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?
#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();  
    getC_input();  
    adjRooms();  
    ->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

```plaintext
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
    | Boolean 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;
        formats : var_decl list;
        locals: var_decl list;
        body : stmt list;
    }

type room_program = room_def *


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 *

type room_program = room_def *


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

```

```
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
fib_func.tbag

```python
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

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

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

while (currentTerm >= fibTerm) {
    endgame;
}
```
Gameplay tests

simulate user input
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

EVERY GROUP PROJECT

DOES 99% OF THE WORK
SAYS HE’S GOING TO HELP BUT HE’S NOT
HAS NO IDEA WHAT’S GOING ON THE WHOLE TIME
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

SAYS HE'S GOING TO HELP BUT HE'S NOT

HAS NO IDEA WHAT'S GOING ON THE WHOLE TIME

DISAPPEAR AT THE VERY BEGINNING AND DOESN'T SHOW UP AGAIN TIL THE VERY END

EVERYONE IS BRADLEY COOPER