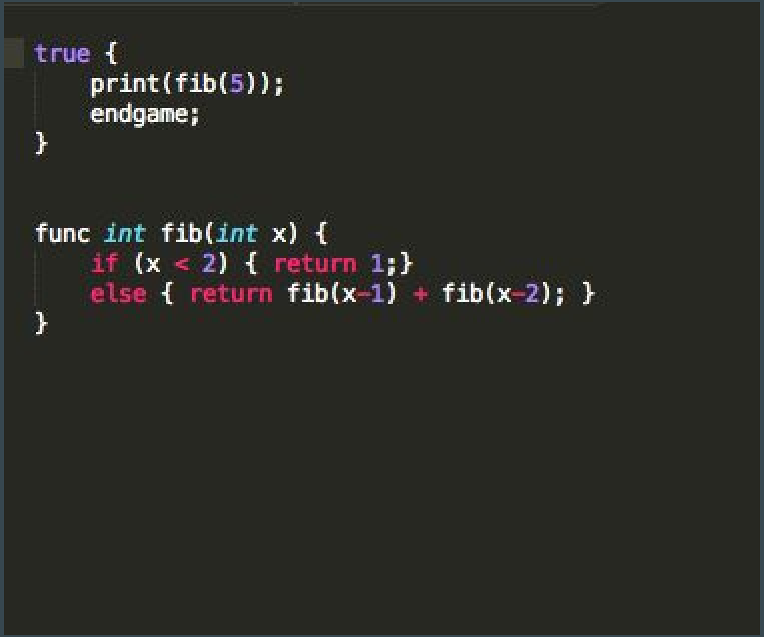
```
st) ^ "}"

semi expr)
") "
 "else" ^ (statement stmt2)
 ^ ") " ^ (statement stmt)
);\n"
```

Testing



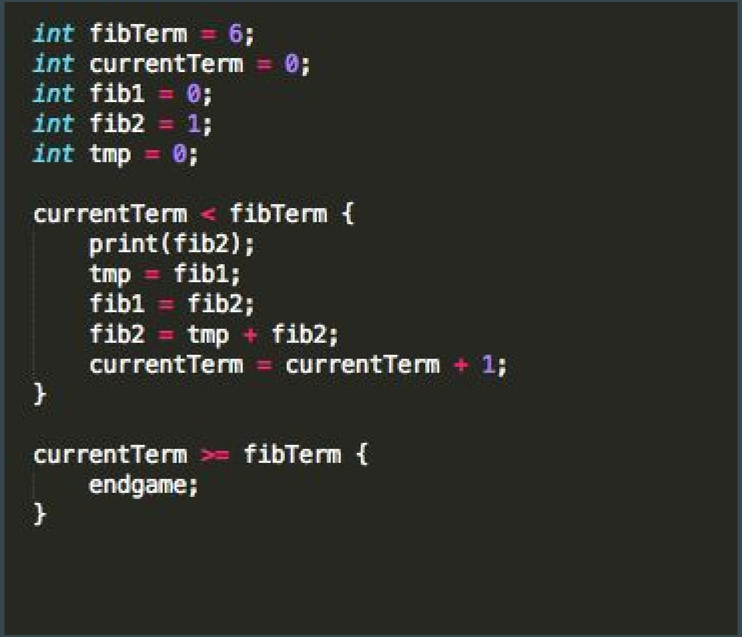
```
fail_arr_assign.out                  fail_ops9.tbag                       test_arr_len_1.tbag                  test_global_var_handler.out
fail_arr_assign.tbag                 fail_pred_expr.out                   test_array_decl_with_int_expr.out    test_global_var_handler.tbag
fail_arr_assign2.out                 fail_pred_expr.tbag                  test_array_decl_with_int_expr.tbag   test_handler1.out
fail_arr_assign2.tbag                fail_pred_expr.tbag~                 test_array_in_func.out               test_handler1.tbag
fail_arr_assign3.out                 fail_room_def.out                    test_array_in_func.tbag              test_handler2.out
fail_arr_assign3.tbag                fail_room_def.tbag                   test_array_in_handler.out            test_handler2.tbag
fail_arr_assign4.out                 fail_room_def.tbag~                  test_array_in_handler.tbag           test_helloworld.out
fail_arr_assign4.tbag                fail_var_assign.out                  test_fib_event.out                   test_helloworld.tbag
fail_arr_decl.out                    fail_var_assign.tbag                 test_fib_event.tbag                  test_helloworld_func.out
fail_arr_decl.tbag                   fail_vdecl_exists.out                test_fib_func.out                    test_helloworld_func.tbag
fail_arr_len.out                     fail_vdecl_exists.tbag               test_fib_func.tbag                   test_if_func.out
fail_arr_len.tbag                    fail_vdecl_ref.out                   test_func.out                        test_if_func.tbag
fail_exist_var.out                   fail_vdecl_ref.tbag                  test_func.tbag                       test_if_func2.out
fail_exist_var.tbag                  fail_void_arr.out                    test_func2.out                       test_if_func2.tbag
fail_fdecl_args.tbag                 fail_void_arr.tbag                   test_func2.tbag                      test_if_func3.out
fail_func_call.out                   fail_void_var.out                    test_game_go_outside_input.in        test_if_func3.tbag
fail_func_call.tbag                  fail_void_var.tbag                   test_game_go_outside_input.out       test_if_func4.out
fail_id_func.out                     fail_void_var2.out                   test_game_go_outside_input.tbag      test_if_func4.tbag
fail_id_func.tbag                    fail_void_var2.tbag                  test_game_hangman_input.in           test_if_handler3.out
fail_notexist_id.out                 test_0npc_0item_2rooms.out           test_game_hangman_input.out          test_if_handler3.tbag
fail_notexist_id.tbag                test_0npc_0item_2rooms.tbag          test_game_hangman_input.tbag         test_local_var_func.out
fail_notexist_var.out                test_0npc_1item_0rooms.out           test_game_mouse_cat_input.in         test_local_var_func.tbag
fail_notexist_var.tbag               test_0npc_1item_0rooms.tbag          test_game_mouse_cat_input.out        test_local_var_handler.out
fail_ops.out                         test_0npc_1item_2rooms.out           test_game_mouse_cat_input.tbag       test_local_var_handler.tbag
fail_ops.tbag                        test_0npc_1item_2rooms.tbag          test_gcd_func.out                    test_loop_event.out
fail_ops2.out                        test_1npc_0item_0rooms.out           test_gcd_func.tbag                   test_loop_event.tbag
fail_ops2.tbag                       test_1npc_0item_0rooms.tbag          test_gcd_func2.out                   test_loop_while_func.out
fail_ops3.out                        test_1npc_0item_2rooms.out           test_gcd_func2.tbag                  test_loop_while_func.tbag
fail_ops3.tbag                       test_1npc_0item_2rooms.tbag          test_gcd_handler1.out                test_loop_while_handler.out
fail_ops4.out                        test_1npc_1item_0rooms.out           test_gcd_handler1.tbag               test_loop_while_handler.tbag
fail_ops4.tbag                       test_1npc_1item_0rooms.tbag          test_gcd_handler2.out                test_ops.out
fail_ops5.out                        test_1npc_1item_2rooms.out           test_gcd_handler2.tbag               test_ops.tbag
fail_ops5.tbag                       test_1npc_1item_2rooms.tbag          test_gcd_handler3.out                test_room_data_w_blank_room_decl.out
fail_ops6.out                        test_add.out                         test_gcd_handler3.tbag               test_room_data_w_blank_room_decl.tbag
fail_ops6.tbag                       test_add.tbag                        test_gcd_handler4.out                test_stdlib.out
fail_ops7.out                        test_arith1.out                      test_gcd_handler4.tbag               test_stdlib.tbag
fail_ops7.tbag                       test_arith1.tbag                     test_global_array_in_handler.out     test_string_literals.out
fail_ops8.out                        test_arith2.out                      test_global_array_in_handler.tbag    test_string_literals.tbag
fail_ops8.tbag                       test_arith2.tbag                     test_global_var_func.out             test_subtract.out
fail_ops9.out                        test_arr_len_1.out                   test_global_var_func.tbag            test_subtract.tbag
iris@Iriss-MacBook-Pro:~/Dropbox/Iris/CS4115/tbag/tbag_compiler/tests$
```

fib_func.tbag



```
true {
    print(fib(5));
    endgame;
}


func int fib(int x) {
    if (x < 2) { return 1;}
    else { return fib(x-1) + fib(x-2); }
}
```

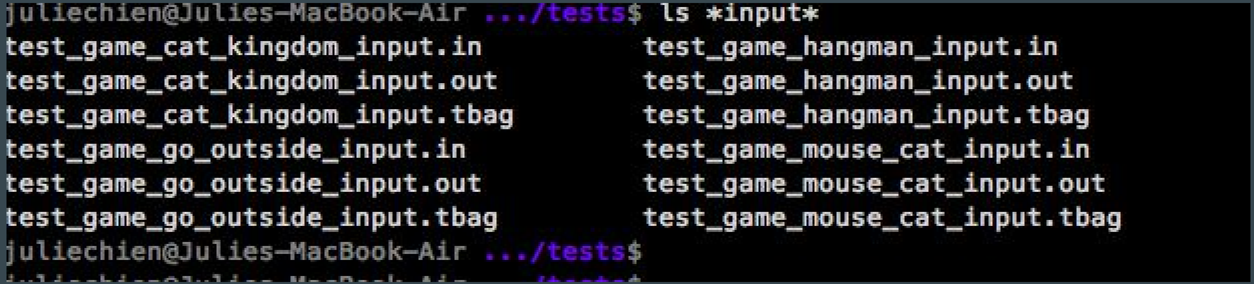
fib_event.tbag



```
int fibTerm = 6;
int currentTerm = 0;
int fib1 = 0;
int fib2 = 1;
int tmp = 0;

currentTerm < fibTerm {
    print(fib2);
    tmp = fib1;
    fib1 = fib2;
    fib2 = tmp + fib2;
    currentTerm = currentTerm + 1;
}

currentTerm >= fibTerm {
    endgame;
}
```

Gameplay tests

simulate user input



```
juliechien@Julies-MacBook-Air .../tests$ ls *input*
test_game_cat_kingdom_input.in          test_game_hangman_input.in
test_game_cat_kingdom_input.out         test_game_hangman_input.out
test_game_cat_kingdom_input.tbag        test_game_hangman_input.tbag
test_game_go_outside_input.in           test_game_mouse_cat_input.in
test_game_go_outside_input.out          test_game_mouse_cat_input.out
test_game_go_outside_input.tbag         test_game_mouse_cat_input.tbag
juliechien@Julies-MacBook-Air .../tests$
```

Planning, Processes, Development, Challenges

- Roles were fluid
- Version control
- Internal deadlines
  - More helpful for planning than for actual results!
- Changes in early December
- Code integration challenges
- Constant group feedback

What We Feared

1 of 1
EVERY GROUP PROJECT
DOES 99% OF THE WORK
HAS NO IDEA WHAT'S GOING ON THE WHOLE TIME
SAYS HE'S GOING TO HELP BUT HE'S NOT
DISAPPEAR AT THE VERY BEGINNING AND DOESN'T SHOW UP AGAIN TIL THE VERY END
IN SCHOOL YOU HAVE EVER DONE

What Really Happened

OUR GROUP PROJECT
DOES 99% OF THE WORK
HAS NO IDEA WHAT'S GOING ON THE WHOLE TIME
SAYS HE'S GOING TO HELP BUT HE'S NOT
DISAPPEAR AT THE VERY BEGINNING AND DOESN'T SHOW UP AGAIN TIL THE VERY END
EVERYONE IS BRADLEY COOPER