ZEN

Final Report

Eleanor Murguia - Systems Architect (egm2142)
Zoë Gordin - Language Specialist, Manager (zeg2103)
Nadia Saleh - Tester (ns3059)

Table of Contents

1 Introduction

1.1 White Paper

ZEN is a simple programming language which helps users easily generate graphics. ZEN follows C-like syntax and gives users access to most standard C types and operators, and has built-in functions which can create graphics and geometric shapes.

1.2 Overview

ZEN is a language that allows users to easily generate graphics. ZEN is best used to algorithmically generate fractals and other geometric patterns, since users have access to most standard C types and operators, as well as built-in functions for creating graphics and geometric shapes.

ZEN outputs graphics to its own output window, and logs textual output to a .out file. ZEN follows a C-like syntax. ZEN's built-in graphics come from the SIGIL C library, and allow users to place rectangles, triangles, circles, lines, and points on the output window.

2 Language Tutorial

2.1 Environmental Setup

ZEN requires OCaml and LLVM. Once those are present in your environment, open the ZEN directory and run

```
$ make install
```

This command downloads and installs SIGIL and all of its dependencies. Then run

```
$ make all
```

2. 2 Getting Started

This command creates the ZEN to LLVM compiler, `zen.native`. To run a ZEN program, use

```
$ ./run program.zen
```

This will compile and run the program. If a graphics are created in the program, the graphical window will open automatically. If textual output is generated, it will be written to `program.out`.

2.3 A Simple Example

The following is a simple program, `example.zen`, which opens a graphical window containing a triangle, and writes “Success!” to the output file.

```
void main() {
      make_window();
      while (keep_open()) {
            make_triangle(100, 450, 50, 50);
            render();
      }
      print("Success!");
      return;
}
```

To run the program, run

```
$ ./run.sh example.zen
```

The graphics window containing the triangle will appear. Upon closing the window, “Success!” will be written to `example.out`.

3 Language Manual

3.1 Lexical Conventions

White Space

White space is not required for parsing by the compiler but can be used to make code more readable by users.

Comments

Comments are single-line, and are indicated with the “#” symbol.

Keywords

ZEN has a number of reserved keywords for program structure.

Keyword	**Description**
`while`	While loop
`for`	For loop
`if`	If statement
`else`	Else statement
`return`	Return function expression

3.2 Syntax

Punctuators

punctuator	**usage**
( )	encloses tuples, defines order of operation, and contains arguments of function calls
[ ]	array initialization, assignment, and

	access
{}	scopes code
;	end of statement
,	separates elements in tuples and arrays, and arguments in function calls

Operators

In order of decreasing precedence:

operator	**associativity**	**description**
`() [] .`	left-to-right	function call, array indexing, tuple access
`! -`	right-to-left	logical and numerical negation
`* / %`	left-to-right	multiplication, division, modulo
`+ -`	left-to-right	addition, subtraction
`!= == <= >= < >`	left-to-right	boolean not equal, equal, less than or equal, greater than or equal, less than, greater than
`and or`	left-to-right	logical AND and OR
`=`	right-to-left	assignment

Declarations

Primitive types and tuples must be declared with the following syntax:

```
    type ID;
```

Arrays must be declared with their length and can only be of type int:

```
    int [length] ID;
```

Control Flow

Statements

Statements include variable declaration and assignment, and always end with a semicolon (";").

```
float flo;
flo = 3.14;
add();
```

Conditionals

If and else statements are implemented for conditionals. If statements do not require an else following them.

```
if(expression){
    statement;
}
else{
    statement;
}
```

Loops

Loops can be achieved through either for loops or while loops:

For Loops:

```
      for (expression; condition; increment expression) {
            statement;
      }
```

While Loops:

```
     while(expression) {
            statement;
     }
```

Functions

Functions must be declared with a return type, which can be any type, including void.

```
type name(parameters) {
      variable declarations;
      statements;
      return type;
}
```

Here is a small example function:

```
int add (int x, int y) {
      sum z;
      z = x + y;
      return z;
}
```

3.3 Types

Primitive Types

type	description
`int`	integer
`float`	float, must include a digit before and after the decimal place
`bool`	single byte boolean
`string`	string
`void`	void

Non-Primitive Types

type	description
`tuple`	a pair of ints
`array`	left-to-right

Tuple Initialization
Tuples are always made up of two ints and are an immutable type. In order to initialize a tuple, the following syntax is used:

```
tup x;
x = (1, 2);
```

Tuple Access
Tuples can be accessed using a dot (.) and the index of the value the user is trying to access (i.e. 0 or 1):

```
int y;
y = x.0;
```

Array Initialization
Arrays are always made up of integers and must be declared with their length.

```
int [4] arr;
arr = [1, 2, 3, 4];
```

Array Access
Arrays can be accessed using square brackets and the index of the value the user is trying to access:

```
int x;
int y;
y = 1;
x = arr[0];
x = arr[y];
```

Array Assign
After initialization, array elements can be assigned using the assign operator.

```
int x;
x = 5;
arr[0] = 5;
arr[1] = x;
```

3.4 Built In Functions

The following functions are built into ZEN in order to create graphics:

- `make_circle(int x, int y, int radius, int vertices)`
- `make_triangle(int x, int y, int height, int width)`
- `make_rectangle(int x, int y, int height, int width)`
- `make_point(int x, int y)`

- `make_line(int x1, int y1, int x2, int y2)`

In order to connect the above functions to a output window, the following utility function must be used:

- `make_window()`
- `keep_open()`
- `render()`
- `close_window()`

For example, the following function uses the `make_line` function to draw the letter Z:

```
void draw_Z(int x, int y){
    make_line(x, y, x+50, y);
    make_line(x, y+100, x+50, y+100);
    make_line(x, y, x+50, y+100);
}
```

In order to call and display the above function, the window utility functions would be used in the following way:

```
void main() {
     make_window();
     while(keep_open()){
          draw_Z();
          render();
     }
     close_window();
     return;
}
```

This code would result in the following output:

The following general use functions are built into ZEN:

- `printi(int i)`
- `printf(float f)`
- `print(string s)`
- `printb(bool b)`

3.5 Library

ZEN's graphic interface is based upon SIGIL (Sound, Input, and Graphics Integration Library). All of ZEN's drawing and window functions call upon functions built into SIGIL.

SIGIL API

4 Project Plan

4.1 Process

4.1.1 Planning

The planning process of creating ZEN involved standard twice-weekly meetings, along with additional times set closer to deadlines. One of those weekly meetings was a work block (typically held on Monday evenings), and the other was a meeting

with our TA, Mark Mazel, on Tuesday afternoons. Our additional meetings typically took place on Friday or over the weekend.

During our meetings with Mark, we would talk about the progress we had made over the last week, our status on completing project goals for upcoming deadlines, and bring up questions and roadblocks we had come across since we had last met with him. Each time we met with Mark, we would set goals for the upcoming week, and make sure that we were on track to complete the relevant project steps.

Our weekly team meetings were typically treated as a work session, where we could check in with each other, talk about our main tasks for the week, questions, roadblocks, and bounce ideas off of each other.

We also communicated day-to-day with short updates and scheduling questions over text message.

4.1.2 Specification

Our original vision for ZEN was a language that would be utilized to create games that relied on shapes — this language was object oriented, and a large variety of special types. Through meeting with Mark, we pared this language down to one that still had the spirit of our original idea — a graphical language that relied on shapes — but removed the aspects that we held no expertise in, like games, or that may have been biting off more than we could chew, such as the language being object-oriented.

4.1.3 Development and Testing

To develop ZEN, features were assigned to team members as they were able to complete their previous tasks. Once a feature was completed from scanner through to codegen, the implementor would write a few tests, but let the Tester create the full test suite and double-check that no prior tests had been broken.

4.2 Style Guide

Our team used the following conventions in our code:

- All variable names are snake_case, all AST types are camel case.
- Tabs, not spaces for indentation
- Keep lines to a reasonable length (around 80 characters)

- Well-commented code blocks

4.3 Timeline

Date	**Milestone**
September 19	Proposal Due
October 1	Revised Proposal
October 12	First commit in Project Repo
October 15	LRM, Parser
October 22	AST
October 30	SAST, Started Semant
November 6	Started Codegen
November 14	Hello World
December 3	External Library Linked
December 18	Tuples, Arrays Completed
December 19	Final Report Completed

This graphic shows the trajectory of our team's contribution to our project repository. Clearly, our team increased the amount of contributions toward the end

of the semester. Though we hit a number of milestones close to the deadline, this history reflects that we were working on several of the milestones (particularly library linking, tuples, and arrays) for several weeks before finally completing them.

4.4 Roles and Responsibilities

Name	Contribution
Zoë Gordin	Arrays, Library Linking
Eleanor Murguia	Tuples, Demo
Nadia Saleh	Tests

4.5 Development Environment

We used the following programming and development environment:

- Libraries and Languages: Ocaml version 4.05.0, including Ocamlyacc version 4.07 and Ocamllex version 4.07, gcc version 7.3.0.
- Software: Team members used a variety of editors including VSCode, SublimeText, and Vim.
- OS: Development was done on OSX 10.13 and on Ubuntu 18.04 within the `numel` Virtual Machine in VirtualBox.

4.6 Project Log

commit d2432fbd2608cb586278ea267c895b7dbbd2b7bd
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Dec 19 19:28:08 2018 -0500

cleanup and added to readme

commit e450cad9d4c0b3005824379c195e4610ed860d2f
Author: Eleanor <eleanormurguia@gmail.com>
Date: Wed Dec 19 19:06:18 2018 -0500

clean up

commit db4cef5fa32d4dfffe31bb4e4f97f477353b5c05
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Wed Dec 19 19:02:05 2018 -0500

clean up

commit 21fb0771f8e033f107cff10d750f3a17ed0365b9
Author: Eleanor <eleanormurguia@gmail.com>
Date: Wed Dec 19 18:58:31 2018 -0500

tuple type updated

commit c208c67cf2551e0edad5a01a75b651913058e958
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 18:05:53 2018 -0500

report example code

commit b87ac21666c41a27ce32c461d03ae11d0a9a6dad
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 16:55:08 2018 -0500

passing tuples to function test

commit b4ed582119d6a82bc583b2696d4987c2c9f2508e
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 16:54:40 2018 -0500

passing array into function test

commit c804de8295f621bc1bde36d557f213c6e7eea263
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 16:43:30 2018 -0500

function returning tuple test

commit b9aa149a6b2976dc89dcab341f67efa1bb1f3b66
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 16:40:11 2018 -0500

    remove single test dependency from Makefile

commit 8330b3b0fbf84ea824d9f63530c786e5bae857ec
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 16:38:56 2018 -0500

    function returning array test

commit 88f2fb5dcaaf1c4ad02637ad51b142234746e92e
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 13:11:48 2018 -0500

    rename test as run

commit b976a276a8e7e7b1cd47c88cc0fa2e3c641c4a01
Merge: defccae 3dc826a
Author: zoeg21 <zoeg21@gmail.com>
Date: Wed Dec 19 10:14:18 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit defccaeb745c3c121a2eefd55a8aeda994538a12
Author: zoeg21 <zoeg21@gmail.com>
Date: Wed Dec 19 10:14:15 2018 -0500

    bye sprites

commit 3dc826ad852b5f79ba82c64db6de0dc12116ba53
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 00:12:22 2018 -0500

    echo make all upon dependency not found

commit 5064c53882dac811b9d7d4553459d67f115feb01
Author: Nadia Saleh <nadiasaleh26@gmail.com>

Date: Wed Dec 19 00:10:42 2018 -0500

    remove sdl_window

commit 06021a3045e4bdc1a4630d3786b8063c5cc7f35f
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Dec 19 00:10:10 2018 -0500

    check for all dependencies in testall

commit cf298ea6d0fb25d87bd15a69b119d632d3dcdf84
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 23:58:25 2018 -0500

    update array test

commit 20435249dda7ac91aed0e34d488c626c7ea80081
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 23:58:04 2018 -0500

    remove printbig from test

commit 0aea378dd740d6ecb23a03f30a7edc32ac01caa7
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 18 20:47:24 2018 -0500

    cleanup

commit 6c961d3d444393f7e1323caae1ebaf6360db6bc8
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 18 20:45:41 2018 -0500

    cleanup

commit 7540323dc8ca0d66827bf0d9212db94b0d88a45b
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 18 20:43:06 2018 -0500

    removed printbig

commit 9bdf5333c97a1d0f1791a4445df4e91922de95f0
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 18 17:56:45 2018 -0500

    cleanup

commit b1b92fcf2491ab5d39a3459dfdedf23e35b7b804
Merge: 37dea33 f58de74
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 18 17:52:41 2018 -0500

    resolving conflicts

commit 37dea3323659c25e5e3c2bfc2d5fe33e58b8c353
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 18 17:50:13 2018 -0500

    ARRAY ASSIGN WORKING and cleanup

commit f58de74bcb82ab298a0395a5df2ae14e9734ae4e
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 17:06:45 2018 -0500

    added y check

commit 47567cd4b4ff800efc3313166fcb7cb6383a2cc8
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 17:02:49 2018 -0500

    clean up tuple tests

commit 3081e9d9aedea5b54a06cad4600b243b996a098a
Merge: 763faf8 3dffa5e
Author: Eleanor <eleanormurguia@gmail.com>
Date: Tue Dec 18 12:19:05 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 763faf8770e27184ea1e3135e6eb3afc23bbe139
Author: Eleanor <eleanormurguia@gmail.com>
Date: Tue Dec 18 12:18:39 2018 -0500

    tuple access tests

commit 3dffa5e8831ea90c099c2cdce2e9d2baa1038e26
Merge: adffa1c 1478c3b
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 12:18:08 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit adffa1c93242f8a79b1ba1328940fc25d46677e8
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 12:17:57 2018 -0500

    add installations

commit 1478c3b9c4670d70dbb0d518ebcad89b215164cd
Merge: 0165c9b 2f6d94a
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Tue Dec 18 12:06:43 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 0165c9b790975b91d56d19d57dce91762ba61410
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Tue Dec 18 12:05:32 2018 -0500

    tuple access

commit 72c91e2882633d8480081b172d1920dcfcd6542f
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 11:55:04 2018 -0500

    add git clones

commit ab39771fa9f0bc7e6676d9bd018a0a7c7bce38c6

Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 18 11:46:21 2018 -0500

    add make installations

commit 2f6d94aef55f8c4f0da6e1b0acb78e5874e5470f
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Mon Dec 17 19:46:44 2018 -0500

    tuple test

commit 05d2746d7c6f7203c5cdfc9c98d9f422e8071b94
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Mon Dec 17 19:41:38 2018 -0500

    update array test

commit bb1440caa29624fb6f698d460b8971f65c0d6ad7
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Mon Dec 17 19:36:23 2018 -0500

    array test

commit ce88e205180de519120bf05e558bf7eeaf0822b1
Merge: 8b19cba 7d15d13
Author: Eleanor <eleanormurguia@gmail.com>
Date: Mon Dec 17 18:25:40 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 8b19cba6452d3572ba0865c22ce08aa31473f473
Author: Eleanor <eleanormurguia@gmail.com>
Date: Mon Dec 17 18:23:43 2018 -0500

    demo

commit 7d15d1315d2f0aa2eda4d117ddcbb599cece3d15
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Mon Dec 17 17:13:06 2018 -0500

```
    mod test

commit 7131023f41c4ef3b12fb723d1a46a1adf3dc172e
Merge: 61eccc3 e1a6d03
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Dec 17 16:53:27 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 61eccc34b5b1ea076bd16f2e26aff69fd8eac6f8
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Dec 17 16:53:22 2018 -0500

    cleanup

commit e1a6d03710a4d76b6c05db8654b29dd798d7b743
Merge: caf6b7f c3f67c1
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date:   Mon Dec 17 16:52:33 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit caf6b7f24313b0e911e627fff9f2e572b3a5a72b
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date:   Mon Dec 17 16:52:25 2018 -0500

    add tests for line point rectangle

commit 745959c3770776962ff82798058b506d318b3a71
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date:   Mon Dec 17 16:52:01 2018 -0500

    add make_rectangle

commit 3f2e3936579355f5b9389289c4cd5e55bac7e419
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date:   Mon Dec 17 16:51:38 2018 -0500
```

```
    typo

commit c3f67c1091ce372846641abbe8d39d4fbaefd69d
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Dec 17 16:32:46 2018 -0500

    cleaning up

commit 6444bc79541f5edaa1d2ce703f7cc17bbd5155f4
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Dec 17 16:29:57 2018 -0500

    cleaning up

commit ebebc3395d07c701403300ecf222fdf2fbd121b2
Merge: a5cb948 1965bf1
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Dec 17 16:29:34 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit a5cb948b0fecde82d1dce66bcc2fdc9f2ee8967f
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Dec 17 16:29:29 2018 -0500

    cleaning up

commit 1965bf1bf71bc07c2157ee155d13a7728fb409f0
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date:   Mon Dec 17 16:26:11 2018 -0500

    update shape tests

commit c9fe007ec7f5b6c5a1203f60d09087c06a7badea
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Dec 17 16:14:11 2018 -0500

    fixed pattern matching and unused var
```

commit 5c4045d40ac355680f5e03a2b0f46ed3ada6a72d
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Mon Dec 17 16:05:58 2018 -0500

    removing unecessary stuff

commit 228cf18b8a3de966c031ee2100a6029b54cd7ded
Merge: 65db24d 5680097
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Mon Dec 17 16:04:53 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 65db24d9d4116eb71cb0c8c828384677ca9404d4
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Mon Dec 17 16:04:49 2018 -0500

    put assign error back in, trying to add array assign

commit 56800974f730ea9db5329b2cea88c28cd17c3caf
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Mon Dec 17 15:37:45 2018 -0500

    update shape tests to immediately close window

commit f065018a36350dd2105fafb90e01a8e62b7e13ef
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Mon Dec 17 15:37:18 2018 -0500

    update test scripts to include render

commit c90345897dce64774e282fdc3e81c10e9e9943db
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Mon Dec 17 15:27:06 2018 -0500

    update move rename shape tests

commit a385ec9f9b421ba538db628f1041806c947f376b
Author: Nadia Saleh <nadiasaleh26@gmail.com>

Date: Mon Dec 17 15:10:15 2018 -0500

    add render to Makefile

commit adb90cb03fb9942cb2525d5b3dc163e96ffcfad9
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Mon Dec 17 15:02:40 2018 -0500

    committing rectangle finally

commit 2a6104228909a204d2698d6897cdbc3abc1cac1a
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Dec 16 14:35:39 2018 -0500

    fix test func1

commit 441704e7afc8bec07b9c5dc018e6d38c4a0efb79
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Dec 16 14:35:26 2018 -0500

    comment out carrots and tuple access

commit 40756b04190e7ae005aefa850927f533261775bb
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Dec 16 13:35:42 2018 -0500

    update single test script

commit c9c60b0a98fada0e4ee25fb14957f525b53566c3
Merge: 7fe5613 37443e9
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Dec 16 13:34:01 2018 -0500

    Merge branch 'master' of github.com:emurguia/ZEN

commit 7fe56130fe0e932f30533fa44cedbaeb22c8acc5
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Dec 16 13:33:56 2018 -0500

    removing make_rectangle from testall run

commit 37443e9b3c53fc9f86eb3c2fde6b86560f441174
Merge: 128f84d 88af039
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Dec 16 13:30:48 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 88af03958342e6868eaf745fcd29db04a8783d81
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sun Dec 16 13:27:06 2018 -0500

    bye tiler

commit 128f84de602ea636b1b7c0a07af102741aa72e61
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Dec 16 13:16:31 2018 -0500

    update line & point links

commit 869552c3ffa193c790079aa435ba792b578ce951
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sat Dec 15 13:51:00 2018 -0500

    fixed pattern matching for arrays

commit 67cdcabeab249b194ef434db98cf64232021eb1d
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 15 13:27:00 2018 -0500

    RCARROT

commit 4ad00eaabc6c65c36e030b47532a4bf21106548b
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 15 13:04:46 2018 -0500

    reduce/reduce error fixed

commit 72c0cd5a32c83900f0feb41c70661292a5ef776f
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 15 12:26:33 2018 -0500

    tuple access is most likely working

commit 99f5cb329794bad03477dfe86c7c2657029a4758
Merge: 431a646 640d8fb
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 15 12:04:08 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 431a6460189556b71ac4f7acda46eb6773bf5c0d
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 15 12:04:02 2018 -0500

    tuple access

commit 640d8fb87537b8e799453f21161a428b0d55b16f
Merge: c301668 55286ca
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sat Dec 15 11:53:28 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit c30166887d58dd288d32961bc7a71ec7b689eea9
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sat Dec 15 11:53:21 2018 -0500

    fixed parser error from accidental comment, fixed array access by removing error checking in check_assign lol

commit 31c3bc10a815d35d71073abe7c0f9ac645f60b42
Author: zoeg21 <zoeg21@gmail.com>
Date: Sat Dec 15 11:53:17 2018 -0500

    added render to rest

commit 3bec3f88dd265c8c41daeced8d1a5e9a9977c489
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sun Dec 16 17:27:18 2018 -0500

    goodbye render

commit dcd5a8fdf25991b7e74aab96295329f6cad1902e
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sun Dec 16 17:21:38 2018 -0500

    render....

commit ad43f7e329517c9b297590bfe939f4ebad3d5240
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sun Dec 16 17:11:23 2018 -0500

    render func

commit 9724be39bb6ff64498eb5d5771a2114f0d757349
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sun Dec 16 17:01:03 2018 -0500

    seperate render function

commit 661ef2113c138f3b65ee2c68b9099eb12dcc6ae0
Merge: 49d3c8e 2a61042
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sun Dec 16 16:58:08 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 49d3c8ee0bbc4e43225e51d9ad7946ae8cb05ef5
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sun Dec 16 16:56:12 2018 -0500

    remove render

commit 55286ca31dd5a268bbf867201a4b3ad85942d998
Merge: 8ad958f e9aace9

Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 15 11:14:22 2018 -0500

    merge conflicts

commit 8ad958ffb781306f1b78a373d78a0872c32fd6be
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 15 11:01:44 2018 -0500

    tuple access...in progress

commit e9aace97cd653443cf5dc196a90b86d318ced1d2
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 13 22:33:10 2018 -0500

    added array creation, only issue is pretty printing in sast, seems to work but probably need access to actually test

commit 9e280f1756f65257695dc65b70943846a963f3eb
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 13 20:50:46 2018 -0500

    fully commented out array stuff to fix shift-reduce issues

commit 8f6da60aa3bfc18b7b467e23573af8e884f1e504
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 13 11:01:00 2018 -0500

    mod and arrays

commit 0adbf609e292f8467604b64e299af26a30b4a89c
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 13 11:00:44 2018 -0500

    added modulo but need to test and fix parse/reduce, messed around with arrays for a while

commit 2c0e93500a4698718025a26cbee477e74871ec66
Author: Zoe Gordin <zoeg21@gmail.com>

Date: Tue Dec 11 18:46:01 2018 -0500

    adding library functions'

commit 1cefd12f9867b1af4f6d1383674ff1c5125d706e
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 18:45:41 2018 -0500

    removing float cast

commit e67bd4c9a4062adc670351505a4780101ef6a429
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 15:50:56 2018 -0500

    horrifying fix for array errors

commit 6150fcf1a140721a82242df5761417639b60a86d
Merge: 840a694 bcab8cb
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 11 14:14:01 2018 -0500

    Merge branch 'master' of github.com:emurguia/ZEN

commit 840a694e59fd4e51239da224b336e6ff0c00cde5
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Dec 11 14:13:56 2018 -0500

    added arrays to scanner parser ast sast

commit bcab8cb98a233e6cde29990e2fd16816708b3666
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 11 13:27:59 2018 -0500

    trying out no float cast

commit 53a1ca9711b601b89d95cfbc59472ab86a558904
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 13:22:17 2018 -0500

```
    fixed triangle test

commit 2aa78732d6fbc60bf7a8ab79aa280bf2a5c78ef5
Author: zoeg21 <zoeg21@gmail.com>
Date:   Tue Dec 11 13:12:11 2018 -0500

    added stuff to clean

commit 57f71f25e9383b9a107da0cd815c1e551198e835
Author: zoeg21 <zoeg21@gmail.com>
Date:   Tue Dec 11 12:50:26 2018 -0500

    fixed int return and added endif

commit bbc1b69e3c1af6ff978662ab78cb76616163e2bc
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Tue Dec 11 12:31:27 2018 -0500

    sprites take 2'

commit 65e172fbd3eb4ae475a2552c05dec80381605e0f
Author: zoeg21 <zoeg21@gmail.com>
Date:   Tue Dec 11 12:26:22 2018 -0500

    well that didn't work

commit 29582f255bb218a3c067bf85e67840d60a9845c2
Merge: 74eba96 dfad2b8
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Tue Dec 11 12:25:26 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 74eba9614816c2d138636b0400c8195ad5873e1a
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Tue Dec 11 12:25:21 2018 -0500

    sprite time baby
```

commit dfad2b874572719a11c6c568806c920a468dccf1
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 11 12:14:32 2018 -0500

    removing print statements

commit 194a02e122abed3b3a4e8d36ff8d1992b26f2d12
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 12:08:43 2018 -0500

    adding window stuff to makefile

commit f14d678948b5547e159c7e98fafd20bd47905177
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 11:54:02 2018 -0500

    fixing test-circle

commit 2828e7f4040e92ee47b461456a01cdd647caaf07
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 11:52:13 2018 -0500

    adding point and line c files

commit e249cdba23ce6749a6c1e4eba4dd47bb7d0ed887
Merge: 5052597 771c015
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 11:35:06 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 5052597554f380f252deaf13621253b5ba7f19ce
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 11:34:55 2018 -0500

    got rid of get_num

commit 90e77a1e093603b0f4dc6be46199aec925b02906
Author: Zoe Gordin <zoeg21@gmail.com>

Date: Tue Dec 11 11:30:15 2018 -0500

    fix funct maps

commit fdf35dfc828cf6654a2b5f90348eaeaec250faa8
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 11:28:44 2018 -0500

    got rid of sdl window

commit 01cd845c242a5d7abfa01e51272e0df14c1f4b52
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 11:27:41 2018 -0500

    fix syntax error, double in

commit 7335fb38b7285236e02517feb006e28472310b44
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 11 11:26:29 2018 -0500

    got rid of unecessary stuff

commit 848a0c87fe8b21397f6d6cec0d06b0201fabbf5c
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Fri Dec 7 11:26:07 2018 -0500

    removed unecessary code

commit b5060d08c9d5c41cabb6f7708b7fb6b5e24f1b5e
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Fri Dec 7 11:25:46 2018 -0500

    adding functions to open close and keep window open

commit 771c01592226492d8b47adaae3fee0e1a0fb4dfb
Author: zoeg21 <zoeg21@gmail.com>
Date: Thu Dec 6 19:41:51 2018 -0500

    fixed extra bracket

commit d0d7340f570dcea5e5d6506689c3add9852a6f09
Merge: e299272 fd87b65
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 19:32:23 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit e2992721e56936d6325f8476c49abdb4bbffc26b
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 19:32:12 2018 -0500

    making lib functions all ints baby

commit fd87b65d7c71e1bb61e97bb20a91f71b1bc8986d
Author: zoeg21 <zoeg21@gmail.com>
Date: Thu Dec 6 19:31:29 2018 -0500

    reverted test-func1, making circle funct take ints

commit 35e4efc1512c602a107a533216c02948f2a779af
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 17:04:37 2018 -0500

    trying this shit out

commit 1738df862a318022cff9ccd81c80dc91ff9a422f
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:54:05 2018 -0500

    int -> float

commit a5e15505e91236be37608b556623b40b67be0d44
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:29:48 2018 -0500

    fix

commit 2479dedb9986d0b66177c5e7acd9c4e79380fe69

Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:26:49 2018 -0500

    fix

commit 0e2969dfc5dbe24f434b09697b59b8fcf3cd6bae
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:24:45 2018 -0500

    fix

commit 57e787bf6056b4d041e6dc87400bb8ae9278f9be
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:19:56 2018 -0500

    fix

commit 29b2f9341984d800893b5f92ed0a2f047093a917
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:19:13 2018 -0500

    fix

commit 2c5daa8767389e735b6907157427cb5c7ca528a9
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:17:36 2018 -0500

    fix

commit ec64f6f8a0a1d27b1e58825194d2f636a2b8d1e8
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:16:57 2018 -0500

    move print arg

commit 48fcdab39fa3e86b61509acd07b91bca2ecc6965
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:11:53 2018 -0500

adding print function for funct params

commit 8f59fb5c3c44dfce74df3800333b93e8e2e5ad78
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:11:14 2018 -0500

adding print function for funct params

commit 7613d5a4bafd4107848b0de0fd2ae856d6006930
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 16:10:27 2018 -0500

adding print function for funct params

commit 61e59e7d49775f8581c597dd0fd8583a1d130911
Merge: ada9b19 d9f3643
Author: zoeg21 <zoeg21@gmail.com>
Date: Thu Dec 6 13:44:42 2018 -0500

correct makefile and testall

commit d9f364370ced2cd16632e7b426e82914d228c29d
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 13:43:37 2018 -0500

removed sdl window bc it is not working

commit ada9b1989dad16284d86e9bd5753dae302242981
Author: zoeg21 <zoeg21@gmail.com>
Date: Thu Dec 6 13:42:59 2018 -0500

goodbye sdl window u did not work

commit 9a5d6c13943f7b284a79a3c4585efe6238ee2f06
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 13:24:38 2018 -0500

adding sdl window test

commit 58a50b4b1d9153acb70968978298cde681be47d4
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Dec 6 13:19:46 2018 -0500

    trying out sdl library

commit 670c43afded25371b800174788384e88a06e236e
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 22:14:30 2018 -0500

    removing unecessary

commit 22a086058166bfd8754b435c1ff0cee4a7845f8f
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 22:13:38 2018 -0500

    printbig test

commit e977bd2e3398ef402071c46bbc943030c91d9a5a
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 22:13:08 2018 -0500

    simplified test circle

commit 8f7485b632c7b5991cf46d35abdc6ad1c444a65c
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 22:12:34 2018 -0500

    added triangle

commit 2ee274f6757c871f53e76e847643dcd0eca5f124
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 22:11:11 2018 -0500

    added triangle

commit 7559ee5b32f061d42f74f01ec73f4b2fd6abc9a4
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 22:10:41 2018 -0500

    triangle files

commit 96621d81e2b7920a0eee020176c1a1308cc4cb68
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 22:01:17 2018 -0500

    fixed triangle and rectangle

commit 590ea525125a2d30368090546753caa5492e88f2
Merge: 43cec69 72c007a
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 22:00:20 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 43cec69b82401bd7bd81a3a01b4e4aba17a7b351
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 22:00:13 2018 -0500

    made triangle and rectangle return int

commit 72c007a035181cff9ffba183c14cea7c09209660
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 19:44:56 2018 -0500

    tests still not working

commit 03f68ac80de5f08ae527c9f9e5ceb117950c8fcb
Merge: ab2fa44 6897da7
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 15:19:02 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 6897da7e9c74132f2c433f8991d57070d70cd2fa
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 15:18:29 2018 -0500

removing unecessary build stuff

commit 5d02ffb53bc75605c827f717f9ab395a407481ba
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 15:16:52 2018 -0500

still working on these

commit 5e76a0296897026caefb1294e3244f2c1bb654ed
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 14:53:04 2018 -0500

committed fixed testall with make_shape.o

commit ab2fa443aed89d002110707aa0d8934e56379a8f
Merge: 81a473d cd92d91
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 14:45:35 2018 -0500

Merge branch 'master' of https://github.com/emurguia/ZEN

commit 81a473d1291b000fd64dc542c8a97dc12d39e84f
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 14:45:26 2018 -0500

bye zen.native

commit cd92d91b8a57a905d7ffba6363685822dfd6946a
Author: zoeg21 <zoeg21@gmail.com>
Date: Tue Dec 4 14:41:54 2018 -0500

removed .o that broke EVERYTHING

commit a87596407956a28f8e264047cb1c9bbcf790529c
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 13:50:55 2018 -0500

oops

commit efb4dc34b8af44ef50cc13a486c9828a95dd5ecc
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 13:50:21 2018 -0500

    trying something with circles

commit 3516124f05e57cda40a750dc2983506732769a32
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 13:41:03 2018 -0500

    added make_circle to testall

commit b9e0d6533006e35f0fea33de0c82f50bfd5afe17
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 13:38:14 2018 -0500

    added make_circle to Makefile

commit a0e5040176722bc100391c2274a0790f03f9c822
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 13:33:05 2018 -0500

    fixed make_circle adding a circle test

commit a92c63e975643ce02364ac614e17e3218e9c9c15
Author: zoeg21 <zoeg21@gmail.com>
Date: Mon Dec 3 22:24:22 2018 -0500

    fixed circle function in VM

commit afb8df48ffbdcfc0f76a9b0ba41e8656c226e33b
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Dec 4 10:16:05 2018 -0500

    circle lib file

commit 71478abec22141bb8314c56285037732f047aff0
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Mon Dec 3 16:22:51 2018 -0500

adding tiler stuff to test in VM

commit f469b04546b2d85c89cfb5e381ce789eb044763d
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Mon Dec 3 11:22:18 2018 -0500

added set functions for tuples

commit dc29eca614b7a96d49e8e6d85d279bb05bbaf772
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sun Dec 2 16:34:07 2018 -0500

added tuple built in functions back in

commit 5e504df7a3a635d01a241ac983f6f7871ad6f314
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 1 18:19:40 2018 -0500

tuples actually working now?

commit e4d0b1cab70467bf05bc5865947dc54388f14360
Merge: cc8043e 41520eb
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 1 18:14:43 2018 -0500

testing

commit cc8043e4562d82f5ac820edf9e6801b665d4e00f
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Sat Dec 1 18:13:03 2018 -0500

almost fixed tuples

commit 41520ebcb8f9ca1d9ea560e428e33fbe6b63e8d9
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sat Dec 1 18:09:11 2018 -0500

move and update tests

commit fec24a8ce0af73f462158fe5cd20f4940d09eba3
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sat Dec 1 18:08:24 2018 -0500

    remove extra list.rev

commit 38a6733b57fe27efebeeb7834bf6306a5733e289
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sat Dec 1 17:53:01 2018 -0500

    add so many tests

commit 9f8aaec8adf99c04cdb396c3e2558aa7ffb414fa
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sat Dec 1 16:12:19 2018 -0500

    more test cleanup

commit 70396107a42530f2b717a922e1331611e7149b0e
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sat Dec 1 15:34:49 2018 -0500

    clean up tests

commit 37f26a8b6fb1ef7a1b4feb6664beeaf0eb497715
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sat Dec 1 15:09:46 2018 -0500

    unfortunately i do not remember what i changed

commit 0c6130a3117fb2e7d31748afc11156b30a9d16da
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sat Dec 1 15:09:28 2018 -0500

    tried to add printb

commit c47107e597ebb9c7fa8bf540871a481993abba48
Author: Zoe Gordin <zoeg21@gmail.com>

Date: Wed Nov 28 21:19:17 2018 -0500

    added string to the scanner lmao

commit 4c022a5153e520d9c5b303f78e949a5cf6e1fe00
Merge: 9ac06b4 79868ec
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Nov 28 17:56:35 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

commit 9ac06b40f3ef14e67da822e7c2ef3da32af3f247
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Nov 28 17:55:29 2018 -0500

    actually FIXED library functions

commit 79868ecff93ed7339c0b08a3713965cb76b03b56
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 16:10:15 2018 -0500

    update int tests

commit e3b1e5b8ecef6ec6bc9cb1428342b5b0cbea82fa
Merge: feafbde 119d66f
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 16:02:19 2018 -0500

    Merge branch 'master' of github.com:emurguia/ZEN

commit feafbde4d9bb59ddef588012480bcde2160eade2
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 16:02:12 2018 -0500

    update tests

commit d80694624211c7c4aaeb4a10891f135044dc0b99
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 16:01:15 2018 -0500

update make clean

commit 119d66ff5ef772f12ca8ae5fddace0ff0b2421bb
Merge: 3a080cd 9fdeb97
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 27 15:58:38 2018 -0500

Merge branch 'master' of https://github.com/emurguia/ZEN

commit 3a080cd5ae66852064a03c1ad6d1e1d415ec3c9e
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 27 15:56:14 2018 -0500

no built ins

commit 68ee584469f95f9dea278a23f7c715058d598759
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 27 15:52:39 2018 -0500

added printi

commit 9fdeb97e5eb6c4fb732f730fe2b97e44b979eb96
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 15:50:01 2018 -0500

fix hello world test

commit b891f36643b41aaa1585d114c3f45c14e282ccd0
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 15:46:21 2018 -0500

committing cleaned directory

commit 5f719157fad39f3c411e34c9daa0ab1196b3d631
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 15:45:16 2018 -0500

put back makefile

commit f77a80d18d31086bddab5be1d62353ff4473cb2f
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 15:44:19 2018 -0500

    add tests

commit 464c733837e6b0d5b3bce4b55f5b56ef493cda46
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 15:23:54 2018 -0500

    update make

commit c38af901e056a53f60fb4c5d2885a3fde0f1c92c
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 27 15:10:48 2018 -0500

    removing built in functions bc syntax error again

commit 14f947f1a02eccc1f5b059ceb6c584c673e392cb
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 27 15:10:30 2018 -0500

    adding built in functions

commit cbeb581fe01097dc553aec6aff7b88ef1f68b8c0
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 14:36:22 2018 -0500

    update testfile naming

commit 79baf45c474380a435d9fe939b6f8cbbf140dea8
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 27 14:35:37 2018 -0500

    update make

commit 4f991dadcae947e9148bfee6e6a62f2880666a0e
Author: Nadia Saleh <nadiasaleh26@gmail.com>

Date: Tue Nov 27 14:35:19 2018 -0500

    add testall

commit d0d840ad8091fe46e67812f9a57db76b376d9863
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 20 15:57:16 2018 -0500

    fixed all add_binds, lists and tuples commented out

commit d2170c9018ae20c8b4e4d670c7146e55429743d2
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 20 15:39:55 2018 -0500

    library function test files

commit 81bb49d7f6ff5e615207f030d356983bed2fdea2
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 20 15:38:21 2018 -0500

    testing library functions

commit 6652a64c19f541722a3e2af25e1615bc762f6036
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 20 15:37:45 2018 -0500

    adding printbig and get_num

commit 89b278ec376abb66cd6af17fb7e0f91cdcbcf200
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 20 15:37:33 2018 -0500

    adding built in functions correctly this time

commit d71970eeaea811c0cc0927b143a12fc4b22acb06
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 20 15:37:13 2018 -0500

    adding printbig

commit f9758c2e51fd30bba30a5972e5bfdf09a02ed588
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Nov 20 15:37:00 2018 -0500

    adding printbig and get_num (test external function)

commit fe49f33828427a976bd93f620a7b8450a5767f86
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Nov 14 11:03:48 2018 -0500

    added readme and zen.native to hello world

commit 1ea66ebe81235144ac512fada81f799974ec9e42
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Nov 14 10:37:40 2018 -0500

    made hello world submission folder

commit eff2f187bfec33903c9cbdcaac46161ae0b63c0d
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Tue Nov 13 17:47:42 2018 -0500

    add string printing and test script

commit b0b6779eeb4d9c29f7ebb7b9a4966b026cdf7120
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sun Nov 11 16:09:24 2018 -0500

    fixed tests to have mains

commit 44ca0b1c852ce1b8da9f997f20e7c9c1376fb62f
Merge: f590726 7dd03b7
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sun Nov 11 16:08:10 2018 -0500

    Merge branch 'master' of https://github.com/emurguia/ZEN

    merge

commit 7dd03b78dba06a763205a547fc610bb23563b2f8
Merge: fc301c0 1bcd382
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Nov 11 15:26:21 2018 -0500

    Merge branch 'master' of github.com:emurguia/ZEN

commit fc301c0871035ab5d26bf4ba66958958155bc5b3
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Nov 11 20:25:46 2018 +0000

    fix errors, got compiler to compile

commit f0d2d5c42c7ed7dd6c9b415790bc052cbf0f25a9
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sun Nov 11 20:25:14 2018 +0000

    basic tests

commit f590726849c42f6713fbcf0f819f600666a56fbb
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sun Nov 11 09:18:59 2018 -0500

    added tests folder and wrote some tests for binops and tuples

commit 1bcd38295402c4ff96c345667f505d2908a734ae
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Sun Nov 11 09:00:46 2018 -0500

    tried to separate out built in functions but kept getting syntax errors

commit fa1a577fea3a85f88aa9810c567660aca0c7a730
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Thu Nov 8 14:06:14 2018 -0500

    added Makefile, started codegen, commented out lists

commit 5e871de0f0535adb5533734e64c4925ed67c5f5b

Author: Zoe Gordin <zoeg21@gmail.com>
Date: Thu Nov 8 12:01:31 2018 -0500

    added comments from Mark

commit e6db961857a02b62abcfa73831c8cdbb5700d95a
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Nov 7 21:02:16 2018 -0500

    removed lists, added separate get functions

commit b28077fb2533ab708940daf402dcf3c35f262d23
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Nov 7 21:02:03 2018 -0500

    changed literal to intliteral

commit 866c3173b4aa8b6ad2c0360316d093296318b1e9
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Nov 7 21:01:32 2018 -0500

    added codegen files

commit 4626aaa1aff9b2ed695c4b958bd3fc7227d6b84b
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Wed Oct 31 19:22:46 2018 -0400

    fixing more build errors

commit b3aa04833afa555ecbe4810bb23866d5bb571bed
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Oct 31 18:51:40 2018 -0400

    add pretty printing sast

commit 34a487fa3be25edfb38bb670d058341d3a8608eb
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Wed Oct 31 18:48:22 2018 -0400

ocaml build errors

commit e6ad4cce5d4f29243dbc4f2d17abaae0fe565cba
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Oct 31 18:34:17 2018 -0400

add zen.ml

commit f60740351e789d817e70b76dcc0710f50cce0ae2
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Oct 31 18:31:15 2018 -0400

test file

commit 6520624002fb168f071f2514cafe05e7f42fb771
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Oct 31 18:27:39 2018 -0400

tuple and list semant updates

commit 74f670dce794055ee24262690d2cab5bd9bc87f5
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Wed Oct 31 17:54:55 2018 -0400

fix $ arguments

commit 20898e286e49831a708dbd95e9a528fcecb9c733
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Wed Oct 31 17:49:32 2018 -0400

removed expr list for slistassign

commit 7d29d1616bff96fd4fd3398e11b9fdebb1fb38b3
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 30 15:24:22 2018 -0400

fixed tuples

commit 2762fc7da56460be8cb365a02c0d4c20f277ae50

Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 30 15:24:08 2018 -0400

    added built in functions and tuples

commit d8e2dfacf6e4836738076c3fb19c3ae0b4e3a1b9
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 30 15:23:22 2018 -0400

    i think i accidentally typed a space in this or something

commit f185ef3c4cf98efbe28d7ab7560e1c4d2865df91
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 30 15:23:02 2018 -0400

    modified pretty printing

commit b174e8cb5c20fa5f9104bbc0fbbddfd977e61454
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 30 15:22:52 2018 -0400

    updated sast

commit e243a31088e5806ce181ac92831d9e33f7d33a45
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 30 14:00:41 2018 -0400

    added semant and sast for zen

commit b7af6cb75286da369fbf10f9f3c7cebb465f718b
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 30 13:45:15 2018 -0400

    adding sast and semant from micro c

commit 18470c12dc34f50f75b222426ad9561d8c9a9599
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Tue Oct 23 13:38:04 2018 -0400

```
    fixed listaccess to be expr instead of int

commit 11818e0e24332394b834c016cd072a06e62f2780
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Oct 22 18:45:00 2018 -0400

    removed dot

commit 2636302d9dee16d301ef36bac67cdbf09561c32e
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Oct 22 18:44:51 2018 -0400

    added exprs and typs

commit e0627084d49f3dfd0f6f682717461614ec1934c6
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date:   Mon Oct 22 18:14:09 2018 -0400

    add ast

commit d9505b86b1d42618cd10aa4e5487af7a66958fd7
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Oct 22 18:12:28 2018 -0400

    fixed end of comment char

commit 39e4d0e2939b84cbf068e52ef57aa090b07d23f7
Author: Zoe Gordin <zoeg21@gmail.com>
Date:   Mon Oct 22 18:11:18 2018 -0400

    updated microC files

commit 55b0d16ec498d59477a9c6680d82cabcf517bd73
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date:   Sat Oct 13 11:18:24 2018 -0400

    add dot

commit 938b810b24da6bd5a63a7a5f277bd2caf85ddd27
```

Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Sat Oct 13 11:16:53 2018 -0400

    fix list indexing

commit 32f916f04b1d2ac166633337a343add92754aa43
Author: Nadia Saleh <nadiasaleh26@gmail.com>
Date: Fri Oct 12 17:11:34 2018 -0400

    move tuple definition to vdecl

commit 087a48a701fc86f9128eef6eff641590212fe4bf
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Fri Oct 12 13:46:41 2018 -0400

    added lists, float,s strings

commit 31b20a3c230cb2a9f76d2e290198e88f9518eea7
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Fri Oct 12 12:17:30 2018 -0400

    adding zen parser

commit 83fda7160af6dd7cae7d233dc69fd9361f1e06e9
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Fri Oct 12 11:51:01 2018 -0400

    zen scanner

commit 7b1df392280a7b59197161cc6b5a1680d39dc759
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Fri Oct 12 11:29:00 2018 -0400

    rename micro c files

commit 15037f873ff0963ca2129296d30636aea1b40d2e
Author: Zoe Gordin <zoeg21@gmail.com>
Date: Fri Oct 12 11:24:30 2018 -0400

adding microc starter files

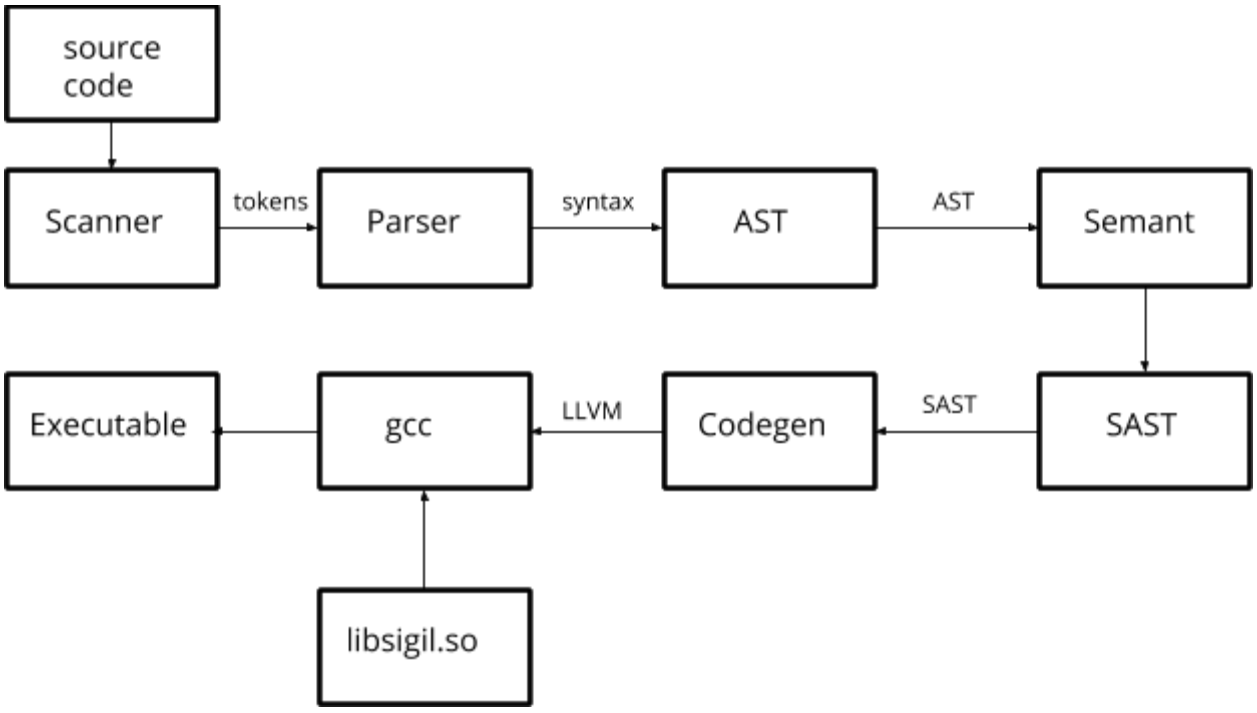
commit 89fd4336e209436f672a5a44ca9536dcd64a9b91
Author: Eleanor Murguia <eleanormurguia@gmail.com>
Date: Fri Oct 12 11:17:10 2018 -0400

first commit

5 Architectural Design

5.1 The Compiler

The architecture of the ZEN compiler consists of: Scanner, Parser, Abstract Syntax Tree (AST), Semantic Checker, Semantic Abstract Syntax Tree (SAST), Code Generator, and a linked C library. The architecture is shown as a diagram below.



The frontend of the ZEN to LLVM compiler `zen.ml` calls each of the above components in order. When a user executes "`make zen.native`" the entire compiler sequence is run. First, ZEN source code is passed through the scanner and parser, generating the AST. The AST is passed through the semantic checker, which

evaluates types and confirms that semantic rules are being followed. The AST is then converted to a SAST, which is passed along to the code generation step, which converts the SAST to LLVM. The LLVM code is passed to the gcc compiler, which links in the C library files and outputs an executable file.

5.1.1 Scanner

The scanner takes a ZEN program as input and generates tokens that can identify keywords and symbols that are further detailed in the LRM. When generating these tokens, whitespace and comments are removed and not passed through to the parser.

5.1.2 Parser

The parser evaluates the tokens output by the scanner and creates the AST through following the grammar in the parser that indicates precedence and matches tokens with AST types.

5.1.3 Semant

The semantic checker walks over the AST to check types, and returns an SAST.

5.1.4 Codegen

The code generator evaluates an SAST and outputs the equivalent in LLVM.

5.2 C Library

The C library that ZEN utilizes to create graphics is the Sound, Input, and Graphics Integration Library (SIGIL)[1], created by Geoff Nagy. SIGIL depends on the Simple DirectMedia Layer (SDL) library[2] for its window creation, as well as other graphical functionality. SDL is a popular library that is intended to provide a layer of abstraction for multimedia components of programs like games. A collection of C files were written that rely on functions within SIGIL, and these files compiled to object files and linked with gcc during compilation of ZEN code.

[1] https://gitlab.com/geoff-nagy/sigil
[2] https://www.libsdl.org/

5.3 Teamwork

We all collaborated to construct the agreed upon architecture, and each team member was responsible for specific features throughout the entire architecture.

6 Test Plan

6.1 Test Suite

All ZEN tests are in the `/tests` directory. These include tests that produce proper output and tests that are intended to fail with a particular error.

Tests which are intended to fail are indicated as `fail-element.zen`. The expected output for these tests is indicated in a `fail-element.err` file.

Tests which are intended to succeed are indicted as `test-element.zen`. The expected output for these tests is indicated in a `test-element.out` file.

6.2 Complications

Automated ZEN tests are capable of comparing actual text output to expected text output, but cannot do the same for graphical output. This is a limitation of SIGIL, which does not include the capability to save the generated graphics to a file. Our test suite thus tests graphical functions by simply ensuring that they run completely, and then print “Success” or “Failure” to an output file as expected. This cannot check that the program produced the correct graphic, but can make sure that all the graphical functions executed without error.

6.3 Automation

ZEN test automation is enabled by the `testall.sh` script, which has been modified from the MicroC script. `testall.sh` compiles and runs all ZEN testing files in the `/tests` directory and compares the output of the file to the corresponding `.err` or `.out` file. The user can then see the results of each test in the terminal where they are running the script - `OK` if the outputs match, `FAIL` if the outputs differ, or a different compilation error if something else went wrong.

6.4 Sample Source Code and Output

In a `tests` directory containing a test file, `test-hello_world.zen`

```
int main() {
      print("Hello World");
      return 0;
}
```

With an expected output file, `test-hello_world.out`

```
Hello World
```

And a failing test file, `fail-assign.zen`

```
int main()
{
  int i;
  float b;

  i = 42;
  b = 6.75;
  i = b;
  return 0;
}
```

With an expected output file, `fail-assign.err`

```
Fatal error: exception Failure("illegal assignment int = float in i =
b")
```

Running

```
$ make test
```

Gives

```
./testall.sh
test-hello_world...OK
fail-assign...OK
```

6.5 Teamwork

Each team member executed testing for features that they implemented, and the Tester completed a robust testing suite once the feature was integrated. The tester also translated many of the MicroC tests to ZEN syntax to ensure basics such as primitive types were working as expected from the beginning.

7 Lessons Learned

7.1 Zoë Gordin

Over the course of this project, I learned a great deal about functional programming (particularly OCaml), LLVM, and about how to approach such a monumental and team-oriented project. One essential skill that I have hopefully improved upon over the course of this project is predicting how long implementing certain features will take. For instance, I had no idea how time-intensive linking SIGIL would be, but because of issues with Docker, I ended up spending a fair amount of time troubleshooting, and eventually switching to a virtual machine. Additionally, I learned a lot about the power of version control, the importance of going to TAs for help, and how to interpret and add to an existing codebase. My advice for future teams is to spend time researching to accurately estimate how long implementing various features will take, as well as examining how groups from past semesters have approached the project!

7.2 Eleanor Murguia

My biggest learning over the course of this project was how much can be accomplished with a few lines of OCaml. I spent a great deal of time, and wrote many, many lines of codes, trying various solutions to get the implementation of

tuples to function. My eventual solution ended up being much simpler than anything I had previously tried and only about three lines of code. My advice to future teams would be to dive into the code as early as possible, as I don't think I would have been able to understand the architecture without trying and failing so much.

7.2 Nadia Saleh

I definitely learned the value of maintaining a test suite. Thorough testing has always been an afterthought in my CS classes, and in internships, I've always had the luxury of testing infrastructure already in place so testing could continue to be an afterthought in my mind. Working on this project really showed me the value of having an extensive lineup of tests and running them often. I really could not have predicted how often tests would fail due to some seemingly unrelated change a week after I'd had that particular test working. I also learned the value of working slowly, little by little. OCaml starts to look like gibberish really quickly when you're trying to spot a bug, and I found myself most productive and successful when I only worked for an hour or two at a time, and then came back to frustrating things another day.

8 Appendix

All files were authored by all three members of the team.

scanner.mll

```
(* Ocamllex scanner for ZEN *)

{ open Parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "#"     { comment lexbuf }           (* Comments *)
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| '['        { LSQUARE }
```

```
| ']'      { RSQUARE }
| ';'      { SEMI }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| '='      { ASSIGN }
| "%"        { MOD }
| "."      { DOT }
| "=="     { EQ }
| "!="     { NEQ }
| "<"      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "and"    { AND }
| "or"     { OR }
| "!"      { NOT }
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "while"  { WHILE }
| "return" { RETURN }
| "int"    { INT }
| "float"  { FLOAT }
| "bool"   { BOOL }
| "string" { STRING }
| "true"   { TRUE }
| "false"  { FALSE }
| "tuple"  { TUPLE }
| "void"   { VOID}
| ['0'-'9']+ as lxm { INT_LITERAL(int_of_string lxm) }
| ['0'-'9']+'.'['0'-'9']+ as lxm { FLOAT_LITERAL(float_of_string
lxm)}

| '"'([^'"']* as lxm)'"' { STRING_LITERAL(lxm) }
```

```
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm {
ID(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped
char)) }


and comment = parse
  "\n" { token lexbuf }
| _    { comment lexbuf }
```

parser.mly

```
/* Ocamlyacc parser for ZEN */

%{
open Ast
%}

%token SEMI LPAREN RPAREN LBRACE RBRACE LSQUARE RSQUARE COMMA
%token PLUS MINUS TIMES DIVIDE ASSIGN NOT
%token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR MOD DOT
%token RETURN IF ELSE FOR WHILE INT BOOL FLOAT STRING VOID
%token TUPLE
%token <int> INT_LITERAL
%token <float> FLOAT_LITERAL
%token <string> STRING_LITERAL
%token <string> ID
%token <float * float> TUPLE_LITERAL
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
```

```
%left TIMES DIVIDE MOD
%right NOT NEG

%start program
%type <Ast.program> program

%%

program:
  decls EOF { $1 }


decls:
   /* nothing */ { [], [] }
  | decls vdecl { (($2 :: fst $1), snd $1) }
  | decls fdecl { (fst $1, ($2 :: snd $1)) }

fdecl:
   typ ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list
RBRACE
     { { typ = $1;
      fname = $2;
      formals = List.rev $4;
      locals = List.rev $7;
      body = List.rev $8 } }

formals_opt:
    /* nothing */ { [] }
  | formal_list   { $1 }


formal_list:
  typ ID                   { [($1,$2)] }
  | formal_list COMMA typ ID  { ($3,$4) :: $1 }

typ:
  INT { Int }
| BOOL { Bool }
| FLOAT { Float }
```

```
| STRING { String }
| TUPLE { Tuple }
| array_t { $1 }
| VOID { Void }

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

vdecl:
   /*local_typ ID SEMI { ($1, $2) }*/
   typ ID SEMI { ($1, $2) }

array_t:
  typ LSQUARE expr RSQUARE { Array($1, $3) }

stmt_list:
    /* nothing */  { [] }
  | stmt_list stmt { $2 :: $1 }

stmt:
    expr SEMI { Expr $1 }
  | RETURN SEMI { Return Noexpr }
  | RETURN expr SEMI { Return $2 }
  | LBRACE stmt_list RBRACE { Block(List.rev $2) }
  | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5,
Block([])) }
  | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7) }
  | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
     { For($3, $5, $7, $9) }
  | WHILE LPAREN expr RPAREN stmt { While($3, $5) }

expr_opt:
    /* nothing */ { Noexpr }
  | expr          { $1 }

expr:
   literals { $1 }
  | ID               { Id($1) }
```

```
  | expr PLUS   expr { Binop($1, Add,   $3) }
  | expr MINUS  expr { Binop($1, Sub,   $3) }
  | expr TIMES  expr { Binop($1, Mult,  $3) }
  | expr DIVIDE expr { Binop($1, Div,   $3) }
  | expr EQ     expr { Binop($1, Equal, $3) }
  | expr NEQ    expr { Binop($1, Neq,   $3) }
  | expr LT     expr { Binop($1, Less,  $3) }
  | expr LEQ    expr { Binop($1, Leq,   $3) }
  | expr GT     expr { Binop($1, Greater, $3) }
  | expr GEQ    expr { Binop($1, Geq,   $3) }
  | expr AND    expr { Binop($1, And,   $3) }
  | expr OR     expr { Binop($1, Or,    $3) }
  | expr MOD    expr { Binop($1, Mod, $3)}
  | MINUS expr %prec NEG { Unop(Neg, $2) }
  | NOT expr         { Unop(Not, $2) }
  | ID ASSIGN expr   { Assign($1, $3) }
  | ID LSQUARE expr RSQUARE ASSIGN expr   { ArrayAssign($1, $3,
$6) }
  | ID LSQUARE expr RSQUARE     { ArrayAccess($1, $3) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | ID DOT INT_LITERAL  { TupleAccess($1, $3)}
  | LPAREN expr COMMA expr RPAREN { TupleLiteral($2,$4) }
  | LPAREN expr RPAREN { $2 }

primitive_literals:
  INT_LITERAL      { IntLiteral($1) }
  | FLOAT_LITERAL    { FloatLiteral(string_of_float $1) }
  | STRING_LITERAL   { StringLiteral($1) }
  | TRUE             { BooleanLiteral(true) }
  | FALSE            { BooleanLiteral(false) }

literals:
  primitive_literals { $1 }
  | LSQUARE array_literal RSQUARE { ArrayLiteral(List.rev $2) }

array_literal:
  primitive_literals { [$1] }
  | array_literal COMMA primitive_literals { $3 :: $1}
```

```
actuals_opt:
    /* nothing */ { [] }
  | actuals_list  { List.rev $1 }

actuals_list:
    expr                        { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }
```

ast.ml

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

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

type uop = Neg | Not

type expr =
    IntLiteral of int
  | FloatLiteral of string
  | BooleanLiteral of bool
  | StringLiteral of string
  | TupleLiteral of expr * expr
  | ArrayLiteral of expr list
  | ArrayAccess of string * expr
  | ArrayAssign of string * expr * expr
  | Id of string
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of string * expr
  | Call of string * expr list
  | TupleAccess of string * int
  | Noexpr

  type typ =
    Int
  | Bool
  | Float
```

```
  | Tuple
  | Array of typ * expr
  | String
  | Void

type bind = typ * string

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

type func_decl = {
    typ : typ;
    fname : string;
    formals : bind list;
    locals : bind list;
    body : stmt list;
  }

type program = bind list * func_decl list

(* Pretty-printing functions *)

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

```
  | Or -> "||"
  | Mod -> "%"

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

let rec string_of_expr = function
    IntLiteral(l) -> string_of_int l
  | FloatLiteral(l) -> l
  | BooleanLiteral(true) -> "true"
  | BooleanLiteral(false) -> "false"
  | StringLiteral(s) -> s
  | TupleLiteral(e1, e2) -> "(" ^ string_of_expr e1 ^ ", " ^
string_of_expr e2 ^ ")"
  | ArrayLiteral(el) -> "[" ^ String.concat ", " (List.map (fun
e -> string_of_expr e) el) ^ "]"
  | TupleAccess(s1, s2) -> s1  ^ "." ^ string_of_int s2
  | Id(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(a, e) -> a ^ " = " ^ string_of_expr e
  | ArrayAccess(a, e) -> a ^ "[" ^ string_of_expr e ^ "]"
  | ArrayAssign(a, e1, e2) -> a ^ "[" ^ string_of_expr e1 ^ "] =
" ^ string_of_expr e2
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el)
^ ")"
  | 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
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^
string_of_stmt s

and string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Float -> "float"
  | Array(t, e) -> string_of_typ t ^ "[" ^ string_of_expr e ^
"]"
  | Tuple -> "tuple"
  | String -> "string"
  | Void -> "void"

let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"

let string_of_fdecl fdecl =
  string_of_typ fdecl.typ ^ " " ^
  fdecl.fname ^ "(" ^ String.concat ", " (List.map snd
fdecl.formals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.locals) ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_program (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs)
```

semant.ml

```
(* Semantic checking for the MicroC compiler *)
```

```
open Ast
open Sast

module StringMap = Map.Make(String)

(* Semantic checking of the AST. Returns an SAST if successful,
   throws an exception if something is wrong.

   Check each global variable, then check each function *)

let check (globals, functions) =

  (* Verify a list of bindings has no void types or duplicate
names *)
  let check_binds (kind : string) (binds : bind list) =
    List.iter (function
     (Void, b) -> raise (Failure ("illegal void " ^ kind ^ " " ^
b))
      | _ -> ()) binds;
    let rec dups = function
        [] -> ()
      |   ((_,n1) :: (_,n2) :: _) when n1 = n2 ->
       raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
      | _ :: t -> dups t
    in dups (List.sort (fun (_,a) (_,b) -> compare a b) binds)
  in

  (**** Check global variables ****)

  check_binds "global" globals;

  (**** Check functions ****)

  (* Collect function declarations for built-in functions: no
bodies *)
  (*change add bind: typ = return type, fname name, formals is
parameters*)
  (*functions that aren't from libraries, manually fill in in
codegen with LLVM generated by OCaml*)
```

```
  let built_in_decls =
    let add_bind map (name, ty) = StringMap.add name {
      typ = Void;
      fname = name;
      formals = [(ty, "x")];
      locals = []; body = [] } map
    in let funct_map = List.fold_left add_bind StringMap.empty [
                    ("print", String);
                                        ("printf", Float);
                                        ("printi", Int);
                    ("printb", Bool);
                               ]
    in
     let add_bind2 map (name, ty1, ty2, ty3, ty4) =
StringMap.add name {
      typ = Int;
      fname = name;
      formals = [(ty1, "x");(ty2, "y");(ty3, "height");(ty4,
"width")];
      locals = []; body = [] } map
    in let funct_map2 = List.fold_left add_bind2 funct_map [
                                ("make_triangle", Int, Int, Int,
Int);
                                ("make_rectangle", Int, Int, Int,
Int);

                                 ]


    in
     let add_bind3 map (name, ty1, ty2, ty3, ty4) =
StringMap.add name {
      typ = Int;
      fname = name;
      formals = [(ty1, "x");(ty2, "y");(ty3, "radius");(ty4,
"vertices")];
      locals = []; body = [] } map
     in
     let funct_map3 = List.fold_left add_bind3 funct_map2 [
```

```
                                ("make_circle", Int, Int, Int,
Int);
                                 ]


  in
     let add_bind4 map (name, ty1, ty2) = StringMap.add name {
      typ = Void;
      fname = name;
      formals = [(ty1, "x");(ty2, "y")];
      locals = []; body = [] } map
    in
    let funct_map4 = List.fold_left add_bind4 funct_map3 [
                                ("make_point", Int, Int);
                                 ]

  in
     let add_bind5 map (name, ty1, ty2, ty3, ty4) =
StringMap.add name {
      typ = Void;
      fname = name;
      formals = [(ty1, "x1");(ty2, "y1");(ty3, "x2");(ty4,
"y2")];
      locals = []; body = [] } map
    in
     let funct_map5 = List.fold_left add_bind5 funct_map4 [
                                ("make_line", Int, Int, Int,
Int);
                                 ]
  in
    let add_bind6 map (name) = StringMap.add name {
      typ = Int;
      fname = name;
      formals = [];
      locals = []; body = [] } map
    in
     let funct_map6 = List.fold_left add_bind6 funct_map5 [
                                ("make_window");
                                ("close_window");
```

```
                                ]
  in
    let add_bind7 map (name) = StringMap.add name {
      typ = Bool;
      fname = name;
      formals = [];
      locals = []; body = [] } map
    in
     List.fold_left add_bind7 funct_map6 [
                                ("keep_open");
                                ("render");
                                 ]



  (* Add function name to symbol table *)

in
  let add_func map fd =
    let built_in_err = "function " ^ fd.fname ^ " may not be
defined"
    and dup_err = "duplicate function " ^ fd.fname
    and make_err er = raise (Failure er)
    and n = fd.fname (* Name of the function *)
    in match fd with (* No duplicate functions or redefinitions
of built-ins *)
         _ when StringMap.mem n built_in_decls -> make_err
built_in_err
       | _ when StringMap.mem n map -> make_err dup_err
       | _ ->  StringMap.add n fd map
  in

  (* Collect all function names into one symbol table *)
  let function_decls = List.fold_left add_func built_in_decls
functions
  in

  (* Return a function from our symbol table *)
  let find_func s =
```

```
    try StringMap.find s function_decls
    with Not_found -> raise (Failure ("unrecognized function " ^
s))
  in

  let _ = find_func "main" in (* Ensure "main" is defined *)

  let check_function func =
    (* Make sure no formals or locals are void or duplicates *)
    check_binds "formal" func.formals;
    check_binds "local" func.locals;

    (* Raise an exception if the given rvalue type cannot be
assigned to
       the given lvalue type *)
    let rec check_assign lvaluet rvaluet err = match rvaluet
with
        Array(t1,_) -> (match t1 with
        Int -> check_assign t1 Int err
        | _ -> raise (Failure err))
      | _ -> if lvaluet = rvaluet then lvaluet else raise
(Failure err)
    in


    (* Build local symbol table of variables for this function
*)
    let symbols = List.fold_left (fun m (ty, name) ->
StringMap.add name ty m)
                    StringMap.empty (globals @ func.formals @
func.locals )
    in

    (* Return a variable from our local symbol table *)
    let type_of_identifier s =
      try StringMap.find s symbols
      with Not_found -> raise (Failure ("undeclared identifier "
^ s))
    in
```

```
    let access_type = function
    Array(t, _) -> t
    | _ -> raise (Failure("illegal array access"))

    in


    (* Return a semantically-checked expression, i.e., with a
type *)
    let rec expr = function
        IntLiteral  l -> (Int, SIntLiteral l)
      | FloatLiteral l -> (Float, SFloatLiteral l)
      | BooleanLiteral l  -> (Bool, SBooleanLiteral l)
      | StringLiteral l -> (String, SStringLiteral l)
      | ArrayLiteral l -> check_array_types l
      | ArrayAccess(a, e) -> check_int_expr e;
(type_of_identifier a, SArrayAccess(a, expr e, access_type
(type_of_identifier a)))
      | ArrayAssign(var, idx, num) -> check_int_expr num;
check_int_expr idx; (type_of_identifier var, SArrayAssign(var,
expr idx, expr num))
      | TupleLiteral (x, y) -> let t1 = expr x and t2 = expr y
in
        (Tuple, STupleLiteral (t1, t2))
      | TupleAccess (s1, s2) ->  (Int, STupleAccess(s1, s2))
      | Noexpr     -> (Void, SNoexpr)
      | Id s       -> (type_of_identifier s, SId s)
      | Assign(var, e) as ex ->
          let lt = type_of_identifier var
          and (rt, e') = expr e in
          let err = "illegal assignment " ^ string_of_typ lt ^ "
= " ^
            string_of_typ rt ^ " in " ^ string_of_expr ex
          in (check_assign lt rt err, SAssign(var, (rt, e')))
      | Unop(op, e) as ex ->
          let (t, e') = expr e in
          let ty = match op with
```

```
            Neg when t = Int || t = Float -> t
          | Not when t = Bool -> Bool
          | _ -> raise (Failure ("illegal unary operator " ^
                                 string_of_uop op ^
string_of_typ t ^
                                 " in " ^ string_of_expr ex))
          in (ty, SUnop(op, (t, e')))
      | Binop(e1, op, e2) as e ->
          let (t1, e1') = expr e1
          and (t2, e2') = expr e2 in
          (* All binary operators require operands of the same
type *)
          let same = t1 = t2 in
          (* Determine expression type based on operator and
operand types *)
          let ty = match op with
            Add | Sub | Mult | Div when same && t1 = Int   ->
Int
          | Add | Sub | Mult | Div when same && t1 = Float ->
Float
          | Equal | Neq            when same               ->
Bool
          | Less | Leq | Greater | Geq
                     when same && (t1 = Int || t1 = Float) ->
Bool
          | And | Or when same && t1 = Bool -> Bool
          | Mod when same && t1 = Int -> Int
          | _ -> raise (
           Failure ("illegal binary operator " ^
                      string_of_typ t1 ^ " " ^ string_of_op op
^ " " ^
                      string_of_typ t2 ^ " in " ^
string_of_expr e))
          in (ty, SBinop((t1, e1'), op, (t2, e2')))
      | Call(fname, args) as call ->
          let fd = find_func fname in
          let param_length = List.length fd.formals in
          if List.length args != param_length then
```

```
            raise (Failure ("expecting " ^ string_of_int
param_length ^
                            " arguments in " ^ string_of_expr
call))
          else let check_call (ft, _) e =
            let (et, e') = expr e in
            let err = "illegal argument found " ^ string_of_typ
et ^
              " expected " ^ string_of_typ ft ^ " in " ^
string_of_expr e
            in (check_assign ft et err, e')
          in
          let args' = List.map2 check_call fd.formals args

          in
          (fd.typ, SCall(fname, args'))

    and get_arr_type e = match e with
      IntLiteral(_) :: ss -> get_arr_type ss
      | [] -> Int
      | _ -> raise (Failure("arrays only ints"))



    and check_array_types e =
      let t = get_arr_type e in
      let check_arr_el e = match e with
        IntLiteral(i) -> if t == Int then expr(IntLiteral(i))
else expr(FloatLiteral(string_of_int i))
        | _ -> raise (Failure("arrays only ints"))
      in (Array (t, IntLiteral(List.length e)),
SArrayLiteral(List.map check_arr_el e, Array(t,
IntLiteral(List.length e))))

    and check_int_expr e =
      let (t', e') = expr e
      and err = "expected Int expression in " ^ string_of_expr e
      in if t' != Int then raise (Failure err) else ignore e'
```

```
  in

    let check_bool_expr e =
      let (t', e') = expr e
      and err = "expected Boolean expression in " ^
string_of_expr e
      in if t' != Bool then raise (Failure err) else (t', e')
    in



    (* Return a semantically-checked statement i.e. containing
sexprs *)
    let rec check_stmt = function
        Expr e -> SExpr (expr e)
      | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1,
check_stmt b2)
      | For(e1, e2, e3, st) ->
       SFor(expr e1, check_bool_expr e2, expr e3, check_stmt st)
      | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
      | Return e -> let (t, e') = expr e in
        if t = func.typ then SReturn (t, e')
        else raise (
       Failure ("return gives " ^ string_of_typ t ^ " expected "
^
            string_of_typ func.typ ^ " in " ^ string_of_expr
e))

         (* A block is correct if each statement is correct and
nothing
           follows any Return statement.  Nested blocks are
flattened. *)
      | Block sl ->
          let rec check_stmt_list = function
              [Return _ as s] -> [check_stmt s]
            | Return _ :: _   -> raise (Failure "nothing may
follow a return")
```

```
            | Block sl :: ss  -> check_stmt_list (sl @ ss) (*
Flatten blocks *)
            | s :: ss         -> check_stmt s :: check_stmt_list
ss
            | []              -> []
          in SBlock(check_stmt_list sl)

    in (* body of check_function *)
    { styp = func.typ;
      sfname = func.fname;
      sformals = func.formals;
      slocals  = func.locals;
      sbody = match check_stmt (Block func.body) with
     SBlock(sl) -> sl
      | _ -> raise (Failure ("internal error: block didn't
become a block?"))
    }
  in (globals, List.map check_function functions)
```

sast.ml

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

open Ast

type sexpr = typ * sx
and sx =
    SIntLiteral of int
  | SFloatLiteral of string
  | SBooleanLiteral of bool
  | SStringLiteral of string
  | STupleLiteral of sexpr * sexpr
  | SArrayLiteral of sexpr list * typ
  | SArrayAccess of string * sexpr * typ
  | SArrayAssign of string * sexpr * sexpr
  | SId of string
  | SBinop of sexpr * op * sexpr
  | SUnop of uop * sexpr
```

```
  | SAssign of string * sexpr
  | SCall of string * sexpr list
  | STupleAccess of string * int
  | SNoexpr

type sstmt =
    SBlock of sstmt list
  | SExpr of sexpr
  | SReturn of sexpr
  | SIf of sexpr * sstmt * sstmt
  | SFor of sexpr * sexpr * sexpr * sstmt
  | SWhile of sexpr * sstmt

type sfunc_decl = {
    styp : typ;
    sfname : string;
    sformals : bind list;
    slocals : bind list;
    sbody : sstmt list;
  }

type sprogram = bind list * sfunc_decl list

(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : " ^ (match e with
    SIntLiteral(l) -> string_of_int l
  | SBooleanLiteral(true) -> "true"
  | SBooleanLiteral(false) -> "false"
  | SStringLiteral(s) -> s
  | SFloatLiteral(l) -> l
  | SArrayLiteral(el, t) -> string_of_typ t ^ "[" ^
String.concat ", " (List.map (fun e -> string_of_sexpr e) el) ^
"]"
  | SArrayAccess(a, e, t) -> string_of_typ t ^ " " ^ a ^ "[" ^
string_of_sexpr e ^ "]"
  | SArrayAssign(a, e1, e2) -> a ^ "[" ^ string_of_sexpr e1 ^ "]
= " ^ string_of_sexpr e2
```

```
  | STupleLiteral(e1, e2) -> "(" ^ string_of_sexpr e1 ^ ", " ^
string_of_sexpr e2 ^ ")"

  | SId(s) -> s
  | SBinop(e1, o, e2) ->
      string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^
string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
  | STupleAccess(s1, s2) -> s1 ^ "." ^ string_of_int s2
  | SCall(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el)
^ ")"
  | SNoexpr -> ""
                      ) ^ ")"

let rec string_of_sstmt = function
    SBlock(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_sstmt stmts)
^ "}\n"
  | SExpr(expr) -> string_of_sexpr expr ^ ";\n";
  | SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n";
  | SIf(e, s, SBlock([])) ->
      "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
  | SIf(e, s1, s2) ->  "if (" ^ string_of_sexpr e ^ ")\n" ^
      string_of_sstmt s1 ^ "else\n" ^ string_of_sstmt s2
  | SFor(e1, e2, e3, s) ->
      "for (" ^ string_of_sexpr e1  ^ " ; " ^ string_of_sexpr e2
^ " ; " ^
      string_of_sexpr e3  ^ ") " ^ string_of_sstmt s
  | SWhile(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^
string_of_sstmt s

let string_of_sfdecl fdecl =
  string_of_typ fdecl.styp ^ " " ^
  fdecl.sfname ^ "(" ^ String.concat ", " (List.map snd
fdecl.sformals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.slocals) ^
```

```
  String.concat "" (List.map string_of_sstmt fdecl.sbody) ^
  "}\n"

let string_of_sprogram (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_sfdecl funcs)
```

codegen.ml

```
(* Code generation: translate takes a semantically checked AST
and
produces LLVM IR

LLVM tutorial: Make sure to read the OCaml version of the
tutorial

http://llvm.org/docs/tutorial/index.html

Detailed documentation on the OCaml LLVM library:

http://llvm.moe/
http://llvm.moe/ocaml/

*)

(* We'll refer to Llvm and Ast constructs with module names *)
module L = Llvm
module A = Ast
open Sast

module StringMap = Map.Make(String)

(* Code Generation from the SAST. Returns an LLVM module if
successful,
   throws an exception if something is wrong. *)
let translate (globals, functions) =
  let context    = L.global_context () in
  (* Add types to the context so we can use them in our LLVM
code *)
```

```
  let i32_t      = L.i32_type    context
  and i8_t       = L.i8_type     context
  and i1_t       = L.i1_type     context
  and float_t    = L.double_type context
  and array_t    = L.array_type
  and void_t     = L.void_type   context
  (* Create an LLVM module -- this is a "container" into which
we'll
     generate actual code *)
  and the_module = L.create_module context "Zen" in

  let str_t = L.pointer_type i8_t in
  let tuple_t = L.named_struct_type context "tuple_t" in
    L.struct_set_body tuple_t [| i32_t; i32_t|] false;

  let int_lit_to_int = function
    A.IntLiteral(i) -> i | _ -> raise(Failure("arrays must be
int"))

  in
  (* Convert MicroC types to LLVM types *)
  let ltype_of_typ = function
      A.Int   -> i32_t
    | A.Bool  -> i1_t
    | A.Float -> float_t
    | A.Void  -> void_t
    | A.String -> str_t
    | A.Tuple -> tuple_t
    | A.Array(typ, size) -> (match typ with
                               A.Int -> array_t i32_t
(int_lit_to_int size)
                               | _ -> raise(Failure("arrays must
be int")))
  in


  (* Declare each global variable; remember its value in a map
*)
  let global_vars : L.llvalue StringMap.t =
```

```
    let global_var m (t, n) =
      let init = match t with
          A.Float -> L.const_float (ltype_of_typ t) 0.0
        | _ -> L.const_int (ltype_of_typ t) 0
      in StringMap.add n (L.define_global n init the_module) m
in
    List.fold_left global_var StringMap.empty globals in

  let printf_t : L.lltype = L.var_arg_function_type i32_t [|
L.pointer_type i8_t |] in
  let printf_func : L.llvalue = L.declare_function "printf"
printf_t the_module in

  let make_triangle_t = L.function_type i32_t [| i32_t; i32_t;
i32_t; i32_t |] in
  let make_triangle_func = L.declare_function "make_triangle"
make_triangle_t the_module in

  let make_rectangle_t = L.function_type i32_t [| i32_t; i32_t;
i32_t; i32_t |] in
  let make_rectangle_func = L.declare_function "make_rectangle"
make_rectangle_t the_module in

  let make_circle_t = L.function_type i32_t [| i32_t; i32_t;
i32_t; i32_t |] in
  let make_circle_func = L.declare_function "make_circle"
make_circle_t the_module in

  let make_point_t = L.function_type i32_t [| i32_t; i32_t; |]
in
  let make_point_func = L.declare_function "make_point"
make_point_t the_module in

  let make_line_t = L.function_type i32_t [| i32_t; i32_t;
i32_t; i32_t |] in
  let make_line_func = L.declare_function "make_line"
make_line_t the_module in

  let make_window_t = L.function_type i32_t [||] in
```

```
  let make_window_func = L.declare_function "make_window"
make_window_t the_module in

  let close_window_t = L.function_type i32_t [||] in
  let close_window_func = L.declare_function "close_window"
close_window_t the_module in

  let keep_open_t = L.function_type i1_t [||] in
  let keep_open_func = L.declare_function "keep_open"
keep_open_t the_module in

  let render_t = L.function_type i1_t [||] in
  let render_func = L.declare_function "render" render_t
the_module in

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

  (* Define each function (arguments and return type) so we can
   * define it's body and call it later *)
  let function_decls : (L.llvalue * sfunc_decl) StringMap.t =
    let function_decl m fdecl =
      let name = fdecl.sfname
      and formal_types =
     Array.of_list (List.map (fun (t,_) -> ltype_of_typ t)
fdecl.sformals)
      in let ftype = L.function_type (ltype_of_typ fdecl.styp)
formal_types in
      StringMap.add name (L.define_function name ftype
the_module, fdecl) m in
    List.fold_left function_decl StringMap.empty functions in

  (* Fill in the body of the given function *)
  let build_function_body fdecl =
    let (the_function, _) = StringMap.find fdecl.sfname
function_decls in
```

```
    let builder = L.builder_at_end context (L.entry_block
the_function) in

    let str_format_str = L.build_global_stringptr "%s\n"
"str_fmt" builder
    and float_format_str = L.build_global_stringptr "%g\n" "fmt"
builder
    and int_format_str = L.build_global_stringptr "%d\n"
"int_fmt" builder 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 (t, n) p =
        let () = L.set_value_name n p in
     let local = L.build_alloca (ltype_of_typ t) n builder in
        let _  = L.build_store p local builder in
     StringMap.add n local m
      in

      (* Allocate space for any locally declared variables and
add the
       * resulting registers to our map *)
      let add_local m (t, n) =
     let local_var = L.build_alloca (ltype_of_typ t) n builder
     in StringMap.add n local_var m
      in

      let formals = List.fold_left2 add_formal StringMap.empty
fdecl.sformals
          (Array.to_list (L.params the_function)) in
      List.fold_left add_local formals fdecl.slocals
    in
```

```
    (* Return the value for a variable or formal argument. First
check
     * locals, then globals *)
    let lookup n = try StringMap.find n local_vars
                   with Not_found -> StringMap.find n
global_vars
    in

    (* Construct code for an expression; return its value *)
    let rec expr builder ((_, e) : sexpr) = match e with
           SIntLiteral i -> L.const_int i32_t i
      | SBooleanLiteral b -> L.const_int i1_t (if b then 1 else
0)
      | SFloatLiteral l -> L.const_float_of_string float_t l
      | SStringLiteral s -> L.build_global_stringptr s "name"
builder
      | SArrayLiteral (l, t) -> L.const_array (ltype_of_typ t)
(Array.of_list (List.map (expr builder) l))
      | SArrayAccess (s, e, _) -> L.build_load
(get_array_acc_address s e builder) s builder
      | SArrayAssign (s, e1, e2) ->
        let lsb = get_array_acc_address s e1 builder
                     in
                     let rsb = expr builder e2 in
                     ignore (L.build_store rsb lsb builder);
rsb
      | STupleLiteral (x, y) ->
        let x' = ensureInt (expr builder x)
        and y' = ensureInt (expr builder y) in
        let t_ptr = L.build_alloca tuple_t "tmp" builder in
        let x_ptr = L.build_struct_gep t_ptr 0 "x" builder in
        ignore (L.build_store x' x_ptr builder);
        let y_ptr = L.build_struct_gep t_ptr 1 "y" builder in
        ignore(L.build_store y' y_ptr builder);
        L.build_load (t_ptr) "t" builder
      | SNoexpr -> L.const_int i32_t 0
      | SId s -> L.build_load (lookup s) s builder
      | SAssign (s, e) -> let e' = expr builder e in
```

```
                          let _  = L.build_store e' (lookup s)
builder in e'
      | SBinop (e1, op, e2) ->
       let (t, _) = e1
       and e1' = expr builder e1
       and e2' = expr builder e2 in
       if t = A.Float 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.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
    | A.Mod     -> L.build_srem
       | A.And | A.Or ->
           raise (Failure "internal error: semant should have
rejected and/or on float")
       ) e1' 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
      | SUnop(op, e) ->
       let (t, _) = e in
```

```
          let e' = expr builder e in
       (match op with
         A.Neg when t = A.Float -> L.build_fneg
       | A.Neg                  -> L.build_neg
          | A.Not                  -> L.build_not) e' "tmp"
builder
      | STupleAccess (s1, s2) ->
        let t_ptr = (lookup s1) in
        let value_ptr = L.build_struct_gep t_ptr s2 ( "t_ptr")
builder in
        L.build_load value_ptr "t_ptr" builder
      | SCall ("make_triangle", [e1; e2; e3; e4]) ->
    L.build_call make_triangle_func [| (expr builder e1); (expr
builder e2); (expr builder e3); (expr builder e4)|]
    "make_triangle" builder
      | SCall ("make_rectangle", [e1; e2; e3; e4]) ->
    L.build_call make_rectangle_func [| (expr builder e1); (expr
builder e2); (expr builder e3); (expr builder e4)|]
    "make_rectangle" builder
      | SCall ("make_circle", [e1; e2; e3; e4]) ->
    L.build_call make_circle_func [| (expr builder e1); (expr
builder e2); (expr builder e3); (expr builder e4)|]
    "make_circle" builder
      | SCall ("make_line", [e1; e2; e3; e4]) ->
    L.build_call make_line_func [| (expr builder e1); (expr
builder e2); (expr builder e3); (expr builder e4)|]
    "make_line" builder
      | SCall ("make_point", [e1; e2]) ->
    L.build_call make_point_func [| (expr builder e1); (expr
builder e2); |]
    "make_point" builder
      | SCall ("make_window", []) ->
    L.build_call make_window_func [||] "make_window" builder
     | SCall ("close_window", []) ->
    L.build_call close_window_func [||] "close_window" builder
     | SCall ("keep_open", []) ->
    L.build_call keep_open_func [||] "keep_open" builder
    | SCall ("render", []) ->
    L.build_call render_func [||] "render" builder
```

```
      | SCall ("printf", [e]) ->
       L.build_call printf_func [| float_format_str ; (expr
builder e) |]
         "printf" builder
      | SCall("print", [e]) ->
    L.build_call printf_func [| str_format_str ; (expr builder
e) |]
      "print" builder
      | SCall("printi", [e]) | SCall("printb", [e]) ->
    L.build_call printf_func [| int_format_str ; (expr builder
e) |]
      "printi" builder
      | SCall (f, args) ->
         let (fdef, fdecl) = StringMap.find f function_decls in
      let llargs = List.rev (List.map (expr builder) (List.rev
args)) in
      let result = (match fdecl.styp with
                          A.Void -> ""
                        | _ -> f ^ "_result") in
         L.build_call fdef (Array.of_list llargs) result builder

    and
     get_array_acc_address s e1 builder = L.build_gep (lookup s)
      [| (L.const_int i32_t 0); (expr builder e1) |] s builder
    in

    (* Each basic block in a program ends with a "terminator"
instruction i.e.
    one that ends the basic block. By definition, these
instructions must
    indicate which basic block comes next -- they typically
yield "void" value
    and produce control flow, not values *)
    (* Invoke "instr builder" if the current block doesn't
already
       have a terminator (e.g., a branch). *)
    let add_terminal builder instr =
                            (* The current block where we're
inserting instr *)
```

```
      match L.block_terminator (L.insertion_block builder) with
     Some _ -> ()
      | None -> ignore (instr builder) in

    (* Build the code for the given statement; return the
builder for
       the statement's successor (i.e., the next instruction
will be built
       after the one generated by this call) *)
    (* Imperative nature of statement processing entails
imperative OCaml *)
    let rec stmt builder = function
     SBlock sl -> List.fold_left stmt builder sl
        (* Generate code for this expression, return resulting
builder *)
      | SExpr e -> let _ = expr builder e in builder
      | SReturn e -> let _ = match fdecl.styp with
                              (* Special "return nothing" instr
*)
                             A.Void -> L.build_ret_void builder
                             (* Build return statement *)
                           | _ -> L.build_ret (expr builder e)
builder
                    in builder
      (* The order that we create and add the basic blocks for
an If statement
      doesnt 'really' matter (seemingly). What hooks them up in
the right order
      are the build_br functions used at the end of the then and
else blocks (if
      they don't already have a terminator) and the
build_cond_br function at
      the end, which adds jump instructions to the "then" and
"else" basic blocks *)
      | SIf (predicate, then_stmt, else_stmt) ->
         let bool_val = expr builder predicate in
         (* Add "merge" basic block to our function's list of
blocks *)
```

```
      let merge_bb = L.append_block context "merge" the_function
in
         (* Partial function used to generate branch to merge
block *)
         let branch_instr = L.build_br merge_bb in

         (* Same for "then" basic block *)
      let then_bb = L.append_block context "then" the_function
in
         (* Position builder in "then" block and build the
statement *)
         let then_builder = stmt (L.builder_at_end context
then_bb) then_stmt in
         (* Add a branch to the "then" block (to the merge
block)
           if a terminator doesn't already exist for the "then"
block *)
      let () = add_terminal then_builder branch_instr in

         (* Identical to stuff we did for "then" *)
      let else_bb = L.append_block context "else" the_function
in
         let else_builder = stmt (L.builder_at_end context
else_bb) else_stmt in
      let () = add_terminal else_builder branch_instr in

         (* Generate initial branch instruction perform the
selection of "then"
         or "else". Note we're using the builder we had access
to at the start
         of this alternative. *)
      let _ = L.build_cond_br bool_val then_bb else_bb builder
in
         (* Move to the merge block for further instruction
building *)
      L.builder_at_end context merge_bb

      | SWhile (predicate, body) ->
```

```
          (* First create basic block for condition instructions
-- this will
          serve as destination in the case of a loop *)
       let pred_bb = L.append_block context "while" the_function
in
          (* In current block, branch to predicate to execute
the condition *)
       let _ = L.build_br pred_bb builder in

          (* Create the body's block, generate the code for it,
and add a branch
          back to the predicate block (we always jump back at
the end of a while
          loop's body, unless we returned or something) *)
       let body_bb = L.append_block context "while_body"
the_function in
          let while_builder = stmt (L.builder_at_end context
body_bb) body in
       let () = add_terminal while_builder (L.build_br pred_bb)
in

          (* Generate the predicate code in the predicate block
*)
       let pred_builder = L.builder_at_end context pred_bb in
       let bool_val = expr pred_builder predicate in

          (* Hook everything up *)
       let merge_bb = L.append_block context "merge"
the_function in
       let _ = L.build_cond_br bool_val body_bb merge_bb
pred_builder in
       L.builder_at_end context merge_bb

      (* Implement for loops as while loops! *)
      | SFor (e1, e2, e3, body) -> stmt builder
         ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr
e3]) ] )
    in
```

```
  (* Build the code for each statement in the function *)
  let builder = stmt builder (SBlock fdecl.sbody) in

  (* Add a return if the last block falls off the end *)
  add_terminal builder (match fdecl.styp with
      A.Void -> L.build_ret_void
    | A.Float -> L.build_ret (L.const_float float_t 0.0)
    | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
in

List.iter build_function_body functions;
the_module
```