# tiler.

2D turn-based gaming language

Manager: Jason Lei (jl3825)
Language Guru: Monica Ting (mst2138)
System Architect: Jiayin Tang (jt2823)
System Architect: Evan Ziebart (erz2109)
Tester: Jacky Cheung (jc4316)

## Table of Contents

# 1. Introduction

## 1.1 Why tiler?

*Tiler* is a language intended to streamline the process of programming 2-dimensional, turn-based games. Its name derives from the fact that the most basic unit of such games is the tile, which is then arranged with other tiles into a grid. *Tiler* can feasibly be used to program for other purposes, but its structure and syntax make it especially intuitive to program turn-based tile games. *Tiler* achieves its goal of simplifying the implementation of games through the following principles:

### 1.1.1 Abstraction of game structure

Any program in *Tiler* is based around the concept of a block. Blocks represent the most universal and foundational aspects of games, such as the initialization of a gameboard, looping of turns, creation of object classes, and checking of end conditions. The built-in abstraction of these aspects allows for faster game design and implementation as well as improved code readability and reuse.

### 1.1.2 Ease of use and learning

*Tiler* is designed to be both familiar to experienced programmers and simple enough for less experienced programmers to easily pick up. This is a consequence of *Tiler*'s Java-like syntax, as well as the aforementioned abstraction of game concepts into blocks. *Tiler*'s language-specific keywords have also been carefully selected to map directly to tangible game components. Thus, the learning curve for a new *Tiler* programmer should be minimal.

### 1.1.3 Handling of I/O

A frequently complicated aspect of programming games is deciding how to collect user input and display game graphics. *Tiler* handles this for programmers by linking directly with SDL (Simple DirectMedia Layer), which is written in C and provides "low-level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D". Specifically, by providing a simple interface for SDL's mouse input and graphics handling, *Tiler* allows programmers to focus on the contents of their games rather than the details of working with a third-party library.

# 2. Language Tutorial

## 2.1 Environment Setup

Tiler requires SDL as well as the Ocaml LLVM library, which can be installed through Opam. To install SDL, download its source code.

On Mac OS, the environment can be easily set up with Homebrew: `brew install ocaml`, then `brew install opam` and `brew install sdl2`.

## 2.2 Getting Started

After downloading tiler files, run in following commands in the command line:

```
$ cd tiler
$ make all
```

You can load up a pre-existing .tile file or write your own called my_awesome_game.tile with the required sprites stored in the /sprites folder. To run:

```
$ make my_awesome_game.game
```

This will produce an executable with the extension .exe, which you can then run:

```
$ ./my_awesome_game.exe
```

## 2.3 Simple Example

The following tiler program is a game which places an object on the grid and then counts to 10. “//” denotes a single-line comment.

```
// declare a class object
class Object {}

// declare a global integer to count the number of turns
int count;

// init block -- code executed at the start
init {
    <Object> obj; // declare an instance of the class
    setSprite(obj, 'foo.bmp');          // set the image for the instance
    tile(3,3);                          // create a 3x3 grid
    grid[0,0] = obj;                    // place the instance on the grid
    count = 0;                          // assign global count
}

// turn -- the code that is run before each check of the end condition
turn {
    count = count + 1;                  // increment global count
    iprint(count);                      // prints the value of count
}
```

```
// end block -- returns if end condition met
end {
    return (count > 9);                  // true when count is 10 or more
}
```

The code first creates an instance of the class object and sets its sprite to 'foo.bmp'. The `setSprite()` function takes an object and a filepath as arguments, so the image will be set according to the data in '`foo.bmp`'. Next, it declares a grid of size 3x3 using the `tile()` function and places the object at location [0,0]. After init, count is 0. The turn block runs 10 times, each time incrementing count and then printing its value using the `iprint()` function. This stops when the end returns true. The following is the program's output.

```
1
2
3
4
5
6
7
8
9
game over
```

## 3. Language Reference Manual

### 3.1 Program Structure

In the tiler language, the .tile program can be organized into four general sections: window parameters, global declarations, user-defined code blocks, and special code blocks.

```
#size 500 250                    // set window width and height
#color 0 0 255                   // set window background color
#title "Hello World"             // set window title

int x;                           // global variables definitions
int y;

int add(...) {...}               // user-defined functions
class Piece {...}                // user-defined classes

init {                           // init block: required
   tile(3, 3);                   // initialize grid to 3 by 3
   background("hello.bmp");      // initialize board image
}
```

```
turn {...}                          // turn block: continuous looping of block
end {...}                           // end block: returns true on game end
```

### 3.1.1 Window Parameters

Parameters for the window display of the game can be customized by the programmer. Customizable parameters include the dimensions of the window, the background color of the window, and the title of the window. These lines must appear at the top of the program file, if you choose to utilize them. Each parameter is specified with a keyword followed by a space separated list. Size takes two integers, where the first integer specifies the width of the window in pixels and the second the height. Color takes three integers between 0 and 255 for the RGB value in decimal code, specifying the window's background color. Title takes a string enclosed in double quotes  for the window's title.

```
#size 500 250
#color 0 0 255
#title "Hello World"
```

### 3.1.2 Global Declarations

Global declarations are made outside of all code blocks and because its scope is global, it is accessible throughout the program in any block. See Section 3.7.1 for more on global declarations.

### 3.1.3 User-Defined Code Blocks

After window parameters and global declarations, the rest of the .tile program is divided into different labeled code blocks, each serving a specific function in defining the larger game. The tiler language allows for user-defined code blocks for functions and classes. These blocks are optional or the programmer can define multiple as long as each has a different name. The scope of each block is marked by a pair of curly braces.

#### Functions

Programmers may define their own reusable functions. Functions may take any primitive types as arguments and also may return any primitive type. Functions may also return `void`. See Section 3.8.1 for more on user-defined functions.

#### Classes

The class block defines the objects that will populate the board during game execution. Each instance of a class implicitly contains the attributes `x` and `y`, specifying its location on the grid. All other attributes can be specified within the class block with the `attr` keyword followed by the type and attribute name. See Section 3.6 for more on user-defined classes.

```
class Piece {
   attr: string player;
   attr: string role;
}
```

### 3.1.4 Special Code Blocks

Finally, the tiler language contains three special code blocks, each of which can appear at most once. Each code block is indicated by a specific keyword in the language: `init, turn, end`. The scope of each special block is also marked by a pair of curly braces.

#### init block

The init block defines how the game should be at the start of execution. This code serves the function of setting up the game board as well as initial placement and states for all the objects involved in the game.

```
init {
   tile(3, 3);                       // set grid size
   background("tictactoe.bmp");      // set optional background image
}
```

turn block

The turn block defines how the game behaves during each of the one or more turns. The game might prompt the player, listen for input, and/or manipulate the objects on the board. This allows the programmer to abstract away the need for a “game loop,” since these code segments will loop continuously during play.

```
turn {
   // developer code here
}
```

end block

The end block defines a series of conditions which indicate that the game has reached a final state. If the end block returns true, the game is over. If it returns false, the game loop continues on by executing the turn block again.

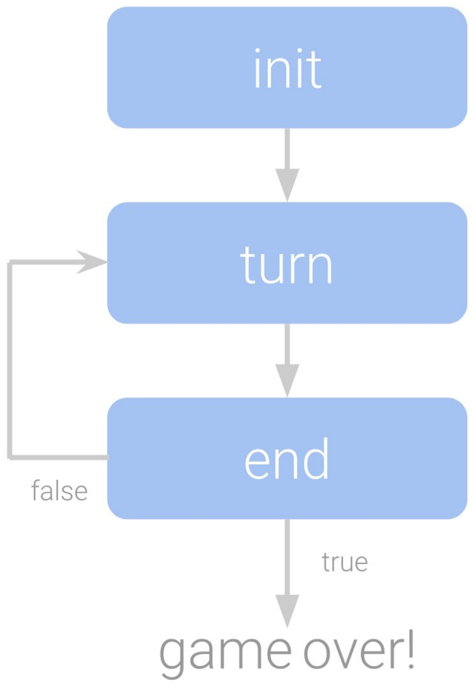
```
end {
   if (endCondition) {
     sprint("Player 1 wins");
     return true;
   }
   return false;
}
```

### 3.1.2 Game Loop

Every game must contain an init block. Every game should contain at most one init block, at most one turn block, and at most one end block. Games can contain any number of user-defined classes and user-defined functions. The order that the blocks appear in the program does not matter.

The tiler language is statically scoped. Variable that are globally declared are globally scoped. Otherwise, all variables declared in the other code blocks have a local scope marked within the curly braces.

The following control flow diagram is an example of a game. The init block is executed first to set up the game. Following the init block, there is an internal game loop that continuously iterates through turn blocks in the game, checking the end block condition after every turn. The game is over when the end block returns `true` indicating end of game.



## 3.2 Lexical Elements

### 3.2.1 Identifiers

Identifiers are strings for identifying variables and functions. They can contain letters, numbers, and underscores. Identifiers always begin with a letter and are case sensitive, generally beginning with lowercase letters, unless they are a class name.

### 3.2.2 Reserved Keywords

Keywords are reserved for specific usages in the language, begin with letters, and cannot be used as identifiers. Keywords are case sensitive.

tiler's keywords include:

```
#title, #size, #color
init, turn, end, class
int, float, string, bool, coord, void
if, else, do, while, for, return
true, false, NULL
grid, gridh, gridw
attr, new
```

`gridh` can be used to get the height of the grid and `gridw` can be used to get the width of the grid. All other keywords are used and explained throughout the Learning Reference Manual.

### 3.2.3 Literals

Each literal has a specific data type corresponding to one of the primitive types. No type casting is allowed. Assigning a literal to a variable of mismatching type will cause an error.

#### Integer literals

Integer literals are optionally-signed sequences of digits (0-9), which represent whole numbers and are in decimal format. A negative integer literal can be indicated using the '-' character.

#### Float literals

Float literals are optionally-signed sequences of digits (0-9), which contains a '.' with at least one digit before it. A negative float literal can be indicated using the '-' character. E.g.

```
0.9  9.0  9.999
```

#### Boolean literals

Boolean literals can take on the values of `true` or `false.`

#### String literals

String literals are sequences of zero or more characters enclosed in double quotes. If a quotation character is desired in the literal, either the non-desire quotation character must be used to contain the literal or the quotation character within the literal must be escaped. Escape characters include:

| | |
|---|---|
| `\n` | for new line |
| `\t` | for tab |
| `\r` | for carriage return |
| `\\` | for backslash |
| `\"` | for double quotes |

### 3.2.4 Delimiters

#### Parentheses

Parentheses are used to enclose arguments to function calls, expressions within control flow statements, and for defining explicit order of operations within arithmetic and logical expressions.

#### Commas

Commas are used to separate elements within coordinates and to separate arguments within function calls.

#### Square brackets

Square brackets are used for coordinate initialization, assignment, and access. They are also used for accessing the grid.

#### Curly braces

Curly braces are used to define the scope of blocks of code, such as tiler's special code blocks and other user-defined function and class blocks.

#### Semicolons

Semicolons are used to indicate end of statement.

#### Period

Periods are used to access attributes of objects.

#### Whitespace

Whitespace is used to separate tokens, but is otherwise ignored by the compiler and used for programmer readability only.

#### Comments

Comments can be made using the single-line notation `//` or multi-line comments can be enclosed within `/*` and `*/`

## 3.3 Expressions

Valid expressions include literals, binary operations, unary operations, assignments, function calls, variables, object instantiation and assignment, and grid assignment and access. The following are binary and unary operators and listed in order of increasing precedence.

### 3.3.1 Binary Operators

Assignment assigns a value to a variable. Assignment is right associative and is supported for all primitive and non-primitive types. E.g.

```
expression = expression
```

And returns true if the left-hand and right-hand expressions are both true and returns false otherwise. And is left associative and is supported for boolean types only. E.g.

```
expression && expression
```

Or returns true if either the left-hand or right-hand expressions is true and returns false otherwise. Or is left associative and is supported for boolean types only. E.g.

```
expression || expression
```

Equals returns true if the values of the left-hand and right-hand expressions are equal and returns false otherwise. Equals is left associative and is supported for integer, float, boolean, and string types. E.g.

```
expression == expression
```

Not equals returns true if the values of the left-hand and right-hand expressions are not equal and returns false otherwise. Not equals is left associative and is supported for integer, float, boolean, and string types. E.g.

```
expression != expression
```

Greater than returns true if the left-hand expression is greater than the right-hand expression and returns false otherwise. Greater than is left associative and is supported for integer and float types only. E.g.

```
expression > expression
```

Less than returns true if the left-hand expression is less than the right-hand expression and returns false otherwise. It is left associative and is supported for integer and float types only. E.g.

```
expression < expression
```

Greater than or equal returns true if the left-hand expression is greater than or equal to the right-hand expression and returns false otherwise. Greater than or equal is left associative and is supported for integer and float types only. E.g.

```
expression >= expression
```

Less than or equal returns true if the left-hand expression is less than or equal to the right-hand expression and returns false otherwise. Less than or equal is left associative and is supported for integer and float types only. E.g.

```
expression <= expression
```

Addition adds the two values on either side of the operator. Addition is left associative and is supported for integer and float types only. E.g.

```
expression + expression
```

Subtraction subtracts the right-hand operand from the left-hand operand. Subtraction is left associative and is supported for integer and float types only. E.g.

```
expression - expression
```

Multiplication multiplies values on either side of the operator. Multiplication is left associative and is supported for integer and float types only. E.g.

```
expression * expression
```

Division divides the left-hand operand by the right-hand operand. Division is is left associative and is supported for integer and float types only. E.g.

```
expression / expression
```

Modulus divides the left-hand operand by the right-hand operand and returns the remainder. Modulus is left associative and is supported for integer types only. E.g.

```
expression % expression
```

### 3.3.2 Unary Operators

| | |
|---|---|
| ! | Logical negation is supported for boolean types only. |
| - | Numeric negation is supported for integer and float types only. |

## 3.4 Statements

Statements in tiler can consist of a single expression or the following control flow statements.

### 3.4.1 Return statement

Returns a value from a from a function. Return may not be followed by a value, such as when a function returns void.

```
return true;
return;
```

### 3.4.2 Conditional statement

Conditionally runs code inside curly braces denoted by these statements.

```
if (condition) { statements }
if (condition) { statements } else { statements }
if (condition) { statements } else if (condition) { statements
}
```

### 3.4.3 `while` loop

Runs code inside curly braces if condition specified by the `while` loop is true.

```
while (condition) { statements }
```

### 3.4.4 `do while` loop

Alternative form of the while loop where the actions are written first followed by the condition.

```
do { statements } while (condition);
```

### 3.4.5 `for` loop

Parameters for iteration are an expression specifying start index, an expression specifying end condition, and an expression to increment/decrement the index.

```
for (i = 0; i < end; i=i+1) { ... }
```

## 3.5 Data Types

### 3.5.1 Primitive Types

int

Can take on the values of integers. An integer is a two's complement signed 32-bit integer number. Operations include: `+, -, *, /, %, ==, !=, <, >, <=, >=`

```
int x; x = 5;
```

#### float

Can take on values of floating point numbers with a mantissa and exponent. It is a two's complement signed 64-bit floating bit number. Operations include: `+, -, *, /, ==, !=, <, >, <=, >=`

```
float y;
y = 6.5;
```

#### bool

Can take on the value of true or false. Operations include: `==, !=, !, &&, ||`

```
bool b;
b = true;
```

#### string

Can take on the value of any sequence of one or more characters. String must always written between double quotes. String comparison is done through `==` and `!=`

```
string s;
s = "Hello World!"
```

#### coord

A `coord` is a built-in datatype representing a pair of integers, representing a coordinate on the grid. The datatype is composed of an x and y value which are enclosed in square brackets and separated by a comma. A coordinate can be declared and assigned to a variable as follows.

```
coord c;
c = [1, 2];
```

The values of a coordinate can be accessed by specifying x or y in square brackets.

`c[x]` and `c[y]`

### 3.5.2 Non-primitive Types

#### Object

An object is anything that can be placed on the grid (see next section on grid). An object is composed of a collection of attributes of primitive types. An object can have any number of attributes, which are defined in the class block which it is an instance of. An example of a class defining a Piece object is as follows:

```
class Piece {
   attr: string player;
   attr: string team;
```

```
    }
```

An object must be declared with its class name in angled brackets, then instantiated and assigned to a variable. E.g.

```
<Piece> piece;
piece = new Piece("Bob", "red");
```

An object's attributes may be accessed as follows:

```
piece.player;
piece.team;
```

An object also contains implicit attributes for its grid location, named x and y. These values will be `NULL` if the object has not been placed on the grid yet. They also may be accessed like attributes. E.g.

```
piece.x;
piece.y;
```

The call to isNull takes an object as argument and returns an integer 1 if the current object is `NULL` and 0 otherwise. E.g.

```
isNull(piece);
```

The call to setSprite takes an object and string as argument. The string contains the filepath to a Bitmap image for the object sprite or display image. E.g.

```
setSprite(piece, "piece.bmp");
```

## Grid

A 2D array of objects. The grid should be initialized in the init block (see Program Structure). On grid initialization, all objects on the grid are `NULL`. E.g.

```
init {
   tile(3, 3);
}
```

To access the object at a grid location:

```
<Piece> piece;
piece = grid[2, 3];
```

To assign an the object to a grid location:

```
grid[2, 3] = piece;
```

To remove the object at a grid location, assign `NULL` to the grid location:

```
grid[2, 3] = NULL;
```

## 3.6 Classes

### 3.6.1 Default Attributes

Each class by default has attributes for x and y, two integers which also store the coordinates of the object on the grid where x is the row and y is the column. Default attributes are only modifiable by the language's built-in functions, not through assignment. However, they can be accessed by the programmer and operated on according to their types.

### 3.6.2 Instantiation

Instantiation uses the `new` keyword. Programmer lists class name, then in parentheses assigns the values of each of the attributes defined in that class, formed as a comma-separated list and in the same order as how attributes are defined in the class. See Section 3.6.2 for more details.

```
     piece = new Piece("Bob", "red");
```

### 3.6.3 Object class type

Programmer can assign an object of any type to a variable of type Object

### 3.6.4 Assignment

Use the = operator for assignment. Syntax: [LHS] = [RHS]. Can only assign an object to an expression which has the same type, or object type.

## 3.7 Scope & Variable Declarations

### 3.7.1 Global declarations

Global declarations are made outside of all code blocks and because its scope is global, it is accessible throughout the program in any block. Declarations consists of two elements: type name and identifier. Globally scoped variables may only be of primitive types, so the type name is one of the reserved words `int, float, string, bool, coord`

### 3.7.2 Local declarations

Local declarations are made inside code blocks and must be made at the top of each code block, before other statements. The scope is the block in which they are declared. In addition to primitive types, objects may also be declared locally. Therefore, the type name is either the reserved word for a primitive type (int, float, string, bool, coord), or it is a class name with angled brackets. E.g.

```
     int count;
     <Piece> piece;
```

### 3.7.3 Attribute declarations

Attribute declarations define attributes of classes using the reserved word `attr`. Attribute declarations may only happen within the class blocks. Attributes may only be of primitive types, and the programmer specifies the type followed by name. E.g.

```
attr string player;
attr int lives;
```

### 3.7.4 Non-primitive type declarations

As mentioned in Section 3.7.2 on Local Declarations, objects can only be declared locally using the class name in angled brackets followed by an identifier. E.g.

```
<Piece> p1;
```

The grid must be declared only once in the program, and declaration must be in the init block. Uses keyword `tile` and takes two integers representing the number of objects in each row and column as arguments.

```
tile(4,4);
```

## 3.8 Functions

### 3.8.1 User-Defined Functions

User can define a function, functionName, that takes a list of arguments and returns a type of returnType. If functionName does not have a return type, returnType should be set to void.

```
int functionName(arguments) {
   // developer code here
}
```

To call the function, use the functionName followed by a comma separated list of parameters in parentheses.

```
functionName(arguments);
```

### 3.8.2 Built-In Functions

#### Board Specific Functions

- `tile(x,y)`
  Generates a board defined by a grid of x tiles wide by y tiles tall.

- `background(“filePath”)`
  Sets the background of the board to an image specified by the file path. Currently, tiler only supports Bitmap images, ending with the extension .bmp

Input-Output Specific Functions

- `capture();`
  Returns the coordinate of the tile that is selected by the mouse cursor

- `iprint(0);      iprint(true);`
  Take an integer or boolean as its single argument and prints to terminal.

- `fprint(1.0);`
  Take a float as its single argument and prints to terminal.

- `sprint(“Hello World”);`
  Take a string as its single argument and prints to terminal.

- `close();`
  Closes the game window.

Object Functions

- `setSprite(obj, imgFile);`
  The call to setSprite takes an object and string as argument. The string contains the file path to a Bitmap image for the object sprite or display image.

- `isNull(obj);`
  The call to isNull takes an object as argument and returns an integer 1 if the current object is `NULL` and 0 otherwise.

## 3.9 Context Free Grammar

```
program      → wflags decls EOF
wflags       → ε wflags wflag
wflag        → TITLE STRING_LITERAL | SIZE INT_LITERAL INT_LITERAL | COLOR
               INT_LITERAL INT_LITERAL INT_LITERAL
Decls        → ε | decls vdecl | decls class_decl | decls block
typ          → INT | FLOAT | BOOL | STRING | COORD | OBJ | VOID
vdecl_list   → ε | vdecl_list vdecl
vdecl        → typ ID SEMI | LT ID GT ID SEMI
```

```
class_decl   → CLASS ID LBRACE attr_decl_list RBRACE
attr_decl_list → ε | attr_decl_list attr_decl
attr_decl    → ATTR COLON typ ID SEMI
block        → init_block | turn_block | end_block | fdecl_block
             init_block → INIT LBRACE vdecl_list stmt_list RBRACE
             turn_block → TURN LBRACE vdecl_list stmt_list RBRACE
fdecl_block → typ ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list
             RBRACE
formals_opt → ε | formal_list
formal_list → typ ID | formal_list COMMA typ ID
end_block    → END LBRACE vdecl_list stmt_list RBRACE
stmt_list    → ε | stmt_list stmt
stmt         → expr SEMI | RETURN SEMI | RETURN expr SEMI | LBRACE stmt_list
             RBRACE | IF LPAREN expr RPAREN stmt %prec NOELSE | IF LPAREN expr
             RPAREN stmt ELSE stmt | FOR LPAREN expr_opt SEMI expr SEMI expr_opt
             RPAREN stmt | DO stmt WHILE LPAREN expr RPAREN SEMI | WHILE LPAREN
             expr RPAREN stmt
expr_opt     → ε | expr
expr         → INT_LITERAL | FLOAT_LITERAL | STRING_LITERAL | TRUE | FALSE | obj
             | expr PLUS expr | expr MINUS  expr | expr TIMES  expr | expr
             DIVIDE expr | expr MOD expr | expr EQ expr | expr NEQ expr | expr
             LT expr | expr LEQ expr | expr GT expr | expr GEQ expr | expr AND
             expr | expr OR expr | MINUS expr %prec NEG | NOT expr | LPAREN expr
             RPAREN | obj ASSIGN expr | NEW ID LPAREN actuals_opt RPAREN | ID
             LPAREN actuals_opt RPAREN | LSQUARE expr COMMA expr RSQUARE | ID
             LSQUARE ID RSQUARE | GRIDW | GRIDH | NULL
             obj → ID | ID PERIOD ID | GRID LSQUARE expr COMMA expr RSQUARE
actuals_opt → ε | actuals_list
actuals_list → expr | actuals_list COMMA expr
```

# 4. Project Plan

## 4.1 Process

### 4.1.1 Planning and development process

Throughout the first two-thirds of the semester, we met in person once or twice a week to iterate over our design, implement more functionality to our language, and delegate assignments for our next meetings. We also spent half an hour each week meeting with our (extremely helpful!) TA Freddy to get input on our designs and deliverables.

At the beginning of the semester, our first few meetings were about what kind of language we wanted to build, what kind of target audience our language would have, and how we might want to implement such a language. We also assigned roles early on, so that people would know what they were accountable for and could start taking initiative on certain parts of the language (although certainly people were still fluid within their roles). We also spent a lot of time writing sample games in our proposed language, to see what functionality would be most important to a potential *tiler* programmer. As the semester wore on, we began to do more work outside of meetings and spend in-person meeting time on things like debugging.

### 4.2.2 Project management

We used Git as our version control system, maintaining a private repo on GitHub with a master branch that required pull request approval from at least one other member of the team before merging. During meetings, careful notes were taken and stored in a group Google Drive so that anyone who missed meetings could get caught up quickly. Furthermore, we maintained a task board using the ZenHub extension in GitHub. This allowed us to efficiently keep track of a backlog of tasks/bugs, who was working on what, whether there were any code dependencies, and general progress on our language.

## 4.2 Roles and Responsibilities

Throughout the course of the project, our team members were flexible about what they were working on. While everyone was assigned a role, we all contributed to the codebase in various ways. Nonetheless, the official role assignments on our team were as follows:

Tester - Jacky
Manager - Jason
System Architects - JY and Evan
Language Guru - Monica

| **Team Member** | **Responsibilities** |
|---|---|
| Jacky Cheung | Testing suite setup and tests, print & comment (scanner/codegen), semant |
| Jason Lei | Initial ast/scanner setup, tests, code blocks (end to end), window flags, Calculator demo |
| Jiayin Tang | Hello World codegen, variable declaration (codegen), classes (end to end), semant, Tic Tac Toe sprites |
| Monica Ting | Language design, variables/primitives/coordinates (end to end), control flow statements (end to end), Hello World demo, Tic Tac Toe demo |
| Evan Ziebart | runtime library, classes, window flags, semant |

## 4.3 Timeline

| **Week** | **Milestones** | **Deliverables** |
|---|---|---|
| Sep 17 | Assign roles<br>Brainstorm language ideas | |
| Sep 24 | Write sample games in language | Project Proposal due on 9/26 |
| Oct 1 | Git repo setup | |
| Oct 8 | Figure out language keywords and operations | |
| Oct 15 | Finalize CFG | LRM due on 10/16 |
| Oct 22 | Scanner and parser can handle .tile program<br>Set up runtime library | |
| Oct 29 | Run tiler-caller program in runtime library | |
| Nov 5 | Connect runtime library functions in codegen | Hello World due on 11/8 |
| Nov 12 | Build test framework | |
| Nov 19 | Add blocks to parser | |
| Dec 3 | Add additional functionality to blocks | |

| | | |
|---|---|---|
| | Add interactivity to tiler programs | |
| Dec 10 | Add functions and classes<br>Add semantic checking<br>Prepare for demo (slides, sample programs)<br>Write final report | Final demo on 12/19<br>Report due on 12/20 |

## 4.4 Software Development Environment

We developed locally on our computers (Mac OS 11 and Windows 10), using a combination of vim and Sublime Text. We developed using OCaml 4.0.6.0, Bash, and C (for the runtime library). As mentioned earlier, we used Git and GitHub for our version control and ZenHub for our task management. We used Google Drive for meeting notes.

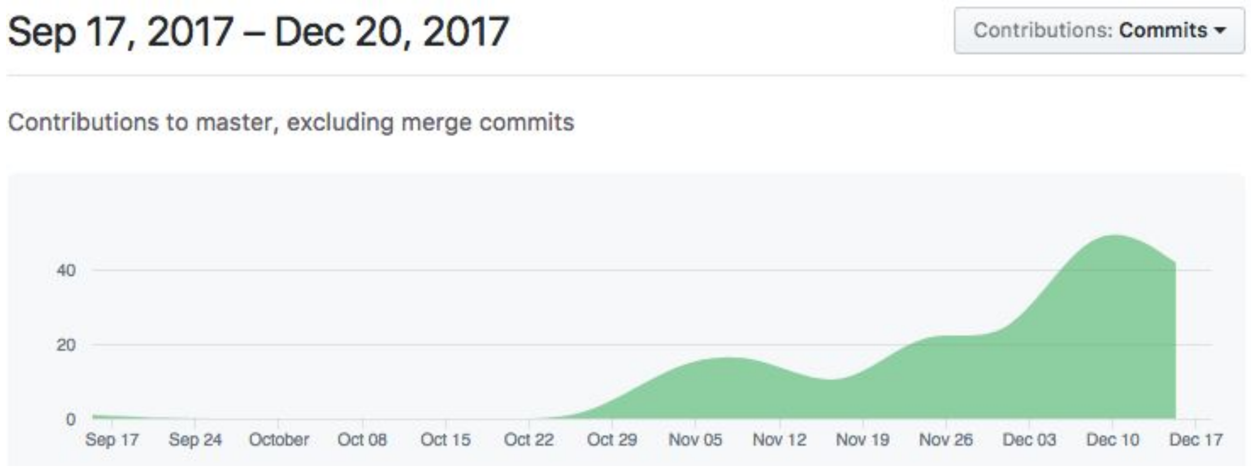
## 4.5 Style Guide

The following style guidelines were used by the group:

- Give variables readable names
- Adhere to Micro-C style for scanner, parser, ast, codegen
- Comment large code blocks and/or code that does not perform obvious functionality
- Indentation size of 2
- Break long lines of code into multiple lines

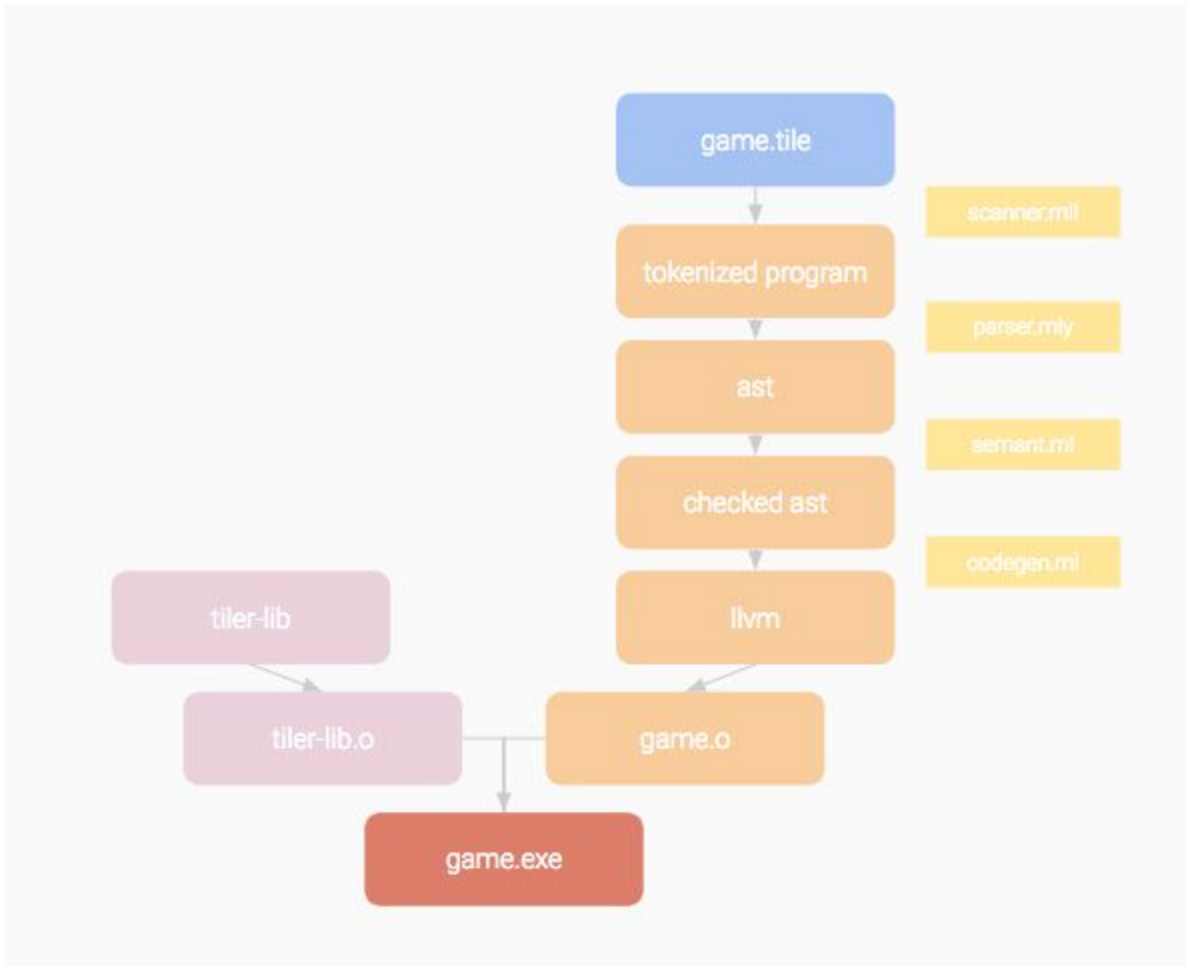
## 4.6 Project Log

Github Timeline:



# 5. Architectural Design

## 5.1 Block Diagram



## 5.2 Compiler Components

The compiler for the tiler language was written in Ocaml and compiled using the ocamlopt compiler. The .tile source code is processed via a series of passes, ending with the generation of code formatted in LLVM IR. The LLVM is then linked with the runtime library and converted into the .game executable file. Below is the ordering of compilation, and a description of the code which corresponds to each step.

1.) Scanning (scanner.mll): This step inputs the .tile source code and converts it into an ordered list of tokens. The .mll file specifies a regular expression to defines each token in the language. Ocamllex is used to generate the ocaml source code.

2.) Parsing (parser.mly): This step inputs the lexical tokens found in the previous step and converts them into an abstract syntax tree for the tiler program. The type names for the abstract syntax tree are defined in ast.ml. The .mly parser file specifies the context-free grammar for the tiler language. Ocamlyacc is used to generate the ocaml source code which uses this grammar to produce an abstract syntax tree from the tokens.
3.) Semantic-Checking (semant.ml): This code inputs the abstract syntax tree and parses it to check for semantic errors. If semantic errors are found (such as duplicate variable or incorrect function arguments), then the program prints an appropriate error message.
4.) Code Generation (codegen.ml): This code inputs the semantically-checked abstract syntax tree and outputs an LLVM IR module containing the corresponding code that the tiler program generates.

The top-level program is contained within tiler.ml. This code inputs the program, calls the scanner and parser to generate an AST, asserts a semantically correct program, and generates the LLVM IR code.

The following options are available:
-a (ast): outputs the AST for the tiler program
-l (llvm): outputs the LLVM IR code for the program
-c (compile): outputs the LLVM IR code to a .ll file to be compiled into an executable

## 5.3 Tiler Runtime Library

The tiler runtime library provides a suite of utilities for the tile game executable code. The generated code links to this library and calls these utilities at runtime. The main use of the tiler runtime library is to manage a game window, which is done with the Simple Direct Media-Layer (SDL) library in C.

### 5.3.1 SDL Summary

The Simple Direct Media-Layer (SDL) Library is a robust suite of tools for working with graphics and windows in the native operating system. The tiler runtime makes use of some of the basic functionalities of this library. These features include interfacing with the window, loading images, rendering graphics on the window, working with threads, and capturing window events and user mouse input. SDL has ports to many different languages, but the tiler runtime library uses the original version written in C.

### 5.3.2 Tiler-lib Functionality

The tiler runtime library supports several important features in the tiler language.

- create the game window
- listen for SDL events
- manage the tile grid
- run the main game loop
- manage memory for objects
- load and render appropriate images to the window

Game Window: The code to create the game window is in tiler-main.c. When runGame() is called, it calls the SDL functions which create the new window and set it to be viewable.

Event Listening: The main event loop is in tiler-main.c. It ensures proper exiting of the window and listens for mouse clicks on the grid. When a mouse click occurs, the program records the grid coordinate which was clicked. The getMouseCoords() function in tiler-mouse.c can be used to access this data.

Grid: The tiler-grid.c file contains the functions for creating and accessing the tiler grid. Objects can be placed on the grid, or removed from the grid, and object at a given grid coordinate can be accessed.

Game Loop: The tiler-main.c code is responsible for running the game loop. The runtime library obtains function pointers to the init, turn, and end blocks through calls to the set functions defined in tiler-functs.c. The gameLoop() function in tiler-main.c calls these in the appropriate order, and returns once the game is over.

Object Memory: The tiler runtime library keeps references to all declared objects in a tiler program. It also keeps track of default attributes for objects, such as xy-coordinate and sprite, and provides accessor functions for these values. Each time an object is declared, the tiler program calls the createObject() function defined in the tiler-object.c file in order to initialize these default attributes and set up a reference to the new object. These references are kept in a list. Any object not on the grid has a reference in this list, and at the end of each turn, the list in cleaned, freeing the memory for each object in it. Thus, the library automatically handles the garbage collection of tiler objects.

Drawing to the Window: The draw() function in tiler-main.c uses an SDL renderer to draw the background and grid objects to the screen. The sprite for each object is automatically drawn at the location on the screen which corresponds to the grid location that it occupies. This rendering process also utilizes back buffering to prevent flickering in tiler games.

# 6. Test Plan and Scripts

## 6.1 Testing Plan

### 6.1.1 Overview

The idea behind testing is to ensure that already designed functionalities continue to work as expected. In addition, test files were made ahead of feature implementation to provide direction to feature implementation. This allows for easy checking of progress on feature implementation and any potential negative effect on already existing features. Fail cases were designed to demonstrate what should not be allowed in the language and needed to be handled by semantic checking. The idea is that that non-compilable/non-executable programs written in tiler fail as intended (in which case it will be regarded as a success in correctly recognizing situations that would lead to failure). Thus, testing was done in parallel with the development of the tiler language to give direction for troubleshooting assessing the progress of feature implementation.

### 6.1.2 Testing Complications

Tiler is intrinsically a language developed for 2D turn-based games. Thus, a significant portion of the language involves graphics and user interaction. An initial problem in automation was that game windows had to manually be closed to proceed. This would make testing semi-manual and rather tedious as someone would have to manually close each window associated with a test case to progress the testing script. Obviously, this would not scale well and a close( ) function was designed to close the game window when it was called as an easy solution. The close( ) function was designed specifically for testing automation and would really not be used elsewhere (except for closing a window automatically after a game ends). As such, all test cases should eventually call the close() function for the sake of automation.

However, such testing could only be done to mainly check that the language's logic/arithmetic functionalities were implemented correctly. The actual game behavior still requires manual testing since the graphical component is the main focus of tiler and difficult to automate. This is mainly because games requires user input/interaction and decision making. The underlying behaviour can be tested for through the use of print functions, but the visuals also require manual testing and examination to ensure that the language is behaving as expected. Thus, although parts of the language can be tested automatically with a script, manual testing is still required.

### 6.1.3 Unit Testing & Integration Testing

Unit testing was performed to ensure that individual functionalities were behaving properly. Initially, these may be done locally in order to focus on the functionality of one aspect of the language. Eventually, this gets incorporated a collection of test and fail cases for integration testing. Integration testing ensure that the addition of new functionalities does not cause older ones to fail. Thus, test cases were designed with and catered to individual features in mind. This was particularly important for control flow elements of the language. Overall, everyone contributed to testing and the use of the test suite helped to facilitate the process by automating regression and integration testing throughout the design process.

## 6.2 Test Suite

### 6.2.1 Design and Description of Testing Script

The tiler language's testing suite took advantage of the regression testing suite established in the provided MicroC. In terms of execution is effectively the same, with minor changes in file location and naming schemes for generated files. Changes were made mostly for housekeeping. To run the test suite, the Makefile associated with tiler must be present as the script takes advantage of "make <file>.test", a phony command designed to properly compile .tile test cases. To run the test suite, ensure that tiler.native is available and run ./testall.sh on the command line in terminal. It can be executed with a -k flag to keep all intermediate files or a -h flag for a help/usage message. Overall, the testing suite allowed the team to do regression testing on older features to ensure these features still worked as intended on test cases; and that the semantic checking is working as intended on fail cases by failing as intended. The locations of all files were intentionally designed for ease of access for troubleshooting when test cases did fail.

Test cases are located in a folder named tests on the same directory as the tiler source code; and named with test-<name>.tile and fail-<name> - where test files are meant to check that features function as intended and fail files are meant to check that compilation fail in an expected why and for testing the language's semantic checking. Like the MicroC testing suite, the output of compiling and executing these test/fail cases are saved in a .gen file and compared with .out/.err files using diff.  If test cases fail to compile, intermediate files (i.e. the associated .gen, .ll, and .diff files) are saved in tests directory for easy access. On success, these intermediate files deleted by default. Logs of the entire testing process is stored in a testall.log file. It is important that the naming scheme is consistent in order for the script to run correct. Specifically, test-<name>.tile and fail-<name>.tile must have a corresponding test-<name>.out and fail-<name>.err respectively. In summary, the testing suite is essentially the MicroC testing suite that was modified and developed by the team tester Jacky.

### 6.2.2 Testing Script Code Listing

```
# Regression testing script for Tiler
# Modified from the MicroC testall.sh

# Time limit for all operations:
```

```
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0

keep=0

# How to use testall.sh:
Usage() {
       echo "Usage: testall.sh [options] [.tile files]"
       echo "-k    Keep intermediate files"
       echo "-h       Print this help"
       exit 1
}

# Report any errors
SignalError() {
    if [ $error -eq 0 ] ; then
       echo "FAILED"
       error=1
    fi
    echo "  $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile.  Differences, if any, are written to difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
       SignalError "$1 differs"
       echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and reports any errors
Run() {
    echo $* 1>&2
    eval $* || {
       SignalError "$1 failed on $*"
       return 1
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
       SignalError "failed: $* did not report an error"
       return 1
    }
    return 0
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
```

```
                             s/.mc//'`
    relPath=`echo $1 | sed 's/.tile//'`
    reffile=`echo $1 | sed 's/.tile$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2
    generatedfiles=""

    generatedfiles="$generatedfiles ${relPath}.ll ${relPath}.o ${relPath}.exe ${relPath}.gen
${relPath}.diff" &&

    # Intermediate files are placed in tests directory

    Run "make ${relPath}.test > /dev/null" &&
    Run "./${relPath}.exe" > "${relPath}.gen" &&
    Compare ${relPath}.gen ${reffile}.out ${relPath}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
       if [ $keep -eq 0 ] ; then
           rm -f $generatedfiles
       fi
       echo "OK"
       echo "###### SUCCESS" 1>&2
    else
       echo "###### FAILED" 1>&2
       globalerror=$error
    fi
}

CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                             s/.mc//'`
    relPath=`echo $1 | sed 's/.tile//'`
    reffile=`echo $1 | sed 's/.tile$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${relPath}.ll ${relPath}.gen ${relPath}.diff" &&
    RunFail "make ${relPath}.test 2> ${relPath}.gen 1> /dev/null" &&
    Compare ${relPath}.gen ${reffile}.err ${relPath}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
       if [ $keep -eq 0 ] ; then
           rm -f $generatedfiles
       fi
       echo "OK"
       echo "###### SUCCESS" 1>&2
    else
       echo "###### FAILED" 1>&2
       globalerror=$error
    fi
}

while getopts kdpsh c; do
    case $c in
       k) # Keep intermediate files
           keep=1
           ;;
       h) # Help
           Usage
           ;;
    esac
done

shift `expr $OPTIND - 1`

if [ $# -ge 1 ]
then
    files=$@
else
    files="tests/test-*.tile tests/fail-*.tile"
fi

for file in $files
do
    case $file in
       *test-*)
           Check $file 2>> $globallog
           ;;
        *fail-*)
           CheckFail $file 2>> $globallog
           ;;
       *)
           echo "unknown file type $file"
           globalerror=1
           ;;
    esac
done

exit $globalerror
```

## 7. Language Evolution

Tiler was originally designed based off of popular 2D tiled board games, such as tic-tac-toe, chess, checkers, Monopoly, etc. Typically for these types of board games, the game set comes with a set number of pieces and the physical types of pieces are always the same. Therefore, we thought it made sense to to enumerate the possible attributes a class could have. Each permutation is a different "state" representing a different piece in the game. For example, in chess, the class would be defined as follows:

```
class Piece {
      attr: string player = ["black", "white"];
      attr: string role = ["pawn", "rook", "knight", "bishop", "queen", "king"];
}
```

However, as we began building up the tiler language, we realized enumerating attribute values did not transfer well over for floating point numbers. It would not be possible for a programmer to specify a range of float, and specifying a range of integers would be a hassle to list out. Additionally, it is not very useful to enumerate Boolean attributes since there are only two states anyways. Therefore, we decided to remove the capability for programmers to enumerate attributes in the class definition. This also simplified the our codegen and runtime library, since we did not need to save information on all possible values for each attribute anymore, and we did not need to do the semantic checking to make sure the user assigned a valid value for the attribute. This change actually made our language more flexible, because now programmers can assign any matching type value to each attribute, similar to classes in Java.

Another design change in our language was our handling of objects on the grid. In our original proposal, we decided to give users the flexibility to put objects in a new grid location in multiple ways. For example, the programmer could specify move on the object:

```
      piece.move(0, 3);
```

Or the programmer could assign an object to a grid location:

```
      grid[0, 3] = piece;
```

Originally, we allowed for this flexibility in the language so that programmers could think from either the perspective of an object or from the perspective of the grid. However, the redundancy actually created more problems as we considered what would happen when the programmer tries to move a piece to a location that already has another object? If it replaces the object on the grid, we also needed to consider what happens to objects that currently do not have a position on the board and how those objects should be kept track of and freed. In the end, we decided to only use the second notation centered around the grid and adjusted the rest of our language similarly. For example, to remove an object from the grid, use:

```
    grid[0, 3] = NULL;
```

rather than:

```
    piece = grid[0, 3];
    piece.remove();
```

# 8. Lessons Learned

## 8.1 Member Reflections

### 8.1.1 Jason

This project was honestly eye-opening in ways that I did not anticipate. Seriously, seriously start early. And by start early, I mean start implementing things early. Iterate quickly, break things quickly. Don't spend all of your time thinking and planning. It is definitely possible to meet once or twice a week for the first half of the semester and still have no clue about what you're actually trying to do. Instead spend that time taking a stab at implementing a scanner and parser. Get familiar with a compiler's structure and ocaml's syntax, so that once you do figure out what you're trying to do, you'll at least have an idea of how you will implement it. Lastly, learn how to delegate tasks properly and accept that everyone has a different style of working. Not everyone likes to meet in person, not everyone likes to work well ahead of the deadline. But as long as you and your group communicate well *and* frequently, things will work out. Don't lose hope!

### 8.1.2 Monica

Generic advice...start early! For me personally, all the new code from microc and past projects was really overwhelming to get all at once, and that's why I think I enjoyed the ideation and design process of the language more fun than actually getting myself to sift through the microc-compiler. The first few weeks was a lot of drawing out ideas on paper and as Language Guru, dishing out different syntactic ways of making our language attractive and intuitive for a gaming language. Instead, time would probably have been better spent on implementing the syntax of our language early on by completing the scanner, parser, and AST all at the beginning.

I think we also could have benefitted from more pair programming; since OCaml is definitely a hard language to pick up within a few weeks, having another pairs of eyes would've definitely sped up that debugging process.

### 8.1.3 Evan

For me, the most important lesson is that figuring out the "how" of something is more important that the "what". At the start of this project, my teammates and I took great care in designing our language to be just how we wanted it, but we were at that point still unaware of how our compiler should work, and what vision was feasible given our time constraints. In the end, our language evolved quite a bit because we found that some things which we thought would be easy turned out to be challenging, and some things we thought would be hard turned out to be simple. From that wasted time at the start, I learned that it's better to have a general idea and a prototype than a super-detailed vision with nothing to show for it.

### 8.1.4 Jacky

As all my teammates already said, start this project early. My advice is to play around with OCaml at the beginning and getting a basic understanding of and feel for the language. You will undoubtedly be looking at a lot of OCaml code, so get comfortable looking at it and understanding it. More likely than not, you will be pretty bad at anything OCaml-related, but it gets better with practice, sort of... That being said, this was the first time I worked on a long-term group project at Columbia and I appreciated the constant communication and celebration of every milestone. In addition, working in pairs tended to work well as you get to have an extra pair of eyes and bounce ideas for troubleshooting and move-forward with another person. Also, have a dedicated workplace that is kept relatively clean and make sure your teammates are fed and hydrated. The project is hard enough as is, so no need to make it any harder. My last word of advice is that OCaml is a ridiculously concise language, so make a habit of looking at any changes to your team's code to stay of top of any changes.

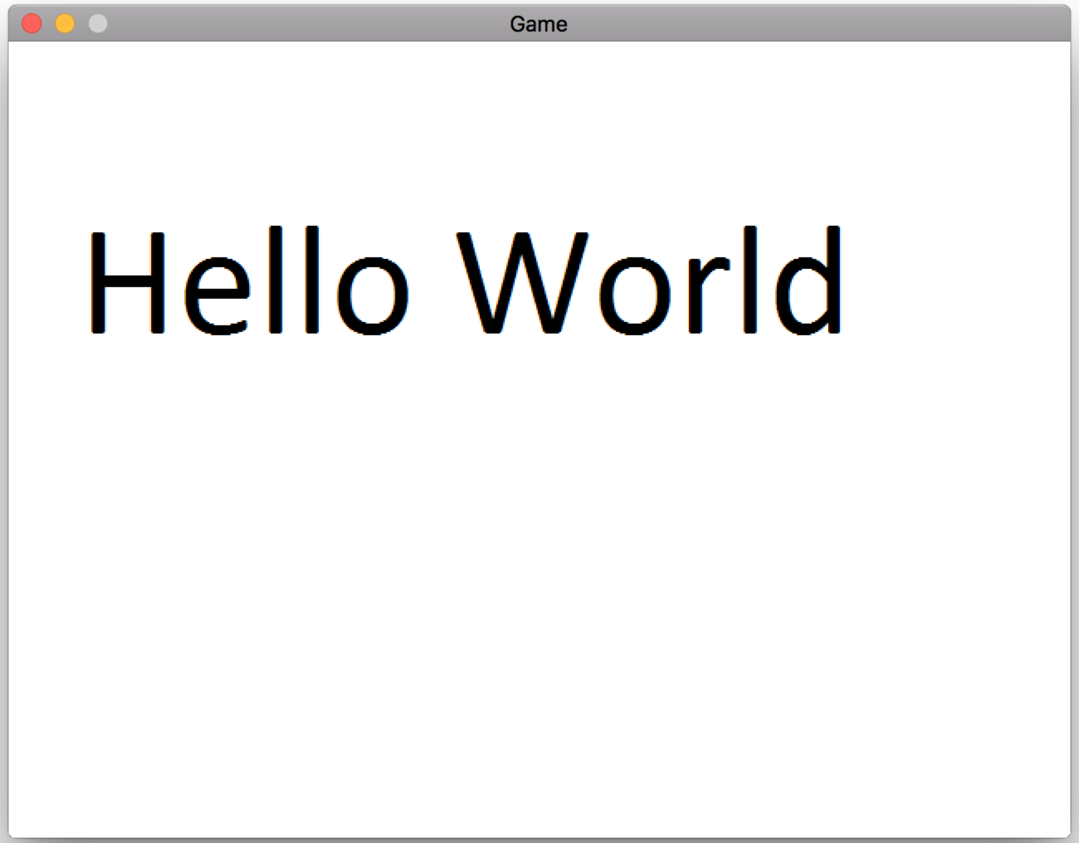
### 8.1.5 JY

Looking back on this semester, I am surprised to realize that we did start rather early...but we spent a lot of time in the ideas phase of our project. We met several times early in the semester and had long discussions about what we wanted our language to do. Everything looked nice on paper once we decided on the details, but the hardest part was implementing it. Our Hello World milestone was one of the hardest things for us as we all struggled with Ocaml and the Codegen. I remember asking Freddy how to do something in Codegen and still feeling lost about how to *actually* write it in Ocaml after hearing his answer. However, once I got over the learning curve I felt much more comfortable. My biggest takeaway from this project is to just jump into the AST/Scanner/Parser as soon as possible to familiarize yourself with the language, and to not be afraid of the Codegen. Even if you write a line that looks incredibly wrong, compile it and see what it actually does and what error it gives. Other advice: pair program, and find a team that communicates well and commits to meeting every week at those times. I really appreciated that my team met at least twice every week without fail. Pair programming was also incredibly helpful near the end, as having a second set of eyes to debug errors made the process more efficient.

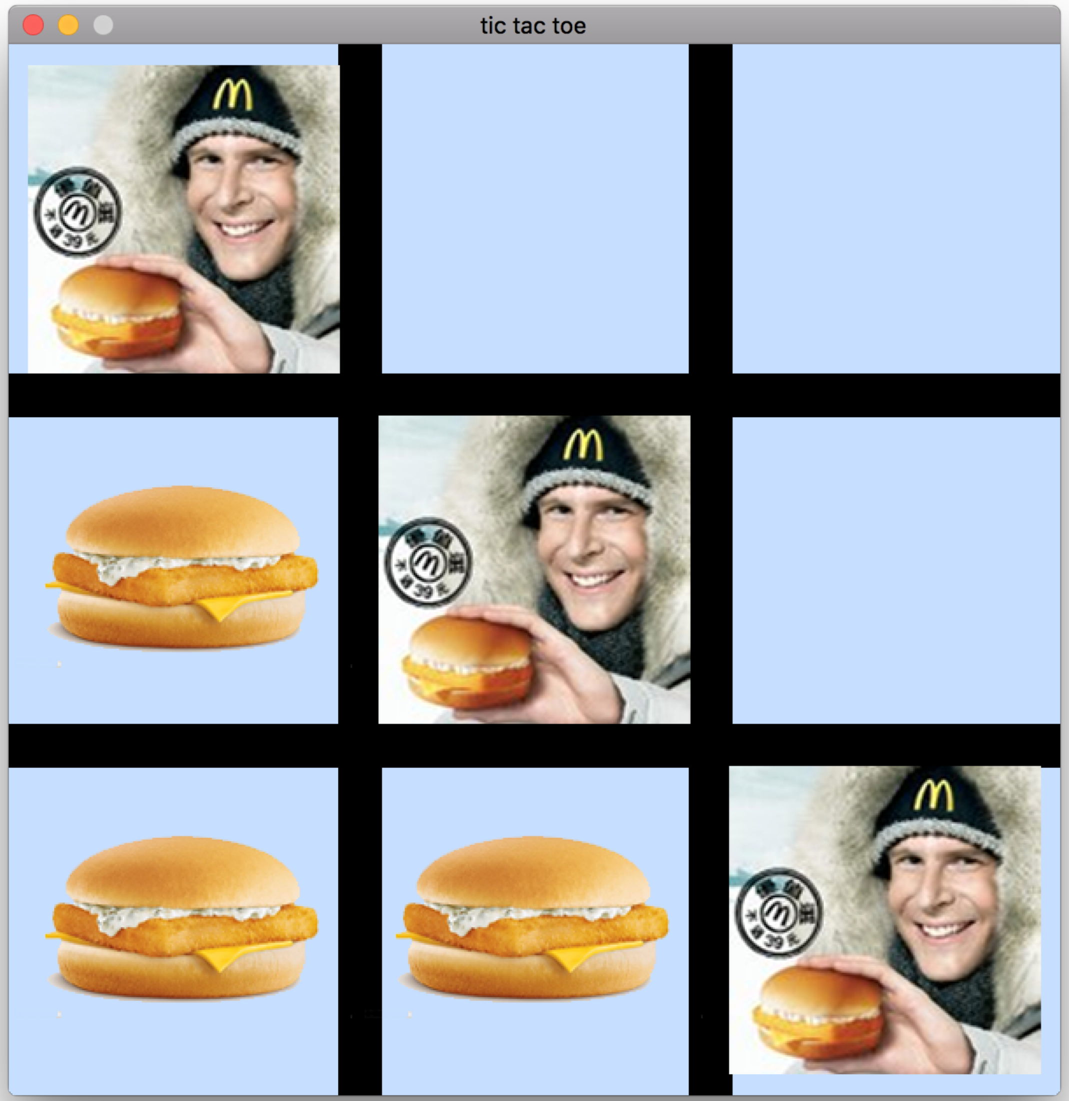
## 9. Demos

### 9.1 helloworld.tile

```
init {
   tile(3, 3);
   background("./sprites/hello.bmp");
}
```



9.2 tictactoe.tile

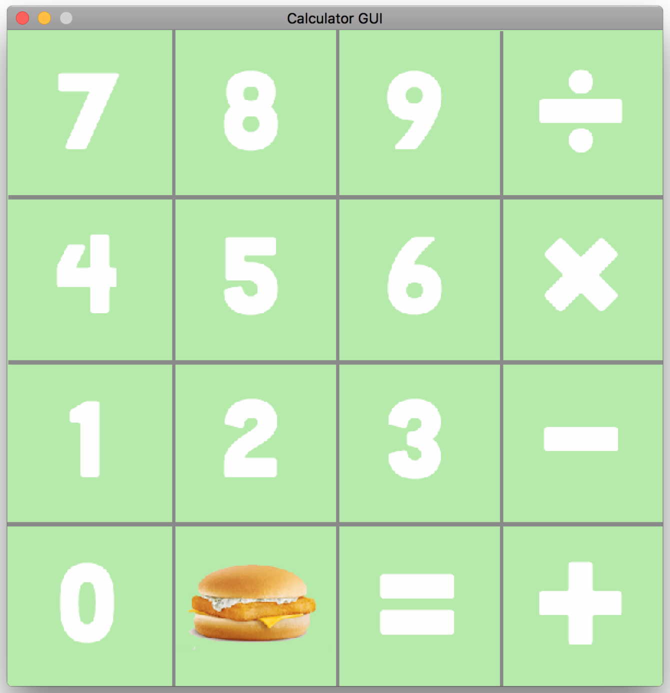


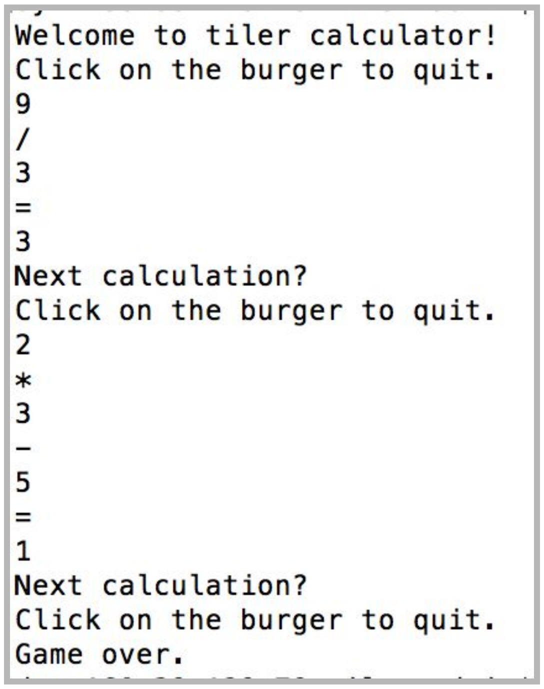
```
#size 600 600
#color 198 222 255
#title "tic tac toe"

coord mouse;
coord endIter;
bool edTurn;
int x; int y;
```

```
class Piece {
   attr: string player;
}

init {
   tile(3, 3);
   background("./sprites/tictactoe_board.bmp");
   edTurn = true;
}

turn {
   <Piece> p;
   do {
      mouse = capture();
   } while(isNull(grid[mouse[x], mouse[y]]) == 0);

   if (edTurn) {
      p = new Piece("burger");
      setSprite(p, "./sprites/burger.bmp");
   } else {
      p = new Piece("Prof. Edwards");
      setSprite(p, "./sprites/lolwards.bmp");
   }

   grid[mouse[x], mouse[y]] = p;
   edTurn = !edTurn;
}

end {
   <Piece> p1; <Piece> p2; <Piece> p3;
   string player; bool isFull;

   for (x=0; x<gridh; x=x+1) {
      p1 = grid[x, 0];
      p2 = grid[x, 1];
      p3 = grid[x, 2];

      if (isNull(p1) == 0 && isNull(p2) == 0 && isNull(p3) == 0) {
         player = p1.player;
         if (player == p2.player && player == p3.player) {
            sprint(player); sprint("wins");
            return 1;
         }
      }
   }

   for (y=0; y<gridw; y=y+1) {
      p1 = grid[0, y];
      p2 = grid[1, y];
      p3 = grid[2, y];
```

```
         if (isNull(p1) == 0 && isNull(p2) == 0 && isNull(p3) == 0) {
            player = p1.player;
            if (player == p2.player && player == p3.player) {
               sprint(player); sprint("wins");
               return 1;
            }
         }
      }

      p1 = grid[0, 0];
      p2 = grid[1, 1];
      p3 = grid[2, 2];

      if (isNull(p1) == 0 && isNull(p2) == 0 && isNull(p3) == 0) {
         player = p1.player;
         if (player == p2.player && player == p3.player) {
            sprint(player); sprint("wins");
            return 1;
         }
      }

      p1 = grid[0, 2];
      p2 = grid[1, 1];
      p3 = grid[2, 0];

      if (isNull(p1) == 0 && isNull(p2) == 0 && isNull(p3) == 0) {
         player = p1.player;
         if (player == p2.player && player == p3.player) {
            sprint(player); sprint("wins");
            return 1;
         }
      }

      isFull = true;
      for (x=0; x<gridh; x=x+1) {
         for (y=0; y<gridw; y=y+1) {
            p1 = grid[x, y];
            if (isNull(p1) == 1) {
               isFull = false;
            }
         }
      }

      if (isFull) {
         sprint("tie!");
         return 1;
      }
   }
```

## 9.3 calculator.tile





```
Welcome to tiler calculator!
Click on the burger to quit.
9
/
3
=
3
Next calculation?
Click on the burger to quit.
2
*
3
-
5
=
1
Next calculation?
Click on the burger to quit.
Game over.
```

```
#size 600 600
#color 181 234 170
#title "Calculator GUI"

int x; int y;
int result;
int current;
int current_power;
string current_op;
bool start_of_num;
bool start_of_expr;
coord mouse;
string p; string s;
string m; string d;
string e; string q;

class Num {
   attr: int value;
}
class Op {
   attr: string op;
```

```
}

init {
   <Num> n0; int count;
   <Num> n1; <Num> n2; <Num> n3;
   <Num> n4; <Num> n5; <Num> n6;
   <Num> n7; <Num> n8; <Num> n9;
   <Op> plus; <Op> minus;
   <Op> divide; <Op> times;
   <Op> equals; <Op> quit;

   start_of_expr = true; start_of_num = true;
   result = 0;
   p = "+"; s = "-"; m = "*";
   d = "/"; e = "="; q = "q";

   tile(4, 4);
   background("./sprites/4x4.bmp");

   n0 = new Num(0);
   setSprite(n0, "./sprites/0.bmp");
   grid[0, 3] = n0;

   n1 = new Num(1);
   setSprite(n1, "./sprites/1.bmp");
   grid[0, 2] = n1;

   n2 = new Num(2);
   setSprite(n2, "./sprites/2.bmp");
   grid[1, 2] = n2;

   n3 = new Num(3);
   setSprite(n3, "./sprites/3.bmp");
   grid[2, 2] = n3;

   n4 = new Num(4);
   setSprite(n4, "./sprites/4.bmp");
   grid[0, 1] = n4;

   n5 = new Num(5);
   setSprite(n5, "./sprites/5.bmp");
   grid[1, 1] = n5;

   n6 = new Num(6);
   setSprite(n6, "./sprites/6.bmp");
   grid[2, 1] = n6;

   n7 = new Num(7);
   setSprite(n7, "./sprites/7.bmp");
   grid[0, 0] = n7;
```

```
    n8 = new Num(8);
    setSprite(n8, "./sprites/8.bmp");
    grid[1, 0] = n8;

    n9 = new Num(9);
    setSprite(n9, "./sprites/9.bmp");
    grid[2, 0] = n9;

    plus = new Op(p);
    setSprite(plus, "./sprites/plus.bmp");
    grid[3, 3] = plus;

    minus = new Op(s);
    setSprite(minus, "./sprites/minus.bmp");
    grid[3, 2] = minus;

    times = new Op(m);
    setSprite(times, "./sprites/multiply.bmp");
    grid[3, 1] = times;

    divide = new Op(d);
    setSprite(divide, "./sprites/divide.bmp");
    grid[3, 0] = divide;

    equals = new Op(e);
    setSprite(equals, "./sprites/equal.bmp");
    grid[2, 3] = equals;

    quit = new Op(q);
    setSprite(quit, "./sprites/burger.bmp");
    grid[1, 3] = quit;

    sprint("Welcome to tiler calculator!");
    sprint("Click on the burger to quit.");
}

void calculate(){
    if (current_op == p){
        result = result + current;
    }
    if (current_op == s){
        result = result - current;
    }
    if (current_op == m){
        result = result * current;
    }
    if (current_op == d){
        result = result / current;
    }
```

```
}

turn {
   <Num> temp_num; <Op> temp_op;
   int val; current = 0;

   mouse = capture();
   while (mouse[x] != 3 && !(mouse[x]==2 && mouse[y]==3) && !(mouse[x]==1 && mouse[y]==3)){
      temp_num = grid[mouse[x], mouse[y]];
      val = temp_num.value;
      current = 10 * current + val;
      mouse = capture();
   }
   if (!(mouse[x]==1 && mouse[y]==3)){
      iprint(current);
   }
   if (start_of_expr){
      current_op = p;
   }
   calculate();
   start_of_expr = false;
   temp_op = grid[mouse[x], mouse[y]];
   current_op = temp_op.op;
   if (current_op == p){
      sprint("+");
   }
   if (current_op == s){
      sprint("-");
   }
   if (current_op == m){
      sprint("*");
   }
   if (current_op == d){
      sprint("/");
   }
   if (current_op == e){
      sprint("=");
      calculate();
      iprint(result);
      result = 0;
      start_of_expr = true;
      sprint("Next calculation?\nClick on the burger to quit.");
   }
}

end {
   if (current_op == q){
      sprint("Done!");
      return 1;
   }
```

```
}
```

## 10. Appendix

Appended to the final report are our project source code, which includes:

- Runtime library: authored by Evan
- Codegen, AST, scanner, parser: authored by everyone
- Makefile: authored by everyone
- Demo programs: authored by Monica and Jason
- Tests: authored by Jacky and Jason

Appended at the end is the git log history.

### 10.1 Tiler Compiler

#### 10.1.1 ast.ml

```
(* Abstract Syntax Tree and functions for printing it *)

type op = Add | Sub | Mult | Div | Mod | Equal | Neq | Less | Leq | Greater | Geq |
          And | Or

type uop = Neg | Not

type typ = Int | Float | Bool | String | Coord | Obj | Void

type bind =
    PrimDecl of typ * string
  | ObjDecl of string * string

type expr =
    Literal of int
  | FloatLit of float
  | BoolLit of bool
  | StringLit of string
  | Id of string
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of expr * expr
  | Access of string * string
  | Call of string * expr list
  | CoordLit of expr * expr
  | CoordAccess of string * string
  | Instant of string * expr list
  | GridAccess of expr * expr
  | GridCall of string
  | Null
  | Noexpr

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

type block = {
  btyp: typ;
  bname : string;
  formals : bind list;
  locals: bind list;
  body : stmt list;
}

type attr_decl = {
  atyp: typ;
  aname: string;
}

(* type rule_decl = RuleSet of expr *)

type class_decl = {
  cname: string;
  attributes: attr_decl list;
}

type flag =
    Title of string
  | Size of int * int
  | Color of int * int * int

type program = flag list * bind list * class_decl list * block list

(* Pretty-printing functions *)

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Mod -> "%"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"

let string_of_uop = function
    Neg -> "-"
  | Not -> "!"

let rec string_of_expr = function
    Literal(l) -> string_of_int l
  | FloatLit(l) -> string_of_float l
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
  | Id(s) -> s
  | StringLit(s) -> s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(s, e) -> string_of_expr s ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | CoordLit(e1, e2) -> "[" ^ string_of_expr e1 ^ ", " ^ string_of_expr e2 ^ "]"
  | CoordAccess(s1, s2) -> s1 ^ "." ^ s2
  | Instant(s, el) ->
      "new " ^ s ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Access(s1, s2) -> s1 ^ "." ^ s2
  | GridAccess(e1, e2) -> "grid[" ^ string_of_expr e1 ^ ", " ^ string_of_expr e2 ^ "]"
  | GridCall(s) -> "grid" ^ s
  | Null -> "NULL"
  | Noexpr -> ""

let rec string_of_stmt = function
    Block(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(expr) -> string_of_expr expr ^ ";\n"
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
      string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | For(e1, e2, e3, s) ->
      "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ; " ^
      string_of_expr e3  ^ ") " ^ string_of_stmt s
  | DoWhile(s, e) -> "do " ^ string_of_stmt s ^ "while (" ^ string_of_expr e ^ ")" ^ ";\n";
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s

let string_of_typ = function
    Int    -> "int"
  | Float  -> "float"
  | Bool   -> "bool"
  | String -> "string"
  | Coord  -> "coord"
  | Obj    -> "obj"
  | Void   -> "void"

let string_of_flag = function
    Title(title) -> "title: " ^ title ^ "\n"
  | Size(w, h) -> "size: " ^ string_of_int w ^ "x" ^ string_of_int h ^ "\n"
  | Color(r, g, b) -> "color: " ^ string_of_int r ^ ", " ^ string_of_int g ^ ", " ^
string_of_int b ^ "\n"

let string_of_vdecl = function
    PrimDecl(t, id) -> string_of_typ t ^ " " ^ id ^ ";\n"
  | ObjDecl(id1, id2) -> "<" ^ id1 ^ "> " ^ id2 ^ ";\n"

let string_of_attr attr = "attr: " ^ string_of_typ attr.atyp ^ " " ^ attr.aname ^ ";\n"

(* let string_of_rule = function
  RuleSet(e) -> "rule: " ^ string_of_expr e ^ ";\n" *)

let string_of_block block =
  string_of_typ block.btyp ^ " " ^
  (* block.bname ^ "(" ^ String.concat ", " (List.map snd block.formals) ^ ")\n{\n" ^ *)
  String.concat "" (List.map string_of_vdecl block.locals) ^
  String.concat "" (List.map string_of_stmt block.body) ^
  "}\n"

let string_of_classdecl cdecl =
  "class " ^ cdecl.cname ^ "{\n" ^
  String.concat "" (List.map string_of_attr cdecl.attributes) ^
  (* String.concat "" (List.map string_of_rule cdecl.rules) ^ *)
  "}\n"

let string_of_program (flags, vars, classes, funcs) =
  String.concat "" (List.map string_of_flag flags) ^ "\n" ^
  String.concat "\n" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "" (List.map string_of_classdecl classes) ^ "\n" ^
  String.concat "\n" (List.map string_of_block funcs)
```

### 10.1.2 scanner.mll

```
(* Ocamllex scanner for tiler *)

{ open Parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"     { multi lexbuf }             (* Multi-Line Comments *)
| "//"     { single lexbuf }             (* Single Line Comments *)
| "init"   { INIT }
| "turn"   { TURN }
| "end"    { END }
| "class"  { CLASS }
| "attr"   { ATTR }
| "new"    { NEW }
| "grid"   { GRID }
| "gridw"  { GRIDW }
| "gridh"  { GRIDH }
| "NULL"   { NULL }
| "#title" { TITLE }
| "#size"  { SIZE }
| "#color" { COLOR }
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| '['     { LSQUARE }
| ']'     { RSQUARE }
| ';'      { SEMI }
| ':'     { COLON }
| ','      { COMMA }
| '.'     { PERIOD }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| '%'     { MOD }
| '='      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "!"      { NOT }
| ">>"    { MOVE }
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "do"    { DO }
| "while"  { WHILE }
| "return" { RETURN }
| "int"    { INT }
| "float"  { FLOAT }
| "bool"   { BOOL }
| "string" { STRING }
| "coord"  { COORD }
| "void"   { VOID }
| "true"   { TRUE }
| "false"  { FALSE }
| ['0'-'9']+ as lxm { INT_LITERAL(int_of_string lxm) }
| ['0'-'9']*'.'['0'-'9']+ | ['0'-'9']+'.'['0'-'9']* as lxm { FLOAT_LITERAL(float_of_string 
lxm)}
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
| '"'([^'"']* as lxm)'"' { STRING_LITERAL(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
(* Ignoring contents between /* and */ of multi-line comments *)
and multi = parse
  "*/"     { token lexbuf }
| _        { multi lexbuf }


(* Ignoring content beyond // on the same line *)
and single = parse
  '\n'     { token lexbuf }
| _        { single lexbuf }
```

### 10.1.3 parser.mly

```
%{
  open Ast

  let first (a,_,_) = a;;
  let second (_,b,_) = b;;
  let third (_,_,c) = c;;
%}

%token INIT TURN END CLASS
%token GRID GRIDW GRIDH NULL
%token NEW ATTR COLON PERIOD MOVE
%token SEMI LPAREN RPAREN LBRACE RBRACE LSQUARE RSQUARE COMMA
%token RETURN IF ELSE FOR DO WHILE
%token TITLE SIZE COLOR
%token INT FLOAT BOOL STRING COORD VOID
%token PLUS MINUS TIMES DIVIDE MOD ASSIGN NOT
%token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
%token <string> ID
%token <int> INT_LITERAL
%token <float> FLOAT_LITERAL
%token <string> STRING_LITERAL
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%right COLON
%left OR
```

```
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE MOD
%right NOT NEG
%left ACCESS

%start program
%type <Ast.program> program

%%

program:
  wflags decls EOF { (List.rev ($1), List.rev (first $2), List.rev (second $2), List.rev
(third $2)) }

wflags:
   /* nothing */  { [] }
 | wflags wflag   { $2 :: $1 }

wflag:
   TITLE STRING_LITERAL                          { Title($2) }
 | SIZE INT_LITERAL INT_LITERAL                  { Size($2, $3) }
 | COLOR INT_LITERAL INT_LITERAL INT_LITERAL  { Color($2, $3, $4) }

decls:
   /* nothing */      { [], [], [] }
 | decls vdecl        { ($2 :: first $1), second $1, third $1 }
 | decls class_decl   { first $1, ($2 :: second $1), third $1 }
 | decls block        { first $1, second $1, ($2 :: third $1) }

typ:
    INT    { Int }
  | FLOAT  { Float }
  | BOOL   { Bool }
  | STRING { String }
  | COORD  { Coord }
  | VOID   { Void }

vdecl_list:
    /* nothing */    { [] }
  | vdecl_list vdecl { $2 :: $1 }

vdecl:
    typ ID SEMI       { PrimDecl($1, $2) }
  | LT ID GT ID SEMI  { ObjDecl($2, $4) }
```

## 10.1.4 semant.ml

```
open Ast

module StringMap = Map.Make(String)

(* Semantic checking of a program. Returns void if successful,
   throws an exception if something is wrong.

   Check each global variable, then check each special block *)

let check (flags, globals, classes, special_blocks) =

  (* Method for finding duplicates *)
  let report_duplicate exceptf list =
    let rec helper = function
        n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf n1))
      | _ :: t -> helper t
      | [] -> ()
    in
    helper (List.sort compare list)
  in

  (* Access types *)
  let get_type ele = match ele with
      PrimDecl(t,_) -> t
    | ObjDecl(c,_) -> Void
  in

  (* Method for checking that assignment is legal by comparing types on both sides *)
  (* let check_assign lvaluet rvaluet err =
    if lvaluet == rvaluet then lvaluet else raise err
  in *)

  (**** Checking flags ****)
  let flag_check wflag = match wflag with
      Title(_) -> ()
    | Size(w,h) ->
        if w <= 0 || h <= 0 then raise (Failure ("size cannot be less than or equal to 0"))
        else ()
    | Color(r,g,b) ->
        if r > 255 || g > 255 || b > 255 || r < 0 || g < 0 || b < 0 then raise (Failure
("invalid RGB values for #color"))
        else ()
  in
  List.iter flag_check flags;

  (**** Checking globals ****)
  let check_gbl l g = match g with
      PrimDecl(_,n) -> n :: l
      (* Major issue of getting the correct type of attribute back *)
    | ObjDecl(_,n) -> raise (Failure ("Cannot declare global object " ^ n))
  in

  let prims = List.fold_left check_gbl [] globals in

  (* check for duplicate globals *)
  report_duplicate (fun n -> "duplicate global variable " ^ n) prims;

  (* Checks for duplicate special blocks *)
  report_duplicate (fun n -> "duplicate blocks " ^ n) (List.map (fun block -> block.bname)
special_blocks);

  (**** Classes ****)
  let class_decls =
    let class_decl m c =
      let name = c.cname in
      (* make map of attr name -> type *)
      let add_attr_typ map attr =
        StringMap.add attr.aname attr.atyp map
      in
      let map_attr_typ = List.fold_left add_attr_typ StringMap.empty c.attributes in

      StringMap.add name map_attr_typ m
    in
    (* create list of classes and their attrs mapped to attr names *)
    List.fold_left class_decl StringMap.empty classes
  in

  (* check that class decls are valid *)
  (* let check_obj_decl n =
    if StringMap.mem n class_decls then ()
    else raise (Failure ("class undefined"))
  in *)

  (*** Check for necessary code blocks ***)

  (* add built-in functions *)
  let built_in_decls = StringMap.add "tile"
    { btyp = Void; bname = "tile"; formals = [PrimDecl(Int, "x"); PrimDecl(Int, "y")];
     locals = []; body = [] }
    (StringMap.singleton "background"
    { btyp = Void; bname = "background"; formals = [PrimDecl(String, "s")];
     locals = []; body = [] })
  in

  let built_in_decls = StringMap.add "capture"
    { btyp = Void; bname = "capture"; formals = [];
      locals = []; body = [] } built_in_decls
  in
  let built_in_decls = StringMap.add "setSprite"
    { btyp = Void; bname = "setSprite"; formals = [ObjDecl("type", "var"); PrimDecl(Int,
"x")];
      locals = []; body = [] } built_in_decls
  in

  let built_in_decls = StringMap.add "isNull"
    { btyp = Int; bname = "isNull"; formals = [ObjDecl("type", "var")];
      locals = []; body = [] } built_in_decls
  in

  let built_in_decls = StringMap.add "iprint"
    { btyp = Void; bname = "iprint"; formals = [PrimDecl(Int, "x")];
      locals = []; body = [] } built_in_decls
  in

  let built_in_decls = StringMap.add "fprint"
    { btyp = Void; bname = "fprint"; formals = [PrimDecl(Float, "x")];
      locals = []; body = [] } built_in_decls
  in

  let built_in_decls = StringMap.add "sprint"
    { btyp = Void; bname = "sprint"; formals = [PrimDecl(String, "x")];
      locals = []; body = [] } built_in_decls
  in

  let built_in_decls = StringMap.add "close"
    { btyp = Void; bname = "close"; formals = [];
      locals = []; body = []} built_in_decls
  in

  (* Create map of all functions - prevent duplicate function names *)
  let function_decls = List.fold_left (fun m fd -> StringMap.add fd.bname fd m)
                     built_in_decls special_blocks
  in

  (* Find function or block given its function name *)
  let function_decl s = try StringMap.find s function_decls
    with Not_found -> raise (Failure ("unrecognized function or block " ^ s))
  in

  let check_function func =

    (* Type of each variable (global, formal, or local *)
    let (symbols,objects) =
      let add_to_lists (s,o) x = match x with
          PrimDecl(t,n) -> (StringMap.add n (get_type x) s, o)
        | ObjDecl(c,n) -> (StringMap.add n (get_type x) s, StringMap.add n c o)
      in
      List.fold_left add_to_lists
      (StringMap.empty, StringMap.empty) (globals @ func.formals @ func.locals )
    in

    (* Get class of object *)
    let class_of_obj s = try StringMap.find s objects with
      Not_found -> raise (Failure ("Object declaration of undefined class"))
    in

    let type_of_identifier s =
      try StringMap.find s symbols
      with Not_found -> raise (Failure ("undeclared identifier" ^ s))
    in

    (* Return the type of an expression or throw an exception *)
    let rec expr = function
      Literal _ -> Int
      | BoolLit _ -> Bool
      | FloatLit _ -> Float
      | StringLit _ -> String
      | Id s -> type_of_identifier s
      | Noexpr -> Void

      | CoordLit (x, y) -> let t1 = expr x and t2 = expr y in
        if t1 = Int && t2 = Int then Coord
        else raise (Failure ("expected integers for type coord"))
      | CoordAccess (_, a) ->
        if a = "x" || a = "y" then Int
        else Coord

      | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2 in
          (match op with
            Add | Sub | Mult | Div when t1 = Int && t2 = Int     -> Int
          | Add | Sub | Mult | Div when t1 = Int && t2 = Float   -> Float
          | Add | Sub | Mult | Div when t1 = Float && t2 = Int   -> Float
          | Add | Sub | Mult | Div when t1 = Float && t2 = Float -> Float
          | Equal | Neq when t1 = t2 -> Bool
          | Less | Leq | Greater | Geq when t1 = Int && t2 = Int   -> Bool
          | Less | Leq | Greater | Geq when t1 = Int && t2 = Float -> Bool
          | Less | Leq | Greater | Geq when t1 = Float && t2 = Int -> Bool
          | Less | Leq | Greater | Geq when t1 = Float && t2 = Float -> Bool
          | And | Or when t1 = Bool && t2 = Bool -> Bool
          | _ -> raise (Failure ("illegal binary operator " ^
                string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
                string_of_typ t2 ^ " in " ^ string_of_expr e)))

      | Unop(op, e) as ex -> let t = expr e in
          (match op with
            Neg when t = Int -> Int
          | Neg when t = Float -> Float
```

```
          | Not when t = Bool -> Bool
          | _ -> raise (Failure ("illegal unary operator " ^ string_of_uop op ^
                string_of_typ t ^ " in " ^ string_of_expr ex)))

      | Call(fname, actuals) as call ->
        if fname = "isNull" then Int
        else
          if fname = "capture" then Coord
          else let fd = function_decl fname in fd.btyp

      | Assign(le, re) as ex ->
        (match re with
          | Instant(c, attrs) ->
            (match le with
              | Id (s) ->
                let obj_class = class_of_obj s in
                if obj_class = c then Void
                else raise (Failure ("class mismatch: assignment " ^ obj_class ^ " to " ^ c))
              | _ -> raise (Failure ("must instantiate a variable of class " ^ c))
            )
          | _ ->
            let t2 = expr re in
            let t1 =
            (match le with
              | Id (s) -> type_of_identifier s
              | Access (s1, s2) ->
                if StringMap.mem s1 symbols then
                  let obj_class = class_of_obj s1 in
                  let attr_decls = StringMap.find obj_class class_decls in
                  if StringMap.mem s2 attr_decls then Void
                  else raise (Failure ("attribute does not exist"))
                else raise (Failure ("object was not declared"))
              | GridAccess (_,_) -> Void
              | _ -> expr le
            ) in
            if t2 == t1 then t2
            else raise (Failure ("Illegal assignment " ^ string_of_typ t1 ^ " = " ^
string_of_typ t2 ^ " in " ^ string_of_expr ex))
        )

      (* | Assign(le, re) as ex -> let t1 = expr le and t2 = expr re in
          if t1 == t2 then t1 else raise (Failure ("illegal assignment " ^ string_of_typ t1 ^
            " = " ^ string_of_typ t2 ^ " in " ^ string_of_expr ex)) *)

      | GridCall(s) -> Int
      | GridAccess (_,_) -> Void
      | Access (s, a) ->
        if StringMap.mem s symbols then
          let obj_class = class_of_obj s in
          let attr_decls = StringMap.find obj_class class_decls in
          if StringMap.mem a attr_decls then StringMap.find a attr_decls
          else raise (Failure ("attribute does not exist"))
        else raise (Failure ("object was not declared"))

      | Instant (_,_) -> Void

      | Null -> Void

    in

    let check_bool_expr e = if expr e != Bool
    then raise (Failure ("expected Boolean expression in " ^ string_of_expr e))
    else () in

    (* Verify a statement or throw an exception *)
    let rec stmt = function
      Block sl -> let rec check_block = function
        [Return _ as s] -> stmt s
        | Block sl :: ss -> check_block (sl @ ss)
        | s :: ss -> stmt s ; check_block ss
        | [] -> ()
      in check_block sl
      | Expr e -> ignore (expr e)
      | Return e -> let t = expr e in if t = func.btyp then () else
         raise (Failure ("return gives " ^ string_of_typ t ^ " expected " ^
                         string_of_typ func.btyp ^ " in " ^ string_of_expr e))

      | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
      | For(e1, e2, e3, st) -> ignore (expr e1); check_bool_expr e2;
                               ignore (expr e3); stmt st
      | While(p, s) -> check_bool_expr p; stmt s
      | DoWhile(s, p) -> check_bool_expr p; stmt s
    in

    stmt (Block func.body)

  in
  List.iter check_function special_blocks
```

### 10.1.5 codegen.ml

```
module L = Llvm
module A = Ast

module StringMap = Map.Make(String)

let translate (wflags, globals, classes, blocks) =
  let context = L.global_context () in
  let the_module = L.create_module context "Tiler"
  and i32_t  = L.i32_type   context
  and i8_t   = L.i8_type    context
  and i1_t   = L.i1_type    context
  and flt_t  = L.double_type context
  and void_t = L.void_type  context in
  let str_t  = L.pointer_type i8_t in
  let obj_t  = L.pointer_type i8_t in

  let coord_t = L.named_struct_type context "coord_t" in
    L.struct_set_body coord_t [| i32_t; i32_t |] false;

  let ltype_of_typ = function
    A.Void -> void_t
    | A.Int -> i32_t
    | A.Bool -> i1_t
    | A.Float -> flt_t
    | A.String -> str_t
    | A.Coord -> coord_t
  in

  let get_init = function
      A.Int -> L.const_int i32_t 0
    | A.Bool -> L.const_int i1_t 0
    | A.Float -> L.const_float flt_t 0.0
    | A.String -> L.const_pointer_null str_t
    | A.Coord -> L.const_named_struct coord_t [|L.const_int i32_t 0; L.const_int i32_t 0|]
    | _ -> L.const_int i32_t 0
  in

  (* Declare grid(), which the grid built-in function will call *)
  let createGrid_t = L.function_type void_t [| i32_t; i32_t |] in
  let createGrid_func = L.declare_function "createGrid" createGrid_t the_module in

  let setTitle_t = L.function_type void_t [| str_t |] in
  let setTitle_func = L.declare_function "setTitle" setTitle_t the_module in

  let setBackground_t = L.function_type void_t [| str_t |] in
  let setBackground_func = L.declare_function "setBackground" setBackground_t the_module in

  let setBackgroundColor_t = L.function_type void_t [| i32_t; i32_t; i32_t |] in
  let setBackgroundColor_func = L.declare_function "setBackgroundColor" setBackgroundColor_t
the_module in

  let setWindow_t = L.function_type void_t [| i32_t; i32_t |] in
  let setWindow_func = L.declare_function "setWindow" setWindow_t the_module in

  let createGame_t = L.function_type void_t [| |] in
  let createGame_func = L.declare_function "createGame" createGame_t the_module in

  (* Game functions *)
  let init_type = L.function_type void_t [| |] in
  let setInit_t = L.function_type void_t [| L.pointer_type init_type |] in
  let setInit_func = L.declare_function "setInit" setInit_t the_module in

  let turn_type = L.function_type void_t [| |] in
  let setTurn_t = L.function_type void_t [| L.pointer_type turn_type |] in
  let setTurn_func = L.declare_function "setTurn" setTurn_t the_module in

  let end_type = L.function_type i32_t [| |] in
  let setEnd_t = L.function_type void_t [| L.pointer_type end_type |] in
  let setEnd_func = L.declare_function "setEnd" setEnd_t the_module in

  let runGame_t = L.function_type void_t [| |] in
  let runGame_func = L.declare_function "runGame" runGame_t the_module in

  (* Method for force closing windows, used mostly for testing purposes *)
  let closeGame_t = L.function_type void_t [| |] in
  let closeGame_func = L.declare_function "closeGame" closeGame_t the_module in

  (* Grid functions *)
  let gridWidth_t = L.function_type i32_t [| |] in
  let gridWidth_func = L.declare_function "gridWidth" gridWidth_t the_module in

  let gridHeight_t = L.function_type i32_t [| |] in
  let gridHeight_func = L.declare_function "gridHeight" gridHeight_t the_module in

  let setGrid_t = L.function_type void_t [| obj_t; i32_t; i32_t|] in
  let setGrid_func = L.declare_function "setGrid" setGrid_t the_module in

  let getGrid_t = L.function_type obj_t [| i32_t; i32_t|] in
  let getGrid_func = L.declare_function "getGrid" getGrid_t the_module in

  (* Object functions *)
  let createObject_t = L.function_type obj_t [| obj_t |] in
  let createObject_func = L.declare_function "createObject" createObject_t the_module in

  let setSprite_t = L.function_type void_t [| obj_t; str_t|] in
  let setSprite_func = L.declare_function "setSprite" setSprite_t the_module in

  let getAttr_t = L.function_type obj_t [| obj_t |] in
  let getAttr_func = L.declare_function "getAttr" getAttr_t the_module in

  let getX_t = L.function_type i32_t [||] in
  let getX_func = L.declare_function "getX" getX_t the_module in

  let getY_t = L.function_type i32_t [||] in
  let getY_func = L.declare_function "getY" getY_t the_module in

  let isNull_t = L.function_type i32_t [| obj_t |] in
  let isNull_func = L.declare_function "isNull" isNull_t the_module in
```

```
  let getNullObject_t = L.function_type obj_t [||] in
  let getNullObject_func = L.declare_function "createNullObject" getNullObject_t the_module in

  (* Mouse function *)
  let getMouseCoords_t = L.function_type void_t [| obj_t |] in
  let getMouseCoords_func = L.declare_function "getMouseCoords" getMouseCoords_t the_module in

  (* Declare printf(), which the print built-in function will call *)
  let printf_t = L.var_arg_function_type i32_t [| str_t |] in
  let printf_func = L.declare_function "printf" printf_t the_module in

  (* Ensures int *)
  let ensureInt c =
  if L.type_of c = flt_t then (L.const_fptosi c i32_t) else c in

  (* Ensures float *)
  let ensureFloat c =
  if L.type_of c = flt_t then c else (L.const_sitofp c flt_t) in

  (* All global variables; remember its value in a map *)
  let global_vars =
    let global_var m decl =
      match decl with
          A.PrimDecl(t, n) ->
            let init = get_init t in
            StringMap.add n (L.define_global n init the_module) m
        | _ -> raise (Failure("can't define global objects"))
    in
    List.fold_left global_var StringMap.empty globals
  in

  (* make a class struct pointer *)
  let class_attr_decls =
    let class_decl m c =


      let name = c.A.cname in
      let add_attr_types m attr =
        Array.append m [| ltype_of_typ attr.A.atyp |]
      in

      (* make array of just attr types *)
      let attr_types = List.fold_left add_attr_types [||] c.A.attributes in

      (* make map of attr name -> index *)
      let add_attr_idx (map, idx) attr =
        (* get index of attr in list attr_types *)
        (StringMap.add attr.A.aname idx map, idx + 1)
      in
      let map_attr_idx = fst (List.fold_left add_attr_idx (StringMap.empty, 0) c.A.attributes)
in
```

```
      (* make struct for class and associate list of (attr name, typ) with class *)
      let class_struct = L.named_struct_type context name in

      L.struct_set_body class_struct attr_types false;
      StringMap.add name (class_struct, map_attr_idx) m
    in
    (* create list of structs for all classes *)
    List.fold_left class_decl StringMap.empty classes
  in

  let block_decls =
    let block_decl m block =
      let name = block.A.bname in
      let map_types ta formal =
        match formal with
            A.PrimDecl(t, _) ->
              Array.append ta [| (ltype_of_typ t) |]
          | _ -> raise (Failure("invalid function formal"))
      in
      let formal_types = List.fold_left map_types [||] block.A.formals in
      let btype = L.function_type (ltype_of_typ block.A.btyp) formal_types in
      StringMap.add name (L.define_function name btype the_module, block) m in
    List.fold_left block_decl StringMap.empty blocks
  in

  (* declares the function to set window flags *)
  let flag_func_ptr =
    let typ = L.function_type void_t [| |] in
    L.define_function "flags" typ the_module in

  (* builds the function to set window flags *)
  let flag_function_builder flags =

let builder = L.builder_at_end context (L.entry_block flag_func_ptr) in
      let flag_build wflag = (match wflag with
        A.Title(str) ->
          let title = L.build_global_stringptr str "title" builder in
          ignore(L.build_call setTitle_func [| title |] "" builder)
      | A.Size(w,h) ->
          let width = L.const_int i32_t w
          and height = L.const_int i32_t h in
          ignore(L.build_call setWindow_func [| width; height |] "" builder)
      | A.Color(r,g,b) ->
          let red = L.const_int i32_t r
          and green = L.const_int i32_t g
          and blue = L.const_int i32_t b in
      ignore(L.build_call setBackgroundColor_func [| red; green; blue |] "" builder)
      ) in
    List.iter flag_build flags;
    ignore(L.build_ret_void builder)
  in

  let build_block_body block =
    let (the_block, _) = StringMap.find block.A.bname block_decls in
    let builder = L.builder_at_end context (L.entry_block the_block) in

    (* Construct the function's "locals": formal arguments and locally
       declared variables.  Allocate each on the stack, initialize their
       value, if appropriate, and remember their values in the "locals" map *)
    let local_vars =
      let add_formal m formal p =
        match formal with
            A.PrimDecl(t, n) ->
              L.set_value_name n p;
              let local = L.build_alloca (ltype_of_typ t) n builder in
              ignore (L.build_store p local builder);
              StringMap.add n local m
          | A.ObjDecl(_,_) -> raise (Failure("invalid function formal"))
      in
      let formals = List.fold_left2 add_formal StringMap.empty block.A.formals (Array.to_list
(L.params the_block)) in

      let add_local (m1, m2) decl =
        match decl with
          A.PrimDecl(t, n) ->
            let local_var = L.build_alloca (ltype_of_typ t) n builder in
            (StringMap.add n local_var m1, m2)
        | A.ObjDecl(c, id) ->
            let (cstruct, _) = StringMap.find c class_attr_decls in
            let cstruct_ptr = L.build_malloc cstruct (c ^ "_class_struct_ptr") builder in
            let cstruct_ptr = L.build_pointercast cstruct_ptr obj_t (c ^ "_ptrcast") builder
in
            let obj_ptr = L.build_call createObject_func [| cstruct_ptr |] "" builder in

            let obj_ptr_ptr = L.build_alloca obj_t id builder in
            ignore(L.build_store obj_ptr obj_ptr_ptr builder);
            (m1, StringMap.add id (c, obj_ptr_ptr) m2)
      in
      List.fold_left add_local (formals, StringMap.empty) block.A.locals
    in

    (* Return the value for a variable or formal argument *)
    let lookup_var n = try StringMap.find n (fst local_vars) with
      | Not_found -> try snd (StringMap.find n (snd local_vars)) with
      | Not_found -> try StringMap.find n global_vars with
      | Not_found -> raise (Failure("unknown variable name " ^ n))
    in

    let lookup_obj n = try StringMap.find n (snd local_vars) with
      | Not_found -> raise (Failure("unknown object name " ^ n))
    in

    (* Construct code for an expression; return its value *)
    let rec expr builder = function
        A.Literal i -> L.const_int i32_t i
      | A.BoolLit b -> L.const_int i1_t (if b then 1 else 0)
      | A.FloatLit f -> L.const_float flt_t f
      | A.StringLit s -> L.build_global_stringptr s "name" builder
       | A.Noexpr -> L.const_int i32_t 0
      | A.CoordLit (e1, e2) ->
          let e1' = ensureInt (expr builder e1)
          and e2' = ensureInt (expr builder e2) in
          let coord_ptr = L.build_alloca coord_t "tmp" builder in
          let x_ptr = L.build_struct_gep coord_ptr 0 "x" builder in
          ignore (L.build_store e1' x_ptr builder);
          let y_ptr = L.build_struct_gep coord_ptr 1 "y" builder in
          ignore (L.build_store e2' y_ptr builder);
          L.build_load coord_ptr "v" builder

      | A.CoordAccess (s1, s2) ->
        let coord_ptr = (lookup_var s1) in
          let idx =
            (match s2 with
                "x" -> 0
              | "y" -> 1
              | _ -> raise (Failure("choose x or y to access coordinate"))
            )
          in
        let value_ptr = L.build_struct_gep coord_ptr idx (s2 ^ "_ptr") builder in
        L.build_load value_ptr s2 builder

      | A.GridAccess (e1, e2) ->
        let e1' = expr builder e1
        and e2' = expr builder e2 in
        L.build_call getGrid_func [| e1'; e2' |] "accessed_obj" builder

      | A.Binop (e1, op, e2) ->
        let e1' = expr builder e1
        and e2' = expr builder e2 in
        if (L.type_of e1' = flt_t || L.type_of e2' = flt_t) then
          (match op with
            A.Add     -> L.build_fadd
          | A.Sub     -> L.build_fsub
          | A.Mult    -> L.build_fmul
          | A.Div     -> L.build_fdiv
          | A.Mod     -> L.build_frem
          | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
          | A.Neq     -> L.build_fcmp L.Fcmp.One
          | A.Less    -> L.build_fcmp L.Fcmp.Olt
```

```
          | A.Leq     -> L.build_fcmp L.Fcmp.Ole
          | A.Greater -> L.build_fcmp L.Fcmp.Ogt
          | A.Geq     -> L.build_fcmp L.Fcmp.Oge
          | _         -> raise (Failure("invalid operands for floating point arguments"))
          ) (ensureFloat e1') (ensureFloat e2') "tmp" builder
        else
          (match op with
              A.Add     -> L.build_add
            | A.Sub     -> L.build_sub
            | A.Mult    -> L.build_mul
            | A.Div     -> L.build_sdiv
            | A.Mod     -> L.build_srem
            | A.And     -> L.build_and
            | A.Or      -> L.build_or
            | A.Equal   -> L.build_icmp L.Icmp.Eq
            | A.Neq     -> L.build_icmp L.Icmp.Ne
            | A.Less    -> L.build_icmp L.Icmp.Slt
            | A.Leq     -> L.build_icmp L.Icmp.Sle
            | A.Greater -> L.build_icmp L.Icmp.Sgt
            | A.Geq     -> L.build_icmp L.Icmp.Sge
          ) e1' e2' "tmp" builder

      | A.Unop(op, e) ->
        let e' = expr builder e in
        (match op with
            A.Neg     -> L.build_neg
          | A.Not     -> L.build_not) e' "tmp" builder

      | A.Assign (le, re) ->
        (match re with
          | A.Instant(c, attrs) ->
            (match le with
              | A.Id (s) ->
                let obj_ptr_ptr = lookup_var s in
                let obj_ptr = L.build_load obj_ptr_ptr "obj" builder in
                let cstruct_ptr = L.build_call getAttr_func [| obj_ptr |] "" builder in
                let (cstruct, _) = StringMap.find c class_attr_decls in
                let new_ptr = L.build_pointercast cstruct_ptr (L.pointer_type cstruct) (c ^
"_ptrcast") builder in
                ignore(List.fold_left (fun idx a ->
                  let attr_ptr = L.build_struct_gep new_ptr idx (c ^ (string_of_int idx) ^
"_attr_ptr") builder in
                  ignore (L.build_store (expr builder a) attr_ptr builder);
                  idx + 1
                ) 0 attrs);
                obj_ptr

              | _ -> raise (Failure ("Must instantiate a variable"))
            )
          | _ -> let e' = expr builder re in
```

```
            (match le with
              | A.Id (s) -> L.build_store e' (lookup_var s) builder
              | A.Access (s1, s2) ->
                let (cname, obj_ptr_ptr) = lookup_obj s1 in
                let obj_ptr = L.build_load obj_ptr_ptr "obj" builder in
                let cstruct_ptr = L.build_call getAttr_func [| obj_ptr |] "" builder in
                let (cstruct, attr_idx) = StringMap.find cname class_attr_decls in
                let new_ptr = L.build_pointercast cstruct_ptr (L.pointer_type cstruct) (s1 ^
"_ptr") builder in
                let idx = StringMap.find s2 attr_idx in
                let attr_ptr = L.build_struct_gep new_ptr idx (s1 ^ "_" ^ s2 ^ "_attr_ptr")
builder in
                L.build_store e' attr_ptr builder;
              | A.GridAccess (e1, e2) ->
                let e1' = expr builder e1
                and e2' = expr builder e2 in
                ignore(L.build_call setGrid_func [| e'; e1'; e2' |] "" builder); e'
              | _ -> L.const_int i32_t 0
            )
        )

      | A.Id s -> L.build_load (lookup_var s) s builder

      | A.Call ("tile", [e1; e2]) ->
       L.build_call createGrid_func [| (expr builder e1); (expr builder e2) |] "" builder

      | A.Call ("background", [e]) ->
        L.build_call setBackground_func [| (expr builder e) |] "" builder

      | A.Call ("setSprite", [e1; e2]) ->
        L.build_call setSprite_func [| (expr builder e1); (expr builder e2) |] "" builder

      | A.Call ("capture", []) ->
        let coord_ptr = L.build_alloca coord_t "tmp" builder in
        let void_ptr = L.build_pointercast coord_ptr (L.pointer_type i8_t) "tmp2" builder in

        let coord_ptr2 =
          ignore(L.build_call getMouseCoords_func [| void_ptr |] "" builder);
          L.build_pointercast void_ptr (L.pointer_type coord_t) "c" builder
        in
        L.build_load coord_ptr2 "v" builder

      | A.Call ("isNull", [e]) ->
        L.build_call isNull_func [| (expr builder e) |] "" builder

       (* Set of print functions that prints to stdout *)
      | A.Call ("iprint", [e]) ->
        let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder in
        L.build_call printf_func [| int_format_str ; (expr builder e) |] "printf" builder
      | A.Call ("fprint", [e]) ->
```

```
        let flt_format_str = L.build_global_stringptr "%f\n" "fmt" builder in
        L.build_call printf_func [| flt_format_str ; (expr builder e) |] "printf" builder
      | A.Call ("sprint", [e]) ->
        let str_format_str = L.build_global_stringptr "%s\n" "fmt" builder in
        L.build_call printf_func [| str_format_str ; (expr builder e) |] "printf" builder

      | A.Call ("close", []) ->
        L.build_call closeGame_func [| |] "" builder

      (* Non-defined functions are defined by this case *)
      | A.Call (f, act) ->
        let (fdef, block) = StringMap.find f block_decls in
        let actuals = List.rev (List.map (expr builder) (List.rev act)) in
        let result = (match block.A.btyp with A.Void -> "" | _ -> f ^ "_result") in
        L.build_call fdef (Array.of_list actuals) result builder

      | A.GridCall (s) ->
        if s = "w" then
          L.build_call gridWidth_func [||] "gridw" builder
        else L.build_call gridHeight_func [||] "gridh" builder

      | A.Access (c, a) ->
        let (cname, obj_ptr_ptr) = lookup_obj c in
        let obj_ptr = L.build_load obj_ptr_ptr "obj" builder in
        let cstruct_ptr = L.build_call getAttr_func [| obj_ptr |] "" builder in
        let (cstruct, attr_idx) = StringMap.find cname class_attr_decls in
        let new_ptr = L.build_pointercast cstruct_ptr (L.pointer_type cstruct) (c ^
"_ptrcast") builder in

        (match a with
          | "x" -> L.build_call getX_func [||] "" builder
          | "y" -> L.build_call getY_func [||] "" builder
          | _ ->
            let idx = StringMap.find a attr_idx in
            let attr_ptr = L.build_struct_gep new_ptr idx (c ^ "_" ^ a ^ "_attr_ptr") builder
in
            L.build_load attr_ptr ("load_" ^ a ^ "_attr") builder
        )
      | A.Instant (_,_) -> raise (Failure ("Instantation needs an assignment"))
        (* this is essentially meaningless in this context w/o LHS *)
      | A.Null -> L.build_call getNullObject_func [||] "" builder
    in

    (* Invoke "f builder" if the current block doesn't already
       have a terminal (e.g., a branch). *)
    let add_terminal builder f =
      match L.block_terminator (L.insertion_block builder) with
      Some _ -> ()
      | None -> ignore (f builder) in
```

```
(* Build the code for the given statement; return the builder for
   the statement's successor *)
let rec stmt builder = function
          A.Block sl -> List.fold_left stmt builder sl
  | A.Expr e -> ignore (expr builder e); builder
  | A.Return e -> ignore (match block.A.btyp with
      A.Void -> L.build_ret_void builder
  | _ -> L.build_ret (expr builder e) builder); builder
  | A.If (predicate, then_stmt, else_stmt) ->
    let bool_val = expr builder predicate in
    let merge_bb = L.append_block context "merge" the_block in
      let then_bb = L.append_block context "then" the_block in
      add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
        (L.build_br merge_bb);

      let else_bb = L.append_block context "else" the_block in
      add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
        (L.build_br merge_bb);

    ignore (L.build_cond_br bool_val then_bb else_bb builder);
    L.builder_at_end context merge_bb

  | A.While (predicate, body) ->
    let pred_bb = L.append_block context "while" the_block in
    ignore (L.build_br pred_bb builder);

    let body_bb = L.append_block context "while_body" the_block in
    add_terminal (stmt (L.builder_at_end context body_bb) body)
      (L.build_br pred_bb);

    let pred_builder = L.builder_at_end context pred_bb in
    let bool_val = expr pred_builder predicate in

    let merge_bb = L.append_block context "merge" the_block in
    ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
    L.builder_at_end context merge_bb

  | A.DoWhile (body, predicate) -> stmt builder
    ( A.Block [ A.Block [ body ] ; A.While(predicate, body) ] )

  | A.For (e1, e2, e3, body) -> stmt builder
  ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ] )
in

(* Build the code for each statement in the function *)
let builder = stmt builder (A.Block block.A.body) in

(* Add a return if the last block falls off the end *)
add_terminal builder (match block.A.btyp with
    A.Void -> L.build_ret_void
```

```
      | t -> L.build_ret (L.const_int (ltype_of_typ t) 0)
    ) in

  List.iter build_block_body blocks;
  flag_function_builder wflags;

  let main_ftype = L.function_type i32_t [| |] in
  let main_function = L.define_function "main" main_ftype the_module in
  let main_builder = L.builder_at_end context (L.entry_block main_function) in
  ignore (L.build_call createGame_func [||] "" main_builder);
  ignore (L.build_call flag_func_ptr [| |] "" main_builder);
  let (init_func_ptr, _) = StringMap.find "init" block_decls in
  ignore (L.build_call setInit_func [| init_func_ptr |] "" main_builder);
  if (StringMap.mem "turn" block_decls)
    then let (turn_func_ptr, _) = StringMap.find "turn" block_decls in
      ignore (L.build_call setTurn_func [| turn_func_ptr |] "" main_builder);
    else ();
  if (StringMap.mem "end" block_decls)
    then let (end_func_ptr, _) = StringMap.find "end" block_decls in
      ignore (L.build_call setEnd_func [| end_func_ptr |] "" main_builder);
    else ();
  ignore (L.build_call runGame_func [| |] "" main_builder);
  ignore (L.build_ret (L.const_int i32_t 0) main_builder);

  the_module
```

### 10.1.6 tiler.ml

```
(* Top-level of the Tiler compiler: scan & parse the input,
   check the resulting AST, generate LLVM IR, and dump the module *)

open Printf

module StringMap = Map.Make(String)

type action = Ast | LLVM_IR | Compile

let _ =
  let action =
  if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [
            ("-a", Ast);      (* Print the AST *)
            ("-l", LLVM_IR);  (* Generate LLVM, don't check *)
            ("-c", Compile) ] (* Generate, check LLVM IR *)
  else Compile in
  let (ic, oc, _) =
    let infile = Sys.argv.(2) in
    let i = String.rindex infile '.' in
    let executable = String.sub infile 0 i in
    let outfile = executable ^ ".ll" in
    (open_in infile, open_out outfile, executable)
  in
  let lexbuf = Lexing.from_channel ic in
  let ast = Parser.program Scanner.token lexbuf in
  Semant.check ast;
  match action with
    Ast -> print_string (Ast.string_of_program ast)
  | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate ast))
  | Compile -> let m = Codegen.translate ast in
    Llvm_analysis.assert_valid_module m;
    fprintf oc "%s" (Llvm.string_of_llmodule m);
    close_out oc
```

## 10.2 Runtime Library

### 10.2.1 TileGame.h

```
#pragma once

// object
struct Object {
       // tiler class type
       int classCode;

       // coordinates on grid
       int x, y;
       int isOnGrid;

       // sprite
       struct SDL_Surface *sprite;

       // attributes
       void *attr;
};

// grid
struct GameGrid {
       // dimensions of grid
       int width, height;

       // display rectangles
       struct SDL_Rect *disp_rects;

       // objects on grid
       struct Object **objs;
};
```

```
// game
struct TileGame {
	// window title
	char *title;

	// window dims
	int width, height;

	// background rgb
	int bg_r, bg_g, bg_b;

	// background surface
	struct SDL_Surface *bg;

	// is running
	int running, exit;
};

// game info
struct TileGame *game;

// grid
struct GameGrid *grid;

// function pointers
struct Blocks {
	// init block
	void(*init_ptr)();

	// turn block
	void(*turn_ptr)();

	// end block
	int(*end_ptr)();
};

// function pointers
struct Blocks *blocks;

// object reference node
struct ObjNode {
	struct Object *obj;
	struct ObjNode *next;
};

// object reference list
struct ObjRefList {
	struct ObjNode *head;
};
```

```
struct ObjRefList obj_refs;

// coordinate on grid
struct Coord {
	int x, y;
};

// mouse event info
struct MouseInfo {
	struct Coord coords;
	int pulled;
};
struct MouseInfo mouseInfo;
```

### 10.2.2 tiler-functs.c

```
#include <stdbool.h>
#include <stdlib.h>
#include "tiler-functs.h"

extern struct Blocks *blocks;

void setInit(void(*init)()) {
	blocks->init_ptr = init;
}

void setTurn(void(*turn)()) {
	blocks->turn_ptr = turn;
}

void setEnd(int(*end)()) {
	blocks->end_ptr = end;
}
```

### 10.2.3 tiler-grid.c

```
#include <stdio.h>
#include <stdlib.h>
#include "SDL.h"
#include "tiler-object.h"
#include "tiler-grid.h"

extern struct TileGame *game;
extern struct GameGrid *grid;

void createNullObjectOnGrid(int x, int y) {
	// create space for object
	struct Object *obj = (struct Object *) malloc(sizeof(struct Object));
	if (!obj) {
		fprintf(stderr, "malloc returned NULL");
```

```
                return;
        }

        // set to be null
        obj->classCode = 0;

        // set sprite to NULL
        obj->sprite = NULL;

        // set the attribute pointer to NULL
        obj->attr = NULL;

        // init to be on grid
        obj->x = x;
        obj->y = y;
        obj->isOnGrid = 1;

        // place on the grid
        int i = x + y * grid->width;
        grid->objs[i] = obj;
}

void createGrid(int width, int height) {
        // allocate space for grid struct
        grid = (struct GameGrid *) malloc(sizeof(struct GameGrid));
        if (!grid) {
                fprintf(stderr, "malloc returned NULL");
                return;
        }

        // initialize width and height
        grid->width = width;
        grid->height = height;

        // allocate space for object ptrs in grid
        int n = width * height;
        grid->objs = (struct Object **) malloc(n * sizeof(struct Object *));
        if (!grid->objs) {
                fprintf(stderr, "malloc returned NULL");
                return;
        }

        // initialize obj ptrs to NULL objects
        for (int i = 0; i < width; i++) {
                for (int j = 0; j < height; j++) {
                        createNullObjectOnGrid(i, j);
                }
        }

        // add display rectangles
```

```
	grid->disp_rects = (struct SDL_Rect *) malloc(n * sizeof(struct SDL_Rect));
	int w, h;
	w = game->width / width;
	h = game->height / height;
	struct SDL_Rect *cur;
	for (int i = 0; i < width; i++) {
		for (int j = 0; j < height; j++) {
			cur = &(grid->disp_rects[i + j*width]);
			cur->x = i*w;
			cur->y = j*h;
			cur->w = w;
			cur->h = h;
		}
	}
}


int gridWidth() {
	return grid->width;
}

int gridHeight() {
	return grid->height;
}

struct Object *removeGrid(int x, int y) {
	// get the object
	int i = x + y*grid->width;
	struct Object *obj = grid->objs[i];

	// remove from grid
	grid->objs[i] = NULL;

	if (obj->classCode) {
		// add back to list
		obj->isOnGrid = 0;
		addObject(obj);
	}
	else {
		// clean NULL object
		freeObject(obj);
	}

	return obj;
}

void setGrid(void *object, int x, int y) {
	struct Object *obj = (struct Object *) object;

	// get the object to place
	struct Object *target;
	if (obj->isOnGrid) {target = removeGrid(obj->x, obj->y);}
	else {target = removeObject(obj);}
	if (!target) {
		fprintf(stderr, "Untracked Object!");
		return;
	}

	// replace
	removeGrid(x, y);
	int i = x + y*grid->width;
	grid->objs[i] = target;

	// change members
	target->isOnGrid = 1;
	target->x = x;
	target->y = y;
}

void clearGrid(int x, int y) {
	// take off
	removeGrid(x, y);

	// replace with NULL object
	createNullObjectOnGrid(x, y);
}

void cleanGrid() {
	int n = grid->width*grid->height;
	for (int i = 0; i < n; i++) {
		freeObject(grid->objs[i]);
	}
}

void *getGrid(int x, int y) {
	int i = x + y*grid->width;
	return (void *) grid->objs[i];
}

int isEmpty(int x, int y) {
	struct Object *obj = (struct Object *) getGrid(x, y);
	return !obj->classCode;
}

void destroyGameGrid() {
	cleanGrid();
	free(grid->objs);
	free(grid->disp_rects);
	free(grid);
}
```

## 10.2.4 tiler-main.c

```
#include <stdio.h>
#include <string.h>
#include <setjmp.h>
#include "SDL.h"
#include "tiler-object.h"
#include "tiler-grid.h"
#include "tiler-mouse.h"
#include "tiler-main.h"

#include <stdlib.h>
//#include <crtdbg.h>

// global values
extern struct TileGame *game;
extern struct GameGrid *grid;
extern struct Blocks *blocks;
extern struct ObjRefList obj_refs;
extern struct MouseInfo mouseInfo;

void createGame() {
	game = (struct TileGame *)malloc(sizeof(struct TileGame));
	if (!game) {
		fprintf(stderr, "malloc returned NULL");
		exit(0);
	}

	// init title
	char *def = "Game";
	size_t n = strlen(def) + 1;
	game->title = (char *)malloc(n);
	if (!game->title) {
		free(game);
		fprintf(stderr, "malloc returned NULL");
		exit(0);
	}
	memcpy(game->title, def, n);

	// init window size
	setWindow(640, 480);

	// init bg color
	setBackgroundColor(255, 255, 255); // white

	// set background to NULL
	game->bg = NULL;

	// set running to false
	game->running = 0;
	game->exit = 0;

	// create the function pointers
	blocks = (struct Blocks *) malloc(sizeof(struct Blocks));
	if (!blocks) {
		SDL_FreeSurface(game->bg);
		free(game->title);
		free(game);
		fprintf(stderr, "malloc returned NULL");
		exit(0);
	}
	blocks->init_ptr = NULL;
	blocks->turn_ptr = NULL;
	blocks->end_ptr = NULL;

	// initialize empty object reference list
	obj_refs.head = NULL;

	// initialize mouse info
	mouseInfo.pulled = 0;
}


void setTitle(const char *str) {
	// free old title
	if (game->title) {
		free(game->title);
	}

	// allocate memory for new title
	size_t n = strlen(str) + 1;
	game->title = (char *)malloc(n);
	if (!game->title) {
		fprintf(stderr, "malloc returned NULL");
		return;
	}

	// copy string
	memcpy(game->title, str, n);
}


void setWindow(int width, int height) {
	game->width = width;
	game->height = height;
}


void setBackgroundColor(int red, int green, int blue) {
	game->bg_r = red;
```

```
        game->bg_g = green;
        game->bg_b = blue;
}


void setBackground(char *filepath) {
        game->bg = SDL_LoadBMP(filepath);
        if (!game->bg) {
                fprintf(stderr, "SDL_LoadBMP Error: %s\n", SDL_GetError());
                return;
        }
}

void draw(SDL_Renderer *renderer) {
        // load background texture
        SDL_Texture *back_tex = NULL;
        if (game->bg) {
                back_tex = SDL_CreateTextureFromSurface(renderer, game->bg);
                if (!back_tex) {
                        fprintf(stderr, "Background Texture Error: %s\n", SDL_GetError());
                }
        }

        // load object textures
        int n = grid->width*grid->height;
        SDL_Texture **obj_tex = (SDL_Texture **) malloc(n*sizeof(SDL_Texture *));
        for (int i = 0; i < n; i++) {
                obj_tex[i] = NULL;
                if (grid->objs[i]->sprite) {
                        obj_tex[i] = SDL_CreateTextureFromSurface(renderer,
grid->objs[i]->sprite);
                        //printf("Loaded Object Sprite");
                }
        }

        // clear back buffer
        SDL_RenderClear(renderer);

        // draw to background img to buffer
        if (back_tex) { SDL_RenderCopy(renderer, back_tex, NULL, NULL); }

        // draw object imgs to buffer
        for (int i = 0; i < n; i++) {
                if (grid->objs[i] && obj_tex[i]) {
                        SDL_RenderCopy(renderer, obj_tex[i], NULL, &(grid->disp_rects[i]));
                }
        }

        // draw buffer to screen
        SDL_RenderPresent(renderer);
```

```
	// free draw resources
	if (back_tex) { SDL_DestroyTexture(back_tex); }
	for (int i = 0; i < n; i++) {
		if (obj_tex[i]) {SDL_DestroyTexture(obj_tex[i]);}
	}
	free(obj_tex);
}


jmp_buf funcBuf;

void exitFunction() {
	longjmp(funcBuf, 1);
}

static int gameLoop(void *vargp) {
	//_CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_DEBUG);

	SDL_Renderer *renderer = (SDL_Renderer *)vargp;

	// set background color
	SDL_SetRenderDrawColor(renderer, game->bg_r, game->bg_g, game->bg_b, 255);

	// run init
	if (!setjmp(funcBuf)) {
		blocks->init_ptr();
	}

	// game loop
	while (game->running && !game->exit) {
		// draw
		draw(renderer);

		// check if end condition already met
		if (blocks->end_ptr) {
			if (!setjmp(funcBuf)) {
				game->exit = blocks->end_ptr();
			}
		}
		else {
			game->exit = 0;
		}


		// perform turns
		while (game->running && !game->exit) {
			// call turn
			if (blocks->turn_ptr) {
				if (!setjmp(funcBuf)) {
```

```
                                        blocks->turn_ptr();
                                }
                        }
                        cleanObjects();

                        // update draw
                        draw(renderer);

                        // test end
                        if (game->running && !game->exit) {
                                if (blocks->end_ptr) {
                                        if (!setjmp(funcBuf)) {
                                                game->exit = blocks->end_ptr();
                                        }
                                }
                        }
                }
        }

        printf("Game over.\n");

        //destroyGame();
        //_CrtDumpMemoryLeaks();

        return 0;
}


void runGame() {
        // initialize SDL
        if (SDL_Init(SDL_INIT_VIDEO)) {
                fprintf(stderr, "SDL_Init Error: %s\n", SDL_GetError());
                return;
        }

        // create window
        SDL_Window *window = SDL_CreateWindow(game->title,
                SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
                game->width, game->height, SDL_WINDOW_SHOWN | SDL_WINDOW_MOUSE_FOCUS);
        if (!window) {
                fprintf(stderr, "SDL_CreateWindow Error: %s\n", SDL_GetError());
                SDL_Quit();
                return;
        }

        // create renderer
        SDL_Renderer *renderer = SDL_CreateRenderer(window, -1,
                SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
        if (!renderer) {
                SDL_DestroyWindow(window);
```

```
		fprintf(stderr, "SDL_CreateRenderer Error: %s\n", SDL_GetError());
		SDL_Quit();
		return;
	}

	// call game logic thread
	game->running = 1;
	SDL_Thread *thread;
	int trv;
	thread = SDL_CreateThread(gameLoop, "game loop", (void *)renderer);

	while (game->running) {
		// event processing
		SDL_Event event;
		while (SDL_PollEvent(&event)) {
			if (event.type == SDL_QUIT) {
				game->running = 0;
			}
			else if (event.type == SDL_MOUSEBUTTONDOWN) {
				// compute the coords on the grid
				int x, y;
				x = event.button.x/(game->width/grid->width);
				y = event.button.y/(game->height/grid->height);
				setMouseInfo(x, y);
			}
		}
	}

	// wait for game thread to complete
	SDL_WaitThread(thread, &trv);

	// release memory
	destroyGame();
	SDL_DestroyRenderer(renderer);
	SDL_DestroyWindow(window);
	SDL_Quit();
}


void closeGame() {
	printf("%s\n", "The game window was successfully launched ...");
	game->running = 0;
	printf("%s\n", "The game window has been exited. Program exited.");
}


void destroyGame() {
	// grid
	if(grid) {
		destroyGameGrid();
	}

	// blocks
	free(blocks);

	// game
	SDL_FreeSurface(game->bg);
	free(game->title);
	free(game);
}
```

### 10.2.5 tiler-mouse.c

```
#include "tiler-main.h"
#include "tiler-mouse.h"

extern struct MouseInfo mouseinfo;
extern struct TileGame *game;

void setMouseInfo(int x, int y) {
	mouseInfo.coords.x = x;
	mouseInfo.coords.y = y;
	mouseInfo.pulled = 0;
}

void pullMouseInfo(struct Coord *tuple_ptr) {
	tuple_ptr->x = mouseInfo.coords.x;
	tuple_ptr->y = mouseInfo.coords.y;
	mouseInfo.pulled = 1;
}

void getMouseCoords(void *tuple_ptr) {
	mouseInfo.pulled = 1;
	while (mouseInfo.pulled) {
		if (!game->running) {
			exitFunction();
		}
	}

	pullMouseInfo((struct Coord *) tuple_ptr);
}
```

### 10.2.6 tiler-object.c

```
#include <stdio.h>
#include "SDL.h"
#include "tiler-object.h"

// global variables
extern struct ObjRefList obj_refs;
```

```
void addObject(struct Object *obj) {
        // allocate space for new node
        struct ObjNode *node = (struct ObjNode *) malloc(sizeof(struct ObjNode));
        if (!node) {
                fprintf(stderr, "malloc returned NULL");
                return;
        }

        // set node values
        node->obj = obj;
        node->next = obj_refs.head;
        obj_refs.head = node;
}

int listIsEmpty() {
        return obj_refs.head == NULL;
}

struct Object* removeHead() {
        struct ObjNode *node = obj_refs.head;
        obj_refs.head = obj_refs.head->next;
        struct Object *obj = node->obj;
        free(node);
        return obj;
}

struct Object *removeObject(struct Object *obj) {
        // check for empty list
        if (listIsEmpty()) {
                return NULL;
        }

        // if object is head of list
        struct ObjNode *cur = obj_refs.head;
        if (cur->obj == obj) {
                return removeHead();
        }

        // object down list
        struct ObjNode *last = cur;
        cur = cur->next;
        while (cur != NULL && cur->obj != obj) {
                cur = cur->next;
                last = last->next;
        }


        if (cur != NULL) {
                struct Object *result = cur->obj;
                last->next = cur->next;
                free(cur);
                return result;
        }
        else { // not found
                return NULL;
        }
}

void* createObject(void *attr) {
        // create space for object
        struct Object *obj = (struct Object *) malloc(sizeof(struct Object));
        if (!obj) {
                fprintf(stderr, "malloc returned NULL");
                return NULL;
        }

        // set to be not null
        obj->classCode = 1;

        // set sprite to NULL
        obj->sprite = NULL;

        // init not on grid
        obj->isOnGrid = 0;
        addObject(obj);

        // set the pointer to attribute struct
        obj->attr = attr;

        return (void *) obj;
}

void *createNullObject() {
        // create space for object
        struct Object *obj = (struct Object *) malloc(sizeof(struct Object));
        if (!obj) {
                fprintf(stderr, "malloc returned NULL");
                return NULL;
        }

        // set to be null
        obj->classCode = 0;

        // set sprite to NULL
        obj->sprite = NULL;

        // init not on grid
        obj->isOnGrid = 0;
        addObject(obj);
        // set the attribute pointer to NULL
        obj->attr = NULL;

        return (void *) obj;
}

void setSprite(void *object, char *sprite_filepath) {
        struct Object *obj = (struct Object *) object;

        // set the sprite
        obj->sprite = SDL_LoadBMP(sprite_filepath);
        if (!obj->sprite) {
                fprintf(stderr, "SDL_LoadBMP Error: %s\n", SDL_GetError());
        }
}

int getX(void *object) {
        struct Object *obj = (struct Object *) object;
        return obj->x;
}

int getY(void *object) {
        struct Object *obj = (struct Object *) object;
        return obj->y;
}

void *getAttr(void *object) {
        struct Object *obj = (struct Object *) object;
        return obj->attr;
}

int isNull(void *object) {
        struct Object *obj = (struct Object *) object;
        return !obj->classCode;
}

int getType(void *object) {
        struct Object *obj = (struct Object *) object;
        return obj->classCode;
}

void freeObject(struct Object *obj) {
        if (obj->attr) { free(obj->attr); }
        SDL_FreeSurface(obj->sprite);
        free(obj);
}

void cleanObjects() {
        while (obj_refs.head) {
                freeObject(removeHead());
```

```
       }
}
```

### 10.2.7 tiler.h

```
#pragma once

#include "tiler-main.h"
#include "tiler-functs.h"
#include "tiler-grid.h"
#include "tiler-object.h"
#include "tiler-mouse.h"
```

## 10.3 Tests

### 10.3.1 helloworld.tile

```
// single line comment

init {
       tile(3, 3);
       background("./sprites/hello.bmp");
}
```

### 10.3.2 helloworld2.tile

```
int x;

init {
   int y;
   int z;

   bool c;
   bool d;

   tile(3, 3);
   background("./sprites/hello.bmp");

   x = 2;
   y = x + 1;
   iprint(y);
   sprint("i love jacky");
   c = true;
   d = !c;
   iprint(d);
   iprint(c);

   // prints 111
   if (d) {
      iprint(000);
   } else if (c) {
      iprint(111);
   }

   // prints 1...10
   for (x=0; x<11; x=x+1) {
      iprint(x);
   }

   // does not print anything
   y = 1;
   while (y < 1) {
      iprint(y);
      y = y + 1;
   }

   // prints 10
   z = 10;
   do {
      iprint(z);
      z = z + 1;
   } while (z < 10);
}

turn {
   int g;
   g = 55;
   iprint(g);
}
```

### 10.3.3 helloworld3.tile

```
#size 500 250
#title "hello world 3"

int x; int y;
coord mouse;
float a; float b;

class Click {
   attr: string cool;
   attr: int bool1;
   attr: string bool2;
}

int add(int a, int b) {
   int x;
   x = 100;
   return a+b+x;
}

init {
   <Click> idk;
   coord point;

   tile(3, 3);
   background("./sprites/hello.bmp");

   x = 1;
   y = 2;

   idk = new Click("yes", 10, "21");
   idk.bool2 = "idk";
   sprint(idk.cool);
   iprint(idk.bool1);
   sprint(idk.bool2);

/*
   point = [0, 9];
   x = point[x];
   iprint(x);
   iprint(point[y]);
*/
   iprint(add(1, 2));
   iprint(7%2);
   a = 3.7;
   b = 2.2;
   fprint(a+b);
}

turn {
   mouse = capture();
   iprint(mouse[x]);
   iprint(mouse[y]);
}
```

### 10.3.4 helloworld4.tile

```
// single line comment

#size 500 250
#color 0 0 255
#title "best game in the whole world"

int num;
string glob;
coord mouse;
```

```
class GridObj {
	attr: bool ok;
}

init {
	<GridObj> obj1;
	<GridObj> obj2;
	<GridObj> obj3;
	string loc;
	tile(3, 3);
	background("./sprites/tictactoe_board.bmp");

	obj1 = new GridObj(true);
	setSprite(obj1, "./sprites/x.bmp");

	obj2 = new GridObj(false);
	obj2 = obj1;
	iprint(obj2.ok);

	num = 5;
	glob = "This is a global string!";
	loc = "this is a local string!";
	sprint(glob);
	sprint(loc);

	grid[1,1] = obj1;

	iprint(isNull(grid[1, 1]));
	obj2 = grid[1,1];
	iprint(isNull(obj2));
	iprint(isNull(obj1));

	iprint(gridw);
	iprint(gridh);

	obj3 = new GridObj(false);
	setSprite(obj3, "./sprites/o.bmp");
	grid[2, 2] = obj3;

	iprint(obj3.x);
	iprint(obj3.y);
}

turn {
	mouse = capture();
	iprint(mouse[x]);
	iprint(mouse[y]);
	sprint(glob);

	grid[1,1] = NULL;
```

```
        iprint(isNull(grid[1,1]));
}
```

### 10.3.5 Listing of All Test Cases

```
// Testing arithmetic operations

int x;

init {
        bool b;
        tile(3,3);
        x = -------9;
        iprint(x);

        b = !!!!!!true;
        iprint(b);
        close();
}


// Testing arithmetic operations

init{
        int a;
        int b;
        int c;
        tile(3,3);

        a = 1;
        b = 2;
        c = 3;
        iprint(a+b/c);
        iprint(b-c);
        iprint(a*c);
        iprint(c*c*c*c*c);
        iprint(b%c);
        close();
}


// This tests that assignment is generally working for int, float, string, bool

init {
        // Type declarations
        int a;
        int b;
        int c;

        string i;
        string j;

        float m;
        float n;

        bool x;
        bool y;

        tile(3,3);

        // Assignment of variables
        a = 42 - 17;
        b = 6 * 3;
        c = b + 64;
        m = 1.12341;
        n = 3.180;
        i = "It's a me Mario";
        j = "Oh hello, Luigi";
        x = true;
        y = !(!x);

        iprint(a);
        iprint(b);
        iprint(c);

        sprint(i);
        sprint(j);

        fprint(m);
        fprint(n);

        if (x) {sprint("true");} else {sprint("false");}
        if (y) {sprint("true");} else {sprint("false");}

        close();

}


// This tests that the do while loop works as expected

init {
        int a;
        int b;

        tile(3,3);
        a = 100;
        b = 4;

        do {
               sprint("Mama Mia~");
        } while(a < 1);

        do {
               iprint(b);
               b = b - 1;
        } while (b > 0);

        close();

}


// tests that end block works as expected

int count;
init{
        count = 0;
        tile(3,3);
}


turn{
        iprint(count);
        count = count + 1;
}


end{
        if (count > 2) {
               close();
               return 1;
        }
}


// testing that float arithmetic between floats works

init {
        float x;
        tile(3,3);
        x = 3.0 + 7.5 * 2.6;
        fprint(x);
        close();
}
```

```
// testing that float arithmetic between floats and ints works

init {
	float x;
	int y;
	tile(3,3);
	y = 7;
	x = 3.0 + 4 * 2.6 + 7;
	fprint(x);
	close();
}


// testing that float comparison between floats and ints works

init {
	float x;
	tile(3,3);
	x = 3.0;
	fprint(x);
	close();
}


// testing that float comparison between floats works

init {
	bool x;
	tile(3,3);
	x = 3.0 < 7.0;
	iprint(x);
	close();
}


// testing that float comparison between floats and ints works

init {
	bool x;
	tile(3,3);
	x = (3.0 < 7);
	iprint(x);
	close();
}


// testing that recursive programmer-defined functions work

int fib(int x) {
        if (x <= 1) {
                return x;
        }
        return fib(x-1) + fib(x-2);
}

init {
        tile(3,3);
        iprint(fib(5));
        close();
}


// testing that function calls have static scope

int a;
int b;

init {
   a = 5;
   b = 6;
   iprint(add(1, 2));
   close();
}

int add(int a, int b) {
   int x;
   x = 100;
   return a+b+x;
}


// This tests that global variables can be assigned locally and updated

int x;

init {
   tile(3,3);
   x = 43;
   iprint(x);
   x = 21;
   iprint(x);
   close();
}


// This tests that global variables that have values assigned in one block can still be
accessed in another block

int x;
int count;

init {
   tile(3,3);
   x = 77;
   count = 0;
   iprint(x);
}

turn {
       iprint(x);
       count = count + 1;

}

end {

       if (count >= 1) {
              close();
              return 1;
       }
}



// This tests that global strings that have values assigned in one block can still be accessed
in another block

string str;
int count;

init {
   tile(3,3);
   str = "test string";
   count = 0;
   sprint(str);
}

turn {
       sprint(str);
       count = count + 1;

}

end {

       if (count >= 1) {
              close();
              return 1;
       }
}



// Tests that the most basic helloworld program can be executed

init {
   tile(3, 3);
   close();
}


// This tests that the various if control flow works as expected

init {
       bool x;
       bool y;

       x = true;
       y = !x;

       tile(3,3);

       if(x){
              sprint("Twinkle, twinkle");
       }

       if(y){
              sprint("Lalala, lalala, Elmo's World");
       }

       if(!y){
              sprint("Little star");
       }

       if(3 > 2){
              sprint("How I wonder what you are");
       }

       if((1+1) == 3){
              sprint("Bow chika wow wow");
       }

       close();

}


// Tests that object attributes acn be accessed
```

```
class Click {
   attr: string str;
   attr: int a;
   attr: string b;
}

init {
   <Click> idk;
   coord point;

   tile(3, 3);

   idk = new Click("yes", 10, "21");
   sprint(idk.str);
   iprint(idk.a);
   sprint(idk.b);

   close();
}


// tests that object attributes can be re-assigned to

class Click {
   attr: string str;
   attr: int a;
   attr: string b;
}

init {
   <Click> idk;
   coord point;

   tile(3, 3);

   idk = new Click("yes", 10, "21");
   idk.a = 77;
   iprint(idk.a);

   close();
}


// tests that booleans can be printed (as ints)

init {
       bool x;
       tile(3,3);
       x = true;
       iprint(x);
```

```
        close();
}


// This tests that print is able to handle expressions

init {
        int x;
        int y;
        float a;
        float b;

        tile(3,3);
        x = 1;
        y = 1;
        a = 3.0;
        b = 4.0;
        iprint(x*y/y-y);
        fprint(b/a);
        close();
}


// tests that floats can be printed

init {
        float x;
        tile(3,3);
        x = 3.3;
        fprint(x);
        close();
}


// tests that ints can be printed

init {
        int x;
        tile(3,3);
        x = 3;
        iprint(x);
        close();
}


// tests that strings can be printed

init {
        string x;
        tile(3,3);
        x = "3.3";
        sprint(x);
        close();
}

// tests that turn block works as expected

int count;

init{
        tile(3,3);
        count = 0;
}

turn{
        iprint(count);
        if (count > 2){
                close();
        }
        count = count + 1;
}

// This tests that the while loop is working as expected

init {
        int x;
        int y;

        tile(3,3);

        x = 7;
        while (x > 0){
                iprint(x);
                x = x -1;
        }

        y = 100;
        while(y < 1){
                sprint("While loop should not have been entered?!?!");
        }

        close();
}
```

## 10.5 Fail Cases

```
// testing that invalid arithmetic operations fail at parser

init{
        grid{3,3};
        int y;
        y = 3/*4;
        iprint(y);
        close();
}

// testing that arithmetic not involving ints/floats dies at semantic checker

bool b;
int x;
int y;
init {
        tile(3,3);
        b = true;
        x = 3;
        y = x + b;
        close();
}

// checks that duplicate end blocks are not allowed

init{
        tile(3,3);
}
end{

}
end{

}

//  Test that the program fails if there are duplicate global variables

int a;
int a; // Duplicate global a

init {

        tile(3,3);

        a = 5;

}


turn {}

end {}

// tests that duplicate turn blocks are not allowed
```

```
init{
       tile(3,3);
}
turn{

}
turn{

}

// Tests error message when functions have incorrect number of arguments

int a;
int b;

init {
   a = 5;
   b = 6;
   iprint(add(2));
   close();
}

int add(int a, int b) {
   int x;
   x = 100;
   return a+b+x;
}
// Tests error message when incorrect if predicate type

init{
       tile(3,3);
       if (3){
              iprint("happy birthday monica !");
       }
}

// Tests that program fails if init is missing

// "init" is missing at this locations
{
   grid(3, 3);
}
// Tests that program fails at parser if undeclared variable used

init{
       tile(3,3);
       y = 30;
       close();
}
```

```
commit cc191bd449d26fd45c95de0ec77fa81e2eb7733c
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Dec 20 14:28:14 2017 -0500

    fixed calc print statements

commit e4104e90c53b4b6ad3a81e8f57f31f4ab8218a59
Author: Monica <monicasyting@gmail.com>
Date:   Wed Dec 20 14:13:02 2017 -0500

    cleaned up scanner/parser

commit ee2fabe33fef2d859295e05b3b41a67a8d6a45dd
Author: Monica Ting <monicasyting@gmail.com>
Date:   Tue Dec 19 18:13:25 2017 -0500

    Update README.md

commit 5f819422deca7ee1f439c50fddf8965154992c0d
Merge: fcfeedc 7fcd741
Author: Monica <monicasyting@gmail.com>
Date:   Tue Dec 19 10:27:05 2017 -0500

    changed naming in tictactoe

commit fcfeedc717aef2766a66c4e35cfcf7c61f8b12d0
Author: Monica <monicasyting@gmail.com>
Date:   Tue Dec 19 10:25:39 2017 -0500

    changed naming in tictactoe

commit 7fcd741d53d26bd3d952525f0677fa72c8350efa
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Dec 19 04:50:38 2017 -0500

    finished burger calculator

commit ccc5d1a8e9f2c57ae84dbeeb567bb72a7d25c217
Author: Monica <monicasyting@gmail.com>
Date:   Tue Dec 19 02:06:35 2017 -0500

    tictactoe working, calculator in progress
```

```
commit b3ba5471341e12d417374e0b1a7ac9d97297bb9e
Merge: 5030504 73063de
Author: Jacky Cheung <Jacky96Cheung@gmail.com>
Date:   Tue Dec 19 09:33:59 2017 -0500

    Merge pull request #73 from jason-lei/testing-prints

    fixing testing print messages v2

commit 73063de04e02a30ef717ba010eacebfeaddcd8df
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Dec 19 09:27:28 2017 -0500

    fixing testing print messages v2

commit a2615d13761ffd55578670e8d1964ab3196a8815
Merge: 0bd8123 63821a6
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Dec 19 04:53:23 2017 -0500

    fixed merge conflicts

commit 0bd8123166bfe6ad4bce59546dcdf014e3ac2d5e
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Dec 19 04:50:38 2017 -0500

    finished burger calculator

commit db92bbcda90173eeac7e2f8db96e7b0a8d8c350e
Author: Monica <monicasyting@gmail.com>
Date:   Tue Dec 19 02:06:35 2017 -0500

    tictactoe working, calculator in progress

commit 5030504d8b44680cab6cb220cc78f2b35235e21a
Merge: 4f6648c 45a5d98
Author: JY <jt2823@columbia.edu>
Date:   Tue Dec 19 03:50:17 2017 -0500

    Merge pull request #70 from jason-lei/testing

    Testing
```

```
commit 4f6648c8073f9c00a1524f3815fc7d6b03aa81f2
Merge: 4004566 aa0d7b0
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Tue Dec 19 02:30:54 2017 -0500

    Merge pull request #72 from jason-lei/xny

    Xny

commit aa0d7b08b411564dc654f89e2f094411bad0e77a
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Dec 19 02:26:10 2017 -0500

    added testing print statements back

commit 9a034c5f1bb070c1e79ee2c5195ef72735b56f30
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Tue Dec 19 02:17:32 2017 -0500

    fixed bug - object list removal

commit 63821a64a1849b0705e9d7bfca366a809358a196
Author: Monica <monicasyting@gmail.com>
Date:   Tue Dec 19 02:06:35 2017 -0500

    tictactoe working, calculator in progress

commit b82d431ab806bd07bf9df2ea380d65b2088f138e
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Tue Dec 19 02:03:23 2017 -0500

    fixed bug - nonsquare grids

commit 45a5d982d106aff6bd6661331ac19233e2992950
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Dec 19 01:49:15 2017 -0500

    added new test cases

commit 4004566c40c5ce30cf68ed4f35e70934b0702fc7
Merge: 472c077 75224e8
Author: Monica Ting <monicasyting@gmail.com>
Date:   Mon Dec 18 22:54:47 2017 -0500
```

```
    Merge pull request #69 from jason-lei/grid-funcs

    added remaining grid functionality

commit 75224e8634f2239b033948fcd8c4b2abf997ef05
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Mon Dec 18 21:20:56 2017 -0500

    get x and y in runtime lib

commit c6876e5021c3632505466181f7f34970221ed10a
Author: Monica <monicasyting@gmail.com>
Date:   Mon Dec 18 22:43:45 2017 -0500

    testing with helloworld4

commit 1a1590b83699a61062db4dde59afb3debf8788f8
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Mon Dec 18 21:20:56 2017 -0500

    get x and y in runtime lib

commit 5d6cabc601eb8118e4253122cc81b604f6e60cb3
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Mon Dec 18 20:33:46 2017 -0500

    added remaining grid functionality

commit b85b2998a47380566b25566cf920997f0b952bd2
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Mon Dec 18 18:18:01 2017 -0500

    arithmetic tests

commit 7e4378a83a140fada43d9f59ec5ebc14e4ccf6ee
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Mon Dec 18 17:06:23 2017 -0500

    added global string test

commit 472c07777f4e2399e4c0bcb359c67b46f3c89aaf
Merge: e8225f7 12abd60
```

```
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Mon Dec 18 18:22:12 2017 -0500

    Merge pull request #64 from jason-lei/add-class

    class declaration, instantiation, and attribute access

commit 12abd60c5f761bfc2a7f6f0a34d878c4f01d66ec
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Mon Dec 18 18:15:24 2017 -0500

    merge with master is fully functional

commit 696af298319bcbda071e79180536151283c96812
Merge: 564cfa0 e8225f7
Author: JY <jt2823@columbia.edu>
Date:   Mon Dec 18 17:04:28 2017 -0500

    Merge branch 'master' into add-class

commit 564cfa051ada33b22a08cb4e7950e53a11597171
Merge: 7780d65 1441644
Author: JY <jt2823@columbia.edu>
Date:   Mon Dec 18 17:02:01 2017 -0500

    Merge branch 'master' into add-class

commit e8225f79841895b62c3df2751481750e508b23e7
Merge: 3259d8e de69154
Author: Monica Ting <monicasyting@gmail.com>
Date:   Mon Dec 18 16:54:28 2017 -0500

    Merge pull request #63 from jason-lei/global-strings

    fixing global strings. need a way to allow programmers to alloc
memor…

commit de69154488670683d6b9adcfe79ddcb813f498a6
Author: Monica <monicasyting@gmail.com>
Date:   Mon Dec 18 16:51:19 2017 -0500

    fixed parsing
```

```
commit e1ec5fdc806329ca1cdcd67fa392ffc1c0b202b0
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Mon Dec 18 15:49:46 2017 -0500

    fixing global strings. need a way to allow programmers to alloc
memory for strings inside blocks

commit 3259d8e53761b693c5ba3677a0d34538f1373d2e
Merge: 277a8e1 e9edbe1
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Mon Dec 18 16:46:19 2017 -0500

    Merge pull request #62 from jason-lei/float-ops

    added float operators and modulus

commit e9edbe1a21badfd975c09a2b8485bc4db454b1e8
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Mon Dec 18 16:44:45 2017 -0500

    added float arithmetic example to helloworld3

commit b0119972d934790eb71015b54abf8469c176625a
Author: Monica <monicasyting@gmail.com>
Date:   Mon Dec 18 14:51:32 2017 -0500

    added float operators and modulus

commit 277a8e1b9835133f98427340bca50405beac06f6
Merge: 1441644 edd76fe
Author: Monica Ting <monicasyting@gmail.com>
Date:   Mon Dec 18 16:39:09 2017 -0500

    Merge pull request #66 from jason-lei/null-obj

    Null obj

commit edd76fe04ce466f0de53b8601d809c31a4aca880
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Mon Dec 18 16:18:11 2017 -0500

    fixed test output messages
```

```
commit 7780d651be7dfa4766eb383824ce6dbc38c713df
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Mon Dec 18 16:00:34 2017 -0500

    class decl, class obj access, and obj instantiation works WHOOO

commit 22a494cf3a1333b11a2bc91e189d5e1fe8d1a33b
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Mon Dec 18 15:46:44 2017 -0500

    null object function

commit 14416443329146cdc374f917607eec04128cf238
Merge: 60893d4 eb39f3e
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Mon Dec 18 01:15:05 2017 -0500

    Merge pull request #60 from jason-lei/functions-2

    Functions 2

commit eb39f3e7b7864fd3f6d624adab747d50f30ea8a6
Author: Monica <monicasyting@gmail.com>
Date:   Mon Dec 18 00:27:46 2017 -0500

    cleaned up functions - now treating functions and blocks the same

commit c7c9e8ce5ebe8d9f9b8a78f7ed881dec7aa302ac
Merge: 600ae6d 2028baa
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Sun Dec 17 22:13:24 2017 -0500

    stupid merge commit after rebasing where i merge this branch into
itself

commit 600ae6d7f4b80681160c85c0385b57f98db5a140
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Sun Dec 17 22:01:17 2017 -0500

    classes almost done, just need to wrap up assign

commit 5cd7710869601eb5d09a7de5c8a554dbc45d47b6
Author: Monica <monicasyting@gmail.com>
```

```
Date:   Sun Dec 17 20:54:24 2017 -0500

    added building functions to codegen

commit 60893d459d60283ec617c4e42810249f6218787d
Merge: 219a7c8 7be8b58
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Sun Dec 17 18:48:05 2017 -0500

    Merge pull request #59 from jason-lei/functions-2

    Functions 2

commit 7be8b580ca04ff6bb13a95ec4779b8a7f96ddd52
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sun Dec 17 18:46:45 2017 -0500

    added semant back into tiler.ml

commit 82972964e59beeff06546c99ff61f9d464d8a3d3
Author: Monica <monicasyting@gmail.com>
Date:   Sun Dec 17 18:35:41 2017 -0500

    fixed leftover merge conflict

commit a9c4438e1dd0f9c820172e5c6a655754b5a69c58
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sun Dec 17 17:51:45 2017 -0500

    Allow us to definie function in the language, but they currently
do not do anything. Just framework

commit f6a9fa479a71c46320bd74f06e03720afd3c88c2
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sun Dec 17 14:25:53 2017 -0500

    Function block is registered, but no different from any other
block currently

commit 219a7c841f5cf8e62c22b33c07c03b4c41bee08e
Merge: 6b00e94 37dc579
Author: Monica Ting <monicasyting@gmail.com>
Date:   Sun Dec 17 17:49:45 2017 -0500
```

```
    Merge pull request #55 from jason-lei/lib-functions

    Lib functions

commit 37dc579f324aadf9a7e0e569bae0d1337cb081e2
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sun Dec 17 17:46:31 2017 -0500

    updated semant input args. should do more checking on flags

commit 6f9a3b2211305a15b2195b847e17c7dcc5ad2159
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sun Dec 17 17:42:01 2017 -0500

    updated codegen lib functions

commit 6a000e9376bb792d8c98a98b638893cb134e464e
Author: Monica <monicasyting@gmail.com>
Date:   Sun Dec 17 13:49:51 2017 -0500

    added all static library functions to codegen

commit 953a9e1e1f6b727bfc050d6b35c4b3d09445bbb9
Author: Monica <monicasyting@gmail.com>
Date:   Sun Dec 17 00:33:02 2017 -0500

    added basic version of chess and tictactoe test games

commit 0d5e9194ab81c1723ba314f7ac8f098f209729a8
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 16 23:32:36 2017 -0500

    added sprites folder

commit 6b00e94c5a8f8c2d23a958b83366c558227097ea
Merge: cd84f9a 4893ae8
Author: Jacky Cheung <Jacky96Cheung@gmail.com>
Date:   Sun Dec 17 16:15:47 2017 -0500

    Merge pull request #54 from jason-lei/adding-flags

    added window flags
```

```
commit 4893ae807c1b90edfcc640aec6aa433784db1695
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sun Dec 17 15:54:43 2017 -0500

    added window flags

commit cd84f9af99b319e4bc88491c6f327a68e63932f4
Merge: 649ef7c e69be94
Author: Monica Ting <monicasyting@gmail.com>
Date:   Sun Dec 17 14:23:02 2017 -0500

    Merge pull request #50 from jason-lei/semantics2

    Semantics2

commit b68456122316a31bffdb81c4544be66715945732
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Sat Dec 16 20:19:39 2017 -0500

    classes in codegen WIP. need to restructure much of it to keep
track of class decls, vdecls, and fdecls separately

commit 77ae19d424aabbf9c5198815e63c9980995d2aa6
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Thu Dec 14 07:23:14 2017 -0500

    fixed reduce/reduce conflicts; took out rules from parser for now

commit 0ecf09d334dc1c2a9e252f70b0b33267665cbb11
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Thu Dec 14 07:19:18 2017 -0500

    tried to clean up some reduce/reduce conflicts

commit a0beb0e9ae046b3ce8dbb1d002b20c36406f7d9c
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Wed Dec 13 17:30:51 2017 -0500

    finished up classes in ast parser and scanner, need to do codegen
now

commit 3da546b8f38149ff717ce9f4cc523fe16e5b93a3
```

```
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Fri Dec 8 19:14:09 2017 -0500

    added a huge chunk of class to parser and ast. still a few ends
that have to be wrapped up here

commit e69be9499efb9236076402c58da7daf5f9e96f4c
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sun Dec 17 00:33:34 2017 -0500

    Fully automated testing, nomore random popup (might have memory
leaking everywhere tho since I am calling exit(0)). Test outputs
updated accordingly

commit ae6d1b120e3b5c011bfdaedf109e60e4016e505e
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sun Dec 17 00:25:31 2017 -0500

    updated the error output so that test is performed correctly

commit 649ef7c65ae603b0f4e6669c65c4cc5d91f3ee0e
Merge: 63a8e1c ca38a9e
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Sat Dec 16 23:26:53 2017 -0500

    Merge pull request #49 from jason-lei/add-coordinate

    Add coordinate

commit 2028baa26391cec736fb373f7ee3e9daccefa531
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Sat Dec 16 20:19:39 2017 -0500

    classes in codegen WIP. need to restructure much of it to keep
track of class decls, vdecls, and fdecls separately

commit ca38a9e20d8d989fbd0b674c3a81c16cb57a53e3
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 16 18:52:33 2017 -0500

    allow global declaration of all variable types

commit 9c5f0cc0e17bdae5a6970e00e57ac734ed3b9bd8
```

```
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sat Dec 16 17:56:20 2017 -0500

    Added test case for checking duplicate local variables

commit 22a0e661a0338c7c9580ccdcb6f357f4c330c3b1
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sat Dec 16 17:52:12 2017 -0500

    Semantic checks for correct assignment and checks for local
variable duplication in blocks

commit 0de8fbb6f4fed0eee7c2662a2ff3d84aaee915eb
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 16 17:11:09 2017 -0500

    updated mouse click in library and codegen

commit 3312e6474da404117ce849da8fb716fc5155365d
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 16 16:10:14 2017 -0500

    tuples working but needs some modification

commit a0430b5093b0c4e15f7dfb2fa0b5573fe0cc858b
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 16 01:00:56 2017 -0500

    add coordinate type and capture function

commit 55df8ff7cf7f4f6ec610bcc6b15be60d25df4519
Merge: 74bd7cf 63a8e1c
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sat Dec 16 16:00:14 2017 -0500

    Merge branch 'master' of https://github.com/jason-lei/tiler into
semantics2

commit 74bd7cfd1bd43eecb530080889d5232f8d161805
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sat Dec 16 15:58:48 2017 -0500

    Semantics checks for duplicate block declaration
```

```
commit cbc5e548193cf66c03a2ce5a44ce3817c1e686c0
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sat Dec 16 15:58:28 2017 -0500

    Added fail case for duplicate globals

commit 324a6d6cceb53cc686b151525afd25abbf0f338d
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sat Dec 16 15:40:26 2017 -0500

    Semantic checking checks for duplicate globals

commit 0e76729b51a5abee5a0038d9752b5f19315521a8
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sat Dec 16 15:39:49 2017 -0500

    Modified Makefile and tiler.ml to support semantic checking

commit 63a8e1cf7d841f0a38c36d09404acd43bf26b900
Merge: 75dc83b b7492ba
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Sat Dec 16 14:45:19 2017 -0500

    Merge pull request #48 from jason-lei/MERGEMONSTER

    Mergemonster

commit b7492ba34ff5b826e37b6b9204a6c2f3d3363136
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sat Dec 16 14:41:48 2017 -0500

    fixing testing print statements

commit 5656a6d61cf61203cf654ce3d2d296b633cb9d69
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Sat Dec 16 12:43:24 2017 -0500

    objects classCode and getType function

commit 8729ffd28b753e52eec9d81bb4298820404d0f1e
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Dec 15 20:40:25 2017 -0500
```

```
    isNull flag in object struct

commit 397ab159b971653d775639378a0f317f04200e34
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Dec 15 18:51:49 2017 -0500

    added isNull for object and Null objects on grid

commit 7af8ed52b4e72dbe10bcee03f68bcafa6333ad8f
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Dec 15 16:42:27 2017 -0500

    attribute management and access

commit 2b85fb1094786fe9a3d3fa8d74eeb60e9ac1456a
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Dec 15 15:07:28 2017 -0500

    no more stdbool

commit 0da50080ca78712a2db3e8d9633fc320eb709ecf
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Thu Dec 14 19:20:31 2017 -0500

    void * objects in interface with runtime

commit 03001c9c94114927bd4c5cf969da4898aee8ca49
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Thu Dec 14 16:34:07 2017 -0500

    separate set sprite function

commit 75dc83b06dea40cacaf9e41ac7bc287b4f8ac0ee
Merge: d26ea22 c22cc3d
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Sat Dec 16 13:30:38 2017 -0500

    Merge pull request #45 from jason-lei/testing

    Testing

commit c22cc3dbd54fb6bec44a4c7899505ca57cd2e17a
```

```
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sat Dec 16 13:26:48 2017 -0500

    changed codegen so that turn and end blocks not mandatory

commit 58af6fed8554898b470c55f9e294d119633369dd
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Fri Dec 15 14:39:12 2017 -0500

    Makefile removes more intermediate files when calling clean

commit 4b728252e3ffb8d46b6e83dd38a8eab2acc88e47
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Thu Dec 14 19:32:43 2017 -0500

    added do while test case

commit e7e9c631709e37212e495ddb14a71632a77c478b
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Thu Dec 14 13:49:05 2017 -0500

    Changed grid() to tile()

commit ef61eb2468e46eb48a8f10c2360dba041a311e8a
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Thu Dec 14 13:33:39 2017 -0500

    Added test case for if testing

commit a4b53885c743a336899082a1d2c9b36f75beacb6
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Thu Dec 14 13:02:36 2017 -0500

    Improved while test case

commit 94f00c16087c364d74a572d49cf5b950db564d39
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Thu Dec 14 12:45:37 2017 -0500

    Added test case for checking while

commit 7ae2bc8090661f1df1d8fa8eb15a1ef969fa3d28
Author: Jacky Cheung <jacky96cheung@gmail.com>
```

```
Date:   Wed Dec 13 22:42:59 2017 -0500

    Test case for testing out assignment on all variables

commit 2a1c70201084670a70fcd64358620911372c50b1
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 22:42:09 2017 -0500

    Tweaked parser to handle expr in parenthesis and codegen to
handle floats properly (L.float -> double

commit d26ea2296397c503bb130dfb50264f1639a93495
Merge: e4e2712 7bde31c
Author: Monica Ting <monicasyting@gmail.com>
Date:   Sat Dec 16 12:56:39 2017 -0500

    Merge pull request #46 from jason-lei/adding-end

    added end blocks

commit 7bde31ce3853bef225ee4c18b310ab5a4ef9ae21
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sat Dec 16 03:12:10 2017 -0500

    added end blocks

commit e4e27123f7e377a74a08fb3350af586ed9ea7f74
Merge: ec0e3a6 c2f8e94
Author: Monica Ting <monicasyting@gmail.com>
Date:   Fri Dec 15 17:10:33 2017 -0500

    Merge pull request #44 from jason-lei/adding-blocks

    Adding turn block

commit c2f8e94128f0a83412654c370bb7d9315b1f18dc
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Thu Dec 14 14:31:32 2017 -0500

    added turn blocks

commit eb70d79aa783d5da433181f425314921a9ac648f
Author: Jason Lei <jason.lei@columbia.edu>
```

```
Date:   Thu Dec 14 13:38:34 2017 -0500

    adding turn blocks

commit ec0e3a63f108719f54716733b28aefb239d4cc57
Merge: 4f59730 8da29de
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Thu Dec 14 13:44:12 2017 -0500

    Merge pull request #43 from jason-lei/tile-function

    changed grid() function to tile()

commit 8da29de3416ad7f51ad72f771fe9f10b00f20c00
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Thu Dec 14 13:41:29 2017 -0500

    changed grid() function to tile()

commit 03d97047acb9c5addee5f03c68b290b55bf334ab
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Thu Dec 14 07:23:14 2017 -0500

    fixed reduce/reduce conflicts; took out rules from parser for now

commit e69452d16f1ed6549007409c7984c0f3503b334d
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Thu Dec 14 07:19:18 2017 -0500

    tried to clean up some reduce/reduce conflicts

commit 4f5973009321555fe060270201b7e55098089ca6
Merge: 01b156a c85c30f
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Wed Dec 13 17:47:05 2017 -0500

    Merge pull request #36 from jason-lei/closing

    Better Automating of Testing

commit c85c30f7525346d2eaa1ccc63351b51f76b5b60d
Merge: b26837f 01b156a
Author: Jason Lei <jason.lei@columbia.edu>
```

```
Date:   Wed Dec 13 17:39:41 2017 -0500

    merging master into closing

commit b9f7c17c7333575f343618da35a07b05ceffbd01
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Wed Dec 13 17:30:51 2017 -0500

    finished up classes in ast parser and scanner, need to do codegen
now

commit 01b156ad98775a714640902b9afc39b74c368fbe
Merge: ce1ca78 2fc0238
Author: Jacky Cheung <Jacky96Cheung@gmail.com>
Date:   Wed Dec 13 17:22:16 2017 -0500

    Merge pull request #40 from jason-lei/add-statements

    Add statements

commit 2fc0238245f58424b2900712a09dfe1531f3f920
Author: Monica <monicasyting@gmail.com>
Date:   Wed Dec 13 17:18:48 2017 -0500

    fixed print statements in helloworld2

commit 5d1611d9295d8fe29690c71bab3720a1a0de251a
Merge: e06faac 5ad1688
Author: Monica <monicasyting@gmail.com>
Date:   Wed Dec 13 17:09:37 2017 -0500

    pulled master and fixed merge conflicts

commit e06faacdcbab122d0598c2b62f3ba235a6070198
Author: Monica <monicasyting@gmail.com>
Date:   Sun Dec 10 00:39:48 2017 -0500

    added do while loop

commit b56e97543d15597b12bdcbd6a66e36828bda4455
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 9 20:59:57 2017 -0500
```

```
    added basic statements if/else/for/return

commit ce1ca783d7060b6a53d67d0d15d07293cd2f5d7b
Merge: b82ef69 2e9272c
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Wed Dec 13 16:43:44 2017 -0500

    Merge pull request #35 from jason-lei/lib-structure

commit 2e9272c242c44a9e9703cdd09cc622bbf5f947bb
Merge: 3ef056a b82ef69
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Dec 13 16:31:14 2017 -0500

    Merge branch 'master' into lib-structure
    added mouse click
    fixed multithreading of turns
    added window force close function
    makes turn block not mandatory

commit b26837fa5fcab26cfd74bd36c117c7aa3348b2cb
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 16:15:51 2017 -0500

    Updated Makefile to clean up after test better

commit eed8ab34fac6e95426fe75249234a743432aa456
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 16:15:29 2017 -0500

    Added global assignment on global level test case

commit ef5e90a6ba6a4153e27d7f2e70487e68ddbac683
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 16:06:40 2017 -0500

    Updated testing and add global test case

commit b08f7cfc70ad5d8ac8cea05883e00d410369565e
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 15:35:29 2017 -0500

    Remove a print statement that was left in
```

```
commit bc4852e4100bfc719e5894db64c03324e78a0faa
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 15:19:11 2017 -0500

    Added a missing semi-colon to tiler-main.c that was accidentally
deleted before committing

commit 0be1d4eaf4adc8124f8a1902a83010de9eafeb2f
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 15:13:52 2017 -0500

    Updated testing library so that it does not require user to close
windows manually

commit dd3a0471d5c7c0a1c79eb92940773db31b6a3518
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 15:08:30 2017 -0500

    Calling close() has a better exit message stating windows was
opened then forced closed

commit 8522babe99ea4d188db14c5cfa1ecd1d54cbce18
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 14:51:41 2017 -0500

    tiler now supports a close() function that force closes the
window. Evan says this will probably not work as intended if the code
has a capturemouse(), but will just not use that function for
regression testing

commit 3ef056a702d9a27081578958f9ec271a8ef8bfe8
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Dec 13 14:44:12 2017 -0500

    no longer seg fault on no turn

commit 742ba330cc270100af43a50bef1cafc3d471c617
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Dec 13 14:30:31 2017 -0500

    remove accidental output.txt file
```

```
commit ca4cb9a55fb592fe640ce99eb56b119064058e92
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Dec 13 14:29:01 2017 -0500

    fix mouse capture in loop

commit 343e5e20be4aaee3cbdaefb537f3bb178f787ce0
Merge: bf0f3ca b82ef69
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 14:19:05 2017 -0500

    Merge branch 'master' of https://github.com/jason-lei/tiler into
closing

commit b82ef69d947d5f79bbcb4490be7998ef1ca25e81
Merge: 5283ea0 d6bd06a
Author: Monica Ting <monicasyting@gmail.com>
Date:   Wed Dec 13 14:11:22 2017 -0500

    Merge pull request #33 from jason-lei/printing2

    Printing functions

commit bf0f3ca2f3c6c64473477b3484722b8786cce002
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 14:06:02 2017 -0500

    Added a closing function to the tiler-lib as specified by Evan -
hopefully no merge conflict later

commit 58a5b87bebb43765e5f9ebda264b047b1af817de
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Dec 13 13:56:04 2017 -0500

    force close window function for testing

commit d6bd06a66a984666b355aa606d0cd3a73b42dd90
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 13 12:16:35 2017 -0500

    codegen now supports i/f/sprint which supports printing ints,
floats, & string respectively
```

```
commit 5ad16881bf0ea70637d9e5a31496be426ee37e40
Author: Monica <monicasyting@gmail.com>
Date:   Sun Dec 10 00:39:48 2017 -0500

    added do while loop

commit 837724e1c7236bbba65eadbea4e2c86e58badc5b
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Sat Dec 9 21:28:37 2017 -0500

    mouse interactivity in tiler-lib

commit 21f0078a5e2dab184f6268f99c5fbf7f4519968c
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 9 20:59:57 2017 -0500

    added basic statements if/else/for/return

commit 409eff81649299a03cf9cb50b80d6408efc23d29
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Dec 8 19:27:01 2017 -0500

    display objects on grid. No more need to call destroyGame in main

commit a9815113de058325ebd668a9a84ff524b478d801
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Fri Dec 8 19:14:09 2017 -0500

    added a huge chunk of class to parser and ast. still a few ends
that have to be wrapped up here

commit 5283ea04516f5e6f9eddc76e1e597e1170016d11
Merge: 1feddf6 5e318bf
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Fri Dec 8 17:12:05 2017 -0500

    Merge pull request #17 from jason-lei/add-primitives-2

    Add primitives and variable declaration/assignment

commit bbccec9ecfbcb5b741b833f9514b1517959fec06
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Dec 8 16:54:11 2017 -0500
```

```
    single turn block

commit 5e318bfe51b911f4c887b003715b22178d1a9e91
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Fri Dec 8 16:36:59 2017 -0500

    updated helloworld2 with more statements. string and bool
printing does not work

commit 35447362aff2241d03973fa0716d9a0c051ab3e3
Author: Monica <monicasyting@gmail.com>
Date:   Wed Dec 6 16:53:19 2017 -0500

    Added assignment and declaration

commit bbeedd53765b13c8edf771e06226c2a7db97f841
Author: Monica <monicasyting@gmail.com>
Date:   Tue Dec 5 22:41:16 2017 -0500

    Begin to add variable assignment

commit b145541798deeda39492277d9a626d22dd2241f5
Author: Monica <monicasyting@gmail.com>
Date:   Sun Dec 3 01:47:28 2017 -0500

    helloworld now works with background image

commit c9fb7e8242925c29910f16c7a34dc8a07a8b75f8
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sun Dec 3 01:08:01 2017 -0500

    fixing seg fault with setEnd

commit 6d7ec823bec4aaa8928deb4cba05c9da90c28f0d
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Sun Dec 3 00:51:33 2017 -0500

    trying to fix seg fault with turn blocks

commit 9716e619bcebbc1f816a797c7b0ed0b527393f88
Author: Monica <monicasyting@gmail.com>
Date:   Sun Dec 3 00:13:55 2017 -0500
```

```
    added some primitives and testing background in codegen

commit 148d862986e117e37425f18cf1ac770b3e280d66
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 2 23:16:45 2017 -0500

    updated attributes, use enum for strings only

commit 6de6e6c2748f4b648c5bfd389f5f80084d3adb6a
Author: Monica <monicasyting@gmail.com>
Date:   Sat Dec 2 23:15:59 2017 -0500

    added float, string

commit 94cfbed462e770cd1427e8732b802ee9d324092c
Author: Monica <monicasyting@gmail.com>
Date:   Wed Nov 29 17:30:37 2017 -0500

    add type to assign in parser

commit 38765a7cc688893453acab5084dc8ed6fd4d95d1
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Nov 29 17:29:46 2017 -0500

    fixed bug - window not closing on exit

commit 24da09160daf0366edd3ce3fe8ea1b5034f7e768
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Nov 29 17:23:14 2017 -0500

    debugging additional datatypes. helloworld2 still rejected by
menhir

commit 44f8b427798856a9b9cc3cd84a9cda459b9e2713
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Nov 29 16:57:49 2017 -0500

    restructuring tiler-lib to multiple threads

commit 82f502391d21d75474c94dc35dbc65510391091b
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Nov 29 11:49:29 2017 -0500
```

```
    added other code blocks to game

commit aea6d99e00fbab141e45d0eb7aa49f152d9a5430
Author: Monica <monicasyting@gmail.com>
Date:   Tue Nov 28 21:08:49 2017 -0500

    updated parser to accept helloworld2 in menhir

commit 67273e423f6c9efc0bc174d1ff63b71b1ecc5fdf
Author: Monica <monicasyting@gmail.com>
Date:   Mon Nov 27 13:36:46 2017 -0500

    added integers and booleans

commit 61ce9403a470d65b999a7fd39e9bc197f3b21259
Author: Monica <monicasyting@gmail.com>
Date:   Fri Nov 24 23:50:39 2017 -0500

    added float, bool, string, tuple to scanner

commit d8868c6378050ab34a73d391e78bca6adfc2cc3c
Author: Monica <monicasyting@gmail.com>
Date:   Fri Nov 24 19:51:41 2017 -0500

    added new test tile games

commit 1feddf65c73fa725df196db5ae93caa5d3fdae20
Merge: 026b21e 5ffb2b2
Author: Monica Ting <monicasyting@gmail.com>
Date:   Wed Dec 6 19:49:12 2017 -0500

    Merge pull request #16 from jason-lei/printing

    Codegen can now support printing to stdout (i.e. the console)
with pr…

commit 5ffb2b2da96776f4eaa339e1d4caba5fcc0e7e5b
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Dec 6 00:49:39 2017 -0500

    Codegen can now support printing to stdout (i.e. the console)
with print()
```

```
commit 9a1dd114836aca9fafaf2c823e49a525fde8d0d4
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Sun Dec 3 15:13:25 2017 -0500

    object creation and memory mgmt
    Edit: fix surface not freeing

commit 026b21e8a09f7fc2e53d10cf3d8d1b747337c618
Merge: 1c5e511 f524c59
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Sun Dec 3 17:03:25 2017 -0500

    Merge pull request #11 from jason-lei/testing

    Regression Testing Script & Makefile Clean Up

commit 1c5e511223b657113a2f835e5cec82707759c56e
Merge: 722b85d d49fcea
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Sun Dec 3 15:45:12 2017 -0500

    Merge pull request #14 from jason-lei/textures

    tiler-lib draws a background image

commit 722b85d0627a85efb7a38a3b12a0602358cf6f62
Merge: 3934464 0bb625a
Author: Monica Ting <monicasyting@gmail.com>
Date:   Sat Dec 2 21:52:36 2017 -0800

    Merge pull request #13 from jason-lei/commenting

    Commenting

commit 0be3a636252604772c9b943d6360876ae0d0216e
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Sun Dec 3 00:51:33 2017 -0500

    trying to fix seg fault with turn blocks

commit 0bb625a013d112d53fa905736a81d6c8949d0b85
Author: Monica <monicasyting@gmail.com>
```

```
Date:   Thu Nov 30 12:58:08 2017 -0500

    added testing comments to helloworld.tile

commit aa91dc2b60f16094d3f25ab82d02d8fd5056cee8
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Thu Nov 30 12:11:02 2017 -0500

    Scanner now supports single and multi line comments as specified
in the LRM

commit f524c591e826160963d77c37fe09e04afa853e84
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 20:43:41 2017 -0500

    testall.sh now checks for failed cases

commit 05aeb7a1b08a005be939a701e9a5500294f5f793
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 20:43:14 2017 -0500

    Test intermediate file removal can be handled by testall.sh on
its own. Default action (forgot to mention on last Makefile commit)

commit 27c5f2c5de4a1367b555e2240b73e1e59af7955f
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 20:40:47 2017 -0500

    Added Fail Cases: No Init

commit 13bd68782d14adff744d2563a88745b11c26dcf6
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 19:52:38 2017 -0500

    Makefile removes testall.log and diff files from test directory

commit d684e828f187e407c84a187434b4a96ccd2a878c
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 19:50:57 2017 -0500

    Commented out CheckFail as it is currently unused. Intermediate
files no do go into test directory
```

```
commit c7f6c0d3c0ec9bc74c82abfcfc4c93a6233bfa6c
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 18:45:26 2017 -0500

    Makefile removes testall.log when calling clean

commit 1e4ffb5d327d936b48ff5004292cd6fb92671f1f
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Nov 29 17:29:46 2017 -0500

    fixed bug - window not closing on exit

commit 23a68b81e35542c20b53ec4dd2b1584222fe54c6
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Nov 29 16:57:49 2017 -0500

    restructuring tiler-lib to multiple threads

commit fed03f2bb31e7359a869ce0961488df84eccf585
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 16:31:16 2017 -0500

    Minor change in Makefile annotation

commit 147ccf0f09ae4db4737b30b92c7ca5d85973cbca
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 16:21:13 2017 -0500

    Regression testing script up and running

commit 60b97cba9d4428ae27f07c49caf9900c4e085675
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 16:19:58 2017 -0500

    Further Makefile cleanup. Addition of .test shorthand for testing

commit 21147d85617a528211eb3c6bce29af3ee8cb1716
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 14:56:45 2017 -0500

    Cleaned up Makefile more with better descriptions/comments

commit 86712b76f306df73ffd93fcfee17e2d690e7529a
```

```
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 13:49:18 2017 -0500

    Cleaned up Makefile further, executables are now .exe files to
allow for easier cleanup

commit 2810f094db183c483e42a6c51a0df716c8b8d2ef
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 12:42:29 2017 -0500

    Cleaned up Makefile to make it less redundant when compiling
.tile files

commit d0afea7388689ae38de389db64a1d148af85ab64
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Wed Nov 29 12:09:07 2017 -0500

    Added Test Case: Hello World

commit b41800bb8d4132293fda3981c0a067b171bde8a0
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Wed Nov 29 11:49:29 2017 -0500

    added other code blocks to game

commit d49fcea2373e918fe6ecc1185c119e85bb51d92a
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Sat Nov 25 14:54:46 2017 -0500

    tiler-lib draws a background image

commit 39344641f582eb5c4cc7066cf3b3002e859dd47e
Merge: eddc07a 45912d4
Author: JY <jt2823@columbia.edu>
Date:   Mon Nov 20 15:27:04 2017 -0500

    Tiler: Hello world

    merge; hello world is done!

commit 45912d43dd0cee2cd706debcdf6a09c00cbc2739
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sun Nov 19 16:49:46 2017 -0500
```

```
    Aesthetic changes to syntax of type matching at the top

commit eabe3148b6f039f3372faba548e4353eeea3301e
Author: Jacky Cheung <jacky96cheung@gmail.com>
Date:   Sun Nov 19 16:23:23 2017 -0500

    make tiler.native now runs without error by removing or
commenting out unused variables and cases. Minor aesthetic
modifications to Makefile

commit ab5de64985053f68a786ec048fcfb4b35caad326
Author: Monica <monicasyting@gmail.com>
Date:   Fri Nov 17 21:24:00 2017 -0500

    updated Makefile to make helloworld

commit 875a4407b5d583a191de2e779f151e0df992bfa8
Author: Monica <monicasyting@gmail.com>
Date:   Fri Nov 17 21:21:06 2017 -0500

    updated README

commit d8e9ffbaa330139d0662ee4fa78ffa48ff973fa8
Author: Monica <monicasyting@gmail.com>
Date:   Fri Nov 17 21:11:08 2017 -0500

    HELLO WORLD WORKS NOW

commit 41c776d83d68b1dfba8706431dcd2199637bd5e5
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Nov 17 17:28:30 2017 -0500

    Fixed looped dependencies issue

commit 47d2406ccaaed726a1cb1c9754728608b9e81388
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Fri Nov 17 17:03:49 2017 -0500

    Updated Makefiles to compile and link tiler games

commit 80feed175b0b4290a70441c928cf447a2cecd2ee
Author: Monica <monicasyting@gmail.com>
```

```
Date:   Wed Nov 15 18:34:05 2017 -0500

    update Makefile for helloworld and matched function calls in
codegen

commit 99cbba39e26833e9d04922efcd267dbb63d1fc79
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Wed Nov 15 17:21:05 2017 -0500

    updated README

commit 8f9a079fa1362a7ed81e7b541dcdac90a145e78a
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Wed Nov 15 16:42:17 2017 -0500

    codegen compiles whoo

commit a867ba521874660ce640399ccd233cfdb2f8cebc
Merge: 55b72ff 74851eb
Author: Monica Ting <monicasyting@gmail.com>
Date:   Wed Nov 15 16:29:23 2017 -0500

    Merge pull request #3 from jason-lei/grid-call

commit 74851eb7e1590977992335a6f2b99a01276d3053
Author: Monica <monicasyting@gmail.com>
Date:   Wed Nov 15 16:27:16 2017 -0500

    match function names

commit a5191f3feb634ca9867df863c3e55f21e0d0a352
Author: Monica <monicasyting@gmail.com>
Date:   Wed Nov 15 16:23:51 2017 -0500

    fixed typos in createGrid

commit 55b72ffa127e8e10d5c3a50352e1bdaba75738c3
Merge: 003ffa0 16280cf
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Wed Nov 15 16:20:17 2017 -0500

    Merge pull request #2 from jason-lei/ptr-genocide-v2
```

```
commit a9a8abbbb4f3421af467befc28a03b2cd7d5929f
Author: Monica <monicasyting@gmail.com>
Date:   Wed Nov 15 16:10:17 2017 -0500

    update library arguments in codegen

commit fbab990969738e6b0ddb72274f5bfc85bac60a76
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Wed Nov 15 15:57:09 2017 -0500

    fixed grid call function

commit 16280cfc285ffdee361d86d07a83a57c6885ecef
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Tue Nov 14 01:33:51 2017 -0500

    Added create and destroy calls to caller

commit 3cfd57263460ecb9b33e334b8c31a398183a4eee
Author: Evan Ziebart <erziebart@gmail.com>
Date:   Tue Nov 14 00:24:33 2017 -0500

    No longer passing around pointers

commit 81e8a3e51456d7261c0b93774de5c62ef12fe9b3
Author:  <evan@evan-laptop.localdomain>
Date:   Mon Nov 13 23:47:37 2017 -0500

    extern init_ptr

commit 003ffa003c0cb49e534cb0cbf51f8551335f6782
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Mon Nov 13 23:20:33 2017 -0500

    still need to figure out call for grid + what to do with game
pointer

commit f361d7118079e40aabc815822476b09c79d8ebd9
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sun Nov 12 19:00:52 2017 -0500

    codegen attempt 2
```

```
commit f557c815c18cdd87c6d3955daf73944e2dc5693e
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Sun Nov 12 18:05:07 2017 -0500

    still chipping away at codegen

commit 3c89ee66e433782e6ce488ffb46e8c935d288653
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Sun Nov 12 17:58:14 2017 -0500

    added blocks to ast and parser

commit 1b53a5b1c473a3e032af6248b5f32e60f09d3412
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Fri Nov 10 17:51:03 2017 -0500

    reverted some changes in ast and parser to fix codegen

commit a550f2d5f4a8b0841b052a438b5c453aacedc367
Author: Monica Ting <monicasyting@gmail.com>
Date:   Fri Nov 10 17:02:18 2017 -0500

    Update README.md

commit 98a4c23da2789e4f43cb82d8bcb58791819d78ac
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Fri Nov 10 16:54:06 2017 -0500

    changed more of codegen

commit e48946afc63e71570e39758b14303ce75a6a967c
Author: Monica <monicasyting@gmail.com>
Date:   Fri Nov 10 16:47:03 2017 -0500

    Updated Makefile to make tiler library

commit 31fc4d5d5df6eaa8a56d042bd31a78efcab02f32
Author: Monica <monicasyting@gmail.com>
Date:   Fri Nov 10 14:40:28 2017 -0500

    Added Makefile and helloworld

commit 5d184d4664cb11bd2f316effd824b7ef22bc95e9
```

```
Merge: bb1e382 eddc07a
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Fri Nov 10 10:39:20 2017 -0500

    Merge branch 'master' of https://github.com/jason-lei/tiler into
jasonHelloWorld

commit eddc07acd7791df9dfe7bc95441a57ff027c9042
Merge: 20fb62c a968370
Author: Monica Ting <monicasyting@gmail.com>
Date:   Fri Nov 10 10:38:16 2017 -0500

    Merge pull request #1 from jason-lei/evan-hello-world

    Adding static c library

commit a96837003480e7b4dbe97fb6d922b583e5255c1c
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Fri Nov 10 10:32:26 2017 -0500

    renamed directory to tiler-lib and updated readme

commit bb1e382b1f2a0c872337a0597919e3b42bea7152
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Fri Nov 10 09:34:01 2017 -0500

    removing extra or symbols

commit 5f47f7fa668789c04f60e5a177eb21a11ca6328e
Author:  <evan@evan-laptop.localdomain>
Date:   Wed Nov 8 19:06:16 2017 -0500

    Fixed seg fault error and better error management

commit 0047f79045616eb4c3bcea47833a7113157f13f8
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Wed Nov 8 16:42:11 2017 -0500

    added codegen

commit 29dc93fc7e57b771ddfc6eef03fbd1f5ae344de1
Author:  <evan@evan-laptop.localdomain>
Date:   Wed Nov 8 16:17:27 2017 -0500
```

```
    Hello World tiler library and test caller

commit 51f1e10ab9a54c0364bd9d21108e861fa4c0a92a
Author: Jiayin Tang <jt2823@columbia.edu>
Date:   Wed Nov 8 15:03:50 2017 -0500

    tweaked parser, now accepted by menhir

commit 8e129b2bb675098ce705f0255b10535df38de7bb
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Nov 8 14:54:55 2017 -0500

    removed grid from scanner tokens

commit 43f36db095d40b06598af27aca2ee529977d52e5
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Nov 8 13:03:58 2017 -0500

    removed typo

commit dfa017e0ea02d7ba2838f7f9c2b0f5a375a7a932
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Nov 8 13:03:44 2017 -0500

    added ID to parser

commit 62d6c94f11268b0ae3db6ab99af8222c0b561765
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Nov 8 12:43:22 2017 -0500

    added ast

commit 4e02c4535ff499a3d6620696f191e45a7a4f70f3
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Wed Nov 8 11:52:33 2017 -0500

    added ID to scanner

commit cb6a6bd7f4de671d8b6a9ddcca63f87d18118c50
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Nov 7 21:16:00 2017 -0500
```

```
    preliminary parser

commit 558c7d8a3fde8c1593de457a1d05486ba334cf4d
Author: Jason Lei <jason.lei@columbia.edu>
Date:   Tue Nov 7 21:12:33 2017 -0500

    removed strings from scanner

commit 757313bc3acf025e05ae7e6dc246c488e7b6d0c6
Author: Jason Lei <jasonlei@dyn-160-39-139-177.dyn.columbia.edu>
Date:   Tue Nov 7 20:34:12 2017 -0500

    preliminary scanner

commit 20fb62cb09a788a1f15f1300e265d6ff0b5466e1
Author: Jason Lei <jason-lei@users.noreply.github.com>
Date:   Mon Sep 18 21:32:00 2017 -0400

    Initial commit
```