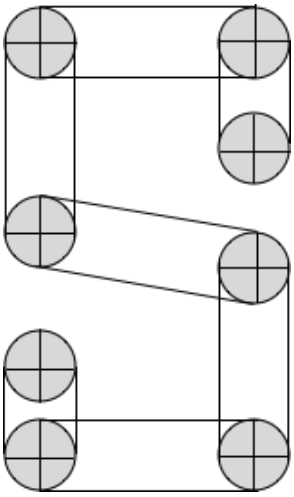


ShapeShifter

Programmatic geometric manipulations made simple

Stephanie Burgos, Ishan Guru, Rashida Kamal,
Eszter Offertaler, and Rajiv Thamburaj

Introduction 4

Language Tutorial 5

Installing dependencies 5

Setting up the Compiler 5

Using the Compiler 5

Getting Started 6

Language Reference Manual 9

1. Lexical Elements 9

1.1. Identifiers 9

1.2. Keywords 9

1.3. Separators 11

1.4. Comments 12

1.5. Operators 12

1.5.1. Arithmetic Operators 12

1.5.2. Relational and Logical Operators 12

1.5.3. Unary Operators 13

1.5.5. Other Operators 13

3. Data Types 14

2.1. Primitive Data Types 14

2.1.1. Integer 14

2.1.2. Double 14

2.1.3. Boolean 14

2.1.4. String 15

2.2. Non-Primitive Data Types 15

2.2.1. Shape 15

2.2.2. Void 16

3. Expressions 16

3.1. Assignment 17

3.2. Unary Operators 17

3.3. Binary Operators 17

4. Statements 17

4.1. Expression Statements 17

4.2. Declarations 17

4.3. Control Flow 18

4.4. Loops 18

5. Functions 18

5.1. Built-in Functions 18

5.2. User Functions 23
5.2.1. Declarations 23
5.2.2. Definitions 23
5.2.3. Calls 24
6. Program Structure 24
7. Sample Program 24

Project Plan **26**
Code Style 26
Project Timeline 27
Team Responsibilities 27
Project Log 27

Architectural Design **56**
How Shapes Work 57
The Compiler 57

Test Plan **58**
Sample .shift → .ll 58
Source: test_basic_transformations.shift 58
Target: test_basic_transformations.shift 59
Source: test_if.shift 59
Target: test_if.shift 60
Source: test_recurse_fibonnaci.shift 61
Target: test_recurse_fibonnaci.shift 61

Lessons Learned **63**

Appendix **65**
ast.ml 65
codegen.ml 68
parser.mly 83
prettyprint.ml 86
scanner.mll 89
semant.ml 91
shapeshifter.ml 96
testall.sh 97
graphics/display/main.cpp 102
graphics/cork/src/cork.cpp (with original code in gray) 114
graphics/cork/src/main.cpp (with original code in gray) 122

Introduction

ShapeShifter is a programmatic geometric modelling language that seeks to give users the ability to manipulate simple primitive shapes in order to create increasingly complex ones. Rather than requiring the user to manage meshes and vertex connectivity information, Shapeshifter abstracts these nitty-gritties away, so that the user only has to call on familiar mathematical shape names with a set of initializing properties to summon them into existence. The user does not need to worry about the underlying mathematical or technical representation of the shape in order to enjoy a visually displayed representation. Shapeshifter relies on OpenGL for the rendering via the GLUT OpenGL toolkit[1]. For mesh manipulation, Shapeshifter uses the Cork Boolean Library[2].

As much of the technical manipulations are abstracted away from the user, this report seeks to instruct the user about the nature of the Shapeshifter language, the specificities of the available lexical elements, operators, program syntax, etc., and provide instruction on how to build up increasingly complex visual representations based on the available primitives.

[1] https://www.opengl.org/resources/libraries/glut/
[2] https://github.com/gilbo/cork

Language Tutorial

ShapeShifter is a programmatic geometry manipulation language that lets you modify and render shapes to the screen. Before running your first ShapeShifter program, it is essential to install dependencies required to run the graphics libraries needed by the compiler.

Installing dependencies

Prior to running the compiler, several dependencies are required to be installed, and this can be done with the following commands:

```
sudo apt-get install freeglut3-dev
sudo apt-get install libgmp3-dev
sudo apt-get install libglew-dev
```

We also rely on a modified version of the Cork Boolean / CSG library, developed by Gilbert Bernstein. We have included it with our source code, but the original is obtainable here:

https://github.com/gilbo/cork

Setting up the Compiler

After installing the necessary dependencies, navigate to the directory you'd like to install Shapeshifter and clone the Shapeshifter repository:

```
git clone https://github.com/nyletara/ShapeShifter.git
```

From here, simply enter the folder using your preferred console and run the Makefile using the command:

```
make
```

This will generate the `shapeshifter` binary that you can use to run the `*.shift` Shapeshifter files, as seen in the following section.

Using the Compiler

Inside the Shapeshifter directory, type make. This creates the Shapeshifter executive, which takes in .shift files and compiles them down to .ll files. The syntax for running the Shapeshifter binary is as follows:

```
./shapeshifter [optional flag] < [source file].shift > [file].ll
```

The above command generates the LLVM IR code that can now be run. In the example above, the LLVM IR code is output into the .ll file that is generated.

In order to run the produced .ll file, use the command:

```
lli <file>.ll
```

Sample files to run can be found in the /shapetests/shapes/ folder, or the quick sample below, which generates a unit sphere. It can be copied into a *.shift file and run as described above.

```
int scene() {
      Shape s = SPHERE;
      Render(s);
}
```

The following flags can also be passed during compilation in order to render different outputs.

```
./shapeshifter -a < … //prints the AST only
./shapeshifter -p < … //PrettyPrint the AST
./shapeshifter -l < … //Generate LLVM IR w/o semantic checking
./shapeshifter -c < … //Compile, check LLVM IR
./shapeshifter -h < … //Show help on how to run
```

Getting Started

Take a look at the language reference manual for a full listing of what ShapeShifter can do, but here's a quick guide to get your feet wet.

Here is the absolute minimal ShapeShifter program:

```
int scene() {
}
```

`scene()` serves as an entry point to a ShapeShifter program. ShapeShifter supports integer and double primitives, as well as printing to the console:

```
int scene() {
      int x = 5;
      double y = 4.0;
      print(x);
      print(y)
}
```

You can also add functions and recursion to make things more interesting:

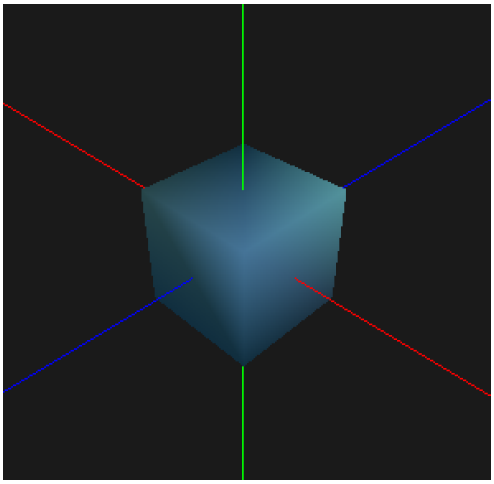
```
int fib(int n) {
      if (n == 1)
            return 1;
      if (n == 2)
            return 1;
```

```
        return fib(n-1) + fib(n-2);
    }

    int scene() {
        print(fib(5));
    }
```

But shapes are the real reason you're using ShapeShifter. We supply a series of primitives that you can use, as well as the Shape type:

```
int scene() {
    Shape cube = CUBE;
    Render(cube);
}
```

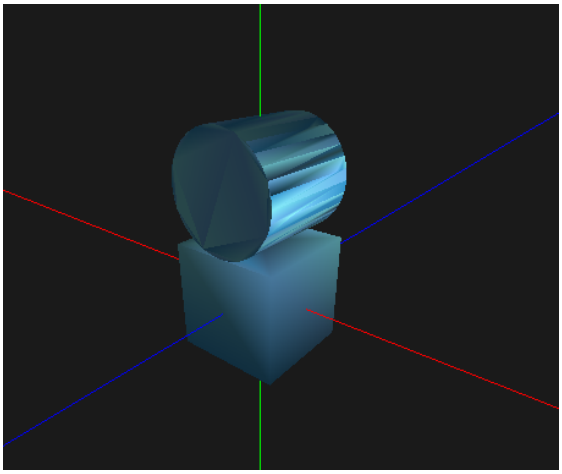
It's that easy! Run the binary that ShapeShifter creates and you will see this:



Use the arrow keys to move around in three dimensional space. Use '+' and '-' to zoom.
Press 'q' or ESC to exit.

ShapeShifter also provides a series of boolean operations that allow you to combine primitive shapes to create exciting composites!

```
int scene() {
    Shape cube = CUBE;
    Shape cylinder = CYLINDER;

    Translate(cylinder, 0.0, 1.0, 0.0);
    Shape composite = Union(cube, cylinder);
    Render(composite);
}
```



That's the end of our brief introduction - consult the LRM or code samples below for a more thorough look at ShapeShifter's capabilities.

Language Reference Manual

1. Lexical Elements

1.1. Identifiers

An identifier consists of a sequence of letters, digits and the underscore (_). The identifier must begin with a letter or underscore, but subsequent characters can be any combination of the previously mentioned characters. By convention, identifiers are are lowercase, and are most frequently used as variable names. Case does matter: abc and aBC are regarded as two unique identifiers.

Legal identifiers include: abc, _abc, abc123, _123abc, ab12c3

Illegal identifiers include: 123b, 67

1.2. Keywords

ShapeShifter has a set of keywords that cannot be used as identifiers. These include data types, booleans, control flow statements, file input/output statements and keywords for pre-defined shapes. Each of these can be seen below (more information regarding each can be seen in the tables below) :

Primitive Data Types: int, double, bool, string
Advanced Data Type: void, Shape
Booleans: true, false
Control Flow: if, else, return, for, while
I/O: Render, Save, Print
Unit Shapes: SPHERE, CUBE, CYLINDER, TETRA, CONE
Built-in Functions: Scale, Rotate, Translate, Reflect, Intersect, Union, Difference, Xor, Copy

The table below shows each of these keywords along with descriptions and sample ShapeShifter syntax.

Primitive Data Types

Keyword	Syntax
int An integer value such as in C. *More information in the Integer Data type section.*	`int x = 7`
double An 8-byte precision value. *More information in the Double Data type section.*	`double x = 7.12345`

bool A representation of the true/false logical values.	`bool x = true`
string A collection of characters supported from the local character set.	`string x = "ShapeShifter"`

Advanced Data Types

Keyword	Syntax

void
A special data type used to represent return types of 'nothing'. *More information in the Data type section.*
	`void makeHat() {`
	`  ...`
	`}`

Shape An advanced data type that represents the underlying mesh representation. *More information in the Data type section.*	`Shape s1 = SPHERE` `Shape s2 = CUBE`

Boolean Keywords

Keyword	Syntax

true
A representation of the logical value true. *More information in the Integer Data type section.*
	`bool x = true`
	`if (x == true){`
	`  ...`
	`}`

false
A representation of the logical value false. *More information in the Double Data type section.*
	`bool x = false`
	`if (x == false){`
	`  ...`
	`}`

Control Flow Keywords

Keyword	Syntax

if
Allows programs to control whether or not certain code needs to be run, provided conditions are met.
	`if(condition) {`
	`  //run this`
	`} else {`
	`  //run this`
	`}`

else Allows programs to control whether or not	`if(condition) {` `  //run this`

certain code needs to be run, provided conditions are not met. This is not optional.	`} else {`
	`  //run this`
	`}`

return
A keyword used to define the return value for a function. Every statement that comes after a return will not be executed. Each branch of execution must include a return statement corresponding to the function return type.
	`Shape makeHat {`
	`  Shape hat = ...`
	`  ...`
	`  return hat;`
	`}`

for
A key word used to represent a loop that runs a certain number of times. Similar to the for loop in C. Variables must be predefined.
	`for (i = 0; i < 10; i++ ){`
	`  ...`
	`}`

while
A keyword used to represent a special type of loop that runs while certain conditions are met.
	`while(condition) {`
	`  /*run this section of code until condition is broken*/`
	`}`

Shape Keywords

Note: Each of the Shape keywords are predefined mesh structures for the unit Shapes

Integer literals are sequences of one or more decimal digits. Negative integer literals are expressed as integer literals with a hyphen prefix.

Examples: -5, 12

Double literals consist of one or more digits, a decimal point, and one or more decimal digits. As with integer literals, negative float literals are expressed by prepending a hyphen.

Examples: 0.1, 123.456, 12.1
Invalid examples: .0, 50.

String literals are double-quoted sequences of zero or more ASCII characters. Special characters in a string can be represented with escape sequences.

Examples: "ShapeShifter is the best", "Line one\nLine two", ""

1.3. Separators

Separators in ShapeShifter are used to separate tokens. Separators include:

`, ; { } ( )`

1.4. Comments

Comments in ShapeShifter are as follows:

- single line comments go from // to the end of the line
- multiline comments are everything in between /* and */. Multiline comments cannot be nested.

For example,

```
// This is a comment in ShapeShifter

/*
      This
      Is
      Also
      a
      Comment
      In
      ShapeShifter
*/
```

1.5. Operators

ShapeShifter uses several operators to compare and combine data. Consult Figure 1 for a complete overview of relative operator precedence.

1.5.1. Arithmetic Operators

The general binary arithmetic operators are supported for numeric types, and are listed below:

```
+     -     *     /
```

Each of these are left associative, however, * and / have a higher precedence level than + and -.

1.5.2. Relational and Logical Operators

The following relational operators are all supported by ShapeShifter on numeric types, are left associative and have the same precedence level:

```
>     <     >=    <=
```

Similarly, the equality operators are also supported by the language, however, have a precedence level slightly lower than the relational operators listed above. These operators are:

```
==    !=
```

Compared to the arithmetic operators defined above, relational operators have a lower precedence.

The following logical operators are also supported:

```
&&    ||
```

These refer to boolean AND/OR, which return either true or false based on whether or not both or none of the comparators are true. These operators are both left associative and have a precedence lower than both equality and binary operators.

Comparisons and logical operators between Shape types are not supported.

1.5.3. Unary Operators

ShapeShifter also supports the following Shape operators for numerics and booleans:

```
!     -
```

Each of these operators are the NOT and NEGATION operators, with the simple functions of negating a boolean, or negating an integer value (multiplying by negative one). The precedence level of unary operators can be seen in relation to all other operators in Fig. 1.0; however, unlike all previously mentioned operators, NOT and NEGATION are right associative.

1.5.5. Other Operators

The following uncategorized operators are also supported by ShapeShifter:

```
( expression )   =
```

These operators are the PARENTHESES and ASSIGN operators, each of which artificially influences precedence levels by evaluating what is inside the parentheses first, and assigning the values to expressions respectively. Their associativity can be seen in Fig 1.

Figure 1: Table of Operators and Associativity
Note: Precedence increases as you go down the table

Operator Symbol	**Name**	**Associativity**
`=`	Assign	Right
`||` `&&`	Or And	Left Left
`==` `!=`	Equals Not Equals	Left Left

`<`
`>`
`<=`
`>=`
	Less than
	Greater than
	Less than or equals
	Greater than or equals
		Left
		Left
		Left
		Left

`+` `-`	Addition Subtraction	Left Left
`*` `/`	Multiplication Division	Left Left
`!` `-`	Not Negation	Right Right
`( )`	Parentheses	Left

3. Data Types

2.1. Primitive Data Types

The standard data types, int, double, string and bool, are all supported by ShapeShifter, along with the generic Shape type and specific types for a number of basic geometric solids. Any of these types can be NULL. Details on each of these types are given below, while the syntax can be seen in the code samples below.

2.1.1. Integer

Integers in ShapeShifter are represented by 32-bit memory chunks in 2's complement. All the standard operators can be applied to integers.

The sample below shows the use of integers in ShapeShifter:

```
int x = 7;
int y = 14;
int z = x + y; // z == 21 and z is an integer value
```

2.1.2. Double

Double values are also supported by ShapeShifter in order to represent more precise values. Each double is required to have a decimal point and numbers both prior to and following the decimal point. They follow the same 8-byte standard as C.

The sample below illustrates the use of double values in ShapeShifter.

```
double x = 7.12345;
double y = 14.12345;
double z = x + y; // z == 21.2469 and z is a double value
```

2.1.3. Boolean

Boolean data types in ShapeShifter are data types with two values, true or false, that represent the logical truth values.

The sample below illustrates the use of boolean values in ShapeShifter.

```
bool x = true;
bool y = false;
bool z = x || y; // z == true
```

2.1.4. String

The string data type in ShapeShifter is a sequence of characters that are declared within double quotes. Strings are converted and represented as an array of characters, and hence, can take the same letters and symbols as characters from the same local character set.

String usage in ShapeShifter can be seen in the sample code below:

```
string s = “ShapeShifter”;
print(s); // Prints out ‘ShapeShifter’ to the console
```

2.2. Non-Primitive Data Types

2.2.1. Shape

The Shape data type is the defining feature of ShapeShifter, since this is where shapes can be declared and modified. Shapes are objects with an underlying triangle mesh representation that the user can manipulate, render to the screen, and save to a file. Hence, they can be modified with either predefined functions or operators, or instantiated from one of the primitive shapes that are built into the language, which are spheres, cubes, cones, cylinders, and tetrahedrons. The shape primitives are all centered at the origin and of unit size, meaning that they tightly fit a cube with radius 1.

Initialization is done by assigning a new Shape variable either to the result of a function with Shape as a return type, or by assigning it to one of the built-in primitive types, specified by the name of that type in uppercase. If an assignment is performed on a variable that already had a definition, that Shape will be completely overwritten. If there are no references to the original Shape, it will be unreachable and deleted at program finish. For instance, the following code sample initializes `ball` to be a unit sphere, and then creates a new Shape clone by invoking the Copy function on it.

```
Shape ball = SPHERE;
Shape clone;
Copy(ball, clone);
```

Assignment of Shape types, when not done using a built-in initializer or the results of a function, is done by reference, not value. For instance, in the following example, both ballA and ballB refer to the same underlying data structure, and the scale affects both.

```
Shape ballA = SPHERE;
Shape ballB = ballA;
Scale(ballB, 1.0, 2.0, 1.0);
```

The programmer must also be careful about scope when assigning by reference. If a Shape variable is assigned to another Shape that goes out of scope before it does, it no longer refers to a safe data structure and the resulting behavior is undefined.

Shape is the data type that represent any of the shapes generated by the language.

```
int scene() {
      Shape s1 = CYLINDER;
      {
            Shape s2 = SPHERE;
            s1 = s2; // s1 now points to the same Sphere as s2
      }
      Render(s1); // The Sphere is out of scope; undefined results
}
```

The sample below illustrates the use of the Shape data type in ShapeShifter:

```
Shape makeHat(double y) {
      Shape brim = CYLINDER;
      Shape top = CYLINDER;
      Scale(top, 1.0, y, 1.0);
      return Union(top, brim);
}
```

2.2.2. Void

The void data type represents a non-existent value that can be returned by a function. In other words, if a function does not return a result, it returns void. Note that void and NULL are not the same thing.

The sample code below illustrates the void data type in action in ShapeShifter:

```
void addShapeProperties() {
      ...
}
```

3. Expressions

Expressions in ShapeShifter include at least one operand and zero or more operators, and can be grouped using parentheses.

Possible expressions include:

```
42
(4 + 2) * 42
```

Note that these are expressions, not statements, so we have excluded trailing semicolons.

3.1. Assignment

Assignment involves the = operator - an assignment expression returns the value that was assigned. Here are possible assignment expressions:

```
i = 50
j = 3.0/2.0
```

3.2. Unary Operators

Expressions can include unary operators, which involve a single operator and operand:

```
-50
! (x > 5)
```

Here is a full list of unary operators in ShapeShifter:

```
! -
```

3.3. Binary Operators

Expressions can include binary operators, which involve a single operator and two operands as follows:

```
3.0 + 4.0
6 + 2
```

The binary operators are:

```
+ - * / == != < <= > >= && ||
```

4. Statements

Statements are single atoms of code execution, delimited by semicolon characters.

4.1. Expression Statements

An expression statement consists of an expression (defined above) followed by a semicolon.

4.2. Declarations

ShapeShifter is a statically typed language. Variable declarations consist of a type, an identifier, and an rvalue such as:

```
int counter = 3;
Shape c = CYLINDER;
```

4.3. Control Flow

We allow for branches with if-else statements. These statements consist of an if block, followed by zero or more elif blocks, followed by an else block (note that the else block is required). The if and elif blocks are preceded by expressions in parentheses that, if true, cause the corresponding block to execute. Here is an example:

```
if (expr) {
      stmt;
} else {
      stmt;
}
```

4.4. Loops

ShapeShifter allows for two loop constructs: while loops and for loops. Of the two options, while loops are more general (the body of the loop executes until the expression in parentheses evaluates to false):

```
while (expr) {
      stmt;
}
```

In addition to while loops, ShapeShifter supports for loops. The parentheses before the block contain three expressions. The first is used for initialization. The body of the loop is called until the second expression evaluates to false. Variables must be pre-defined. The third expression is called every time the body of the loop is executed:

```
for (expr; expr; expr) {
      Stmt;
}
```

5. Functions

5.1. Built-in Functions

ShapeShifter has a variety of built-in functions that can be used to manipulate shapes that have already been defined. These are built in order to simplify common functions that users may want to use while developing scenes and making geometric manipulations. Built-in functions include: `Scale()`, `Translate()`, `Rotate()`, `Reflect()`, `Union()`, `Intersect()`, `Render()`, `Copy()`. The purpose of this duplication is to allow users to have different approaches to organizing their shape-building process, as more interesting programs will rely on the creativity of the programmer to make complex, reusable shapes.

Each of these functions and their descriptions can be found below, along with sample code to illustrate their use in ShapeShifter.

The Scale Function: `Scale(Shape x, double x, double y, double z)`

The Scale function takes in four arguments, a shape to operate on, as well as three double values that the determine the scaling for the shape. It multiplies each of the mesh coordinates of the given shape by the appropriate scale factor in the corresponding X, Y, or Z directions. A snippet of code using the Scale function in ShapeShifter can be seen below.

```
Shape s1 = SPHERE;
Scale(s1, 3.0, 3.0, 3.0); // uniform scaling
Render(s2); //renders sphere with radius 3 on display
```

The Rotate Function: `Rotate(Shape s, double x, double y, double z)`

The Rotate function takes in two arguments, a shape to operate on, as well as a double value that the shape needs to be rotated by for each axis. Function takes the double value mod 360.0 and rotates the shape that many degrees counter-clockwise in the same axis. A snippet of code using the Rotate function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Rotate(c1, 180.0, 0, 0);
Render(c1); /*renders the unit cone rotated 180 degrees CCW
around the X-axis. By default the unit cone is rendered with the
point pointing out of the screen*/
```

The Translate Function: `Translate(Shape x, double x, double y, double z)`

The Translate function takes in four arguments, a shape to operate on, as well as three double values that the shape needs to be translated by. The second argument is translation in the x-axis, the third argument is translation in the y-axis, and the fourth argument is the translation in the z-axis. A snippet of code using the Rotate function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Translate(c1, 9.0, 8.0, 7.0);
Render(c1); /*renders the unit cone translated (9.0, 8.0, 7.0)
units respectively*/
```

The Reflect Function: `Reflect(Shape x, double a, double b, double c)`

The Reflect function takes in four arguments, a shape to operate on, as well as three doubles that determine the plane of reflection: a*x + b*y + c*z = 0.
A snippet of code using the Rotate function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Reflect(c1, 0, 1.0, 0.0);
Render(c1); /*renders the unit cone reflected over the xz
plane.*/
```

The Union Function: `Union(Shape x, Shape y)`

The Union function takes in two arguments, two shapes to operate on. The function takes the union of these shapes and returns an entirely new shape. The input shapes are not modified in any way. The programmer is permitted to assign the result to a Shape being used as one of the arguments. By default, an operator mapping to this function is also available to the programmer in order to simply use.

```
Shape c1 = CONE;
Shape s1 = SPHERE;
Shape sc1 = Union(s1, c1);
Render(sc1); /*renders the union of these two shapes. Think ice
cream cone with the ball of ice cream sticking through the cone*/
```

The Intersect Function: `Intersect(Shape x, Shape y)`

The Intersect function takes in two arguments, two shapes to operate on. The function takes the intersection of these shapes and returns the new shape. The input shapes are not modified in any way. The programmer is permitted to assign the result to a Shape being used as one of the arguments. By default, an operator mapping to this function is also available to the programmer in order to simply use. A snippet of code using the Intersect function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Shape s1 = SPHERE;
Shape sc1 = Intersect(s1, c1);
Render(sc1); /*renders the intersection of these two shapes*/
```

The Difference Function: `Difference(Shape x, Shape y)`

The Difference function takes in two arguments, two shapes to operate on. The function takes the difference of these shapes and returns the new shape. The second shape is 'subtracted' from the first to form the new shape. The inputs are not modified. The programmer is permitted to assign the result to a Shape being used as one of the arguments. By default, an operator mapping to this function is also available to the programmer in order to simply use. A snippet of code using the Difference function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Shape s1 = SPHERE;
Shape sc1 = Difference(s1, c1);
Render(sc1); /*renders the difference of these two shapes*/
```

The Copy Function: `Copy(Shape src, Shape dest)`

The Copy function takes in two arguments, the source shape and the destination shape. It makes an exact copy of the first shape, with the same underlying mesh, and stores the copy in the second. The source shape is not modified. A snippet of code using the Copy function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Shape s1;
Copy(s1, c1);
Render(s1); /*renders the unit cone*/
```

The Render Function: `Render(Shape x)`

The Render function takes in one argument, the shape to render to the display. The function takes the given shape value and renders the shape to the display using the defining shape properties. A snippet of code using the Render function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Render(c1); /*renders the unit cone to the display based on the
defining values it contains*/
```

The Save Function: `Save(Shape x, string filename)`

The Save function takes in two arguments, the shape to save to file and the file to save it to. The function saves the underlying properties of the shape to a standard .off file that the user can then import in order to retrieve the shape. By default, the file is saved to the same directory that the user is working in. A snippet of code using the Save function in ShapeShifter can be seen below.

```
Shape c1 = CONE;
Save(s1, "cone.off"); /*saves the unit cone and all underlying
properties to a txt file that can later be imported by the user*/
```

Built-in Functions

Note: More information on the functions can be found in the Built-In functions section of this reference manual

Keyword	Syntax

void Scale(Shape x, double x, double y, double z)
This is a predefined function that is used to scale shapes. The x,y, z components scale the Shape by the input values, which are positive-valued doubles.
	`Shape s1 = SPHERE;`
	`Scale(s1, 2.0, 2.0, 2.0);`
	`Shape c1 = CONE;`
	`Scale(c1, 2.0, 1.0, 2.0);`

void Rotate(Shape x, double x, double y, double z)
This is a predefined function that is used to rotate shapes. It takes in a shape, three double values that correspond to counterclockwise rotation about the X, Y, and Z axes. The values
	`Shape s1 = SPHERE;`
	`Rotate(s1, 90.0, 0.0, 0.0);`
	`Cone c1 = CONE;`
	`Rotate(c1, 0.0, -90.0, 0.0);`

correspond to degrees and the effective values are mod 360.0	

void Translate(Shape x, double x, double y, double z)
This is a predefined function that is used to translate shapes. It takes in the shape to translate and the input x,y,z values represent the offset in the corresponding directions.
	`Shape s1 = SPHERE;`
	`Translate(s1, 10.0, 0.0, 10.0);`
	`Shape c1 = CONE;`
	`Translate(c1, 0.0, -1.0, 0.0);`

void Reflect(Shape x, double a, double b, double c)
This is a predefined function that is used to reflect shapes. It takes in a shape, as well as 3 doubles that define the plane of reflection: **a**x + **b**y + **c**z = 0.
	`Shape s1 = SPHERE;`
	`Reflect(s1, 0.0, 1.0, 0.0);`
	`Shape c1 = CONE;`
	`Reflect(c1, 1.0, 0.0, 0.0);`

Shape Union(Shape x, Shape y)
Shape Union ([Shape x, Shape y, …])
This is a predefined function that is used to union shapes. It takes in two shapes and returns the union.
	`Shape s1 = SPHERE;`
	`Shape c1 = CUBE;`
	`Shape c2 = CONE;`
	`Shape us = Union(s1, c1);`
	`Shape us2 = Union(c1, c2);`

Shape Intersect(Shape x, Shape y)
Shape Intersect ([Shape x, Shape y, …])
This is a predefined function that is used to intersect shapes.
	`Shape s1 = SPHERE;`
	`Shape c1 = CUBE;`
	`Shape c2 = CONE;`
	`Shape is = Intersect(s1, c1);`
	`Shape is2 = Intersect(c1, c2);`

Shape Difference(Shape x, Shape y) Shape Difference ([Shape x, Shape y, …])
This is a predefined function used to take the difference of shapes. The operator DU maps to this function. It returns the result of y subtracted from x as a new shape.
	`Shape s1 = SPHERE;`
	`Shape c1 = CUBE;`
	
	`Difference(s1, c1);`

Shape Copy(Shape x)
This is a predefined function that is used to copy shapes. It returns a copy of the input shape.
	`Shape s1 = SPHERE;`
	`Shape c1 = NULL;`
	`c1 = Copy(s1);`
	`Shape s2 = Copy(c1);`

void Render()
This is a predefined function that is used to display a shape on the screen. The function takes in a Shape. Must be called in scene().
	`int scene() {`
	`  …`
	`  Shape s1 = SPHERE;`
	`  Render(s1);`
	`}`

void Save()	`int scene() {`

This is a predefined function that is used to save the current scene into a file. The function takes in a Shape.	… Shape s1 = SPHERE; Save(s1, “sphere.off”); }

5.2. User Functions

5.2.1. Declarations

Functions are declared in the following format:

```
return_type function_name(arg_type arg_name, arg_type arg_name,
...)
```

Sample code for a function that creates a shape might look like:

```
Shape makeHat(double y)
```

Shapeshifter does not require users to declare functions in a separate statement from actual function definitions.

5.2.2. Definitions

Function definitions are included in `{ }` after the function declaration. Each statement within the body of a function definition is punctuated with `;` as is the case with all statements through a Shapeshifter program. The use of whitespace is encouraged to increase the legibility of a program and make scope more apparent. A sample function definition may look like:

```
Shape makeHat(double y) {
      Shape brim = CYLINDER;
      Shape top = CYLINDER;
      Scale(top, 1.0, y, 1.0);
      return Union(top, brim);
}
```

Function names can not be repeated; attempting to define a function with the same name will result in a compiler error.

A value of the type corresponding to the function’s return type must be returned using the return keyword. If there is conditional branching, each branch must have a return statement.
If the function has a void return type, the return statement is not followed by anything, as such:

```
return;
```

5.2.3. Calls

Function calls behave like other statements within a Shapeshifter program. A sample function call may look like:

```
Shape hat = makeHat(6.9);
```

In this case, the returned Shape is automatically stored in a variable called `hat`.

6. Program Structure

There are some particularities within the language. The body of each Shapeshifter program must call `int scene()`, which is the entry point for the program. Scene refers to the abstract coordinate system environment that all following shapes exist in. Function definitions must occur prior to the `scene()`. Shapeshifter, however, does not house function declarations separately - users are meant to be encouraged to construct custom shapes by creating a function and then use the space of `scene()` to produce intermediary or a final visual representation. This workflow would be most relevant if the user seeks to create custom shapes that they mean to reuse throughout their program.

Each `Shape` is initialized as a specific subtype: a `SPHERE`, `CONE`, `CUBE`, `CYLINDER`, or `TETRA` (tetrahedron). Successive transformations on these shapes which can then in turn result in more complex shapes whose properties are not limited to those of the basic subtypes.

Note that shapes can be initialized within the body of a function definition as well as the body of `scene()`. The scope of each object or variable is contained within the block (denoted by `{ }`) that it exists.

Statements, expressions, and variable definitions must be contained inside a user-defined function or scene(). There is no support for global variables.

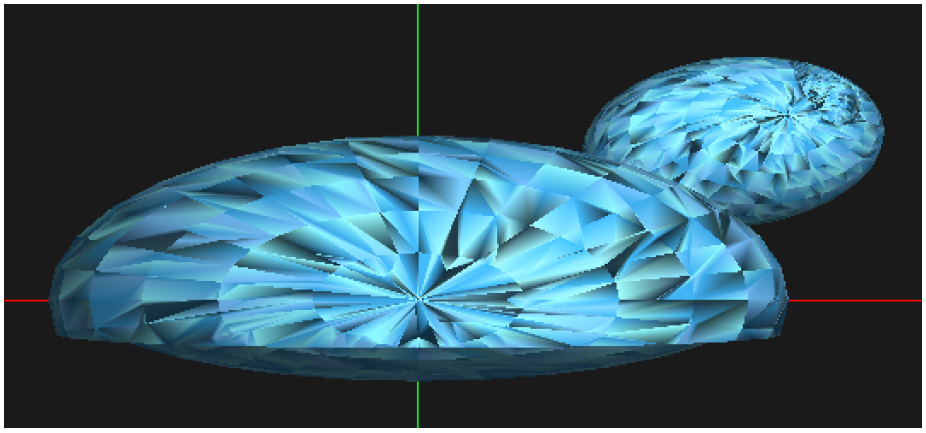
7. Sample Program

This program creates a turtle programmatically through primitive shapes and boolean operations, then renders the result to the screen.

```
int scene() {
    Shape turtle;
    Shape shell1 = SPHERE;
    Scale(shell1, 3.0, 2.0, 3.0);
    Shape shellc = CUBE;
    Scale(shellc, 5.0, 5.0, 5.0);
    Translate(shellc, 0, -2.7, 0);
    shell1 = Difference(shell1, shellc);
    Shape shellb = SPHERE;
    Scale(shellb, 2.8, 1.0, 2.8);
    shell1 = Union(shell1, shellb);
```

```
    Scale(shell1, 1.5, 1.0, 1.0);

    Shape head = SPHERE;
    Scale(head, 1.3, 0.75, 1.0);
    Shape leye = SPHERE;
    Scale(leye, 0.3, 0.3, 0.3);
    Shape reye;
    Copy(leye, reye);

    Translate(leye, 0.2, 0.1, 0.4);
    Translate(reye, 0.2, 0.1, -0.4);
    Shape eyes = Union(leye, reye);
    head = Difference(head, eyes);

    Scale(head, 1.4, 1.4, 1.4);

    Translate(head, 2.0, 1.0, 0.0);

    turtle = Union(head, shell1);
    Render(turtle);
}
```

The result looks like this:



Project Plan

The Shapeshifter team met regularly throughout the semester in addition to their meetings with PLT teaching assistant, Jacob Graff. During these meetings, the team conceptualized what they hoped their programming language would accomplish for its users and they planned what to implement next. The team implemented version control by hosting their project on Github. They maintained a TODO list within their repository, which captured a running list of what they hoped to implement, issues they had encountered, and the insights that had been gained in their last interaction with the code. In addition to the regular weekly meetings, some team members met on Thursday evenings and Saturdays to work on the project.

The team had the guidance of TA Graff to identify project milestones -- such as Hello World -- and meet them in a timely manner.

For implementing their compiler, the Shapeshifter team used the following tools:

- **Github / git**: used for version control
- **OCaml**: language used to implement our compiler
- **Cork**: used to performed mesh transformations and manipulate our primitive shape types
- **OpenGL + glut**: used to render and display shapes to the screen
- **Blender + Meshlab:** used to create the primitive shapes and export them as .OFF files

In addition to in-person meetings, the team maintained regular communication through Facebook Messenger and used Google Docs to collaborate on submitted materials and documentation for the project.

The project was tested using the set of tests in the Test Suite (and run by `./testall`) -- while the tests attempted to capture potential issues that may be encountered in the process of development, we encountered several issues as more of the compiler was being built out -- we used those moments to then design specific tests to catch the problem in future instances.

Code Style

While we did not establish an official style guide for our project, we kept these principles in mind as we wrote and modified our compiler:

- Aim for consistency - use a style similar to that established by the MicroC compiler.
- Use indents to organize blocks of code, with newlines between sections of code as appropriate.
- Try to keep columns below 100 characters wide.
- Use descriptive variable names.
- Comment tricky sections so that everyone can get acquainted with new code quickly

Beyond this, we were not too strict about style - we tried to keep our code readable and self-documenting to help our teammates.

Project Timeline

As seen in the github graph of the repository above, work was consistently being done throughout the semester on the project; however, as expected, more and more was being done closer to the project submission deadline. There was a significant ramp-up in late November as we began to do more work with the graphics libraries in preparation for our Hello World demo.

Team Responsibilities

Team Member	Responsibility
Stephanie Burgos	Tester
Ishan Guru	Project Manager
Rashida Kamal	Tester
Eszter Offertaler	Language Guru
Rajiv Thamburaj	System Architect

Project Log

```
commit 7aa5b5dd8177c4e1c66a5c52aeac9a267af4286f
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Tue Dec 20 00:48:35 2016 -0500

     Cleans up tmp directory on program finish :)

commit d82c7b8f4b62383c933c21fd929bd7cd14431e95
Merge: 974de22 17763b1
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Tue Dec 20 00:43:21 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 974de2294d0664ece18e0d111a7dc6fdaeab0a3a
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Tue Dec 20 00:43:19 2016 -0500
```

```
    Program creates .tmp :)

commit 17763b17c665ccf693aadad452515a746ec232df
Merge: ffc45d2 66c8ad3
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 22:45:28 2016 -0500

    Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit ffc45d2127aa91d9c19bd47d03fde9a8f4ee957a
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 22:45:11 2016 -0500

    tests

commit 66c8ad3499c2119aac515c21ebcf13f386c2e092
Merge: 4b56042 c6a51b1
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 19:47:17 2016 -0500

    Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 4b56042ca90fe1cab14e9071f49831ba9b679a2b
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 19:47:14 2016 -0500

    Updated TODO with current list of priorities, added tests

commit c6a51b16712f4ec05d32e23711f7f2ec57a8041f
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 19:40:36 2016 -0500

    test

commit b184252799b37336a5e87ba003c1dc363e07c598
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 11:41:40 2016 -0500

    sad beakless bird

commit 101870c906d637aca91a19deac1c12f1076400f7
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 11:41:15 2016 -0500

    tests

commit b2cc22894bd6b741e355de7e88d58c836d4698c3
```

```
Merge: 2c05fe7 a275125
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 09:02:18 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 2c05fe7d25d497b13db4900aa7c27c6e14a73b01
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 09:02:14 2016 -0500

     Added Copy(s1, s2)

commit a27512525f0d756075f8a20d901135e1c72b573f
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 08:40:16 2016 -0500

     sandbox

commit 7c166fb8ee15ee1569e3bb983ce42d586d944cdc
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 08:11:34 2016 -0500

     more test fixes

commit 2be43742a8a85ca1d282e60bf125069758a5577f
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 07:50:57 2016 -0500

     Edits to test suite

commit 8d88714fae2f14081d701f93e1398a0fd4a25a34
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 03:56:31 2016 -0500

     Moved light a bit

commit 32d0a7a474548c8759dba46d091fe3f5c1ee066b
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 03:52:18 2016 -0500

     WE HAVE LIGHTING

commit af5556bfdacdb2cd8afe6cca12224c89839c1f1c
Merge: 53e2ab8 e27fe52
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 03:17:08 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter
```

```
commit 53e2ab8114ba9c02b6a9e1d6c3c451ee079f8f92
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 19 03:13:21 2016 -0500

     Trying to add Shape->string support

     Adds the const strings (also tracked in hashtable shstr_map)...
But
     have no idea how to then use those llvm strings in the system
command...
     some memory/load/stores?

commit e27fe529e13f9e75561cc0599b611838ad68f50e
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 02:44:33 2016 -0500

     more test stuff

commit a09af3fd40510bf60696d4b72b718e2183404942
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 02:13:35 2016 -0500

     renamed/reorganized

commit 5125bf7212c2d0f6e5b6de582783cc590a9ef067
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 01:49:21 2016 -0500

     outfiles for tests

commit 3e4ef988490cf1c2ee5d9522235be81fd1296452
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Mon Dec 19 01:39:33 2016 -0500

     test edits

commit cdc0c859c077d4880b8fa7b7b6b9419112c013bd
Merge: 02db6b0 16c0fa8
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sun Dec 18 23:56:01 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit 02db6b0026c1c64b5210a5e8eaf10b5b8c9443ae
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sun Dec 18 23:55:40 2016 -0500
```

```
      edits to test suite

commit 16c0fa83f7b56c7fa8c435f6dbae53eda1d60ffc
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 22:27:19 2016 -0500

      Moved test_sphere_union to the right spot xD

commit e8229f81dd10d70e4f36d00c5cff0ec036445dea
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 22:21:42 2016 -0500

      Moved test

commit b79c27aa5dcc52200d486904b74b989e7ec24330
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sun Dec 18 22:20:49 2016 -0500

      starting to reorganize test-suite

commit 0366fee3a5dbfd1cea51192350d6a551304d8b8b
Merge: 162efbe b79c27a
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 22:19:33 2016 -0500

      Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 162efbe18fd71ad259a4479805e1d10deee35b14
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 22:19:30 2016 -0500

      Can do Sphere s; s = SPHERE/Union...

commit 13dacf332340c3684018ea8e51a6789a68eaaef9
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 20:56:11 2016 -0500

      References work inline as well

commit a48850a5b059c36c73a0ccdc89c0b4b930218c52
Merge: 94abef7 fedcb70
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 20:33:09 2016 -0500

      Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 94abef7430b19fba99c8c67c1a90e946c7e1d98c
Author: Eszter Offertaler <eo2309@columbia.edu>
```

```
Date:   Sun Dec 18 20:33:05 2016 -0500

     Shape aliases work now

commit fedcb705ec0378f3d4aaee169651f15c504bd932
Merge: 48f3f3b 297115d
Author: Stephanie Burgos <stephburgos96@gmail.com>
Date:   Sun Dec 18 18:34:05 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit 48f3f3b62f4edb3bc547a50edf13f27ce4560ef4
Author: Stephanie Burgos <stephburgos96@gmail.com>
Date:   Sun Dec 18 18:33:44 2016 -0500

     Adds function, string, and array tests

commit 297115d613c2e707eca4b0d1fbe568dc0aca2200
Merge: 9569c62 84217be
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 17:33:02 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 9569c621e12b641e1406277895a76c7fba933ead
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 17:32:57 2016 -0500

     Still tweaking display; not really sure why cone/sphere/cylinder
are streaked

commit 84217bea6a576e9df930172c9fe8750a106a8c41
Merge: e7cf9db 9956bd7
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sun Dec 18 17:17:14 2016 -0500

     Merge conflicts on pull

commit e7cf9dbed00f70c78d367dd76ade177e57c1f5d9
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sun Dec 18 17:11:41 2016 -0500

     Fixed all warnings during make

commit 9956bd7b5a32f9397e1e335341c77cbfa393982f
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sun Dec 18 17:08:23 2016 -0500
```

```
      help flag

commit 9c816d4ed5355718b68f635fad81299e06557a94
Merge: e25273b aa04d7a
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sun Dec 18 17:03:35 2016 -0500

      Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit e25273bc8d7873e476044be825ab7b88aead60c7
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sun Dec 18 17:03:14 2016 -0500

      added help flag to front end

commit 1a894ef16cee14334cf02333f4e0a6449b9722b3
Merge: 65073f8 aa04d7a
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 17:01:19 2016 -0500

      Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 65073f8d8b67ecd061a8f69cbcc58b7080793a66
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 17:01:15 2016 -0500

      Fixed missing face in tetra; tweaking display

commit aa04d7a7689061f5e745a377be252745f341ba71
Merge: c50884b 79d8f45
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sun Dec 18 15:25:27 2016 -0500

      Fixed merge conflicts

commit c50884b0b7ec7c0ad628f4e55380fa41dda48823
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sun Dec 18 15:22:35 2016 -0500

      Negative numbers tests modified

commit 227e275480939c838ebcbba07582a5fbafd6462d
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sun Dec 18 15:18:59 2016 -0500

      Support for Unop strings in string_of_expr

commit d4e042986888c5c1807051b571224e3d63087a4e
```

```
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sun Dec 18 14:52:22 2016 -0500

     Tests for assigning and printing negative integer

commit 15dafa47f986efe4133b8a7348e2e39494629d6c
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sun Dec 18 14:51:14 2016 -0500

     Negative integers now supported

commit 79d8f45a8b2b02aaf727212db585021fa73d7f46
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sun Dec 18 14:13:00 2016 -0500

     an incorrect attempt at negative numbers

commit 6b02e7d52b9d66008a9fc62c5e7a9960ac47455b
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 13:32:51 2016 -0500

     Make normals displayable in debug only; fixed compiler warnings

commit a3abfd91f58cba1ff1ad16c7d2c54aa8c2d84a64
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 01:41:52 2016 -0500

     Tetra primitive file improved

commit c6c8d8f2b595a657a8d5c35bb87e5bf6f27b589f
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 18 01:01:58 2016 -0500

     Display seems fixed; remaining artifacts probably because of
repeated vertices in actual shape mesh files

commit 889b692f85aa18ae7e2306f075a5e0c2c41bbbfc
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Dec 17 22:59:57 2016 -0500

     Camera flipping fixed; lighting still screwy

commit e9bae2fec9910499465d666fef7601372d86cf4d
Merge: 7ad17fd 431e831
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Dec 17 20:39:56 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter
```

```
commit 7ad17fdf046326c2559402672d6c6d686fd1365b
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Dec 17 20:39:53 2016 -0500

     Finally the normals are calculated correctly; display still looks
weird :(

commit 9ca8d20c3dc1647472f98ab6e2d1dc8fc0ebdb62
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Dec 17 19:52:02 2016 -0500

     Better cube

commit 24472b60a140bc69f98f7d549a0ed7258652adac
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Dec 17 18:29:26 2016 -0500

     Fixing norms

commit 431e8311eb0bed2ce7605310f89896426d847e40
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sat Dec 17 15:57:18 2016 -0500

     few more tests

commit 0f88a4753ec154094e708ed81f5e58f6b81a0cbf
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sat Dec 17 15:31:57 2016 -0500

     testing logical operators

commit 92089aab318b932f8e10d349973e2fe911a30365
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sat Dec 17 12:58:32 2016 -0500

     simple test cases for transformations

commit 7843ec2cb6f5820a3b4e8258ecb49a66933200f5
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sat Dec 17 02:12:20 2016 -0500

     Deleted duplicate todo file

commit edec6c17b068e9ddcd079fb87eefeb6b60bc9816
Merge: 5c8128b 5dea213
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sat Dec 17 01:44:31 2016 -0500
```

```
    Merge branch 'master' of https://github.com/nyletara/ShapeShifter

    Merge on pull

commit 5c8128b4cb47ddbfab8cad15b102b8b293d169c6
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sat Dec 17 01:44:14 2016 -0500

    Sample recursive test; string type test

commit 24d689dcd6535f6755841c456bd00a2964f43c89
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sat Dec 17 01:42:15 2016 -0500

    Infunction binop support added to test

commit 53ceedd60c2840703f343d8dfb772086a95577e8
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sat Dec 17 01:40:02 2016 -0500

    Support for recursive functions; support for binops in functions

commit 5dea213e547dca41e1c738b2074f477739fc5fc0
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Dec 17 00:18:57 2016 -0500

    Added test for union

commit e1605832d17887e83d85a21603aeb7d17a4555bb
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 23:14:21 2016 -0500

    Removed extraneous prints

commit 014f1ccfb7b40bf0cbc4a145a927f4da689fb4f5
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 23:12:21 2016 -0500

    Fixed memory bug :(

commit e8e0447be489eb8aa2f8ddb27c99d780c5ad5a4c
Merge: 9735539 e3d1e29
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 22:03:50 2016 -0500

    Merge branch 'master' of github.com:nyletara/ShapeShifter
```

```
commit e3d1e290ab8ce79edc5f350ca090218f0901ed91
Merge: d0930bc 3c98030
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 16 22:14:07 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit d0930bcce4ac2b3480d89ecaf762d3d772e1e565
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 16 22:13:41 2016 -0500

     automatically make cork & display

commit 97355395fd4121f40b372998c1e29cd6a1b86788
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 22:03:43 2016 -0500

     Seems to create new shape for boolean ops, but render fails in
runtime and save generates a not_found exception

commit 3c9803071a8302dfa4594a9f43836045d114bfad
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Dec 16 21:01:47 2016 -0500

     Whoops, semantic analysis isn't complete yet. Comment it out for
now.

commit 8fe51be89139e448f6fa7b8050f628dd07ad8b10
Merge: acd0bd7 93b3c70
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Dec 16 20:44:31 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit acd0bd7b75164f15f471b4a7100403a1179e56e3
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Dec 16 20:44:22 2016 -0500

     Modify pretty printer so shapeshifter.native compiles.

commit 8347426b2732ed0d1e54b7e1c400c016499857c9
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Dec 16 20:43:45 2016 -0500

     Add semantic checking.

commit 3b0d063259ae79b62f56dff6f75f0c08db6073e7
Author: RajivJT <rjt2132@columbia.edu>
```

```
Date:   Fri Dec 16 20:43:24 2016 -0500

     Update AST and Parser with removal of null.

commit 93b3c70e99675b93622960559ac1fb972163bf0f
Merge: 554c138 2c0cdbf
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Dec 16 20:27:02 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

     Merge on pull

commit 554c1385e0ef3d4821547eb3ff2d0c44261229a4
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Dec 16 19:47:33 2016 -0500

     Fixed simple if, while tests

commit 0838bfec58f7dfc23ef3dd97df065d4f0e7cb719
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Dec 16 18:55:03 2016 -0500

     Removed null and comments

commit 2c0cdbf0d136b766295a20a0933dc21bef515a5c
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 17:16:05 2016 -0500

     Added rest of transformation functions (untested); boolean
operators currently don't work due to missing last arg

commit af981ba75cca74a3d088d770d2eda770dff0a2bc
Merge: 2748223 3b01015
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Dec 16 15:47:52 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

     Commit for merge

commit 2748223be1a78998e80943df76632bf9350fcfa1
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Dec 16 15:47:20 2016 -0500

     Modified test cases

commit 84472932dfc9fceb75abf68d7f66e88de888f996
```

```
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Dec 16 15:45:52 2016 -0500

     Refactored int/double binops to remove duplicate code

commit 3b0101549656271866ede657c916d1907768aef3
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 15:38:44 2016 -0500

     Added all primitives; actual file representations may be
problematic

commit af053833a58d06eeb522be82741a10a2bedf7145
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 14:42:08 2016 -0500

     Generalized primitive constructors

commit b07e833434cc381a742b954e383d604a19c2a3e1
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 14:25:21 2016 -0500

     Generalized shapeshifter function calls

commit 9f85db81d30ba8f8ef1420541970730f24609013
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 02:26:38 2016 -0500

     trying to generalize cork functions; commented out since
returning wrong type

commit b479f18e7bdf3569e0ab9dd71e1308556d60e6c7
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 16 02:21:13 2016 -0500

     for some moronic reason get_cork_cmd is returning a function
instead of a damn string so that's fun

commit 7c145b964ff54da39cbf067e34e8bbd461cb1c06
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Thu Dec 15 15:39:19 2016 -0500

     Looks like we need the A.Assign and the first A.Id - fixed issue
with arithmetic

commit d062d815d5399b0c3977b27c8e0f1fcf117796a1
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Wed Dec 14 22:28:14 2016 -0500
```

```
    Added constructor for Cube; save and translate use the actual
parameters now

commit 511c66f76859e74689c0ad732bb813f7b400e555
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Wed Dec 14 00:19:57 2016 -0500

    Add thing to shape_map only if of type Shape

commit d48cf640db5e9fdbea6fa14a92f9ee309747598a
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 12 01:59:15 2016 -0500

    Map shapes to names; test with hacky version of Save

commit 4c0682dcb892e43be4d802ef1cc8a46f01144d53
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Dec 12 00:16:07 2016 -0500

    String quotes from string literals

commit 6ade0551514fa8413e2ca4e40a2ef6b9afb59d6a
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 11 23:42:01 2016 -0500

    Add test for printing string variables

commit 94a16965b3a626fef3d5908de02de9eb855de36a
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 11 23:09:23 2016 -0500

    Can print string IDs :)

commit 3e22ae70856e8a480f85d755de8ba6413be36cbc
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sun Dec 11 16:32:25 2016 -0500

    Seem to have fixed prettyprint for locals

commit 86e09780f0a2a2d05431e9f67a700122060f4844
Author: RajivJT <rjt2132@columbia.edu>
Date:   Sun Dec 11 01:09:30 2016 -0500

    Support mixed statements and declarations, along with
simultaneous assignments and declarations.

commit bc97bc7e705d172dd98bd451c08a08d3f82d807c
```

```
Merge: 5acd390 8d2619f
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sat Dec 10 15:38:59 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

     Merge commit

commit 5acd390eb27e9fd5d4751b622def59104807b249
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Sat Dec 10 15:38:37 2016 -0500

     Arithmetic and print now works with Integers and Doubles

commit 8d2619fd10e5cce948764811fb9ebb8dcd5c49c2
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 19:28:14 2016 -0500

     Added 3D acceleration warning to graphics readme

commit d5aede3c013f4e5121c08b449ae6f537f0b92f64
Merge: c9b8e94 f100962
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 19:27:03 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter

commit c9b8e94e46b651d01e6997a538a1d5063ba0a9a2
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 19:27:00 2016 -0500

     Renamed primitive types to 'Prim' instead of 'Obj'

commit f100962c4ac863ff256af6c4853b17d9c576d077
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Dec 9 19:21:46 2016 -0500

     Install these libraries from README to get the display working

commit 9244f5c4314979342e0cdc4bfe2b8d7fe9eade6f
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 19:02:04 2016 -0500

     Added translate test

commit 2cf634d085db2757c2e8b241907819a071ae8723
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 17:53:18 2016 -0500
```

```
    hacky version of render added

commit 62f554be2edd4f782cac3a77520d9fbfb00554b5
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 17:05:41 2016 -0500

    Translate calls cork executable and actually translates the thing

commit dbb3ea62a913e95a4a9e0cded5fa8767cae92412
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 16:25:59 2016 -0500

    IT CALLS LS HECK YEAH

commit 037c017cf6f2a9273f7e0635b4f5214c45936885
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 16:16:50 2016 -0500

    Well it compiles but execl doesn't seem to actually do anything

commit 4c84003a936d019b025c34b3b214a30ec8b14d62
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 15:38:57 2016 -0500

    So that's how to get multiple arguments; still dummy what
function

commit f50e95cacc4f08f3410366f03df4f4e1da984c05
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 15:35:03 2016 -0500

    Added temporary what function to muck about with

commit 18f10a9d621828c800851dd09c3acedc315df91d
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 13:48:11 2016 -0500

    Compiles; doesn't actually run Test; Shape temporarily just an
int to test function

commit 85db532b037dc17883a9d042b3edae7afb621b2f
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 9 02:03:20 2016 -0500

    Trying to add trivial call to Translate; hard to get pointers to
strings, even for hardcoded stuff
```

```
commit 6541acca988242b03333cf087c6f036ade84b940
Merge: 8d8ae92 af90d70
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Thu Dec 8 20:59:35 2016 -0500

     fixed conflict on graphics/display/main by using local changes;
one in repo seemed to be outdated

commit 8d8ae923b251c839d6022afeec309a72eb451d18
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Thu Dec 8 20:39:35 2016 -0500

     Add execv function declaration to codegen

commit af90d70ee2bb6eaf716b7808c84023c8b310307a
Author: Stephanie Burgos <stephburgos96@gmail.com>
Date:   Thu Dec 8 19:59:46 2016 -0500

     Adds arithmethic and control flow tests, splits shape test into
another folder until we're ready to test them

commit 4dc4128baffc4c4566215bc3fd50165d005279fd
Author: Stephanie Burgos <stephburgos96@gmail.com>
Date:   Sat Dec 3 12:29:11 2016 -0500

     Hello world test passes

commit cd97b4b4b00684cb1ccb6e31016f4af2ca10af9e
Merge: 037aafc cbaf0ac
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 2 18:37:02 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit 037aafc54c499118b00d08fb4bb0155698e3a9a5
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 2 18:36:39 2016 -0500

     testall

commit cbaf0ac2924dd521a38143547206b3707ac81d9a
Merge: ec47cfc c582ad4
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 2 18:17:08 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter

commit c582ad408ce4b7f5f4bcbb7c1723f0494427712b
```

```
Merge: ad3d315 e056c2e
Author: Stephanie Burgos <stephburgos96@gmail.com>
Date:   Fri Dec 2 17:59:35 2016 -0500

     Merge tests
     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit ad3d31579e1ffaf82312b73be406f8a6cb13818b
Author: Stephanie Burgos <stephburgos96@gmail.com>
Date:   Fri Dec 2 17:55:31 2016 -0500

     Adds passing test

commit ec47cfcfc5fc4e2b73f142387fd541b9a134fc23
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 2 17:40:27 2016 -0500

     Lighting still screwy; but at least can differentiate faces

commit e056c2e37243d494215021581ab8512f6bb9b7c1
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 2 17:38:55 2016 -0500

     test readme

commit b6ce256fa206631a6ec91e69a5af82e9222e14e9
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 2 17:36:55 2016 -0500

     fixed tetra coordinates

commit fc12a1a30a1145a60075135806749b6c2af1de4b
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 2 17:32:27 2016 -0500

     demo

commit 167c708c73af010597c13bf30484672287a210cb
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 2 17:07:18 2016 -0500

     Can't tell if the normals are wrong or the lighting is :(

commit 578161bf4ee7f98eb51e444d0fcb7f4ae10ea3c5
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 2 16:53:25 2016 -0500

     fleshing out the design of our test suite!
```

```
commit ad0fff5812b2fadc9a765a3b9ca46f17be35a186
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Dec 2 16:39:53 2016 -0500

     lighting doesn't move with camera; normals added to pipeline with
dummy value

commit 51187c4c94bcaf405bf50af9cb1a29b417f99c07
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Fri Dec 2 16:39:05 2016 -0500

     arithmetic tests for int

commit 7a7d63e55ad65b9c852f6d2e90b117cd5a8e5cc0
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Thu Dec 1 23:31:25 2016 -0500

     comments added

commit f9e821468ea5e08811b59136cf1bc857b65feb93
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Thu Dec 1 23:19:47 2016 -0500

     add line to rm .err files in make clean

commit c66ba190f347c8e669f7233dbaf2401f536d4259
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Thu Dec 1 23:10:57 2016 -0500

     renamed fail tests and edited make clean

commit 4468e188e07d4cb092b7ed61379908d667706598
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Thu Dec 1 23:02:08 2016 -0500

     renamed

commit d1232dc1e9eb88af2a573b0d94f6e58b6fd160fe
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Tue Nov 29 23:22:28 2016 -0500

     Changes to print + binop for IntLit and DblLits

commit 86a3a308315d16022dc65e7b4bbbe43549816ab0
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Tue Nov 29 13:46:47 2016 -0500
```

```
    Mapped scene() to main()

commit 269c6b8eebc79018a2f5149c9cbe7b316c0e30e8
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Nov 19 16:03:51 2016 -0500

    Added camera zoom; use +/- keys

commit f3444100955747d2c0b2cefb2f55fda0c67bcd08
Merge: 89a1b84 4477717
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Nov 19 15:48:50 2016 -0500

    Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 89a1b84ab01d1e00b58dc12b97718b80d89ff32f
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Nov 19 15:48:47 2016 -0500

    Camera coordinates now in spherical

commit 4477717d224e8331970263e410c607c3a5340fe8
Merge: ad738dd 5a29e52
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sat Nov 19 15:36:32 2016 -0500

    Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit ad738ddb1d18e6817ce81ea52e8b7e8f7ce84383
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sat Nov 19 15:36:21 2016 -0500

    adding sample prog from lrm

commit 5a29e5215ecf16ea9a3e8d5be7d02e6253146b8c
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Sat Nov 19 15:12:21 2016 -0500

    Some camera movement; possibly incorrect primitives

commit d3c54547e4acba3778b5c740078f487e75e409d1
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Sat Nov 19 12:24:52 2016 -0500

    hello world demo script

commit 2f024a3e3ec5d57023711a3f098903590cb1a055
Author: rashidakamal <rsk2161@columbia.edu>
```

```
Date:   Sat Nov 19 11:50:14 2016 -0500

     adding demo directory

commit 2320c3ed7e0c9671d270426db5f3079c73cefe88
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Nov 18 20:17:47 2016 -0500

     renamed pp.ml to prettyprint.ml

commit 735093637fe1695128833d86638ae5eed27c1822
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 18 19:46:48 2016 -0500

     Update .gitignore to ignore *.ll, *.out

commit 0bda5bca8d8182b9676b93efdcbc9fa218c46572
Author: nyletara <iguru95@gmail.com>
Date:   Fri Nov 18 19:40:24 2016 -0500

     Added TODO.txt

commit 48583e6d043108c82da786320a141d0c7aaf1212
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 18 19:07:01 2016 -0500

     Added graphics README; removed samples from repo

commit fb8da22712270d4303af50254cceca6e018f3950
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 18 19:05:40 2016 -0500

     added graphics README

commit 752aa679cd0c6bd29f92d9cf730af6c613cb440d
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 18 17:44:38 2016 -0500

     Set up camera projection; lighting still wonky

commit d790b0d75b95cf6a6421c1c7732902a6c179a04b
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 18 17:10:57 2016 -0500

     lighting is weird :(

commit bc1782ac702e614d7048160dc397f7b4b820c8cf
Author: Eszter Offertaler <eo2309@columbia.edu>
```

```
Date:   Fri Nov 18 16:30:13 2016 -0500

     Display clipping fixed; near/far clipping was too small

commit d7ec84930525e315ccb43ec2d40a8165aaec1d22
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Thu Nov 17 00:22:48 2016 -0500

     C++ interface first pass doneish; not really tested

commit b726c2377bd7dc13d0aeb856412682fd42c590a1
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Wed Nov 16 22:35:47 2016 -0500

     Bulking out C++ interface; not yet tested besides translate

commit 68b9a98a025ac354377cdfce7bbdc21ca2d51605
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Wed Nov 16 01:23:37 2016 -0500

     Starting C interface for cork

commit 8fd7cff8f29d576a71c62f65c95af85b00f20573
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Wed Nov 16 00:24:43 2016 -0500

     Reflect added; untested

commit 8f55b86ffbf2c35f65f68c4adcd8823cb5ff2f98
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Tue Nov 15 21:57:53 2016 -0500

     Added rotate (around world origin); not thoroughly tested

commit bd529e17ef84cf88f5380cd54596f9efb5234d32
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Tue Nov 15 20:58:11 2016 -0500

     Scale added; not thoroughly tested

commit a89cd9f227f67b32509f1d95453f82d1540573fb
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Tue Nov 15 16:59:08 2016 -0500

     Tweak the gitignore a bit

commit f969c91346d4f567dafb81e704aedbb705995c87
Author: Eszter Offertaler <eo2309@columbia.edu>
```

```
Date:   Mon Nov 14 00:41:44 2016 -0500

     Added translate to cork executable

commit 00147f78abde1835fa0a7d3d25317cc279ad7cbe
Merge: 35824fb 232e59b
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 11 17:37:58 2016 -0500

     Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 35824fbd7c0c8cbb37599d2465147516f7e5a2f2
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 11 17:37:55 2016 -0500

     Basic display worksish; not all models show up correctly for some
reason??

commit 232e59bebfda9e0f756d4b8dad686cf58c4ff43a
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Fri Nov 11 00:03:24 2016 -0500

     Added StringLiteral printing

commit aeb529fbda45fc18151d646d25f3962aeeaf4623
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Thu Nov 10 23:07:56 2016 -0500

     Added separate print functions for doubles and integers

commit 3a2262037b9635508a1e6a3da8c2c7cdb009eeed
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Thu Nov 10 22:51:43 2016 -0500

     Modified expr builder to support different literal types

commit bce8475ba03d8d05bfd1ba7fde15527f8092d9ef
Merge: 37b8cd7 6dec998
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Thu Nov 10 22:01:46 2016 -0500

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit 37b8cd7b7bbbb00bfcfcc4015c53bbc45767a068
Author: Ishan Guru <ig2333@columbia.edu>
Date:   Thu Nov 10 22:01:20 2016 -0500

     Adding DoubleLiteral support to codegen
```

```
commit 6dec9989f0bc93b895458633da06b666fe06193f
Author: rashidakamal <rsk2161@columbia.edu>
Date:   Thu Nov 10 21:49:39 2016 -0500

     double added to codegen

commit 5aa131bb63cf74d3929a8541195467e84d6f5c20
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Wed Nov 9 00:52:54 2016 -0500

     display almost works; indices are weird

commit 4aa9abfbbe04fb90d5ce99a38b7bb45ab11e0286
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Tue Nov 8 10:20:01 2016 -0500

     Linked llibcork.a properly for display

commit c17b3611ac1041b53a8d1d6bb57808cd380273d3
Merge: f536242 a364aba
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Nov 4 18:25:28 2016 -0400

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit f536242819459b0e152a513c5254a3f6cd1ef860
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Nov 4 18:25:12 2016 -0400

     Remove non-generic types for now.

commit a364aba7cde21aeff167743e859ca64271226030
Merge: 632708c f5c83ee
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 4 18:24:01 2016 -0400

     Merge branch 'master' of github.com:nyletara/ShapeShifter

commit 632708c435edb042a8e0b0b0626b27b64cec2d06
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 4 17:51:27 2016 -0400

     Trying to link libcork fail

commit f5c83ee848f24fa7261ad3cf7ceeb5e3205598e8
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Nov 4 17:46:28 2016 -0400
```

```
    Minor fixes for semantic analysis (still disabled).

commit 03938d983726a2ccfa27c94be7848d1fdff0248c
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 4 17:20:37 2016 -0400

    Trying to hook Cork up so we can use their file I/O

commit 4e1be840752fc790c072d576442b4abd06934571
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 4 16:56:47 2016 -0400

    Make display executable; displays whatever is in .OFF input file

commit e9a5b69128540019cadf1b0382cb5adf29388b47
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Nov 4 16:18:26 2016 -0400

    Adding virgin Cork library

commit 443d1f32facdf4f76476059c4ea75ba5ffb65f0a
Author: RajivJT <rjt2132@columbia.edu>
Date:   Fri Nov 4 13:57:21 2016 -0400

    Add AST pretty-printing functionality.

commit b41b257d3b2bd32600a02c7a89a8ef981ad25d1c
Merge: 3bd93b7 f373c32
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 23:57:13 2016 -0400

    Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit 3bd93b7fbf679611633727473303137bb8a0bddd
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 23:56:10 2016 -0400

    Add support for single-line comments.

commit f373c326b2935bb70032d88812d78a284ecbc3cd
Author: nyletara <iguru95@gmail.com>
Date:   Thu Nov 3 22:52:40 2016 -0400

    Forgot to declare array with all other variables

commit 13c4954db6bef15941e34a4331bc3de0dc8e8382
Merge: 38c1bed 86e3c4f
```

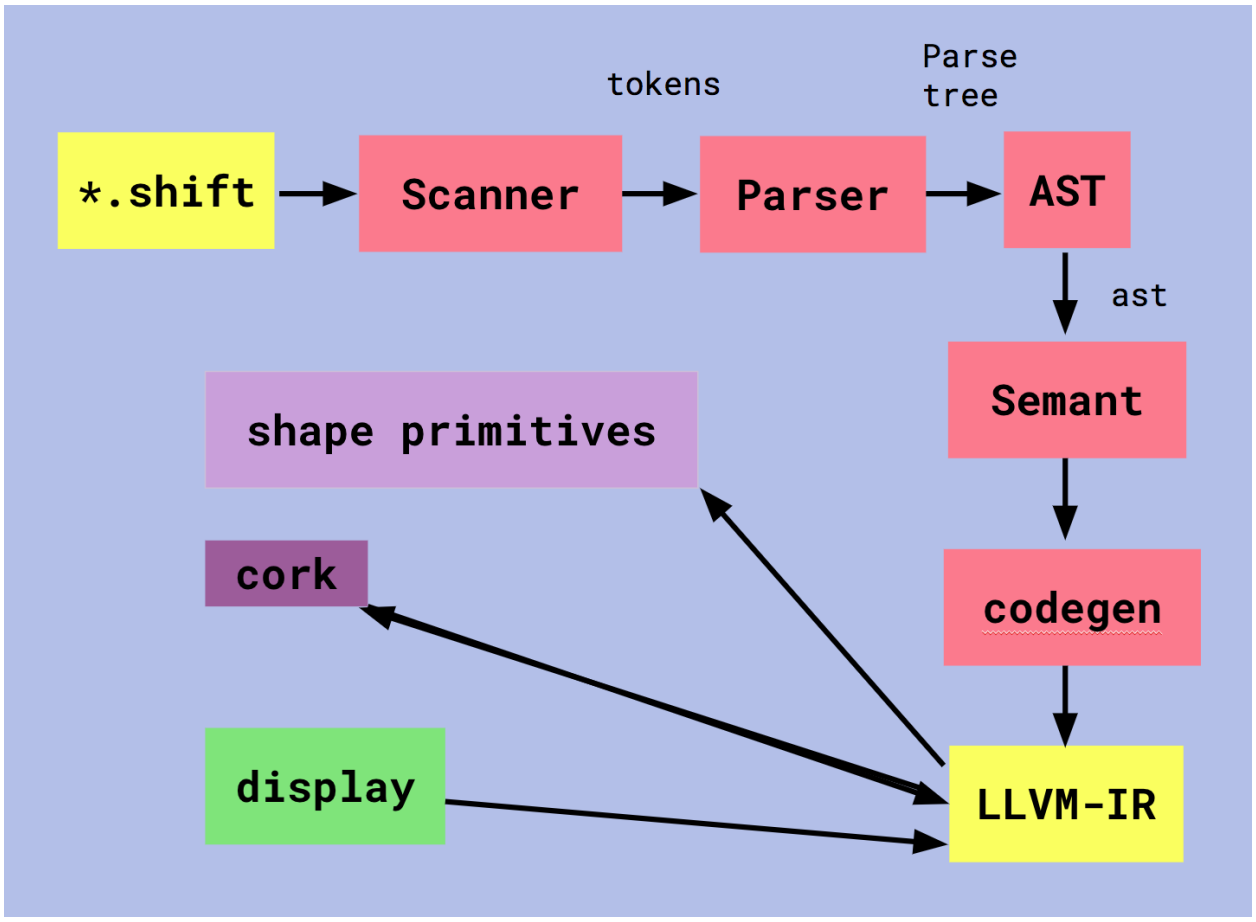
```
Author: nyletara <iguru95@gmail.com>
Date:   Thu Nov 3 22:44:39 2016 -0400

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit 38c1bed3469d6fea36eb94a594f9db86ce8d1dd8
Author: nyletara <iguru95@gmail.com>
Date:   Thu Nov 3 22:44:32 2016 -0400

     Modified shapetests to support AST

commit 86e3c4f15ea0cd5e188105c69018a3becfe32180
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 22:28:09 2016 -0400

     Modify testing script to use shapetests instead of tests.

commit 0d5747088444a1cf92a4df799fa317cd3c36d4cc
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 21:42:01 2016 -0400

     Remove semantic checking for now, and modify a file to test
pretty-printing.

commit 5b7761236b609fe12435cda13cae4ad55fb18c00
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 21:17:52 2016 -0400

     Modify files so that shapeshifter builds (however, pattern
matching is not exhaustive).

commit fc0910c78316de5497c87c861e23afe3792b2bb2
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 21:15:07 2016 -0400

     Update AST to match scanner and parser format.

commit 00ef5b90039c61bd6d12c5ffbf5d4d9b4b16a432
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 20:45:30 2016 -0400

     Change toplevel name from microc to shapeshifter.

commit 21cb9b5997ef5e86b6beaeb8eb0f21f894670793
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 20:38:33 2016 -0400

     Update parser to match LRM.
```

```
commit a8931004725dafc15cde95079f0d08a1aab6f2e9
Author: RajivJT <rjt2132@columbia.edu>
Date:   Thu Nov 3 18:18:27 2016 -0400

     Update scanner to match LRM.

commit 618bc55bf5a868e5a8060cb8912a71b5b29b5da5
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Oct 31 16:55:29 2016 -0400

     Adding bool operator tests, invalid return type tests

commit 65788976c2aa4880323f7050c6a882a74095f135
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Mon Oct 31 15:39:03 2016 -0400

     Updated tests to reflect LRM

commit f9c9502cd7b29a58895cfbf45736b3a347b69657
Author: RajivJT <rjt2132@columbia.edu>
Date:   Wed Oct 26 02:27:00 2016 -0400

     Update parser to accept new scanner tokens.

commit 385f12475c5d65b4c170b0463a0ccc5fc306cba1
Author: RajivJT <rjt2132@columbia.edu>
Date:   Tue Oct 25 23:39:06 2016 -0400

     Modify scanner.mll to compile with ocamllex.

commit 50a651db89fcfe26eb5a9c120968b07e95ba86b6
Merge: bd98222 26c4e0a
Author: nyletara <iguru95@gmail.com>
Date:   Fri Oct 21 18:56:56 2016 -0400

     Merge branch 'master' of https://github.com/nyletara/ShapeShifter

commit bd98222723025671ae3e568ef881ca0d8f3c14ae
Author: nyletara <iguru95@gmail.com>
Date:   Fri Oct 21 18:56:17 2016 -0400

     Added tokens to Scanner for specfific shapes

commit 26c4e0a41f60a004b614de2b84fe4d14807359dd
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Oct 21 18:45:44 2016 -0400
```

```
      Shape->Scene test type

commit e706979adceb4db5116eafb10a105ab1d6d82cb5
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Oct 21 17:38:28 2016 -0400

      More basic tests; loops, function

commit bf84d6ecaea11b14d1d21b47dbd992d93e4e7265
Author: Eszter Offertaler <eo2309@columbia.edu>
Date:   Fri Oct 21 17:03:28 2016 -0400

      Super simple test cases

commit 6af1bbbcf140b799f9eafe1a3417fab6340b2c0d
Author: nyletara <iguru95@gmail.com>
Date:   Thu Oct 20 01:35:17 2016 -0400

      Modified scanner + ast for ShapeShifter

commit 03e33e407d811e1c2935e94f01dbdfdad0bb13a6
Author: nyletara <iguru95@gmail.com>
Date:   Wed Oct 19 01:45:24 2016 -0400

      Modified scanner and parser

commit cc70cb085da9728d26b22d9b0b858e783a6c8a0b
Author: nyletara <iguru95@gmail.com>
Date:   Mon Oct 17 02:29:36 2016 -0400

      Adding all microc files

commit a51c648b4733f02b900db03bd8fd10584802d26c
Author: nyletara <nyletara@users.noreply.github.com>
Date:   Fri Oct 14 18:20:04 2016 -0400

      Initial commit
```

Architectural Design

The system architecture for Shapeshifter can be broken down into the parts shown, along with how they are linked, in the diagram below:



The ShapeShifter compiler begins by scanning the source file to produce a series of tokens. These tokens are then passed to the parser, which generates an AST (abstract syntax tree) that fully defines the program. The semantic analyzer checks the AST to ensure that the tree represents a logical program, and the AST is then passed to codegen. The codegen module converts the AST to LLVM IR, which can then be invoked with the `lli` command.

At runtime, a ShapeShifter binary communicates with several other entities to create, modify, and display shapes. ShapeShifter has a primitives folder (graphics/.primitives) that contains files with mesh data for each of the five basic primitive shapes, which are combined to form new shapes in Shapeshifter. These were created by the team using Blender and converted to the appropriate format in meshlab. Cork is the library that we use to perform transformations on shapes - at runtime, our LLVM code calls Cork functions on temporary shape files to create new shapes. Finally, our display program is used to render shapes to the screen. We wrote the display program ourselves, and we added several functions for rigid transformations to the Cork library to get the behavior we wanted.

How Shapes Work

Since Cork operates primarily on files, we had to devise a system to deal with Shape objects programmatically. When a program starts, we create a temporary file directory to store intermediate files. When a new Shape variable is created, we copy a primitive from the primitives folder to the temporary file directory. Our runtime establishes a mapping between variables and file names (which are created via a random hash). Thus, as far as our LLVM IR is concerned, Shape objects are simply strings that correspond to file names. When the program terminates, the temporary directory is deleted. If the user wants to store a shape file permanently, they need to invoke the Save() function.

The Compiler

- **cork library:** Eszter
- **rendering program:** Eszter
- **.off shape primitives**: Eszter
- **codegen.ml**: Eszter, Ishan, Rashida, and Rajiv
- **parser.mly**: Ishan and Rajiv
- **scanner.mll**: Ishan and Rajiv
- **semant.ml**: Rajiv
- **shapeshifter.ml**: Everyone
- **test suite**: Rashida and Stephanie

Test Plan

There were two main sets of tests within the test suite. The primary test suite was broken down into components, each of which focused on testing one specific feature of the language. The components were: arithmetic operations, boolean operations, comparative operations, control flow structures, data types, functions, logical operators, shapes, shape transformations (like `Translate()` and `Rotate()`), shape boolean transformations (like `Union()` and `Intersect()`). While the testers attempted to be imaginative about the tests written for this suite, we found that several of our tests were also prompted by encountered bugs directly.

The second set of tests were contained in what we called the "sandbox." The sandbox contained programs that were beyond the ambitions of the regular rest of the test suite. The programs here either attempted to accomplish a much more complex series of transformations (which may or may not encounter errors more related to our Cork external library rather than our language) or were samples of what not-yet-implemented features (such as arrays) would look like. The test suite was much more incremental in its testing.

The testing suite was automated -- typing `./testall.sh` ran all the folders specified in the script. If the output of each program matched what we expected the output to be, that test was noted as "OK" in stdout. `testall.log` noted all the commands that were run and captured all related errors -- including what had been read out to stderr, if something had been outputted. `./sandbox` ran only the script over the programs that had been isolated in the related folder, but provided the same analysis.

The team also made use of the Render() function to obtain visual cues for the correctness of Shape-related tests. These were not automated, but used in a casual manner to get quick feedback for whether the created and transformed Shapes 'looked right'.

Sample .shift → .ll

Source: test_basic_transformations.shift

```
// Translate, rotate, scale a sphere

int scene() {
      Shape sph;
      sph = SPHERE;
      Scale(sph, 2.0, 1.0, 1.5);
      Rotate(sph, 0.0, 0.0, -90.0);
      Translate(sph, 0.0, 2.0, 0.0);

      print("Basic transformations work.");
}
```

Target: test_basic_transformations.shift

```
; ModuleID = 'ShapeShifter'

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@sph = private unnamed_addr constant [23 x i8]
c"./.tmp/00043945109.off\00"
@0 = private unnamed_addr constant [60 x i8] c"cp
./graphics/.primitives/sphere.off ./.tmp/00043945109.off\00"
@1 = private unnamed_addr constant [65 x i8]
c"./graphics/cork/bin/cork -scale ./.tmp/00043945109.off 2. 1. 1.5\00"
@2 = private unnamed_addr constant [67 x i8]
c"./graphics/cork/bin/cork -rotate ./.tmp/00043945109.off 0. 0.
-90.\00"
@3 = private unnamed_addr constant [68 x i8]
c"./graphics/cork/bin/cork -translate ./.tmp/00043945109.off 0. 2.
0.\00"
@4 = private unnamed_addr constant [28 x i8] c"Basic transformations
work.\00"

declare i32 @system(i8*)

declare i32 @printf(i8*, ...)

define i32 @main() {
entry:
  %sph = alloca i8*
   %sphereprimf = call i32 @system(i8* getelementptr inbounds ([60 x
i8], [60 x i8]* @0, i32 0, i32 0))
   %scalef = call i32 @system(i8* getelementptr inbounds ([65 x i8],
[65 x i8]* @1, i32 0, i32 0))
   %rotatef = call i32 @system(i8* getelementptr inbounds ([67 x i8],
[67 x i8]* @2, i32 0, i32 0))
   %translatef = call i32 @system(i8* getelementptr inbounds ([68 x
i8], [68 x i8]* @3, i32 0, i32 0))
  %str_printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds
([28 x i8], [28 x i8]* @4, i32 0, i32 0))
  ret i32 0
}
```

Source: test_if.shift

```
int scene() {
     int x = 10;
```

```
      int y = 20;
      int z = 1;

      if(x < y) {
            print(x);
      }
      else {
            print(y);
      }
}
```

Target: test_if.shift

```
; ModuleID = 'ShapeShifter'

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%f\0A\00"

declare i32 @system(i8*)

declare i32 @printf(i8*, ...)

define i32 @main() {
entry:
  %x = alloca i32
  store i32 10, i32* %x
  %y = alloca i32
  store i32 20, i32* %y
  %z = alloca i32
  store i32 1, i32* %z
  %x1 = load i32, i32* %x
  %y2 = load i32, i32* %y
  %tmp = icmp slt i32 %x1, %y2
  br i1 %tmp, label %then, label %else

merge:                                            ;   preds   =   %else,
%then
  ret i32 0

then:                                             ; preds = %entry
  %x3 = load i32, i32* %x
  %int_printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds
([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %x3)
  br label %merge

else:                                             ; preds = %entry
```

```
  %y4 = load i32, i32* %y
    %int_printf5 = call i32 (i8*, ...) @printf(i8* getelementptr
inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %y4)
  br label %merge
}
```

Source: test_recurse_fibonnaci.shift

```
int fib(int x) {
     if (x == 1)
          return 1;
     if (x == 0)
          return 1;

     return fib(x-1) + fib(x-2);
}

int scene() {
     print(fib(10));
}
```

Target: test_recurse_fibonnaci.shift

```
; ModuleID = 'ShapeShifter'

@0 = private unnamed_addr constant [17 x i8] c"mkdir -p ./.tmp/\00"
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@1 = private unnamed_addr constant [15 x i8] c"rm -rf ./.tmp/\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@fmt.3 = private unnamed_addr constant [4 x i8] c"%f\0A\00"

declare i32 @system(i8*)

declare i32 @printf(i8*, ...)

define i32 @main() {
entry:
   %mkdir_tmpf = call i32 @system(i8* getelementptr inbounds ([17 x
i8], [17 x i8]* @0, i32 0, i32 0))
  %x = alloca i32
  %fib_result = call i32 @fib(i32 5)
  store i32 %fib_result, i32* %x
  %x1 = load i32, i32* %x
```

```
  %int_printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds
([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %x1)
   %rmdir_tmpf = call i32 @system(i8* getelementptr inbounds ([15 x
i8], [15 x i8]* @1, i32 0, i32 0))
  ret i32 0
}

define i32 @fib(i32 %x) {
entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %x2 = load i32, i32* %x1
  %tmp = icmp eq i32 %x2, 1
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %x3 = load i32, i32* %x1
  %tmp4 = icmp eq i32 %x3, 0
  br i1 %tmp4, label %then6, label %else7

then:                                             ; preds = %entry
  ret i32 1

else:                                             ; preds = %entry
  br label %merge

merge5:                                           ; preds = %else7
  %x8 = load i32, i32* %x1
  %tmp9 = sub i32 %x8, 1
  %fib_result = call i32 @fib(i32 %tmp9)
  %x10 = load i32, i32* %x1
  %tmp11 = sub i32 %x10, 2
  %fib_result12 = call i32 @fib(i32 %tmp11)
  %tmp13 = add i32 %fib_result, %fib_result12
  ret i32 %tmp13

then6:                                            ; preds = %merge
  ret i32 1

else7:                                            ; preds = %merge
  br label %merge5
}
```

Lessons Learned

Stephanie Burgos: My biggest lesson learned is that you cannot assume a feature works simply because one basic test has been passed. Along with that I achieved a greater appreciation for what testing can reveal in this course because my previous experience with testing had made me feel as though unit tests were just a hoop to jump through before your code could be allowed into production.

Advice: I recommend writing sample programs that you would possibly want to use ahead of time and tailoring your tests to make sure all the features that those programs require definitely work. In this way you can write better, more specific tests and when it comes time to prepare your program for the final demo you know that everything you need for it is ready.

Ishan Guru: As a first major project in a functional programming language, learning the nitty-gritty details of how OCaml works in general was a huge hurdle to overcome before we could even begin properly working on the compiler. Combining that with compiling down to LLVM simply resulted in hours and hours of research. I think the research phase took up the majority of time, which is why not having firm "code based" targets that we could achieve came back to bite, since although lots of research was done, codegen.ml and OCaml still seemed alien. I feel like once smaller features were added bit by bit, I got the hang of how OCaml works more, which really got the project going. Having code based targets to achieve early on would be helpful in learning OCaml, so even if it wouldn't have an impact on the project immediately, it would be more helpful in the long run. Another huge learning point from this project was the exposure it gave to graphics and graphics libraries simply because of how diverse our team was with previous CS knowledge. It was great learning from other members on the team and coming up with a creative project together was an amazing experience. I feel like the willingness to learn and try something new was something everyone in the team had, which is what made this project as enjoyable of a learning experience as it was.

Advice: It is essential to set up multiple, significant milestones for the language; dedicate yourself to achieving them, and then test, and test, and test. Testing even the smallest of things in the most obscure way possible is absolutely essential, since you never know what little feature you've forgotten to support in the language.

Rashida Kamal: This was my first CS course as an M.S. student who did not study Computer Science in undergrad -- it was definitely a challenge. As a former linguistics student, I thoroughly enjoyed learning about how another discipline thought about some similar topics. I'm so glad to have had the team that I did -- I think we did a great job supporting and encouraging each other. I preferred to share any insights or research related to problems we were encountering as often as possible -- I found that this was helpful in keeping us engaged with the problems at hand even when we felt "stuck."

Advice: Create systems to capture insights gained from one session to the next, one team member to the next. Github's Issues lends itself particularly well to identifying and tracking problems that the whole team can be aware of (and help solve).

Eszter Offertaler: This was my first time programming in something that isn't a C-like or OOP language. It was rough getting used to the functional programming paradigm, particularly due to the lack of explicit return types. The poetic quote we were shown at the beginning of class is totally valid though; Ocaml does so much with so little, once you manage to get everything to compile.

I also learned that integration testing is at least as important as unit testing. Our method for Shape storage - assign randomly hashed filenames for each Shape, which the compiler stores and changes as appropriate with assignments and the like - worked in our simple tests. However, once we started combining the Shape stuff with functions and looping, we found that it actually wasn't robust to these behaviors, and we'd end up trying to run executables on files that didn't exist. We probably should have realized this either beforehand, when planning out the language, or earlier testing.

Integrating the graphics and boolean libraries was also an interesting challenge. We ended up settling on writing executables to call in the LLVM IR, which proved an elegant if hacky solution. I regret not doing more stress testing of the boolean library though; we chose it for its simplicity and ease-of-use, and it seemed to behave well under our simple tests. However, once we were trying to create cool shapes in our language, we found, to our bitter disappointment, that it had instabilities and memory leaks that would be highly compromised after 5 shape interactions or so. Boolean mesh operations are a hard problem, and it was nice to not have to implement that ourselves, but it still stung a little. It was definitely a satisfying project though, and having a good, cohesive team was one of the best things we had going for us.

Advice: Write tests! Lots of tests. Moreover, make sure they're the right kind of tests; unit tests are important when starting out or isolating issues, but we found the most crippling bugs only when we tried writing full-fledged programs incorporating multiple aspects of our language. As soon as you get multiple components working, write tests integrating them.

Also, if using external libraries, stress test them beforehand to make sure they hold up to your intended use.

Rajiv Thamburaj: This was my first significant project in a functional programming language. While the learning curve was steep, I found features like type inference, pattern matching, and forced immutability to be very helpful when I needed to reason about my code. I thought the MicroC compiler was a great starting point for our project - it helped us establish a basic structure for our codebase, but it was also limited enough

that it allowed us to move significantly beyond the original implementation with some novel features.

Advice: Learn OCaml (and maybe even LLVM) as early as possible. The better you know the tools you're working with, the more time you have to spend designing your language rather than looking up syntax errors.

Appendix

ast.ml - Rajiv

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

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

type uop = Neg | Not

type typ = Int | Dbl | Bool | String | Shape | Void

type bind = typ * string

type expr =
     IntLit of int
  | DblLit of float
  | StrLit of string
  | BoolLit of bool
  | SpherePrim
  | CubePrim
  | CylinderPrim
  | TetraPrim
  | ConePrim
  | Id of string
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of string * expr
  | Call of string * expr list
  | Noexpr

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
  | Local of typ * string * expr

type func_decl = {
     typ : typ;
     fname : string;
```

```
      formals : 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 -> "||"
  | Union -> "UN"
  | Intersect -> "IN"
  | Difference -> "DI"

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

let rec string_of_expr = function
  | IntLit(i) -> string_of_int i
  | DblLit(d) -> string_of_float d
  | StrLit(l) -> l
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
  | SpherePrim -> "SPHERE"
  | CubePrim -> "CUBE"
  | CylinderPrim -> "CYLINDER"
  | TetraPrim -> "TETRA"
  | ConePrim -> "CONE"
  | 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(v, e) -> v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
```

```
  | Noexpr -> ""

let string_of_typ = function
     Int -> "int"
  | Bool -> "bool"
  | Dbl -> "double"
  | String -> "string"
  | Shape -> "Shape"
  | Void -> "void"

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
  | Local(t, s, e) -> if (String.length (string_of_expr e)) = 0 then
     string_of_typ t ^ " " ^ s ^ ";\n" else
     string_of_typ t ^ " " ^ s ^ " = " ^ string_of_expr e ^ ";\n"

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_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)
```

codegen.ml - Eszter, Ishan, Rajiv, Rashida

```
module L = Llvm
module A = Ast

module StringMap = Map.Make(String)

let _ = Random.self_init()

let local_vars:(string, L.llvalue) Hashtbl.t = Hashtbl.create 50
let type_map:(string, A.typ) Hashtbl.t = Hashtbl.create 50
(* Store the underlying filenames for each shape *)
let shape_map:(string, string) Hashtbl.t = Hashtbl.create 50
let shstr_map:(string, L.llvalue) Hashtbl.t = Hashtbl.create 50


let translate (globals, functions) =
  let context = L.global_context () in
  let the_module = L.create_module context "ShapeShifter"
  and i32_t      = L.i32_type  context
  and i8_t       = L.i8_type   context
  and i1_t       = L.i1_type   context
  and void_t   = L.void_type context
  and double_t = L.double_type context in
  (* and i64_t  = L.i64_type  context in *)
  (* let i64_pt   = L.pointer_type i64_t  *)
  (* and i32_pt   = L.pointer_type i32_t  *)
  let i8_pt     = L.pointer_type i8_t in

  let ltype_of_typ = function
      A.Int -> i32_t
      | A.Bool -> i1_t
      | A.Void -> void_t
      | A.String -> i8_pt
      | A.Dbl -> double_t
      | A.Shape -> i8_pt (*Make Shape an int just to be able to test
translate *)  in

  (* Declare each global variable; remember its value in a map *)
  let global_vars =
      let global_var m (t, n) =
      let init = 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

  (* Declare system functions we need *)
```

```
  let system_t = L.function_type i32_t [| i8_pt |] in
  let system_func = L.declare_function "system" system_t the_module in

  (* Declare Shapeshifter functions (need to finish enumerating) *)

  let shflen = 11 in (* fixed-size shape file string length *)

  (* Declare printf(), which the print built-in function will call *)
   let printf_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t
|] in
  let printf_func = L.declare_function "printf" printf_t the_module in

   (* Define each function (arguments and return type) so we can call
it *)
  let function_decls =
      let function_decl m fdecl =
      let name =
            if fdecl.A.fname = "scene" then "main" else fdecl.A.fname
      and formal_types =
      Array.of_list   (List.map   (fun   (t,_)   ->   ltype_of_typ   t)
fdecl.A.formals)
      in  let  ftype  =  L.function_type  (ltype_of_typ  fdecl.A.typ)
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 name =
      if fdecl.A.fname = "scene" then "main" else fdecl.A.fname in
      let (the_function, _) = StringMap.find name function_decls in
      let   builder   =   L.builder_at_end   context   (L.entry_block
the_function) in

      let tmp_folder = "./.tmp/" in

      let make_tmp_cmd = "mkdir -p "^tmp_folder in
      let rm_tmp_cmd = "rm -rf "^tmp_folder in (* Wow is this command
dangerous ;) *)
      let make_temp_dir =
      (* Add mkdir command here if name = "main" *)
      (match name with
      "main" ->
      let  string_head  =  L.build_global_stringptr  make_tmp_cmd  ""
builder in
      let zero_const = L.const_int i32_t 0 in
```

```
     let str = L.build_in_bounds_gep string_head [| zero_const |] ""
builder in
     ignore(L.build_call system_func [| str |] "mkdir_tmpf" builder);
     | _ -> ()
     ) in
     make_temp_dir;

     (* Cork executable and commands *)
     let get_cork_cmd func args =
     let cork_exec = "./graphics/cork/bin/cork"
     and render_exec = "./graphics/display/sshiftdisplay" in

     let cork_cmd f =
     (match f with
          "Translate"     -> cork_exec ^ " -translate"
          | "Reflect"     -> cork_exec ^ " -reflect"
          | "Rotate"      -> cork_exec ^ " -rotate"
          | "Scale"       -> cork_exec ^ " -scale"
          | "Union"       -> cork_exec ^ " -union"
          | "Difference"  -> cork_exec ^ " -diff"
          | "Intersect"   -> cork_exec ^ " -isct"
          | "Save"        -> "cp"
          | "Copy"        -> "cp"
          | "Render"      -> render_exec
          | "Xor"         -> cork_exec ^ " -xor"
          | _ -> raise (Failure "Incorrect cork_cmd")
     )
     in
     (cork_cmd func) ^ " " ^ args

     in

     let get_prim_file p =
     match p with
     A.ConePrim      -> "./graphics/.primitives/cone.off"
     | A.CubePrim    -> "./graphics/.primitives/cube.off"
     | A.CylinderPrim-> "./graphics/.primitives/cylinder.off"
     | A.SpherePrim  -> "./graphics/.primitives/sphere.off"
     | A.TetraPrim   -> "./graphics/.primitives/tetra.off"
     | _ -> raise (Failure "Shape file for specified primitive not
found")
     in

     (* Create a temp file for each shape from random int; collisions
not checked but unlikely yolo *)
     let add_shape_type ty na=
     match ty with
     A.Shape ->
```

```
      let shnum = string_of_int(Random.int 100000000) in
      let pad0 = String.make (shflen - String.length shnum) '0' in
      let shape_file = tmp_folder ^ (pad0) ^ shnum ^ ".off" in
      ignore (Hashtbl.add shape_map na shape_file);
      ignore   (Hashtbl.add   shstr_map   na   (L.build_global_stringptr
shape_file na builder));

      | _ -> () in

      let   int_format_str   =   L.build_global_stringptr   "%d\n"   "fmt"
builder in
      let   dbl_format_str   =   L.build_global_stringptr   "%f\n"   "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 _ =
      ignore (Hashtbl.clear local_vars);
      ignore (Hashtbl.clear type_map);
      ignore (Hashtbl.clear shape_map);
      let add_formal (t, n) p =
      add_shape_type t n;
      L.set_value_name n p;
      let local = L.build_alloca (ltype_of_typ t) n builder in
           ignore (L.build_store p local builder);
           ignore (Hashtbl.add local_vars n local);
      in
      List.iter2  add_formal  fdecl.A.formals  (Array.to_list  (L.params
the_function))
      in

      (* Add types as necessary *)
      let rec string_of_expr = function
      | A.Id(s)       -> s
      | A.DblLit(s)   -> string_of_float s
      | A.IntLit(s)   -> string_of_int s
      | A.StrLit(s)   -> s
      | A.ConePrim    -> "coneprim"
      | A.CubePrim    -> "cubeprim"
      | A.CylinderPrim  -> "cylinderprim"
      | A.SpherePrim  -> "sphereprim"
      | A.TetraPrim   -> "tetraprim"
      | A.Unop(o, z)  ->
           (match z with
           | A.DblLit(s) -> A.string_of_uop o ^ string_of_float s
```

```
          | _ -> raise (Failure "Invalid"))
      | A.Binop(e1, _, _) -> string_of_expr e1
      | _ -> raise (Failure "Invalid")
      in

      let build_string s n expr =
       let string_head = expr builder (A.StrLit s) in
      let zero_const = L.const_int i32_t 0 in
      let  str  =  L.build_in_bounds_gep  string_head  [|  zero_const  |]
(n^"_str") builder in
      L.build_call system_func [| str |] n builder
      in

      let lookup n =
      try Hashtbl.find local_vars n
      with Not_found -> StringMap.find n global_vars
      in

      let _ =
      let add_type (t, n) =
      Hashtbl.add type_map n t
      in
      List.iter add_type fdecl.A.formals
      in

      let lookup_type n =
      Hashtbl.find type_map n
      in

      let integer_ops op =
      (match op with
      A.Add      -> L.build_add
      | A.Sub    -> L.build_sub
      | A.Mult   -> L.build_mul
      | A.Div    -> L.build_sdiv
      | 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
      | _ -> raise (Failure "Integer operator not found")
      )
      in

      let double_ops op =
```

```
      (match op with
      A.Add      -> L.build_fadd
      | A.Sub    -> L.build_fsub
      | A.Mult   -> L.build_fmul
      | A.Div    -> L.build_fdiv
      | A.And    -> L.build_and
      | A.Or     -> L.build_or
      | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
      | A.Neq    -> L.build_fcmp L.Fcmp.One
      | A.Less   -> L.build_fcmp L.Fcmp.Ult
      | A.Leq    -> L.build_fcmp L.Fcmp.Ole
      | A.Greater -> L.build_fcmp L.Fcmp.Ogt
      | A.Geq    -> L.build_fcmp L.Fcmp.Oge
      | _ -> raise (Failure "Double operator not found")
      )
      in

      (* Construct code for an expression; return its value *)
      let rec expr builder = function
      | A.DblLit d -> L.const_float double_t d

      | A.StrLit s -> L.build_global_stringptr s "" builder

       | A.IntLit i -> L.const_int i32_t i

      | A.BoolLit b -> L.const_int i1_t (if b then 1 else 0)

      | A.Noexpr -> L.const_int i32_t 0

      | A.Id s -> L.build_load (lookup s) s builder
      | A.Binop (e1, op, e2) ->
      let e1' = expr builder e1
      and e2' = expr builder e2 in

      (match e1 with
         | A.IntLit _ -> (integer_ops op) e1' e2' "tmp" builder
           | A.DblLit _ -> (double_ops op) e1' e2' "tmp" builder
           | A.Id id ->
           (* We need to match with each type that ID can take, use
StringMap for storing types *)
           let variable_type = lookup_type id in
           (
           match variable_type with
           | A.Dbl ->  (double_ops op) e1' e2' "tmp" builder
           | A.Int ->  (integer_ops op) e1' e2' "tmp" builder
           | _ -> raise (Failure "A.Id variable type not found")
           )
```

```
          | A.Call (st, [_]) -> (let fdecl = snd (StringMap.find st
function_decls)
                                in match fdecl.A.typ with
                                      | A.Dbl ->  (double_ops op) e1'
e2' "tmp" builder
                                      | A.Int ->  (integer_ops op) e1'
e2' "tmp" builder
                                      | _ -> raise (Failure "Type for
call not found"))
          | _ -> raise (Failure "No match found for specified binop")

     )

     | A.Unop(op, e) ->
     let e' = expr builder e in
     (match op with
          | A.Neg    ->
          (match e with
          | A.IntLit _ -> L.build_neg
          | A.DblLit _ -> L.build_fneg
          | A.Id id ->
               let variable_type = lookup_type id in
               (
                    match variable_type with
                    | A.Dbl -> L.build_fneg
                    | A.Int -> L.build_neg
                    | _ -> raise (Failure "A.Id type not found for
A.Neg")
               )
          | _ -> raise (Failure "Invalid Operator")
          )
          | A.Not    -> L.build_not) e' "tmp" builder
          (* | _ -> raise (Failure "Unop not supported")) e' "tmp"
builder *)

     | A.Assign (s, e) ->
     let make_prim_cmd p n =
     (
          let prim_cmd = get_cork_cmd "Save" (String.concat " "
                         [(get_prim_file p);
                         (Hashtbl.find shape_map n)]) in
          ignore  (build_string  prim_cmd  ((string_of_expr  p  )^"f")
expr);
          (*Hashtbl.find shstr_map n*)
          L.const_int i32_t 0
     )
     in
```

```
     let make_boolop_cmd op s1 s2 n =
          let cmd_str = get_cork_cmd op (String.concat " "
                              [(Hashtbl.find                    shape_map
(string_of_expr(s1)));
                              (Hashtbl.find                     shape_map
(string_of_expr(s2)));
                              (Hashtbl.find shape_map n)]) in
          ignore (build_string cmd_str (op^"f") expr);
          (*Hashtbl.find shstr_map n*)
          L.const_int i32_t 0
     in

     (match e with
          A.Id id ->
          let var_type = lookup_type id in
          (match var_type with
          A.Shape ->
          let sf = Hashtbl.find shape_map id in
          ignore (Hashtbl.add shape_map s sf);
          L.const_int i32_t 0
          (*Hashtbl.find shstr_map s *)
          | _ ->
          let e' = expr builder e in
          ignore (L.build_store e' (lookup s) builder); e'
          )
          (* Call primitive constructors *)
          | A.ConePrim    ->
          make_prim_cmd A.ConePrim s
          | A.CubePrim    ->
          make_prim_cmd A.CubePrim s
          | A.CylinderPrim->
          make_prim_cmd A.CylinderPrim s
          | A.SpherePrim  ->
          make_prim_cmd A.SpherePrim s
          | A.TetraPrim   ->
          make_prim_cmd A.TetraPrim s
          (* Boolean shape operations create a new shape*)
          | A.Call ("Union", [s1; s2]) ->
          make_boolop_cmd "Union" s1 s2 s
          | A.Call ("Intersect", [s1; s2]) ->
          make_boolop_cmd "Intersect" s1 s2 s
          | A.Call ("Difference", [s1; s2]) ->
          make_boolop_cmd "Difference" s1 s2 s
          | A.Call ("Xor", [s1; s2]) ->
          make_boolop_cmd "Xor" s1 s2 s
     (*
          | A.Call (f, act) ->
 *)
```

```
          | _ ->
          let e' = expr builder e in
          ignore (L.build_store e' (lookup s) builder); e'
     )

     | A.Call ("print", [e]) ->
     let print_strlit s =
          let string_head = expr builder (s) in
          let zero_const = L.const_int i32_t 0 in
          let str = L.build_in_bounds_gep string_head [| zero_const |]
"str_printf" builder in
          L.build_call printf_func [| str |] "str_printf" builder in

          (match List.hd[e] with
        | A.StrLit _ ->
          print_strlit (List.hd[e])
        | A.DblLit _ -> L.build_call printf_func [| dbl_format_str ;
(expr builder e) |] "dbl_printf" builder
        | A.IntLit _ -> L.build_call printf_func [| int_format_str ;
(expr builder e) |] "int_printf" builder
        | A.Id id    ->
          let variable_type = lookup_type id in
          (
          match variable_type with
               | A.Dbl    ->     L.build_call     printf_func     [|
dbl_format_str ; (expr builder e) |] "dbl_printf" builder
                  | A.Int ->     L.build_call     printf_func     [|
int_format_str ; (expr builder e) |] "int_printf" builder
          | A.String -> print_strlit (List.hd[e])
          | _ -> raise (Failure "Print not specified for this type")

          )
                   | _ -> L.build_call printf_func [| int_format_str ;
(expr builder e) |] "int_printf" builder
          )

     (* Transformation calls *)
     | A.Call ("Reflect", [s; a; b; c]) ->
     let refle_cmd = get_cork_cmd "Reflect" (String.concat " "
                          [(Hashtbl.find                    shape_map
(string_of_expr(s)));
                          string_of_expr(a); string_of_expr(b);
                          string_of_expr(c)]) in
     build_string refle_cmd "reflf" expr;

     | A.Call ("Rotate", [s; x; y; z]) ->
     let rotat_cmd = get_cork_cmd "Rotate" (String.concat " "
```

```
                            [(Hashtbl.find                    shape_map
(string_of_expr(s)));
                            string_of_expr(x); string_of_expr(y);
                            string_of_expr(z)]) in
     build_string rotat_cmd "rotatef" expr;

     | A.Call ("Scale", [s; x; y; z]) ->
     let scale_cmd = get_cork_cmd "Scale" (String.concat " "
                            [(Hashtbl.find                    shape_map
(string_of_expr(s)));
                            string_of_expr(x); string_of_expr(y);
                            string_of_expr(z)]) in
     build_string scale_cmd "scalef" expr;

     | A.Call ("Translate", [s; x; y; z]) ->
     let trans_cmd = get_cork_cmd "Translate" (String.concat " "
                            [(Hashtbl.find                    shape_map
(string_of_expr(s)));
                            string_of_expr(x); string_of_expr(y);
                            string_of_expr(z)]) in
     build_string trans_cmd "translatef" expr;

     | A.Call ("Save", [s; n]) ->
     let save_cmd = get_cork_cmd "Save" (String.concat " "
                            [(Hashtbl.find                    shape_map
(string_of_expr(s)));
                            string_of_expr(n)]) in
     build_string save_cmd "savef" expr;
     | A.Call ("Copy", [s1; s2]) ->
     let copy_cmd = get_cork_cmd "Copy" (String.concat " "
                            [(Hashtbl.find                    shape_map
(string_of_expr(s1)));
                            (Hashtbl.find                     shape_map
(string_of_expr(s2)));]) in
     build_string copy_cmd "copyf" expr;

     | A.Call ("Render", [s]) ->
     let  rend_cmd  =  get_cork_cmd  "Render"  (Hashtbl.find  shape_map
(string_of_expr(s))) in
     build_string rend_cmd "rendf" expr;

     | A.Call (f, act) ->
     let (fdef, fdecl) = StringMap.find f function_decls in
           let actuals = List.rev (List.map (expr builder) (List.rev
act)) in
         let result = (match fdecl.A.typ with A.Void -> ""
                                             | _ -> f ^ "_result") in
     L.build_call fdef (Array.of_list actuals) result builder
```

```
      | _ -> raise (Failure "Match not found in expr builder")

      in


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

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

            let then_bb = L.append_block context "then" the_function in
             add_terminal  (stmt  (L.builder_at_end  context  then_bb)
then_stmt)
            (L.build_br merge_bb);

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

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

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

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

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

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

     | A.For (e1, e2, e3, body) -> stmt builder
     ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ]
)
     | A.Local (t, n, e) ->
     let local = L.build_alloca (ltype_of_typ t) n builder in
          ignore (Hashtbl.add local_vars n local);
          ignore (Hashtbl.add type_map n t);

          add_shape_type t n;

(*
          let build_string2 sp s n expr =
          let string_head = expr builder (A.StrLit s) in
          let zero_const = L.const_int i32_t 0 in
          let str = L.build_in_bounds_gep string_head [| zero_const |]
(n^"_str") builder in
          let ip = L.const_in_bounds_gep
          L.build_store
          L.build_call system_func [| str |] n builder
          in

          let make_prim_cmd p n =
          (
          let prim_cmd = get_cork_cmd "Save" (get_prim_file p) in
          ignore (build_string2 n prim_cmd ((string_of_expr p )^"f")
expr);
          builder
          )
          in
  *)

          let make_prim_cmd p n =
          (
          let prim_cmd = get_cork_cmd "Save" (String.concat " "
                         [(get_prim_file p);
                         (Hashtbl.find shape_map n)]) in
          ignore  (build_string  prim_cmd  ((string_of_expr  p  )^"f")
expr);
          builder
          )
```

```
          in


          let make_boolop_cmd op s1 s2 n =
          let cmd_str = get_cork_cmd op (String.concat " "
                              [(Hashtbl.find                    shape_map
(string_of_expr(s1)));
                              (Hashtbl.find                     shape_map
(string_of_expr(s2)));
                              (Hashtbl.find shape_map n)]) in
          ignore (build_string cmd_str (op^"f") expr);
          builder
          in

          match e with
          (* Call primitive constructors *)
          | A.ConePrim    ->
          make_prim_cmd A.ConePrim n
          | A.CubePrim    ->
          make_prim_cmd A.CubePrim n
          | A.CylinderPrim ->
          make_prim_cmd A.CylinderPrim n
          | A.SpherePrim  ->
          make_prim_cmd A.SpherePrim n
          | A.TetraPrim   ->
          make_prim_cmd A.TetraPrim n
          (* Boolean shape operations create a new shape*)
          | A.Call ("Union", [s1; s2]) ->
          make_boolop_cmd "Union" s1 s2 n
          | A.Call ("Intersect", [s1; s2]) ->
          make_boolop_cmd "Intersect" s1 s2 n
          | A.Call ("Difference", [s1; s2]) ->
          make_boolop_cmd "Difference" s1 s2 n
          | A.Call ("Xor", [s1; s2]) ->
          make_boolop_cmd "Xor" s1 s2 n
(*
          | A.Call (f, act) ->
               (*
          let (fdef, fdecl) = StringMap.find f function_decls in
               let  actuals  =  List.rev  (List.map  (expr  builder)
(List.rev act)) in
               let result = (match fdecl.A.typ with A.Void -> ""
                                        | _ -> f ^ "_result") in
          L.build_call fdef (Array.of_list actuals) result builder
               *)
          print_string("sad");
          let e' = expr builder e in
          ignore (L.build_store e' (lookup n) builder);
```

```
          builder
  *)
          | A.Noexpr -> builder
          | A.Id id->
          let variable_type = lookup_type id in
          (match variable_type with
               A.Shape ->
               (* Set both shapes to have same file representation *)
               let sf = Hashtbl.find shape_map id in
               ignore (Hashtbl.add shape_map n sf);
               let e' = expr builder e in
               ignore (L.build_store e' (lookup n) builder);
               builder
               | _ ->
               let e' = expr builder e in
               ignore (L.build_store e' (lookup n) builder);
               builder
          )
          | _ ->
          let e' = expr builder e in
          ignore (L.build_store e' (lookup n) builder);
          builder
     in

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

     (* Add rm -rf .tmp command if name = main*)
     let rm_temp_dir =
     (match name with
     "main" ->
     let string_head = L.build_global_stringptr rm_tmp_cmd "" builder
in
     let zero_const = L.const_int i32_t 0 in
     let str = L.build_in_bounds_gep string_head [| zero_const |] ""
builder in
     ignore(L.build_call system_func [| str |] "rmdir_tmpf" builder);
     | _ -> ()
     ) in
     rm_temp_dir;

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

  List.iter build_function_body functions;
```

```
the_module
```

parser.mly

```
%{
open Ast
%}

%token SEMI LPAREN RPAREN LBRACE RBRACE LBRACK RBRACK COMMA
%token PLUS MINUS TIMES DIVIDE ASSIGN NOT
%token UNION INTERSECT DIFFERENCE
%token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
%token RETURN IF ELSE FOR WHILE
%token INT BOOL DBL STRING VOID
%token SHAPE
%token SPHERE_PRIM CUBE_PRIM CYLINDER_PRIM TETRA_PRIM CONE_PRIM
%token <int> INT_LIT
%token <string> ID
%token <string> STR_LIT
%token <float> DBL_LIT
%token NULL
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left UNION INTERSECT DIFFERENCE
%left PLUS MINUS
%left TIMES DIVIDE
%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 stmt_list RBRACE
     {
     {
           typ = $1;
           fname = $2;
           formals = $4;
           body = List.rev $7
     }
     }

vdecl:
  | typ ID SEMI { ($1, $2) }

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

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

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

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

typ:
  | INT { Int }
  | DBL { Dbl }
  | BOOL { Bool }
  | STRING { String }
  | SHAPE { Shape }
  | VOID { Void }

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) }
  | typ ID SEMI { Local($1, $2, Noexpr) }
  | typ ID ASSIGN expr SEMI { Local($1, $2, $4) }

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

expr:
  | literal { $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 UNION  expr { Binop($1, Union, $3) }
  | expr INTERSECT expr { Binop($1, Intersect, $3) }
  | expr DIFFERENCE expr { Binop($1, Difference, $3) }
  | MINUS expr %prec NEG { Unop(Neg, $2) }
  | NOT expr          { Unop(Not, $2) }
  | ID ASSIGN expr   { Assign($1, $3) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN { $2 }

literal:
  | INT_LIT           { IntLit($1) }
  | DBL_LIT           { DblLit($1) }
  | STR_LIT           { StrLit($1) }
  | SPHERE_PRIM       { SpherePrim }
  | CUBE_PRIM         { CubePrim }
  | CYLINDER_PRIM     { CylinderPrim }
  | TETRA_PRIM        { TetraPrim }
  | CONE_PRIM         { ConePrim }
  | TRUE              { BoolLit(true) }
  | FALSE             { BoolLit(false) }
  | ID                { Id($1) }
```

prettyprint.ml

```
open Ast   (* this should probably be specified @ command line *)
    (* Generated by ast-pp *)

    (* indentation utilities *)

    type pp_indentation = {
      pp_indentation_curr : string;
      pp_indentation_prev : pp_indentation option
    }

    let pp_indentation_level = ref {
      pp_indentation_curr = "\n";
      pp_indentation_prev = None
    }

    (* balence push_indent with pop_indent *)

    let push_indent _ =
      pp_indentation_level := {
                               pp_indentation_curr                    =
!pp_indentation_level.pp_indentation_curr  ^ "   ";
           pp_indentation_prev = Some(!pp_indentation_level)
      }

    let pop_indent _ =
      match !pp_indentation_level.pp_indentation_prev with
              None -> ()    (* this is actually an error, but we just
leave newline *)
            | Some(prev) -> pp_indentation_level := prev

    let inner_indent f =
      let str = (push_indent(); f()) in (pop_indent(); str)

    (* get the indentation with newline().
     The default is just a newline, additional pushes are tabs.
     This way, anytime you want a new line, it will be properly
indented *)

    let newline _ = !pp_indentation_level.pp_indentation_curr

    (* builtin functions *)

    let enum _ = "enum"
```

```
    let tuple _  = "tuple"
    let string_of_string str = "\"" ^ str ^ "\""
    let string_of_list f l =
       "["
       ^ (inner_indent (fun _ -> newline() ^
            String.concat (";"^newline()) (List.map f l)))
       ^ newline() ^ "]"

    let string_of_option f = function
        None -> "None"
      | Some(x) -> f(x)

     let string_of_ref f r = f (!r)

    let rec  string_of_op obj = (((function Difference -> "Difference"
| Intersect -> "Intersect"
| Union -> "Union"
| Or -> "Or"
| And -> "And"
| Geq -> "Geq"
| Greater -> "Greater"
| Leq -> "Leq"
| Less -> "Less"
| Neq -> "Neq"
| Equal -> "Equal"
| Div -> "Div"
| Mult -> "Mult"
| Sub -> "Sub"
| Add -> "Add")) obj)

 and  string_of_uop obj = (((function Not -> "Not"
| Neg -> "Neg")) obj)

 and  string_of_typ obj = (((function Void -> "Void"
| Shape -> "Shape"
| String -> "String"
| Bool -> "Bool"
| Dbl -> "Dbl"
| Int -> "Int")) obj)

 and  string_of_bind obj = (((fun (obj1, obj2) -> "(" ^ (string_of_typ
obj1) ^ ", " ^ (string_of_string obj2) ^ ")")) obj)

 and  string_of_expr obj = (((function Noexpr -> "Noexpr"
|  Call(obj1,  obj2)  ->  "Call("  ^  (string_of_string  obj1)  ^  ",  "  ^
((string_of_list (string_of_expr)) obj2) ^ ")"
|  Assign(obj1,  obj2)  ->  "Assign("  ^  (string_of_string  obj1)  ^  ",  "  ^
(string_of_expr obj2) ^ ")"
```

```
| Unop(obj1, obj2) -> "Unop(" ^ (string_of_uop obj1) ^ ", " ^
(string_of_expr obj2) ^ ")"
| Binop(obj1, obj2, obj3) -> "Binop(" ^ (string_of_expr obj1) ^ ", " ^
(string_of_op obj2) ^ ", " ^ (string_of_expr obj3) ^ ")"
| Id(x) -> "Id(" ^ string_of_string x^")"
| ConePrim -> "ConePrim"
| TetraPrim -> "TetraPrim"
| CylinderPrim -> "CylinderPrim"
| CubePrim -> "CubePrim"
| SpherePrim -> "SpherePrim"
| BoolLit(x) -> "BoolLit(" ^ string_of_bool x^")"
| StrLit(x) -> "StrLit(" ^ string_of_string x^")"
| DblLit(x) -> "DblLit(" ^ string_of_float x^")"
| IntLit(x) -> "IntLit(" ^ string_of_int x^")")) obj)

 and  string_of_stmt obj = (((function Local(obj1, obj2, obj3) ->
"Local(" ^ (string_of_typ obj1) ^ ", " ^ (string_of_string obj2) ^ ",
" ^ (string_of_expr obj3) ^ ")"
| While(obj1, obj2) -> "While(" ^ (string_of_expr obj1) ^ ", " ^
(string_of_stmt obj2) ^ ")"
| For(obj1, obj2, obj3, obj4) -> "For(" ^ (string_of_expr obj1) ^ ", "
^ (string_of_expr obj2) ^ ", " ^ (string_of_expr obj3) ^ ", " ^
(string_of_stmt obj4) ^ ")"
| If(obj1, obj2, obj3) -> "If(" ^ (string_of_expr obj1) ^ ", " ^
(string_of_stmt obj2) ^ ", " ^ (string_of_stmt obj3) ^ ")"
| Return(x) -> "Return(" ^ string_of_expr x^")"
| Expr(x) -> "Expr(" ^ string_of_expr x^")"
| Block(x) -> "Block(" ^ (string_of_list (string_of_stmt)) x^")")) 
obj)

 and  string_of_func_decl obj = (((fun s -> "{" ^
      inner_indent (fun _ -> newline() ^ ("typ = " ^ string_of_typ
s.typ)
 ^ newline() ^ ("fname = " ^ string_of_string s.fname)
 ^ newline() ^ ("formals = " ^ (string_of_list (string_of_bind))
s.formals)
 ^ newline() ^ ("body = " ^ (string_of_list (string_of_stmt)) s.body))
^ newline() ^ "}")) obj)

 and  string_of_program obj = (((fun (obj1, obj2) -> "(" ^
((string_of_list (string_of_bind)) obj1) ^ ", " ^ ((string_of_list
(string_of_func_decl)) obj2) ^ ")")) obj)
```

scanner.mll

```
{
  open Parser

  let strip_string s =
      String.sub s 1 ((String.length s) - 2)


}

let whitespace = [' ' '\t' '\r' '\n']
let digit = ['0'-'9']
let integer = digit+
let double = digit+ '.' digit+
let str_lit = ['\"'] [^'\"']* ['\"']
let id = ['a'-'z' 'A'-'Z' '_'] ['a'-'z' 'A'-'Z' '0'-'9' '_']*

rule token = parse
  | whitespace  { token lexbuf } (* Whitespace *)
  | "//"        { single_line_comment lexbuf }
  | "/*"        { multi_line_comment lexbuf } (* Comments *)
  | '('         { LPAREN }
  | ')'         { RPAREN }
  | '{'         { LBRACE }
  | '}'         { RBRACE }
  | '['         { LBRACK }
  | ']'         { RBRACK }
  | ';'         { SEMI }
  | ','         { COMMA }
  | '+'         { PLUS }
  | '-'         { MINUS }
  | '*'         { TIMES }
  | '/'         { DIVIDE }
  | '='         { ASSIGN }
  | "=="        { EQ }
  | "!="        { NEQ }
  | '<'         { LT }
  | "<="        { LEQ }
  | ">"         { GT }
  | ">="        { GEQ }
  | "&&"        { AND }
  | "||"        { OR }
  | "!"         { NOT }
```

```
  | "UN"         { UNION }
  | "IN"         { INTERSECT }
  | "DI"         { DIFFERENCE }
  | "if"         { IF }
  | "else"       { ELSE }
  | "for"        { FOR }
  | "while"      { WHILE }
  | "return"     { RETURN }
  | "int"        { INT }
  | "double"     { DBL }
  | "string"     { STRING }
  | "bool"       { BOOL }
  | "Shape"      { SHAPE }
  | "SPHERE"     { SPHERE_PRIM }
  | "CUBE"       { CUBE_PRIM }
  | "CYLINDER"   { CYLINDER_PRIM }
  | "TETRA"      { TETRA_PRIM }
  | "CONE"       { CONE_PRIM }
  | "void"       { VOID }
  | "true"       { TRUE }
  | "false"      { FALSE }
  | integer as lxm  { INT_LIT(int_of_string lxm) }
  | double as lxm   { DBL_LIT(float_of_string lxm) }
  | str_lit as lxm  { STR_LIT(strip_string lxm) }
  | id as lxm    { ID(lxm) }
  | _ as char    { raise (Failure("illegal character " ^ Char.escaped
char)) }
  | eof          { EOF }

and single_line_comment = parse
  | '\n' { token lexbuf }
  | _ { single_line_comment lexbuf }

and multi_line_comment = parse
  | "*/" { token lexbuf }
  | _ { multi_line_comment lexbuf }
```

semant.ml

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

open Ast

module StringMap = Map.Make(String)

let global_idents:(string, typ) Hashtbl.t = Hashtbl.create 50

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

   Check each global variable, then check each function *)

let check (globals, functions) =
  (* Raise an exception if the given list has a duplicate *)
  let report_duplicate exceptf list =
    let rec helper = function
      n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf n1))
     | _ :: t -> helper t
     | [] -> ()
    in helper (List.sort compare list)
  in

  (* Raise an exception if a given binding is to a void type *)
  let check_not_void exceptf = function
    | (Void, n) -> raise (Failure (exceptf n))
    | _ -> ()
  in

  (* Raise an exception if the given rvalue type cannot be assigned to
     the given lvalue type *)
  let check_assign lvaluet rvaluet err =
     if lvaluet = rvaluet then lvaluet else
     if rvaluet = Void then lvaluet else raise err
  in

  (**** Checking Global Variables ****)
   List.iter (check_not_void (fun n -> "illegal void global " ^ n))
globals;
   report_duplicate (fun n -> "duplicate global " ^ n) (List.map snd
globals);

  (**** Checking Functions ****)

  if List.mem "print" (List.map (fun fd -> fd.fname) functions)
```

```
  then raise (Failure ("function print may not be defined")) else ();

  report_duplicate (fun n -> "duplicate function " ^ n)
    (List.map (fun fd -> fd.fname) functions);

  (* Function declaration for a named function *)
  let built_in_decls =
      List.fold_left (fun map (key, value) ->
          StringMap.add key value map
      ) StringMap.empty [
           ("Copy", { typ = Void; fname = "Copy"; formals = []; body =
[] });
            ("Render", { typ = Void; fname = "Render"; formals = [];
body = [] });
           ("Save", { typ = Void; fname = "Save"; formals = []; body =
[] });
           ("Copy", { typ = Void; fname = "Copy"; formals = []; body =
[] });
          ("Difference", { typ = Void; fname = "Difference"; formals =
[]; body = [] });
           ("Intersect", { typ = Void; fname = "Intersect"; formals =
[]; body = [] });
           ("Reflect", { typ = Void; fname = "Reflect"; formals = [];
body = [] });
           ("Union", { typ = Void; fname = "Union"; formals = []; body
= [] });
           ("Scale", { typ = Void; fname = "Scale"; formals = []; body
= [] });
            ("Rotate", { typ = Void; fname = "Rotate"; formals = [];
body = [] });
           ("Translate", { typ = Void; fname = "Translate"; formals =
[]; body = [] });
            ("print", { typ = Void; fname = "print"; formals = [(Int,
"x")]; body = [] });
          ("printb", { typ = Void; fname = "printb"; formals = [(Bool,
"x")]; body = [] });
      ]
  in

   let function_decls = List.fold_left (fun m fd -> StringMap.add
fd.fname fd m)
    built_in_decls functions
  in

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

```
  let _ = function_decl "scene" in (* Ensure "scene" is defined *)

  let _ =
      let add_symbol (t, n) =
          Hashtbl.add global_idents n t
      in
      List.iter add_symbol globals
  in

  let check_function func =
    List.iter (check_not_void (fun n -> "illegal void formal " ^ n ^
      " in " ^ func.fname)) func.formals;
     report_duplicate (fun n -> "duplicate formal " ^ n ^ " in " ^
func.fname)
      (List.map snd func.formals);

    (* Type of each variable (global, formal, or local *)
    let local_idents:(string, typ) Hashtbl.t = Hashtbl.create 50 in
    let _ =
        let add_symbol (t, n) =
            Hashtbl.add local_idents n t
        in
        List.iter add_symbol (func.formals)
    in

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

    (* Return the type of an expression or throw an exception *)
    let rec expr = function
        | IntLit _ -> Int
      | DblLit _ -> Dbl
      | StrLit _ -> String
      | BoolLit _ -> Bool
      | Id s -> type_of_identifier s
      | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2 in
          (match op with
            | Add | Sub | Mult | Div when t1 = Int && t2 = Int -> Int
            | Add | Sub | Mult | Div when t1 = Dbl && t2 = Dbl -> Dbl
            | Equal | Neq when t1 = t2 -> Bool
             | Less | Leq | Greater | Geq when t1 = Int && t2 = Int ->
Bool
             | Less | Leq | Greater | Geq when t1 = Dbl && t2 = Dbl ->
Bool
```

```
            | And | Or when t1 = Bool && t2 = Bool -> Bool
            | _ -> raise (Failure ("illegal binary operator " ^
                          string_of_typ t1 ^ " " ^ string_of_op op ^ "
" ^
                            string_of_typ t2 ^ " in " ^ string_of_expr
e)))
      | Unop(op, e) as ex -> let t = expr e in
          (match op with
              Neg when t = Int -> Int
            | Not when t = Bool -> Bool
                | _ -> raise (Failure ("illegal unary operator " ^
string_of_uop op ^
                            string_of_typ t ^ " in " ^ string_of_expr
ex)))
      | Noexpr -> Void
      | Assign(var, e) as ex -> let lt = type_of_identifier var
                                and rt = expr e in
                check_assign lt rt (Failure ("illegal assignment " ^
string_of_typ lt ^
                                    " = " ^ string_of_typ rt ^ " in " ^
                                    string_of_expr ex))
      | Call(fname, _) -> let fd = function_decl fname in fd.typ
      | SpherePrim -> Shape
      | CubePrim -> Shape
      | CylinderPrim -> Shape
      | TetraPrim -> Shape
      | ConePrim -> Shape
    in

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

    (* Verify a statement or throw an exception *)
    let rec stmt = function
        | Block sl -> let rec check_block = function
            | [Return _ as s] -> stmt s
               | Return _ :: _ -> raise (Failure "nothing may follow a
return")
            | Block sl :: ss -> check_block (sl @ ss)
            | s :: ss -> stmt s; check_block ss
            | [] -> ()
          in check_block sl
      | Expr e -> ignore (expr e)
      | Return e -> let t = expr e in if t = func.typ then () else
```

```
              raise (Failure ("return gives " ^ string_of_typ t ^ "
expected " ^
                                    string_of_typ func.typ ^ " in " ^
string_of_expr e))
      | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
      | For(e1, e2, e3, st) -> ignore (expr e1); check_bool_expr e2;
                               ignore (expr e3); stmt st
      | While(p, s) -> check_bool_expr p; stmt s
      | Local(t, s, e) -> ignore (Hashtbl.add local_idents s t);
          if expr e = Void then () else
          if expr e = t then () else
           raise (Failure ("expression has type " ^ string_of_typ (expr
e) ^
          " expected " ^ string_of_typ t))
    in
    stmt (Block func.body)
  in
  List.iter check_function functions
```

shapeshifter.ml

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

type action = Ast | PrettyPrint | LLVM_IR | Compile | Help


let get_info = (
    "Usage: ./shapeshifter [optional flag] < <source file>\n")

let _ =
  (*try *)
      let action =
       if Array.length Sys.argv > 1 then
                  List.assoc Sys.argv.(1) [ ("-a", Ast);     (* Print
the AST only *)
                                            ("-p",    PrettyPrint);    (*
Pretty-print the AST *)
                                           ("-l", LLVM_IR);   (* Generate
LLVM, don't check *)
                                           ("-c", Compile); (* Generate,
check LLVM IR *)
                                               ("-h", Help) ] (* Usage &
option info *)
       else Compile in

    let lexbuf = Lexing.from_channel stdin in
      let ast = Parser.program Scanner.token lexbuf in
      Semant.check ast;

    match action with
            | Help -> print_string get_info
           | Ast -> print_string (Ast.string_of_program ast)
                    |        PrettyPrint        ->        print_string
(Prettyprint.string_of_program ast)
              |   LLVM_IR   ->   print_string   (Llvm.string_of_llmodule
(Codegen.translate ast))
            | Compile -> let m = Codegen.translate ast in
                         Llvm_analysis.assert_valid_module m;
                         print_string (Llvm.string_of_llmodule m)
```

testall.sh

```
# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"

# Path to the ShapeShifter compiler.
# Try "_build/shapeshifter.native" if ocamlbuild was unable to create
a symbolic link.
SHAPE="./shapeshifter"
#SHAPE="_build/shapeshifter.native"

# Set time limit for all operations
ulimit -t 30

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

keep=0

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

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

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

```
    echo "FAILED $1 differs from $2" 1>&2
      }
}

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

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

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

      echo -n "$basename..."

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

      generatedfiles=""

      generatedfiles="$generatedfiles  ${basename}.ll  ${basename}.out"
&&
      Run "$SHAPE" "<" $1 ">" "${basename}.ll" &&
      Run "$LLI" "${basename}.ll" ">" "${basename}.out" &&
      Compare ${basename}.out ${reffile}.out ${basename}.diff

      # Report the status and clean up the generated files

      if [ $error -eq 0 ] ; then
```

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

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

      echo -n "$basename..."

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

      generatedfiles=""

      generatedfiles="$generatedfiles ${basename}.err ${basename}.diff"
&&
      RunFail "$SHAPE" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
      Compare ${basename}.err ${reffile}.err ${basename}.diff

      # Report the status and clean up the generated files

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

while getopts kdpsh c; do
      case $c in
    k) # Keep intermediate files
      keep=1
```

```
      ;;
    h) # Help
      Usage
      ;;
      esac
done

shift `expr $OPTIND - 1`

LLIFail() {
  echo "Could not find the LLVM interpreter \"$LLI\"."
  echo "Check your LLVM installation and/or modify the LLI variable in
testall.sh"
  exit 1
}

which "$LLI" >> $globallog || LLIFail


if [ $# -ge 1 ]
then
      files=$@
else
      files="test-suite/shapes/test_*.shift
test-suite/arithmetic/test_*.shift
test-suite/control_flow/test_*.shift
test-suite/shape_bool_transf/test_*.shift
test-suite/data_types/test_*.shift         test-suite/misc/fail_*.shift
test-suite/comp_ops/test_*.shift      test-suite/functions/test_*.shift
test-suite/shape_transf/test_*.shift"
fi

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

```
exit $globalerror
```

graphics/display/main.cpp

```
#define GL_GLEXT_PROTOTYPES

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/glext.h>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>

#include "files.h"
#include "cork.h"

#define PI 3.14159265

void check_GL_Error(const char *file, int line)
{
    GLenum err = glGetError();
    while (err != GL_NO_ERROR) {
         fprintf(stderr, "OpenGL Error %x at %s, line %d\n", err, file,
line);
        err = glGetError();
    }
}

// OpenGL Error checking
#define debug
#if defined(debug)
#define CHECK_GL(func) func; \
    check_GL_Error(__FILE__, __LINE__);
#else
#define CHECK_GL(func) func;
#endif

static void die(const char *message)
{
    fprintf(stderr, "%s\n", message);
    exit(EXIT_FAILURE);
}

int windowWidth = 600;
```

```
int windowHeight = 600;

double winL = -10; // value at left of window
double winR = 10; // right
double winB = -10;  // bottom
double winT = 10; // top
double winN = -10; // near
double winF = 10;

double camTheta = 0.0;
double camPhi = 90.0;
double camR = 4.0;
double camX, camY, camZ;
double upX = 0.0;
double upY = 1.0;
double upZ = 0.0;
CorkTriMesh shape;
GLuint vbo; // vertex buffer object, stores triangle vertex info
GLuint ibo; // index buffer object, stores indices of triangles

// Set up lighting and material properties
GLfloat lightPos0[] = {10.000001, 10.00001, 0.000001, 10.0};
GLfloat lightAmb0[] = {0.1, 0.1, 0.1, 1.0};
GLfloat lightDiff0[] = {1.0, 1.0, 1.0, 1.0};
GLfloat lightSpec0[] = {1.0, 1.0, 1.0, 1.0};


float xTrans, yTrans, zTrans;

#if defined(debug)
float *norms;
float *fnorms;
bool drawNorms = 0;
#endif

void reshape(int w, int h)
{
    windowWidth = std::max(w, 1);
    windowHeight = std::max(h, 1);
    CHECK_GL(glViewport(0, 0, windowWidth, windowHeight));
    CHECK_GL(glMatrixMode(GL_PROJECTION));
    CHECK_GL(glLoadIdentity());
        CHECK_GL(gluPerspective(80,  (GLfloat)windowWidth/windowHeight,
.001, 1000));
    CHECK_GL(glMatrixMode(GL_MODELVIEW));
    glutPostRedisplay();
}
```

```
void keyboard(unsigned char key, int x, int y)
{
    double dcam = .05;
    double theta = camTheta*PI/180.0;
    double phi = camPhi*PI/180.0;

    switch (key) {
    case  27:
    case 'q':
    case 'Q':
        exit(0);
        break;
    case '+':
        camR = std::max(.01, camR-dcam);
        camX = camR * cos(theta) * cos(phi);
        camY = camR * sin(theta);
        camZ = camR * cos(theta) * sin(phi);
        glutPostRedisplay();
        break;
    case '-':
        camR += dcam;
        camX = camR * cos(theta) * cos(phi);
        camY = camR * sin(theta);
        camZ = camR * cos(theta) * sin(phi);
        glutPostRedisplay();
        break;
#if defined(debug)
    case 'n':
        drawNorms = !drawNorms;
        glutPostRedisplay();
        break;
#endif
    default:
        break;
    }

      //fprintf(stdout, "cam: (%f, %f, %f): %f\n", camX, camY, camZ,
std::sqrt(camX*camX + camY*camY + camZ*camZ));

    // add other key functions - zoom? translate? animate?
    // toggle axis display
}

void special(int key, int x, int y)
{
    double dcam = 1;
    switch (key) {
    // Want to not flip the damn thing when camTheta does the thing;
```

```
    // Change up?
    case GLUT_KEY_UP:
        camTheta -= dcam;
        break;
    case GLUT_KEY_DOWN:
        camTheta += dcam;
        break;
    case GLUT_KEY_LEFT:
        camPhi -= dcam;
        break;
    case GLUT_KEY_RIGHT:
        camPhi += dcam;
        break;
    }

    double theta = camTheta*PI/180.0;
    double phi = camPhi*PI/180.0;
    camX = camR * cos(theta) * cos(phi);
    camY = camR * sin(theta);
    camZ = camR * cos(theta) * sin(phi);
    if(cos(theta) < 0)
        upY = -1.0;
    else
        upY = 1.0;

    glutPostRedisplay();

}



// Upon mouse interaction
void mouse(int button, int state, int x, int y)
{
    // zoom, translate camera, ...


}

// Run when not displaying...
void idle()
{
    //glutPostRedisplay();

}

void calcNormals(float *pos, unsigned int *ind, float *norms, int
ntri, int nv)
```

```
{
    // triangle vertex positions
    float u0, u1, u2, v0, v1, v2, w0, w1, w2;
     float a0, a1, a2, ad, b0, b1, b2, bd, c0, c1, c2, cd; // edge
vectors uv, uw
    float n0, n1, n2;
    float d = 0;
    float au, av, aw; // angles
    // For each triangle, calc normal, set for each vertex
    for(int i = 0; i < ntri; ++i) {
        u0 = pos[ind[i*3]*3];
        u1 = pos[ind[i*3]*3+1];
        u2 = pos[ind[i*3]*3+2];
        v0 = pos[ind[i*3+1]*3];
        v1 = pos[ind[i*3+1]*3+1];
        v2 = pos[ind[i*3+1]*3+2];
        w0 = pos[ind[i*3+2]*3];
        w1 = pos[ind[i*3+2]*3+1];
        w2 = pos[ind[i*3+2]*3+2];

        // uv
        a0 = v0 - u0;
        a1 = v1 - u1;
        a2 = v2 - u2;
        ad = std::sqrt(a0*a0 + a1*a1 + a2*a2);
        // uw
        b0 = w0 - u0;
        b1 = w1 - u1;
        b2 = w2 - u2;
        bd = std::sqrt(b0*b0 + b1*b1 + b2*b2);
        // vw
        c0 = w0 - v0;
        c1 = w1 - v1;
        c2 = w2 - v2;
        cd = std::sqrt(c0*c0 + c1*c1 + c2*c2);

        // calc angles
        au = std::acos((a0*b0 + a1*b1 + a2*b2)/(ad*bd));
        if (au > PI/2.0)
            au = PI - au;
        av = std::acos((a0*c0 + a1*c1 + a2*c2)/(ad*cd));
        if (av > PI/2.0)
            av = PI - av;
        aw = std::acos((b0*c0 + b1*c1 + b2*c2)/(bd*cd));
        if (aw > PI/2.0)
            aw = PI - aw;

        // calc normals
```

```
        n0 = a1 * b2 - a2 * b1;
        n1 = a2 * b0 - a0 * b2;
        n2 = a0 * b1 - a1 * b0;

        // Add normal for all 3 vertices
        d = std::max(.0001f, std::sqrt(n0*n0 + n1*n1 + n2*n2));
        n0= n0/d;
        n1 = n1/d;
        n2 = n2/d;

#if defined(debug)
        fnorms[i*3] = n0;
        fnorms[i*3+1] = n1;
        fnorms[i*3+2] = n2;
#endif

        // weigh by angle

        norms[ind[i*3]*3] += n0*au;
        norms[ind[i*3]*3+1] += n1*au;
        norms[ind[i*3]*3+2] += n2*au;
        norms[ind[i*3+1]*3] += n0*av;
        norms[ind[i*3+1]*3+1] += n1*av;
        norms[ind[i*3+1]*3+2] += n2*av;
        norms[ind[i*3+2]*3] += n0*aw;
        norms[ind[i*3+2]*3+1] += n1*aw;
        norms[ind[i*3+2]*3+2] += n2*aw;
    }

    for (int i = 0; i < nv; ++i) {
        n0 = norms[i*3];
        n1 = norms[i*3+1];
        n2 = norms[i*3+2];
        d = std::max(.0001f, std::sqrt(n0*n0 + n1*n1 + n2*n2));
        norms[i*3] = n0/d;
        norms[i*3+1] = n1/d;
        norms[i*3+2] = n2/d;
    }
}



void calcCols(float *cols, int n)
{
     float colors[][3] = {{.43, .72, .94}, {0.21, .60, 0.82}, {0.51,
0.91, .96}};
```

```
     //float colors[][3] = {{.43, .72, .94}, {0.43, .72, 0.94}, {0.43,
0.72, .94}};

    for (int i = 0; i < n; ++i) {
        cols[i*3] = colors[i%3][0];
        cols[i*3+1] = colors[i%3][1];
        cols[i*3+2] = colors[i%3][2];
    }
}


void uploadMeshData()
{
    CHECK_GL(glGenBuffers(1, &vbo));
    CHECK_GL(glGenBuffers(1, &ibo));
    CHECK_GL(glBindBuffer(GL_ARRAY_BUFFER, vbo));
    CHECK_GL(glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, ibo));

                                //CHECK_GL(glBufferData(GL_ARRAY_BUFFER,
shape.n_vertices*3*sizeof(float), shape.vertices, GL_STATIC_DRAW));

    // Allocate enough buffer space for positions + normals
                                  CHECK_GL(glBufferData(GL_ARRAY_BUFFER,
shape.n_vertices*9*sizeof(float), 0, GL_STATIC_DRAW));
    // Upload position data
                       CHECK_GL(glBufferSubData(GL_ARRAY_BUFFER,     0,
shape.n_vertices*3*sizeof(float), shape.vertices));
    // Calc and upload normal data

#if defined(debug)
    norms = (float *)malloc(shape.n_vertices*3*sizeof(float));
    fnorms = (float *)malloc(shape.n_triangles*3*sizeof(float));
    if (!norms || !fnorms) {
        die("malloc failed");
    }
#else
    float *norms = (float *)malloc(shape.n_vertices*3*sizeof(float));
    if (!norms) {
        die("malloc failed");
    }
#endif

    memset(norms, 0, shape.n_vertices*3*sizeof(float));
               calcNormals(shape.vertices,    shape.triangles,    norms,
shape.n_triangles, shape.n_vertices);
                                CHECK_GL(glBufferSubData(GL_ARRAY_BUFFER,
shape.n_vertices*3*sizeof(float),     shape.n_vertices*3*sizeof(float),
norms));
```

```
#if !defined(debug)
    free(norms);
#endif

    float *cols = (float *)malloc(shape.n_vertices*3*sizeof(float));
    calcCols(cols, shape.n_vertices);
                                CHECK_GL(glBufferSubData(GL_ARRAY_BUFFER,
shape.n_vertices*6*sizeof(float),     shape.n_vertices*3*sizeof(float),
cols));
    free(cols);

                            CHECK_GL(glBufferData(GL_ELEMENT_ARRAY_BUFFER,
shape.n_triangles*3*sizeof(uint), shape.triangles, GL_STATIC_DRAW));

    CHECK_GL(glBindBuffer(GL_ARRAY_BUFFER, 0));
    CHECK_GL(glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0));
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    CHECK_GL(glLightfv(GL_LIGHT0, GL_POSITION, lightPos0));
    CHECK_GL(glEnable(GL_DEPTH_TEST));
    CHECK_GL(glEnable(GL_NORMALIZE));

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    CHECK_GL(gluLookAt(camX, camY, camZ, 0, 0, 0, upX, upY, upZ));

    // Draw axes
    glLineWidth(1.0);
    glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0); // x
    glVertex3f(-100.0, 0.0, 0.0);
    glVertex3f(100.0, 0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0); // y
    glVertex3f(0.0, -100.0, 0.0);
    glVertex3f(0.0, 100.0, 0.0);
    glColor3f(0.0, 0.0, 1.0); // z
    glVertex3f(0.0, 0.0, -100.0);
    glVertex3f(0.0, 0.0, 100.0);
    glEnd();

    glColor3f(.7, .7, .7);
    CHECK_GL(glEnable(GL_LIGHTING));
    CHECK_GL(glEnable(GL_LIGHT0));
    glEnable(GL_COLOR_MATERIAL);
```

```
    //glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
    //CHECK_GL(glFrontFace(GL_CW));

    CHECK_GL(glBindBuffer(GL_ARRAY_BUFFER, vbo));
    CHECK_GL(glEnableClientState(GL_VERTEX_ARRAY));
    CHECK_GL(glVertexPointer(3, GL_FLOAT, 0, 0));
    CHECK_GL(glEnableClientState(GL_NORMAL_ARRAY));
    CHECK_GL(glEnableClientState(GL_COLOR_ARRAY));
                    CHECK_GL(glNormalPointer(GL_FLOAT,     0,     (char
*)((intptr_t)(shape.n_vertices*3*sizeof(float)))));
                  CHECK_GL(glColorPointer(3,    GL_FLOAT,    0,    (char
*)((intptr_t)(shape.n_vertices*6*sizeof(float)))));

    CHECK_GL(glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, ibo));
           CHECK_GL(glDrawElements(GL_TRIANGLES,   shape.n_triangles*3,
GL_UNSIGNED_INT, (void*)0));

#if defined(debug)
    // draw normals yo
    if(drawNorms) {
        glLineWidth(3.0);
        float px, py, pz;
        glBegin(GL_LINES);
        glColor3f(1.0, 1.0, 1.0);
        for (int i = 0; i < (int)shape.n_vertices; ++i) {
            px = shape.vertices[i*3];
            py = shape.vertices[i*3+1];
            pz = shape.vertices[i*3+2];
            glVertex3f(px, py, pz);
                            glVertex3f(px+norms[i*3],  py+norms[i*3+1],
pz+norms[i*3+2]);
        }
        glEnd();
/*
    // draw face normals
    glColor3f(0.0, 1.0, 1.0);
    glBegin(GL_LINES);
    for (int i = 0; i < (int)shape.n_triangles; ++i) {
        px = shape.vertices[shape.triangles[i*3]*3];
        py = shape.vertices[shape.triangles[i*3]*3+1];
        pz = shape.vertices[shape.triangles[i*3]*3+2];
        glVertex3f(px, py, pz);
                        glVertex3f(px+fnorms[i*3],   py+fnorms[i*3+1],
pz+fnorms[i*3+2]);
    }
    glEnd();
*/
    }
```

```
#endif

    CHECK_GL(glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0));
    CHECK_GL(glDisableClientState(GL_VERTEX_ARRAY));
    CHECK_GL(glDisableClientState(GL_NORMAL_ARRAY));
    CHECK_GL(glDisableClientState(GL_COLOR_ARRAY));
    CHECK_GL(glBindBuffer(GL_ARRAY_BUFFER, 0));

    CHECK_GL(glDisable(GL_LIGHTING));
    CHECK_GL(glDisable(GL_LIGHT0));
    CHECK_GL(glDisable(GL_DEPTH_TEST));

    glutSwapBuffers();

}

void initOpenGLandGLUT(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DEPTH|GLUT_DOUBLE|GLUT_RGBA);
    glutInitWindowSize(windowWidth, windowHeight);

    // Position the window in the center of the screen
    int screenW = glutGet(GLUT_SCREEN_WIDTH);
    int screenH = glutGet(GLUT_SCREEN_HEIGHT);
    if (screenW > 0 && screenH > 0) {
        glutInitWindowPosition(windowWidth/2, windowHeight/2);
    }

    glutCreateWindow("Shapeshifter");
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    //glutMotionFunc(motion);
    glutMouseFunc(mouse);
    glutIdleFunc(idle);
    glutSpecialFunc(special);

  // Set background color to gray
    CHECK_GL(glClearColor(.1, .1, .1, 1.0));

    CHECK_GL(glLightfv(GL_LIGHT0, GL_POSITION, lightPos0));
    CHECK_GL(glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmb0));
    CHECK_GL(glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiff0));
    CHECK_GL(glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpec0));
    CHECK_GL(glLightModelf(GL_LIGHT_MODEL_LOCAL_VIEWER, 1));
    CHECK_GL(glShadeModel(GL_SMOOTH));
```

```
    // Set camera coordinates
    double theta = camTheta*PI/180.0;
    double phi = camPhi*PI/180.0;
    camX = camR * cos(theta) * cos(phi);
    camY = camR * sin(theta);
    camZ = camR * cos(theta) * sin(phi);

    reshape(windowWidth, windowHeight);
}


void loadMesh(std::string fileName, CorkTriMesh *out)
{
    Files::FileMesh fileMesh;
    if(Files::readTriMesh(fileName, &fileMesh) > 0) {
                fprintf(stderr,  "Unable  to  load  file  from  %s\n",
fileName.c_str());
        exit(1);
    }

    // Copy data from FileMesh
    out->n_vertices = fileMesh.vertices.size();
    out->n_triangles = fileMesh.triangles.size();

    out->triangles = new uint[(out->n_triangles) * 3];
    out->vertices = new float[(out->n_vertices) * 3];
    for (uint i = 0; i < out->n_triangles; ++i) {
        (out->triangles)[3*i+0] = fileMesh.triangles[i].a;
        (out->triangles)[3*i+1] = fileMesh.triangles[i].b;
        (out->triangles)[3*i+2] = fileMesh.triangles[i].c;
    }

    for (uint i = 0; i < out->n_vertices; ++i) {
        (out->vertices)[3*i+0] = fileMesh.vertices[i].pos.x;
        (out->vertices)[3*i+1] = fileMesh.vertices[i].pos.y;
        (out->vertices)[3*i+2] = fileMesh.vertices[i].pos.z;
    }
}


int main(int argc, char **argv)
{
    if (argc != 2) {
        fprintf(stdout, "usage: sshapedisplay [shapefile.OFF]\n");
        return 0;
    }

    // read the thing
```

```
    initOpenGLandGLUT(argc, argv);

    loadMesh(argv[1], &shape);

    uploadMeshData();
    glutMainLoop();
}
```

graphics/cork/src/cork.cpp (with original code in gray)

```
//
+-----------------------------------------------------------------------
----
// | cork.cpp
// |
// | Author: Gilbert Bernstein
//
+-----------------------------------------------------------------------
----
// | COPYRIGHT:
// |    Copyright Gilbert Bernstein 2013
// |    See the included COPYRIGHT file for further details.
// |
// |    This file is part of the Cork library.
// |
// |    Cork is free software: you can redistribute it and/or modify
// |    it under the terms of the GNU Lesser General Public License as
// |    published by the Free Software Foundation, either version 3 of
// |    the License, or (at your option) any later version.
// |
// |    Cork is distributed in the hope that it will be useful,
// |    but WITHOUT ANY WARRANTY; without even the implied warranty of
// |    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// |    GNU Lesser General Public License for more details.
// |
// |    You should have received a copy
// |    of the GNU Lesser General Public License
// |    along with Cork.  If not, see <http://www.gnu.org/licenses/>.
//
+-----------------------------------------------------------------------
----
#include "cork.h"

#include "mesh.h"
#include <cmath>

#define PI 3.14159265

void freeCorkTriMesh(CorkTriMesh *mesh)
{
    delete[] mesh->triangles;
    delete[] mesh->vertices;
    mesh->n_triangles = 0;
    mesh->n_vertices = 0;
```

```
}


struct CorkTriangle;

struct CorkVertex :
    public MinimalVertexData,
    public RemeshVertexData,
    public IsctVertexData,
    public BoolVertexData
{
    void merge(const CorkVertex &v0, const CorkVertex &v1) {
        double                              a0 = 0.5;
        if(v0.manifold && !v1.manifold)     a0 = 0.0;
        if(!v0.manifold && v1.manifold)     a0 = 1.0;
        double a1 = 1.0 - a0;

        pos         = a0 * v0.pos       + a1 * v1.pos;
    }
    void interpolate(const CorkVertex &v0, const CorkVertex &v1) {
        double a0 = 0.5;
        double a1 = 0.5;
        pos         = a0 * v0.pos       + a1 * v1.pos;
    }


    void isct(IsctVertEdgeTriInput<CorkVertex,CorkTriangle> input)
    {
        Vec2d       a_e     = Vec2d(1,1)/2.0;
        Vec3d       a_t     = Vec3d(1,1,1)/3.0;
        a_e /= 2.0;
        a_t /= 2.0;
    }
    void isct(IsctVertTriTriTriInput<CorkVertex,CorkTriangle> input)
    {
        Vec3d       a[3];
        for(uint k=0; k<3; k++) {
            a[k]    = Vec3d(1,1,1)/3.0;
            a[k] /= 3.0;
        }
        for(uint i=0; i<3; i++) {
          for(uint j=0; j<3; j++) {
        }}
    }
    void isctInterpolate(const CorkVertex &v0, const CorkVertex &v1) {
        double a0 = len(v1.pos - pos);
        double a1 = len(v0.pos - pos);
        if(a0 + a1 == 0.0) a0 = a1 = 0.5; // safety
```

```
        double sum = a0+a1;
        a0 /= sum;
        a1 /= sum;
    }
};

struct CorkTriangle :
    public MinimalTriangleData,
    public RemeshTriangleData,
    public IsctTriangleData,
    public BoolTriangleData
{
    void merge(const CorkTriangle &, const CorkTriangle &) {}
    static void split(CorkTriangle &, CorkTriangle &,
                      const CorkTriangle &) {}
    void move(const CorkTriangle &) {}
    void subdivide(SubdivideTriInput<CorkVertex,CorkTriangle> input)
    {
        bool_alg_data = input.pt->bool_alg_data;
    }
};

// SHAPESHIFTER

// Multiply a 3x3 matrix and a 3-dim vector
void matMult3(float *mat, float *vin, float *vout)
{
    vout[0] = mat[0]*vin[0] + mat[1]*vin[1] + mat[2]*vin[2];
    vout[1] = mat[3]*vin[0] + mat[4]*vin[1] + mat[5]*vin[2];
    vout[2] = mat[6]*vin[0] + mat[7]*vin[1] + mat[8]*vin[2];
}

void reflectCork(CorkTriMesh *mesh, float a, float b, float c)
{
    float k = 1.0f/(a*a + b*b + c*c);
    if (k == 0)
        return;

    float refmat[9];
    refmat[0] = k * (-a*a + b*b + c*c);
    refmat[1] = k * (-2.0f*a*b);
    refmat[2] = k * (-2.0f*a*c);
    refmat[3] = refmat[1];
    refmat[4] = k * (a*a - b*b + c*c);
    refmat[5] = k * (-2.0f*b*c);
    refmat[6] = refmat[2];
    refmat[7] = refmat[5];
    refmat[8] = k * (a*a + b*b - c*c);
```

```
    float ppos[3];
    for (uint i = 0; i < mesh->n_vertices; ++i) {
        ppos[0] = mesh->vertices[i*3];
        ppos[1] = mesh->vertices[i*3+1];
        ppos[2] = mesh->vertices[i*3+2];
        matMult3(refmat, ppos, &(mesh->vertices[i*3]));
    }
}

void rotateCork(CorkTriMesh *mesh, float x, float y, float z)
{
    float cosX = (float)cos(x * PI / 180.0);
    float sinX = (float)sin(x * PI / 180.0);
    float cosY = (float)cos(y * PI / 180.0);
    float sinY = (float)sin(y * PI / 180.0);
    float cosZ = (float)cos(z * PI / 180.0);
    float sinZ = (float)sin(z * PI / 180.0);

    float rotx[] = {1, 0, 0, 0, cosX, -sinX, 0, sinX, cosX};
    float roty[] = {cosY, 0, sinY, 0, 1, 0, -sinY, 0, cosY};
    float rotz[] = {cosZ, -sinZ, 0, sinZ, cosZ, 0, 0, 0, 1};
    float ppos[3];
    for (uint i = 0; i < mesh->n_vertices; ++i) {
        ppos[0] = mesh->vertices[i*3];
        ppos[1] = mesh->vertices[i*3+1];
        ppos[2] = mesh->vertices[i*3+2];
        matMult3(rotx, ppos, &(mesh->vertices[i*3]));
        ppos[0] = mesh->vertices[i*3];
        ppos[1] = mesh->vertices[i*3+1];
        ppos[2] = mesh->vertices[i*3+2];
        matMult3(roty, ppos, &(mesh->vertices[i*3]));
        ppos[0] = mesh->vertices[i*3];
        ppos[1] = mesh->vertices[i*3+1];
        ppos[2] = mesh->vertices[i*3+2];
        matMult3(rotz, ppos, &(mesh->vertices[i*3]));
    }
}


void scaleCork(CorkTriMesh *mesh, float x, float y, float z)
{
    float scale[] = {x, 0, 0, 0, y, 0, 0, 0, z};
    float ppos[3];
    for (uint i = 0; i < mesh->n_vertices; ++i) {
        ppos[0] = mesh->vertices[i*3];
        ppos[1] = mesh->vertices[i*3+1];
        ppos[2] = mesh->vertices[i*3+2];
```

```
        matMult3(scale, ppos, &(mesh->vertices[i*3]));
    }
}
void translateCork(CorkTriMesh *mesh, float x, float y, float z)
{
    for (uint i = 0; i < mesh->n_vertices; ++i) {
        mesh->vertices[i*3] += x;
        mesh->vertices[i*3+1] += y;
        mesh->vertices[i*3+2] += z;
    }
}


// END SHAPESHIFTER


//using RawCorkMesh = RawMesh<CorkVertex, CorkTriangle>;
//using CorkMesh = Mesh<CorkVertex, CorkTriangle>;
typedef RawMesh<CorkVertex, CorkTriangle>   RawCorkMesh;
typedef Mesh<CorkVertex, CorkTriangle>      CorkMesh;

void corkTriMesh2CorkMesh(
    CorkTriMesh in,
    CorkMesh *mesh_out
) {
    RawCorkMesh raw;
    raw.vertices.resize(in.n_vertices);
    raw.triangles.resize(in.n_triangles);
    if(in.n_vertices == 0 || in.n_triangles == 0) {
        CORK_ERROR("empty mesh input to Cork routine.");
        *mesh_out = CorkMesh(raw);
        return;
    }

    uint max_ref_idx = 0;
    for(uint i=0; i<in.n_triangles; i++) {
        raw.triangles[i].a = in.triangles[3*i+0];
        raw.triangles[i].b = in.triangles[3*i+1];
        raw.triangles[i].c = in.triangles[3*i+2];
        max_ref_idx = std::max(
                        std::max(max_ref_idx,
                                 in.triangles[3*i+0]),
                        std::max(in.triangles[3*i+1],
                                 in.triangles[3*i+2])
                      );
    }
    if(max_ref_idx > in.n_vertices) {
```

```
            CORK_ERROR("mesh input to Cork routine has an out of range
reference "
                "to a vertex.");
            raw.vertices.clear();
            raw.triangles.clear();
            *mesh_out = CorkMesh(raw);
            return;
        }

        for(uint i=0; i<in.n_vertices; i++) {
            raw.vertices[i].pos.x = in.vertices[3*i+0];
            raw.vertices[i].pos.y = in.vertices[3*i+1];
            raw.vertices[i].pos.z = in.vertices[3*i+2];
        }

        *mesh_out = CorkMesh(raw);
}
void corkMesh2CorkTriMesh(
        CorkMesh *mesh_in,
        CorkTriMesh *out
) {
        RawCorkMesh raw = mesh_in->raw();

        out->n_triangles = raw.triangles.size();
        out->n_vertices  = raw.vertices.size();

        out->triangles = new uint[(out->n_triangles) * 3];
        out->vertices  = new float[(out->n_vertices) * 3];

        for(uint i=0; i<out->n_triangles; i++) {
            (out->triangles)[3*i+0] = raw.triangles[i].a;
            (out->triangles)[3*i+1] = raw.triangles[i].b;
            (out->triangles)[3*i+2] = raw.triangles[i].c;
        }

        for(uint i=0; i<out->n_vertices; i++) {
            (out->vertices)[3*i+0] = raw.vertices[i].pos.x;
            (out->vertices)[3*i+1] = raw.vertices[i].pos.y;
            (out->vertices)[3*i+2] = raw.vertices[i].pos.z;
        }
}


bool isSolid(CorkTriMesh cmesh)
{
        CorkMesh mesh;
        corkTriMesh2CorkMesh(cmesh, &mesh);
```

```
    bool solid = true;

    if(mesh.isSelfIntersecting()) {
        CORK_ERROR("isSolid() was given a self-intersecting mesh");
        solid = false;
    }

    if(!mesh.isClosed()) {
        CORK_ERROR("isSolid() was given a non-closed mesh");
        solid = false;
    }

    return solid;
}

void computeUnion(
    CorkTriMesh in0, CorkTriMesh in1, CorkTriMesh *out
) {
    CorkMesh cmIn0, cmIn1;
    corkTriMesh2CorkMesh(in0, &cmIn0);
    corkTriMesh2CorkMesh(in1, &cmIn1);

    cmIn0.boolUnion(cmIn1);

    corkMesh2CorkTriMesh(&cmIn0, out);
}

void computeDifference(
    CorkTriMesh in0, CorkTriMesh in1, CorkTriMesh *out
) {
    CorkMesh cmIn0, cmIn1;
    corkTriMesh2CorkMesh(in0, &cmIn0);
    corkTriMesh2CorkMesh(in1, &cmIn1);

    cmIn0.boolDiff(cmIn1);

    corkMesh2CorkTriMesh(&cmIn0, out);
}

void computeIntersection(
    CorkTriMesh in0, CorkTriMesh in1, CorkTriMesh *out
) {
    CorkMesh cmIn0, cmIn1;
    corkTriMesh2CorkMesh(in0, &cmIn0);
    corkTriMesh2CorkMesh(in1, &cmIn1);

    cmIn0.boolIsct(cmIn1);
```

```
    corkMesh2CorkTriMesh(&cmIn0, out);
}

void computeSymmetricDifference(
    CorkTriMesh in0, CorkTriMesh in1, CorkTriMesh *out
) {
    CorkMesh cmIn0, cmIn1;
    corkTriMesh2CorkMesh(in0, &cmIn0);
    corkTriMesh2CorkMesh(in1, &cmIn1);

    cmIn0.boolXor(cmIn1);

    corkMesh2CorkTriMesh(&cmIn0, out);
}

void resolveIntersections(
    CorkTriMesh in0, CorkTriMesh in1, CorkTriMesh *out
) {
    CorkMesh cmIn0, cmIn1;
    corkTriMesh2CorkMesh(in0, &cmIn0);
    corkTriMesh2CorkMesh(in1, &cmIn1);

    cmIn0.disjointUnion(cmIn1);
    cmIn0.resolveIntersections();

    corkMesh2CorkTriMesh(&cmIn0, out);
}
```

graphics/cork/src/main.cpp (with original code in gray)

```
//
+------------------------------------------------------------------------
----
// | main.cpp
// |
// | Author: Gilbert Bernstein
//
+------------------------------------------------------------------------
----
// | COPYRIGHT:
// |    Copyright Gilbert Bernstein 2013
// |    See the included COPYRIGHT file for further details.
// |
// |    This file is part of the Cork library.
// |
// |    Cork is free software: you can redistribute it and/or modify
// |    it under the terms of the GNU Lesser General Public License as
// |    published by the Free Software Foundation, either version 3 of
// |    the License, or (at your option) any later version.
// |
// |    Cork is distributed in the hope that it will be useful,
// |    but WITHOUT ANY WARRANTY; without even the implied warranty of
// |    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// |    GNU Lesser General Public License for more details.
// |
// |    You should have received a copy
// |    of the GNU Lesser General Public License
// |    along with Cork.  If not, see <http://www.gnu.org/licenses/>.
//
+------------------------------------------------------------------------
----

// This file contains a command line program that can be used
// to exercise Cork's functionality without having to write
// any code.

#include "files.h"

#include <iostream>
using std::cout;
using std::cerr;
using std::endl;
#include <sstream>
using std::stringstream;
```

```
using std::string;

using std::ostream;
#include <stdlib.h>

#include "cork.h"


void file2corktrimesh(
    const Files::FileMesh &in, CorkTriMesh *out
) {
    out->n_vertices = in.vertices.size();
    out->n_triangles = in.triangles.size();

    out->triangles = new uint[(out->n_triangles) * 3];
    out->vertices  = new float[(out->n_vertices) * 3];

    for(uint i=0; i<out->n_triangles; i++) {
        (out->triangles)[3*i+0] = in.triangles[i].a;
        (out->triangles)[3*i+1] = in.triangles[i].b;
        (out->triangles)[3*i+2] = in.triangles[i].c;
    }

    for(uint i=0; i<out->n_vertices; i++) {
        (out->vertices)[3*i+0] = in.vertices[i].pos.x;
        (out->vertices)[3*i+1] = in.vertices[i].pos.y;
        (out->vertices)[3*i+2] = in.vertices[i].pos.z;
    }
}

void corktrimesh2file(
    CorkTriMesh in, Files::FileMesh &out
) {
    out.vertices.resize(in.n_vertices);
    out.triangles.resize(in.n_triangles);

    for(uint i=0; i<in.n_triangles; i++) {
        out.triangles[i].a = in.triangles[3*i+0];
        out.triangles[i].b = in.triangles[3*i+1];
        out.triangles[i].c = in.triangles[3*i+2];
    }

    for(uint i=0; i<in.n_vertices; i++) {
        out.vertices[i].pos.x = in.vertices[3*i+0];
        out.vertices[i].pos.y = in.vertices[3*i+1];
        out.vertices[i].pos.z = in.vertices[3*i+2];
    }
}
```

```
void loadMesh(string filename, CorkTriMesh *out)
{
    Files::FileMesh filemesh;

    if(Files::readTriMesh(filename, &filemesh) > 0) {
        cerr << "Unable to load in " << filename << endl;
        exit(1);
    }

    file2corktrimesh(filemesh, out);
}
void saveMesh(string filename, CorkTriMesh in)
{
    Files::FileMesh filemesh;

    corktrimesh2file(in, filemesh);

    if(Files::writeTriMesh(filename, &filemesh) > 0) {
        cerr << "Unable to write to " << filename << endl;
        exit(1);
    }
}



class CmdList {
public:
    CmdList();
    ~CmdList() {}

    void regCmd(
        string name,
        string helptxt,
        std::function< void(std::vector<string>::iterator &,
                            const std::vector<string>::iterator &) >
body
    );

    void printHelp(ostream &out);
    void runCommands(std::vector<string>::iterator &arg_it,
                     const std::vector<string>::iterator &end_it);

private:
    struct Command {
          string name;     // e.g. "show" will be invoked with option
"-show"
        string helptxt; // lines to be displayed
```

```
        std::function< void(std::vector<string>::iterator &,
                       const std::vector<string>::iterator &) > body;
    };
    std::vector<Command> commands;
};

CmdList::CmdList()
{
    regCmd("help",
    "-help                  show this help message",
    [this](std::vector<string>::iterator &,
           const std::vector<string>::iterator &) {
        printHelp(cout);
        exit(0);
    });
}

void CmdList::regCmd(
    string name,
    string helptxt,
    std::function< void(std::vector<string>::iterator &,
                        const std::vector<string>::iterator &) > body
) {
    Command cmd = {
        name,
        helptxt,
        body
    };
    commands.push_back(cmd);
}

void CmdList::printHelp(ostream &out)
{
    out <<
    "Welcome to Cork.  Usage:" << endl <<
    "  > cork [-command arg0 arg1 ... argn]*" << endl <<
    "for example," << endl <<
    "  > cork -union box0.off box1.off result.off" << endl <<
    "Options:" << endl;
    for(auto &cmd : commands)
        out << cmd.helptxt << endl;
    out << endl;
}

void CmdList::runCommands(std::vector<string>::iterator &arg_it,
                          const std::vector<string>::iterator &end_it)
{
    while(arg_it != end_it) {
```

```
        string arg_cmd = *arg_it;
        if(arg_cmd[0] != '-') {
            cerr << arg_cmd << endl;
            cerr << "All commands must begin with '-'" << endl;
            exit(1);
        }
        arg_cmd = arg_cmd.substr(1);
        arg_it++;

        bool found = true;
        for(auto &cmd : commands) {
            if(arg_cmd == cmd.name) {
                cmd.body(arg_it, end_it);
                found = true;
                break;
            }
        }
        if(!found) {
               cerr << "Command -" + arg_cmd + " is not recognized" <<
endl;
            exit(1);
        }
    }
}


std::function< void(
    std::vector<string>::iterator &,
    const std::vector<string>::iterator &
) >
genericBinaryOp(
     std::function< void(CorkTriMesh in0, CorkTriMesh in1, CorkTriMesh
*out) >
        binop
) {
    return [binop]
    (std::vector<string>::iterator &args,
     const std::vector<string>::iterator &end) {
        // data...
        CorkTriMesh in0;
        CorkTriMesh in1;
        CorkTriMesh out;

        if(args == end) { cerr << "too few args" << endl; exit(1); }
        loadMesh(*args, &in0);
        args++;

        if(args == end) { cerr << "too few args" << endl; exit(1); }
```

```
        loadMesh(*args, &in1);
        args++;

        binop(in0, in1, &out);

        if(args == end) { cerr << "too few args" << endl; exit(1); }
        saveMesh(*args, out);
        args++;

        freeCorkTriMesh(&out);

        delete[] in0.vertices;
        delete[] in0.triangles;
        delete[] in1.vertices;
        delete[] in1.triangles;
    };
}


int main(int argc, char *argv[])
{
    initRand(); // that's useful

    if(argc < 2) {
        cout << "Please type 'cork -help' for instructions" << endl;
        exit(0);
    }

    // store arguments in a standard container
    std::vector<string> args(argc);
    for(int k=0; k<argc; k++) {
        args[k] = argv[k];
    }

    auto arg_it = args.begin();
    // *arg_it is the program name to begin with, so advance!
    arg_it++;

    CmdList cmds;

    // add cmds
    cmds.regCmd("solid",
      "-solid in                    Determine whether the input mesh
represents\n"
     "                              a solid object.  (aka. watertight)
(technically\n"
      "                                          solid  ==  closed  and
non-self-intersecting)",
```

```
    [](std::vector<string>::iterator &args,
       const std::vector<string>::iterator &end) {
        CorkTriMesh in;
        if(args == end) { cerr << "too few args" << endl; exit(1); }
        string filename = *args;
        loadMesh(*args, &in);
        args++;

        bool solid = isSolid(in);
        cout << "The mesh " << filename << " is: " << endl;
        cout << "    " << ((solid)? "SOLID" : "NOT SOLID") << endl;

        delete[] in.vertices;
        delete[] in.triangles;
    });

    // SHAPESHIFTER

    cmds.regCmd("translate",
    "-translate in x y z    Translate the input shape by x,y,z and \n"
     "                            and write the result back to the same
file",
    [](std::vector<string>::iterator &args,
        const std::vector<string>::iterator &end) {
        CorkTriMesh in;
           if(args == end) { cerr << "too few args for translate" <<
endl; exit(1); }
        string filename = *args;
        loadMesh(*args, &in);
        args++;

        float x, y, z;
           if(args == end) { cerr << "too few args for translate" <<
endl; exit(1); }
        x = strtof((*args).c_str(), NULL);
        args++;
           if(args == end) { cerr << "too few args for translate" <<
endl; exit(1); }
        y = strtof((*args).c_str(), NULL);
        args++;
           if(args == end) { cerr << "too few args for translate" <<
endl; exit(1); }
        z = strtof((*args).c_str(), NULL);
        args++;

        translateCork(&in, x, y, z);

        saveMesh(filename, in);
```

```
        delete[] in.vertices;
        delete[] in.triangles;
    });

    cmds.regCmd("reflect",
     "-reflect in a b c      Reflect the input shape across the plane
\n"
    "                       defined by ax + by + cz = 0 \n"
     "                           and write the result back to the same
file",
    [](std::vector<string>::iterator &args,
        const std::vector<string>::iterator &end) {
        CorkTriMesh in;
         if(args == end) { cerr << "too few args for reflect" << endl;
exit(1); }
        string filename = *args;
        loadMesh(*args, &in);
        args++;

        float x, y, z;
         if(args == end) { cerr << "too few args for reflect" << endl;
exit(1); }
        x = strtof((*args).c_str(), NULL);
        args++;
         if(args == end) { cerr << "too few args for reflect" << endl;
exit(1); }
        y = strtof((*args).c_str(), NULL);
        args++;
         if(args == end) { cerr << "too few args for reflect" << endl;
exit(1); }
        z = strtof((*args).c_str(), NULL);
        args++;

        reflectCork(&in, x, y, z);

        saveMesh(filename, in);
        delete[] in.vertices;
        delete[] in.triangles;
    });

    cmds.regCmd("rotate",
    "-rotate in x y z       Rotate the input shape around the\n"
    "                       x, y, and z axes and write the result\n"
    "                       back to the same file",
    [](std::vector<string>::iterator &args,
        const std::vector<string>::iterator &end) {
        CorkTriMesh in;
```

```
         if(args == end) { cerr << "too few args for rotate" << endl;
exit(1); }
        string filename = *args;
        loadMesh(*args, &in);
        args++;

        float x, y, z;
         if(args == end) { cerr << "too few args for rotate" << endl;
exit(1); }
        x = strtof((*args).c_str(), NULL);
        args++;
         if(args == end) { cerr << "too few args for rotate" << endl;
exit(1); }
        y = strtof((*args).c_str(), NULL);
        args++;
         if(args == end) { cerr << "too few args for rotate" << endl;
exit(1); }
        z = strtof((*args).c_str(), NULL);
        args++;

        rotateCork(&in, x, y, z);

        saveMesh(filename, in);
        delete[] in.vertices;
        delete[] in.triangles;
    });

    cmds.regCmd("scale",
    "-scale in x y z        Scale the input shape by x,y,z and \n"
     "                           and write the result back to the same
file",
    [](std::vector<string>::iterator &args,
        const std::vector<string>::iterator &end) {
        CorkTriMesh in;
         if(args == end) { cerr << "too few args for scale" << endl;
exit(1); }
        string filename = *args;
        loadMesh(*args, &in);
        args++;

        float x, y, z;
         if(args == end) { cerr << "too few args for scale" << endl;
exit(1); }
        x = strtof((*args).c_str(), NULL);
        args++;
         if(args == end) { cerr << "too few args for scale" << endl;
exit(1); }
        y = strtof((*args).c_str(), NULL);
```

```
        args++;
          if(args == end) { cerr << "too few args for scale" << endl;
exit(1); }
        z = strtof((*args).c_str(), NULL);
        args++;

        scaleCork(&in, x, y, z);

        saveMesh(filename, in);
        delete[] in.vertices;
        delete[] in.triangles;
    });

    // END SHAPESHIFTER

    cmds.regCmd("union",
      "-union in0 in1 out      Compute the Boolean union of in0 and
in1,\n"
    "                        and output the result",
    genericBinaryOp(computeUnion));
    cmds.regCmd("diff",
     "-diff in0 in1 out      Compute the Boolean difference of in0 and
in1,\n"
    "                        and output the result",
    genericBinaryOp(computeDifference));
    cmds.regCmd("isct",
     "-isct in0 in1 out       Compute the Boolean intersection of in0
and in1,\n"
    "                        and output the result",
    genericBinaryOp(computeIntersection));
    cmds.regCmd("xor",
    "-xor in0 in1 out       Compute the Boolean XOR of in0 and in1,\n"
    "                        and output the result\n"
    "                        (aka. the symmetric difference)",
    genericBinaryOp(computeSymmetricDifference));
    cmds.regCmd("resolve",
    "-resolve in0 in1 out   Intersect the two meshes in0 and in1,\n"
      "                              and output the connected mesh with
those\n"
      "                                 intersections made explicit and
connected",
    genericBinaryOp(resolveIntersections));


    cmds.runCommands(arg_it, args.end());

    return 0;
}
```