## Contents

1. **Introduction**  
   1.1 Concept ................................................. 10  
   1.2 Motivation ........................................... 10

2. **Language Tutorial**  
   2.1 Writing Hello World .................................. 10  
   2.2 Control flow statements ............................... 11  
   2.3 Built-in Objects ...................................... 11  
   2.4 Function calls ........................................ 12  
   2.5 runGame() Loop ....................................... 12

3. **LRM**  
   3.1 Lexical .................................................. 12  
      3.1.1 Identifiers ......................................... 12  
      3.1.2 Comments .......................................... 13  
      3.1.3 Separators ......................................... 13  
      3.1.4 White Space ....................................... 13  
      3.1.5 Reserved Keywords ................................. 13  
   3.2 Primitive types ....................................... 14  
      3.2.1 Integer ............................................ 14  
      3.2.2 Float .............................................. 14  
      3.2.3 Char ............................................... 14  
      3.2.4 Boolean .......................................... 14  
   3.3 Literals ................................................ 15  
      3.3.1 Integer Literals ................................... 15  
      3.3.2 Float Literals ..................................... 15  
      3.3.3 Bool Literals ...................................... 15  
      3.3.4 Char Literals ...................................... 16  
   3.4 Expressions ............................................ 16  
      3.4.1 Primary Expressions ............................... 16  
      3.4.2 Precedence of Operators ........................... 16  
      3.4.3 Expressions formed from function calls .......... 17  
      3.4.4 Expressions formed from object function calls .. 17  
      3.4.5 Statements ........................................ 18  
      3.4.6 Declarations ...................................... 18  
      3.4.7 Assignments ........................................ 18  
      3.4.8 Object Instantiations ............................... 19  
      3.4.9 Blocks and Control Flow ......................... 19  
   3.5 Operators ............................................... 19  
      3.5.1 Arithmetic Operators ............................. 19  
      3.5.2 Comparison Operators ............................ 20  
      3.5.3 Negation Operator ................................ 20  
      3.5.4 Logical Operators ............................... 21  
      3.5.5 Assignment Operators ............................ 21
### Program Structure
- **3.6.1 The if Statement**
- **3.6.2 The while Statement**
- **3.6.3 The for Statement**
- **3.6.4 The runGame() Statement**
- **3.6.5 How to use runGame()**
- **3.6.6 What rungame() does internally**
- **3.6.7 Scope**

### Standard Functions
- **3.7.1 Function Declarations**
- **3.7.2 Function Calls**
- **3.7.3 Return Statements**

### Built-in Functions
- **3.8.1 The initsdl() function**
- **3.8.2 The isKeyPressed(<int>) function**
- **3.8.3 Printing functions**

### Built-in Classes
- **3.9.1 Constructing Built-in Objects**
- **3.9.2 Using Functions of Built-in Objects**
- **3.9.3 The Entity Class**
- **3.9.4 Instance Variables**
- **3.9.5 Constructor**
- **3.9.6 Functions**
- **3.9.7 The Player Class**
- **3.9.8 Player Instance Variables**
- **3.9.9 Player Constructor**
- **3.9.10 Player Functions**

### Context Free Grammar

### Project Plan
- **4.1 Planning**
- **4.2 Testing**
- **4.3 Style Guide**
- **4.4 Timeline**
- **4.5 Roles and Responsibilities**
- **4.6 Software Development Environment**
- **4.7 Project Log**

### Architectural Design
- **5.1 Block Diagram**
- **5.2 Compiler**
  - **5.2.1 Scanner**
  - **5.2.2 Parser and AST**
  - **5.2.3 Semantic Checking**
  - **5.2.4 Code Generation**

### Sample Code
5.2.5 Linking with C (Runtime) and integrating SDL functionality 48
5.3 Architecture Design Responsibility 49

6 Testing Plan 49
6.1 Source To Target 49
6.2 Test Suite 61
6.3 Test Automation 64
6.4 Testing Responsibility 64

7 Lessons Learned 64
7.1 Elliott Morelli 64
7.2 Cherry Chu 65
7.3 Robert Becker 65
7.4 Michael Fagan 65

8 Appendix 65
8.1 Testall.sh 65
8.2 scanner.ml 69
8.3 ast.ml 71
8.4 Luciferparse.mly 74
8.5 sast.ml 77
8.6 semant.ml 78
8.7 codegen.ml 83
8.8 lucifer.ml 93
8.9 C files 94
8.9.1 entity.c 94
8.9.2 player.c 95
8.10 Test file bodies & outputs 98
8.10.1 fail-after-return.luc 98
8.10.2 fail-after-return.err 98
8.10.3 fail-arith1.luc 98
8.10.4 fail-arith1.err 98
8.10.5 fail-arith2.luc 98
8.10.6 fail-arith2.err 98
8.10.7 fail-arith3.luc 98
8.10.8 fail-arith3.err 98
8.10.9 fail-assign1.luc 99
8.10.10 fail-assign1.err 99
8.10.11 fail-assign2.luc 99
8.10.12 fail-assign2.err 99
8.10.13 fail-assign3.luc 99
8.10.14 fail-assign3.err 99
8.10.15 fail-comment.luc 100
8.10.16 fail-comment.err 100
8.10.17 fail-controlPlayer-rungame.luc 100
8.10.18 fail-controlPlayer-rungame.err 100
8.10.19 fail-declEntity-changeXY.luc . . . . . . . . . . . . . . . . 100
8.10.20 fail-declEntity-changeXY.err . . . . . . . . . . . . . . . . 101
8.10.21 fail-entity-uninitialized.luc . . . . . . . . . . . . . . . . . . 101
8.10.22 fail-entity-uninitialized.err . . . . . . . . . . . . . . . . . . 101
8.10.23 fail-entityInvalidName1.luc . . . . . . . . . . . . . . . . . 101
8.10.24 fail-entityInvalidName1.err . . . . . . . . . . . . . . . . . 101
8.10.25 fail-entityInvalidName2.luc . . . . . . . . . . . . . . . . . 101
8.10.26 fail-entityInvalidName2.err . . . . . . . . . . . . . . . . . 101
8.10.27 fail-entityInvalidTexture.luc . . . . . . . . . . . . . . . . . 102
8.10.28 fail-entityInvalidTexture.err . . . . . . . . . . . . . . . . . 102
8.10.29 fail-entityInvalidX.luc . . . . . . . . . . . . . . . . . . . . 102
8.10.30 fail-entityInvalidX.err . . . . . . . . . . . . . . . . . . . . 102
8.10.31 fail-entityInvalidY.luc . . . . . . . . . . . . . . . . . . . . 102
8.10.32 fail-entityInvalidY.err . . . . . . . . . . . . . . . . . . . . 102
8.10.33 fail-expr1.luc . . . . . . . . . . . . . . . . . . . . . . . . . 102
8.10.34 fail-expr1.err . . . . . . . . . . . . . . . . . . . . . . . . . 102
8.10.35 fail-for1.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.10.36 fail-for1.err . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.10.37 fail-for2.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.10.38 fail-for2.err . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.10.39 fail-fun1.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.10.40 fail-fun1.err . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.10.41 fail-fun10.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.10.42 fail-fun10.err . . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.10.43 fail-fun2.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.10.44 fail-fun2.err . . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.10.45 fail-fun3.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.10.46 fail-fun3.err . . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.10.47 fail-fun4.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.10.48 fail-fun4.err . . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.10.49 fail-fun5.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 105
8.10.50 fail-fun5.err . . . . . . . . . . . . . . . . . . . . . . . . . . 105
8.10.51 fail-fun6.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 105
8.10.52 fail-fun6.err . . . . . . . . . . . . . . . . . . . . . . . . . . 105
8.10.53 fail-fun7.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 105
8.10.54 fail-fun7.err . . . . . . . . . . . . . . . . . . . . . . . . . . 105
8.10.55 fail-fun8.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 106
8.10.56 fail-fun8.err . . . . . . . . . . . . . . . . . . . . . . . . . . 106
8.10.57 fail-fun9.luc . . . . . . . . . . . . . . . . . . . . . . . . . . 106
8.10.58 fail-fun9.err . . . . . . . . . . . . . . . . . . . . . . . . . . 106
8.10.59 fail-hello-world-nofun.luc . . . . . . . . . . . . . . . . . . . 106
8.10.60 fail-hello-world-nofun.err . . . . . . . . . . . . . . . . . . . 106
8.10.61 fail-hello-world-nosemi.luc . . . . . . . . . . . . . . . . . . 106
8.10.62 fail-hello-world-nosemi.err . . . . . . . . . . . . . . . . . . 107
8.10.63 fail-hello-world-openstring.luc . . . . . . . . . . . . . . . . . 107
8.10.64 fail-hello-world-openstring.err . . . . . . . . . . . . . . . . . 107
8.10.11 test-addPlayerControl2.luc .......................... 113
8.10.11 test-addPlayerControl2.out .......................... 113
8.10.11 test-addPlayerHitBox.luc ............................. 113
8.10.11 test-addPlayerHitBox.out ............................ 114
8.10.11 test-arith1.luc ........................................ 114
8.10.11 test-arith1.out .................................... 114
8.10.11 test-arith2.luc ........................................ 114
8.10.11 test-arith2.out .................................... 114
8.10.11 test-arith3.luc ........................................ 114
8.10.11 test-arith3.out .................................... 114
8.10.12 test-changeEntityXY.luc ............................ 115
8.10.12 test-changeEntityXY.out ............................ 115
8.10.12 test-changePlayerXY.luc ............................ 115
8.10.12 test-changePlayerXY.out ............................ 115
8.10.12 test-comment.luc ...................................... 115
8.10.12 test-comment.out .................................... 115
8.10.12 test-entity.luc ......................................... 116
8.10.12 test-entity.out ...................................... 116
8.10.12 test-entityExprArg.luc .............................. 116
8.10.12 test-entityExprArg.out .............................. 116
8.10.13 test-fib.luc ........................................... 116
8.10.13 test-fib.out ......................................... 117
8.10.13 test-for1.luc .......................................... 117
8.10.13 test-for1.out ....................................... 117
8.10.13 test-for2.luc .......................................... 117
8.10.13 test-for2.out ....................................... 117
8.10.13 test-fun1.luc ......................................... 118
8.10.13 test-fun1.out ...................................... 118
8.10.13 test-fun2.luc ......................................... 118
8.10.13 test-fun2.out ...................................... 118
8.10.14 test-fun3.luc ......................................... 118
8.10.14 test-fun3.out ...................................... 118
8.10.14 test-fun4.luc ......................................... 118
8.10.14 test-fun4.out ...................................... 118
8.10.14 test-fun5.luc ......................................... 119
8.10.14 test-fun5.out ...................................... 119
8.10.14 test-fun6.luc ......................................... 119
8.10.14 test-fun6.out ...................................... 119
8.10.14 test-fun7.luc ......................................... 120
8.10.14 test-fun7.out ...................................... 120
8.10.15 test-fun8.luc ......................................... 120
8.10.15 test-fun8.out ...................................... 120
8.10.15 test-fun9.luc ......................................... 120
8.10.15 test-fun9.out ...................................... 120
8.10.15 test-gcd.luc ........................................... 121
8.10.15 test-gcd.out ........................................ 121
1 Introduction

1.1 Concept

Our language proposal was to design and implement a language that will make it easier to develop 2D video games efficiently. We plan to have our implemented language syntactically similar to Java, in addition to being strongly and statically typed, have static scoping, and pass by value. Lucifer will also feature direct integration with the SDL2 library. SDL is a software development library that provides low level access to audio, keyboard, mouse, joystick and graphical hardware. This library is commonly used in the design of indie video games such as Cave Story, Angry Birds, and Dwarf Fortress, and its support for OCaml allows us to use it to mediate hardware interactions. We include functionality from SDL directly into our language in order to create a more streamlined language that is hardware independent.

1.2 Motivation

Our language is built with the purpose of efficiently creating video games. In order to keep the scope of the project within a reasonable margin, our language was developed with a focus on 2D games in mind. Examples of what one could implement are simpler games such as Pong, Space Invaders, and Pac-Man. That being said, 2D games of any type will be able to be implemented. As our language will also be able to produce console output and perform arithmetic, programs performing arithmetic operations will also be possible (although these would usually be used to move an entity around in a game).

2 Language Tutorial

To set up the compiler first run `make SDL` followed by `make` while in the Lucifer source file. Lucifer is syntactically similar to C with some Java-like characteristics. To illustrate the structure of a typical Lucifer program we will run through a "hello world" program. Afterwards we will give a rundown of some Lucifer-specific features to get you started.

2.1 Writing Hello World

To write hello world we will first declare our main function:

```lucifer
fun main void(){
```

`fun` is a keyword which tells the compiler you are declaring a function, `main` is our function's name, and `void` is the function's return type. `()` is where we would put our function parameters, and `}` opens the body of our function. Our next line of code would be:
string hello;

string hello; is a variable declaration statement with hello being the name of the variable and string being its type. Variables in Lucifer are either of primitive type e.g. int, string, float, bool, char

```c
hello = "Hello World!";
print(hello);
```

Here our last couple lines of code first assign the string literal "Hello World!" to our variable hello, Then we make a call to the built-in function print() to print out the string variable.

### 2.2 Control flow statements

In addition to the above features, Lucifer also has for and while loops. These are declared in the same way they would be in C:

```c
for ( stmt ; stmt ; stmt ) stmt
// or
while ( stmt ) stmt
```

stmt here can be any other statement as defined in LRM e.g. an expression or a block of statements surrounded by curly braces. Lucifer also features if statements

### 2.3 Built-in Objects

Lucifer has two built-in objects: the Player Object and the Entity Object.

Here is how to declare and define a Entity object:

```c
Entity e;
e = new Entity(0, 0, "entityImage.png");
```

The above code generates a Entity object named e on the upper left corner. The Entity e will be rendered as the image supplied by entityImage.png.

Here is how to declare and define a Player object:

```c
Player p1;
p1 = new Player(10, 10, "texture.png", 10);
```

The above code generates a Player object named p1 with position (10, 10) and will be rendered as the image supplied by texture.png. The last argument is the size of the player control array, which contains a set of keyboard keys that is associated with the object.

Note that the size argument is expected to be at least 4.
2.4 Function calls

Functions in Lucifer can be called as a global function, or a member function. A global function is called like this:

```latex
print(123);
```

And member function is called like this:

```latex
Player p;
p.addPlayerHitBox(10, 10);
```

Also, a member function can be called as a global function by adding the object as the first argument of the function. Here is a rewrite of the member function call above as a global function.

```latex
Player p;
addPlayerHitBox(p, 10, 10);
```

2.5 runGame() Loop

Here is how to call a runGame Loop.

```latex
int x; Entity e;
x = 100;
e = new Entity(100, 100, "bat.png")
runGame(x > 0; 1) {
    x = x - 1;
    e.changeEntityX(-10);
}
```

The Entity e will start at position (100, 100). Then the program will check the condition given in the first runGame() argument. Entity e will move up 10 unit every loop as long as the condition is valid. Note that an object cannot be instantiated within a Rungame loop as this would cause SDL to error. Thus, the Lucifer Parser throws an error is this happens.

3 LRM

3.1 Lexical

3.1.1 Identifiers

Identifiers in Lucifer are sequences of characters are composed of ASCII letters, decimal digits and the underscore character \_.

The first character of an identifier has to be a letter.
3.1.2 Comments

Lucifer supports both multi-line and single-line comments. Comments are ignored by parser.

Single-line comment starts with double backslash //.

```c
int r = 1; // r is radius of a circle
```

Multi-line comment are enclosed by /* and */.

```c
float x = 2.91;
/*
   This is a multi line comment
*/
```

3.1.3 Separators

Separators separate tokens. Lucifer accepts ( ) { } [ ] ; , . as separators.

Example of separator usage

```c
int x = (1 + 2) * (2 + 3); x + 1;
```

3.1.4 White Space

White Space characters are separator that are discarded during parsing. Valid white space characters include space characters and newline characters. Tab characters are invalid.

3.1.5 Reserved Keywords

```c
return
if
else
noelse
for
while
int
bool
true
false
float
```
Lucifer also accepts the predefined SDL Scancodes as keywords. They can be looked up in [https://wiki.libsdl.org/SDLScancodeLookup](https://wiki.libsdl.org/SDLScancodeLookup). Currently Lucifer supports the SDL Scancodes for alphanumerical characters, up down left and right arrows, and the space bar.

### 3.2 Primitive types

Lucifer supports four primitive types of objects: integers, character literals, single-precision floating point numbers and booleans.

#### 3.2.1 Integer

Integer has 4 bytes. The 32-bit int data type can hold integer values in the range of -2,147,483,648 to 2,147,483,647.

Here is an example to declare and define integer variable.

```java
int a = 1;
```

#### 3.2.2 Float

Float has 8 bytes. Float data type has size ranges from 1e-37 to 1e37.

An example to declare and define float variable is

```java
float foo = 3.14;
```

#### 3.2.3 Char

Char has 1 byte and it can be any ASCII characters.

#### 3.2.4 Boolean

Boolean has 1 byte. It is a binary data type with either true or false value. Boolean values are represented by the reserved keyword `true` and `false`. 
3.3 Literals
Lucifer supports integer literals, float literals, char literals and bool literals.

3.3.1 Integer Literals
An integer literals is a sequence of decimal digits.
The regular expression for integer literal is

\[[0-9]+\]

Examples of integer literals are

72
0
1327

The context free grammar for integer literal in the parser is:

\texttt{expr: LITERAL}

3.3.2 Float Literals
A float Literal consists of an integer part, a decimal point, a fraction part, an e
and an optionally signed integer exponent.
The regular expression representing the float literal is

\(((([0-9]+[.]([0-9]*))|([.][0-9]+))(([e][+-]?[0-9]+))|([0-9]+[e][+-]?[0-9]+))\)

Examples of float literals are

0.5e+15
8.3
1.
13e9

The context free grammar for float literal in the parser is:

\texttt{expr: FLIT}

3.3.3 Bool Literals
A bool literal is represented by keywords \texttt{true} and \texttt{false}.
The context free grammar for bool literal in the parser is:

\texttt{expr: BLIT}
3.3.4 Char Literals

A char literal is a sequence of ASCII characters enclosed by double quotation characters (" "). If the char literal content contains any double quotation characters, the character should be preceded by the escape character \.

The regular expression for string literals is

"([^"\]|\.)*)"

Example of char literals are

"Hello World!"
"foo"
"bar\"baz"
"abc345@#&"

The context free grammar for char literal in the parser is:

expr: CLIT

3.4 Expressions

3.4.1 Primary Expressions

Primary expressions involving "." and function calls group left to right. Expressions include the following:

1. Literals, whose grammar rules are described in 3.1

2. Identifiers, who have the token ID

3.4.2 Precedence of Operators

Operators in Lucifer follow rules of precedence and associativity that determine the order in which expressions will be evaluated.

Table 2 lists each operator and indicates the precedence and associativity of each. Table 3 shows some simple examples of precedence and associativity. Parentheses can be used to override these rules.

The list below shows the grammar rules for operator expressions:

expr:

expr + expr
expr - expr
expr * expr
expr / expr
expr % expr
expr == expr
expr != expr
expr <= expr
expr > expr
expr >= expr
expr && expr
expr || expr
-expr
!expr

<table>
<thead>
<tr>
<th>Tokens (From High to Low Priority)</th>
<th>Operators</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>! -</td>
<td>Unary negation</td>
<td>R-L</td>
</tr>
<tr>
<td>* / %</td>
<td>Multiplicative</td>
<td>L-R</td>
</tr>
<tr>
<td>+ -</td>
<td>Additive</td>
<td>L-R</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;=</td>
<td>Relational comparisons</td>
<td>L-R</td>
</tr>
<tr>
<td>== !=</td>
<td>Equality comparisons</td>
<td>L-R</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND</td>
<td>L-R</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>Assignment</td>
<td>R-L</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
<td>L-R</td>
</tr>
</tbody>
</table>

Table 1: Operator Precedence and Associativity

3.4.3 Expressions formed from function calls

func-call-expr:
  identifier.(actualparams-list)

3.4.4 Expressions formed from object function calls

obj-func-call-expr:
  identifier.expression(actualparams-list)
<table>
<thead>
<tr>
<th>Expression</th>
<th>Results</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 + 2 * 5</td>
<td>13</td>
<td>Multiplication is done before division</td>
</tr>
<tr>
<td>3 + (2 * 5)</td>
<td>13</td>
<td>Same order as before, but parentheses provide clarification</td>
</tr>
<tr>
<td>(3 + 2) * 5</td>
<td>25</td>
<td>Parentheses override the precedence rules</td>
</tr>
<tr>
<td>true</td>
<td></td>
<td>true &amp;&amp; false</td>
</tr>
<tr>
<td>true</td>
<td></td>
<td>(true &amp;&amp; false)</td>
</tr>
<tr>
<td>(true</td>
<td></td>
<td>true) &amp;&amp; false</td>
</tr>
</tbody>
</table>

Table 2: Precedence and Associativity Examples

3.4.5 Statements

3.4.6 Declarations

Syntax for declaring variables takes the following form:
<type> <identifier> ;

where a <type> is any of the following:

type:

int
bool
float
char

Rules for function declarations can be found in Section 8.1

3.4.7 Assignments

Assignments must be done after declarations. Primitive Variable Assignment syntax:

<type> <varidentifier> ;
<varidentifier> = <expression> ;
where the left side of the statement is the variable’s identifier and the right hand side is an expression.

### 3.4.8 Object Instantiations

Object Instantiations take the following form:

```
<objtyp> <objidentifier>;
<objidentifier> = new <objtyp> (args_opt);
```

Where `<objtyp>` is the keyword name of a built-in class (Entity, Player), `<objidentifier>` is the identifier for the object, `new` is the keyword for instantiation, and `args_opt` is an optional list of actual parameters that the constructor takes.

### 3.4.9 Blocks and Control Flow

If statements, While Statements, For Statements, and the `runGame()` statement are all considered *statements* in Lucifer.

Their rules and syntax can be found in Section 7.

### 3.5 Operators

#### 3.5.1 Arithmetic Operators

Lucifer supports standard two-operands arithmetic operations: addition, subtraction, multiplication, division, and modulo. Two-operands operations require the two operands to be of the same type. For the modulo operation, both operands must be of type integer.

```java
/* Addition */
a = 1 + 3;
b = 3.4 + 5.2;
c = a + b;

/* Subtraction */
a = 1 - 3;
b = 3.4 - 5.2;
c = a - b;

/* Multiplication */
a = 1 * 3;
b = 3.4 * 5.2;
```
### Table 3: Comparison Operators

<table>
<thead>
<tr>
<th>operator</th>
<th>meaning</th>
<th>usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equal to</td>
<td>foo == bar</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
<td>foo != bar</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>foo &gt; bar</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>foo &lt; bar</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
<td>foo &gt;= bar</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
<td>foo &lt;= bar</td>
</tr>
</tbody>
</table>

```plaintext
c = a * b;

/* Division */
a = 1 / 3; // resulting quotient is truncated to integer value
b = 3.4 / 5.2;
c = 20.3 / b;

/* Modulo */
a = 17 % 3;
b = 5 % a;
```

Lucifer also supports single-operand arithmetic negation.

```plaintext
/* Negation */
int a = -2;
float b = -3.14;
```

### 3.5.2 Comparison Operators

Comparison operators compare the two operands and determines how they relates to each other. The two operands are expression tokens that must evaluate to the same type, and the result of comparison is a boolean type.

Details of each operators and their usage are illustrated in Table 1.

### 3.5.3 Negation Operator

Negation operator is denoted by the character `!` and is applied to a single operand of boolean type to negate its value.

Example of negation operator usage

```plaintext
if (!(foo == 1)) prints("hello world.");
```
3.5.4 Logical Operators

Lucifer support the OR and AND logical operators. Logical operator evaluates the truth value of the two operands of boolean type.

The OR operator \texttt{||} evaluates to true if either of the operands is true, false otherwise.

```java
if ((bar == 2) || (foo == 1)) print("bar is 2 and foo is 1.");
```

The AND operator \texttt{&&} evaluates to true if both operands are true, false otherwise.

```java
if ((bar == 2) && (foo == 1)) print("Either bar is 2 or foo is 1.");
```

3.5.5 Assignment Operators

Assignment operator is denoted by the character =. It is used to define or assign value to a variable.

The left operand is a variable and the right operand is a value to be stored in the variable on the left.

Example of assignment operator usage

```java
int a;
a = 3 + 2;
```

3.6 Program Structure

A typical program structure in Lucifer contains function definitions, followed by a \texttt{main()} function.

In Lucifer, statements allowed in global scope are variable declarations and function definitions.

The starting execution point for a program in Lucifer is the \texttt{main()} function which is required to be a function definition in the program file.

3.6.1 The if Statement

The if/else statement in Lucifer supports branching of program flow based on the evaluation of a boolean expression. An if/else statement introduces a new scope and therefore must be surrounded by curly brackets.

The if statement has the following general form

```java
if (condition) {then-statement}
else {else-statement}
```
When there is no action for the else part, the statement takes the following form.

```plaintext
if (condition) {then-statement}
noelse
```

The context free grammar for if statement is:

```plaintext
stmt: IF (expr) stmt ELSE stmt
stmt: IF (expr) stmt NOELSE
```

### 3.6.2 The while Statement

The while loop statement has the following syntax:

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

The statement within the loop is executed repeatedly so long as the value of the expression evaluates to true. The value of the expression is tested before each execution of the statement.

The context free grammar for while statement is:

```plaintext
stmt: WHILE (expr) stmt
```

### 3.6.3 The for Statement

The for loop statement has the following syntax:

```plaintext
for (init; condition; increment)
{statement}
```

Only the condition part is required, while the init and increment parts are optional. The increment part updates the condition after every loop. The statement within the for loop is executed repeatedly as long as the condition is true.

The context free grammar for for statement is:

```plaintext
stmt: FOR (expr_opt; expr; expr_opt) stmt
```

### 3.6.4 The runGame() Statement

The runGame() loop in Lucifer is responsible for making internal calls to SDL library functions in order to render Entity and Player objects in a window.
3.6.5 How to use runGame()

The runGame() loop should be the last statement in a Lucifer program, within the main() method. It can contain expressions.

runGame() takes one boolean argument and one integer argument. runGame() will continue to iterate as long as the boolean evaluates to true. The integer argument indicates desired delay in ms.

runGame(<bool> expression, <int> framerate) b_stmt

Note: an b_stmt is a helper grammar which is identical to a regular stmt but it cannot produce an object instantiation stmt

3.6.6 What rungame() does internally

1. runGame() calls loadTexture() and SDL_QueryTexture() for each Player and Entity that has been declared and instantiated before the runGame() loop starts.

2. runGame() internally calls SDL_SetRenderDrawColor() and SDL_RenderClear() to prepare the window for rendering.

3. runGame() internally calls SDL_PollEvent() in order to update the keyboard scancode array, which is accessible through the checkkey() function.

4. runGame() runs all code written within its function body. This should be game logic that updates positions and states of Players and Entities.

5. runGame() draws any Entities and Players that are global or local to the same function as itself (these were queued on the first iteration) by calling SDL_QueryTexture() and SDL_RenderCopy().

6. runGame() renders the updated scene by calling SDL_RenderPresent().

7. runGame() caps the framerate of the game by calling SDL_Delay(), passing in its second argument as the framerate in ms.

3.6.7 Scope

Variables declared within a function have the scope of that function, and are accessible only within that function’s body. Subscopes are designated using curly brackets: {}.

3.7 Standard Functions

3.7.1 Function Declarations

Functions in Lucifer have the following syntax:
fun <functionId> <returnType> (<type> arg1,<type> arg2...<type> argn){function body}

Where functionId is a unique identifier and returnType is the type that the function returns. The arguments are variable declarations that are passed into the function body when the function is called. Arguments are optional, but must have their type stated in the function definition.

Example of function declaration:

fun add int (int a, int b){
    return a + b;
}

In this example, add is the functionId, int is the returnType, and int a, int b are the arguments.

3.7.2 Function Calls

When calling a function, the function call must have the same number, ordering, and type of actual parameters as the arguments declared in the function declaration.

To supply actual parameters to a function and run the function body, the following syntax is used:

functionId(actual parameters);

Example of a function call:

int g = 5;
int h = 10;
int res = add(g,h);

In this example, add(g,h); runs the function body of the add function and supplies g and h as actual parameters.

3.7.3 Return Statements

The return statement allows a function to return an expression of the type specified in its function declaration. If the type of the expression in the return statement does not match the return type of the function, the compiler will
throw an error. The keyword `void` is used in the function declaration if the function does not return anything.

```c
return ( expr_opt );
```

### 3.8 Built-in Functions

#### 3.8.1 The `initsdl()` function

The `initsdl()` function in Lucifer is responsible for initializing SDL and internally creates its window. It internally calls `SDL_CreateWindow()`, `SDL_CreateRenderer()`, `SDL_SetHint()`, and `IMG_Init()`.

It must be called in the `main()` in order for a `runGame()` loop to render Entities and Players. Without this call, `runGame()` will not show graphical output.

```c
main()
{
    initsdl();
}
```

#### 3.8.2 The `isKeyPressed(<int>)` function

The `isKeyPressed(<int>)` function in Lucifer returns true if the key matching the scancode passed to it has been pressed since its last call.

```c
main()
{
    isKeyPressed(<int>);
}
```

#### 3.8.3 Printing functions

Lucifer has differently typed print functions for different data types:

- `prints(<string>): // for strings`
- `print(<int>): // for ints`
- `printf(<bool>): // for bools`
- `printf(<float>): // for floats`

### 3.9 Built-in Classes

Lucifer has built-in Entity and Player classes which may be used to construct objects.
3.9.1 Constructing Built-in Objects

Built-in Objects are instantiated with Javalike syntax, using the `new` keyword. Objects must be declared and then forward-instantiated. All declarations must come before instantiations.: Lucifer does not allow objects to be instantiated within a runGame() loop, in order for their textures to be properly rendered.

```
Declaration:  <classid> <objectid> ;
Instantiation:  <objectid> = new <classid> (args_opt);
```

Example:

```
Player p;
Entity e;
p = new Player(75,100, "texture.png",2);
e = new Entity(75,100, "texture.png");
```

After constructing an object, refer to it by its variable name to access its variables and functions.

3.9.2 Using Functions of Built-in Objects

To call an object’s function, use the '.' character before the function name and two parentheses () after the function name. Actual parameters of the function, if any, should be in between the parentheses:

```
<objectid>.<functionid>(actual_list_opt);
```

Example:

```
p.addHitbox(50,100);
```

3.9.3 The Entity Class

An Entity in Lucifer describes any game object that will be rendered on the SDL Window.

3.9.4 Instance Variables

Entity stores a 2D (x, y) position, the name of a texture file(JPG,PNG) and a 2D hitbox (hx,hy).
3.9.5 Constructor

The expected parameters for the Entity constructor are its starting (center) x position, its starting y position, and its texture file for its sprite.

Entity e = new Entity(50,50, "texture.png");

3.9.6 Functions

fun void setEntityX(int x);
   Updates Entity position x.

fun void setEntityY(int y);
   Updates Entity position y.

fun void addEntityHitbox(int x, int y);
   Updates Entity hitbox variables hx and hy.

fun void changeEntityX(int dx);
   Updates Entity x to x+dx.

fun void changeEntityY(int dy);
   Updates Entity y to y+dy.

3.9.7 The Player Class

In Lucifer, Player Objects are sprites that have an array of preset controls by hardware input.
3.9.8 Player Instance Variables

Player objects have the same Entity instance variables of x, y, texture, hx and hy. Player objects also contain an array of integers that store the keycodes for player controls.

```java
int x;
int y;
String texture;
int hx;
int hy;

int[] controls;
```

3.9.9 Player Constructor

The Player constructor has four arguments: starting (top-left) x position, its starting y position, its texture file for its sprite, and an int value used to designate the size of its internal, array, which is an array of SDL Scancodes. The size of this array must be greater than or equal to 4. Internally, the first four indices of the controls array correspond to up, down, left and right movement.

SDL Scancodes refer to hardware input from specific keyboard keys. Their corresponding integer values can be found in this reference:

https://wiki.libsdl.org/SDLScancodeLookup

Either an integer value or the SDL Scancode value is accepted, as shown below:

```java
Player p = new Player(75,100, "texture.png",4);
p.addControl(0,SDL_SCANCODE_UP);
Player p = new Player(75,100, "texture.png",4);
p.addControl(0,82);
```

3.9.10 Player Functions

Player has all of the same functions of Entity, which include:

```java
fun void setPlayerX(int x);

Updates Player position x.
```
fun void setPlayerY(int y);
    Updates Player position y.

fun void addPlayerHitbox(int x, int y);
    Updates Player hitbox variables hx and hy.

fun void changePlayerX(int dx);
    Updates Player x to x+dx.

fun void changePlayerY(int dy);
    Updates Player y to y+dy.

Player also contains addPlayerControl() function.

fun void addPlayerControl(int i, int newcontrol);
    This function replaces the scancode in Player's controls array at
    index i with the newcontrol int value.

3.10 Context Free Grammar

The terminals for this grammar are written in ALLCAPS. They represent tokens
that are passed to the parser by the scanner. Some include ID, SEMI, or ADD.
program:
  decls EOF

decls:
  /* nothing */
  | decls vdecl
  | decls fdecl

fdecl:
  FUN ID typ LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE

formals_opt:
  /* nothing */
  | formal_list

formal_list:
  typ ID
  | formal_list COMMA typ ID

typ:
  INT
  | BOOL
  | FLOAT
  | STRING
  | VOID
  | PLAYER
  | ENTITY

vdecl_list:
  /* nothing */
  | vdecl_list vdecl

vdecl:
  typ ID SEMI

stmt_list:
  /* nothing */
  | stmt_list stmt

b_stmt_list:
  /* nothing */
  | b_stmt_list b_stmt

stmt:
  expr SEMI
  | RETURN expr_opt SEMI
  | LBRACE stmt_list RBRACE
  | IF LPAREN expr RPAREN stmt %prec NOELSE
  | IF LPAREN expr RPAREN stmt ELSE stmt
  | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
  | WHILE LPAREN expr RPAREN stmt
  | RUNGAME LPAREN expr SEMI expr RPAREN b_stmt

b_stmt:
  niexpr SEMI
  | RETURN niexpr_opt SEMI
  | LBRACE b_stmt_list RBRACE
  | IF LPAREN niexpr RPAREN b_stmt %prec NOELSE
  | IF LPAREN niexpr RPAREN b_stmt ELSE b_stmt
  | FOR LPAREN niexpr_opt SEMI niexpr SEMI niexpr_opt RPAREN b_stmt
  | WHILE LPAREN niexpr RPAREN b_stmt
  | RUNGAME LPAREN niexpr SEMI niexpr RPAREN b_stmt

expr_opt:
  /* nothing */
expr

niexpr_opt:
    /* nothing */
    | niexpr

eexpr:
    LITERAL
    | FLIT
    | BLIT
    | CLIT
    | ID
    | expr PLUS expr
    | expr MINUS expr
    | expr TIMES expr
    | expr DIVIDE expr
    | expr EQ expr
    | expr NEQ expr
    | expr LT expr
    | expr LEQ expr
    | expr GT expr
    | expr GEQ expr
    | expr AND expr
    | expr OR expr
    | expr MOD expr
    | MINUS expr %prec NOT
    | NOT expr
    | ID ASSIGN expr
    | ID LPAREN args_opt RPAREN
    | LPAREN expr RPAREN
    | NEW PLAYER LPAREN expr COMMA expr COMMA expr RPAREN
    | NEW ENTITY LPAREN expr COMMA expr COMMA expr RPAREN
    | ID DOT ID LPAREN args_opt RPAREN

niexpr:
    LITERAL
    | FLIT
    | BLIT
    | CLIT
    | ID
    | niexpr PLUS niexpr
    | niexpr MINUS niexpr
    | niexpr TIMES niexpr
    | niexpr DIVIDE niexpr
    | niexpr EQ niexpr
    | niexpr NEQ niexpr
    | niexpr LT niexpr
    | niexpr LEQ niexpr
    | niexpr GT niexpr
    | niexpr GEQ niexpr
    | niexpr AND niexpr
    | niexpr OR niexpr
    | niexpr MOD niexpr
    | MINUS niexpr %prec NOT
    | NOT niexpr
    | ID ASSIGN niexpr
    | ID LPAREN nargs_opt RPAREN
    | LPAREN niexpr RPAREN
    | ID DOT ID LPAREN nargs_opt RPAREN

args_opt:
    /* nothing */
    | args_list
args_list:
expr
| args_list COMMA expr

nargs_opt:
/* nothing */
| nargs_list

nargs_list:
niexpr
| nargs_list COMMA niexpr
3.11 Sample Code

Here is the code for running a simple game with an obstacle and a player that displays the player and entity as stationary textures.

```java
fun main void () {
    Entity rock;
    Player p;
    int x;
    initsdl();
    rock = new Entity(50, 50, "rock.png");
    p = new Player(75, 100, "player.png", 10);
    x = 100;
    p1. addPlayerControl(0, 81);
    runGame(x > 0; 1) {
        x = x - 1;
        rock.changeEntityX(-10);
    }
}
```

4 Project Plan

4.1 Planning

For most of the semester, our group met twice a week, with both meetings on Tuesdays. We would attend office hours with our TA Wonhyuk (Harry) Choi, and then have our group meeting afterwards on the same day. Attending office hours with Harry was a great opportunity to ask questions if we were struggling with a particular concept or bug. Meeting again as a group on the same day also allowed us to debrief what we learned during our TA meeting and come up with plans and action items for the following week.

In the last four to five weeks of the project, we increased our weekly meetings to include working meetings on Fridays and Sundays, in which available group members would pop in and out of zoom calls throughout the day as we each worked on our current compiler action item.

4.2 Testing

We were able to do full stack testing (excluding SDL) in the last five weeks of the course, because we began generating code shortly after completing the Hello World milestone. As we were collaborating on the same GitHub repository, group members would communicate about which features they were pushing, describe the tests that they had already written, and update the other members of the team before pushing.

After integrating SDL successfully into our language in the last three weeks of the course, we also shifted to visual testing for some of our test cases. This was for several reasons: Firstly, initializing SDL had to be done locally due to the fact that we needed to run an XServer in order for SDL to properly initialize while running one of our generated .exe files. Secondly, the testing had to be visual because of the way that entities and players moved across the graphics
window when affected by hardware input; since each execution could have a
different output, there was no way to reliably automate that.

If one team member struggled to debug something on their local branch, they
would create a new branch describing the error, so that other group members
would be able to provide more support to that person. Our "origin" branch was
reserved for code that was passing test cases or otherwise behaving as expected.

4.3 Style Guide

Our group used the following code style conventions:

1. Lines of code should not be longer than 80 characters
2. Each file only uses 4-space indentation
3. Use underscores for naming variables

4.4 Timeline

\begin{center}
\begin{tabular}{|c|l|}
\hline
Date & Description \\
Feb 23, 2021 & Submitted Project Proposal \\
Feb. 21, 2021 & First commit, creation of project repository \\
Feb. 24, 2021 & Completed LRM and Parser \\
Mar. 24, 2021 & First code generation (Hello World) \\
April 3rd, 2021 & Integrated SDL with Codegen \\
April 25th, 2021 & Project presentation and submission of report \\
\hline
\end{tabular}
\end{center}

4.5 Roles and Responsibilities

Our group did Zoom code reviews where members who had just implemented
features would demonstrate and run through their code, so that everyone in the
group would be up to date on the current "version" of the compiler.

Elliott Morelli: Code Generation, Semantic Checking, SDL Integration/ Envi-
ronment Testing (Language Guru)

Cherry Chu: Code Generation, Semantic Checking, Built-in object functional-
ity, Feature Testing (System Architect)

Robert Becker: Test suite maintenance, Unit tests (Tester)

Michael Fagan: Parsing, Scanner, Arithmetic Operators, Makefile and script
(Manager)

4.6 Software Development Environment

Our Stack:

1. Github- Hosted Git Repository
2. OCaml
3. GCC - for building the C output
4. C - for maintaining the "app" struct that stores the SDL_Renderer and SDL_Window, as well as C side "Entity" and "Player" structs
5. SDL2 - headers and libraries for output to cross-platform GPUs

Our testing environment:
For standard pass/fail tests, these were run in Docker using the testall.sh bash script.

Visual tests were compiled using Docker, with testall.sh, but then their .exes were run on Windows 10 (WSL) with an xServer (XLaunch Vcxsrv) running.

### 4.7 Project Log

Below is our project log with 130 commits, starting on February 21st and ending on April 25th.

<table>
<thead>
<tr>
<th>Commit</th>
<th>Author</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7304e2ad4650ce95fbf810489b2048f4494d1955</td>
<td>ElliottMorelli <a href="mailto:gmn2123@barnard.edu">gmn2123@barnard.edu</a></td>
<td>Sun Apr 25 12:07:23 2021</td>
<td>updated scanner and fixed testall</td>
</tr>
<tr>
<td>f4c9a5993b93c35a49225457bc40506feabb91a</td>
<td>ElliottMorelli <a href="mailto:gmn2123@barnard.edu">gmn2123@barnard.edu</a></td>
<td>Sun Apr 25 12:04:39 2021</td>
<td>Merge branch 'master' of <a href="https://github.com/ElliottMorelli/Parser">https://github.com/ElliottMorelli/Parser</a></td>
</tr>
<tr>
<td>09fc6471794e8116969c6ebeb9e45b77e7f2f4b4</td>
<td>ElliottMorelli <a href="mailto:gmn2123@barnard.edu">gmn2123@barnard.edu</a></td>
<td>Sun Apr 25 12:04:08 2021</td>
<td>accepted scanner change</td>
</tr>
<tr>
<td>4e7494c0f1e3b00ecf025ef75579a003f0bfa77</td>
<td>Michael Fagan <a href="mailto:mef2224@columbia.edu">mef2224@columbia.edu</a></td>
<td>Sun Apr 25 12:03:45 2021</td>
<td>Added spacebar as a scancode in the scanner</td>
</tr>
<tr>
<td>46208f46851c48e4782bd4f0ecfe3696202865c7</td>
<td>ElliottMorelli <a href="mailto:gmn2123@barnard.edu">gmn2123@barnard.edu</a></td>
<td>Sun Apr 25 12:01:00 2021</td>
<td>removed struct decls and signed some files</td>
</tr>
<tr>
<td>cd3ad99115151acff292f347fe0a6146950f37a</td>
<td>Michael Fagan <a href="mailto:mef2224@columbia.edu">mef2224@columbia.edu</a></td>
<td>Sun Apr 25 11:57:36 2021</td>
<td></td>
</tr>
</tbody>
</table>
Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 8a6732c8090be68cf05d3a7547f7b3e4752fb3bf
Author: Michael Fagan <mef2224@columbia.edu>
Date:  Sun Apr 25 11:57:32 2021 -0400

Modified parser to make it throw an error when initializing object in Rungame

commit e462ed10940f777aa95754c4971525b82b6e8a898
Author: ElliottMorelli <gnm2123@barnard.edu>
Date:  Sun Apr 25 10:16:27 2021 -0400

removed printbig

commit 7c3264e8e363a48ffcf7589761ab71116c010a530
Merge: 7db487b ae22196
Author: ElliottMorelli <gnm2123@barnard.edu>
Date:  Sun Apr 25 10:11:59 2021 -0400

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 7db487bcac72f4a84bb515f5647e9260ed947a76
Author: ElliottMorelli <gnm2123@barnard.edu>
Date:  Sun Apr 25 10:11:46 2021 -0400

removed printbig

commit ae22196e431266ad5e26ef9f877b0f335fbd63c0
Author: Rob Becker <rtbecker99@gmail.com>
Date:  Sun Apr 25 03:36:17 2021 -0400

finishing up tests

commit a40285c127061824eca89f62b6cdd53ae127f968
Author: Rob Becker <rtbecker99@gmail.com>
Date:  Sun Apr 25 03:07:51 2021 -0400

added testing for conditionals

commit 067b0f937c2594b482aadbb1138e12b38c6388f1
Author: Rob Becker <rtbecker99@gmail.com>
Date:  Sun Apr 25 02:09:48 2021 -0400

added testing on loops and common algos

commit 2be9ebf9333f2d1106247f82e3cab35b03dbd76
Author: ElliottMorelli <gnm2123@barnard.edu>
Date:  Sat Apr 24 23:47:17 2021 -0400

changed to lucifer.native, finalized style

commit 238653c993f08391914d833c78c244a13244f00b9
Merge: 7a6c226 f32d896
Author: ElliottMorelli <gnm2123@barnard.edu>
Date:  Sat Apr 24 23:30:20 2021 -0400

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 7a6c2261bfb0c5462a203614c4b6b17a854b
Author: ElliottMorelli <gnm2123@barnard.edu>
Date:  Sat Apr 24 23:30:04 2021 -0400

style and removing comments

commit f32d8964a7a8c16675765331f8c1d44e08c76f
Author: Cherry Chu <ccc2207@columbia.edu>
Date:  Sat Apr 24 23:17:36 2021 -0400
match coding style.
commit 20a38bd46d9fc8d56f080df1b87d7a2ad39ff4d
Merge: 982cb87 8dd9991
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 23:15:01 2021 -0400
    Merge branch 'master' of https://github.com/ElliottMorelli/Parser
commit 982cb877b52d7ea144d3f6f51126dfcfff87219a6
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 23:14:40 2021 -0400
codegen style and comments
commit 8dd99142cdc1fa56f2b77fe14ad87e79d863f9b
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 24 23:06:38 2021 -0400
    added single line comment and test.
commit 8201869f9ba2781975f90031c6082eab5ddada3
Merge: 041697c 4704633
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 24 22:40:09 2021 -0400
    merge master
commit 041697ce48dbe06135e4771836a680837d96df
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 24 22:39:22 2021 -0400
    change sdl tests to visual and produce only exe.
commit 4704633f846555815292a9c239ef36eeac6a6a
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 21:42:47 2021 -0400
    updated initsdl() to take 0 args
commit 5f77a68b5e97fa9a5d0f71861ee2ad201cdd911
Merge: 5661988 8ee488d3
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 20:54:16 2021 -0400
    Merge branch 'master' of https://github.com/ElliottMorelli/Parser
commit 5661988e74f2ee1f94e2a384c7b05bff4b7a3
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 20:53:59 2021 -0400
    fixing llvm alloc for player and working on style
commit 8a38d320c64d5e6821304636a2d1d5fb2b0d4663
Merge: f6b3f6b 20ba090f
Author: Rob Becker <rtbecker9@gmail.com>
Date: Sat Apr 24 18:16:48 2021 -0400
    Merge branch 'master' of https://github.com/ElliottMorelli/Parser
commit f6b3f6b3b68b6b6a0c690ea7293600c4ed3e3674
Author: Rob Becker <rtbecker9@gmail.com>
Date: Sat Apr 24 18:16:42 2021 -0400
    added more function testing
commit 20ba090f7010774f51d95e74fa39beba6c0188d3
Removed pattern-match warning on Mod

commit 0 c2b2897863d1d0d1531422257de48e785c315dc
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 18:04:41 2021 -0400
Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 7ce54b15115721495c7d2b99244df9ee57f59079
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 17:00:00 2021 -0400

code cleanup and citation

commit e27ae188bd1fd9f1d4e23b06c8ff82709629a7ad3
Merge: c7a36e4 68dd512
Author: Rob Becker <rtbecker99@gmail.com>
Date: Sat Apr 24 14:47:00 2021 -0400
Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit c7a36e49790f7dda102b7488f7ebd9367f5c4615
Author: Rob Becker <rtbecker99@gmail.com>
Date: Sat Apr 24 14:46:43 2021 -0400

temporary fix for current mc tests

commit 68dd512be0c6b37b4457d9410847306947872963
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 13:09:55 2021 -0400
updated demo program to .luc, code cleanup

commit 538d32b92115cc021db5f5446a33520666d06e
Merge: 340b2e9 5f8e313
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 12:58:40 2021 -0400
fixing conflict

commit 340b2e9560d258227f930d9aa9dbccf938b8ac7db
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 12:56:57 2021 -0400
finished demo program

commit 1a3d46800b03c4807e66e9347607109e7cb7bb6c
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 24 12:14:14 2021 -0400

added setplayerx and setplayery

commit 5f8e313427a9eb893ba1902e3c842d461bdeen
Author: Rob Becker <rtbecker99@gmail.com>
Date: Sat Apr 24 00:42:42 2021 -0400
renamed all .mc files to .luc, all current tests pass in new shell script

commit fec2ef3cd1bc1fa5a4c888947b19de3998947c1
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Fri Apr 23 20:38:56 2021 -0400

updated semantic error
commit 74977dad785729dfc6c0605538150d38346ffbf
Author: Michael Fagan <mef2224@columbia.edu>
Date: Fri Apr 23 16:17:10 2021 -0400

changed target of testall.sh from .mc files to .luc

commit 69d24a25655e0501ac3bd5518f779e324d171ef2
Merge: 805af65 3682a0e
Author: Michael Fagan <mef2224@columbia.edu>
Date: Fri Apr 23 16:55:44 2021 -0400

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 3682a0e7d470113e123bb68ace9cf46f55ade9
Merge: 06af25d a06cfe6
Author: ElliottMorelli <gmn2123@barnard.edu>
Date: Fri Apr 23 12:00:51 2021 -0400

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 06af25df91aca6aa0586d622c3714dbaa5c19cd1
Author: ElliottMorelli <gmn2123@barnard.edu>
Date: Fri Apr 23 12:00:31 2021 -0400

added err message in semant for uninitialized obj

commit a06cfe6e00e1189ba14a1e99e9c807b8bc303abe
Author: Rob Becker <rtbecker99@gmail.com>
Date: Thu Apr 22 20:41:52 2021 -0400

first batch of function tests

commit 0e748224344ec5bbc6a8dbab08c22b0acff8869
Author: Rob Becker <rtbecker99@gmail.com>
Date: Thu Apr 22 19:24:07 2021 -0400

added tests on basic arith

commit 71f88848e2eb019818f2b8b2a7642227e06c262915
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Thu Apr 22 16:22:29 2021 -0400

updated player tests control array to minimum size 4

commit d07d4a403db64b5bc35abed50086db1bdced3
Author: ElliottMorelli <gmn2123@barnard.edu>
Date: Thu Apr 22 16:18:52 2021 -0400

updated err msg for cntrl arr size

commit 173fec87a61f8381e363b1cbda6263ae04fe28e20
Author: ElliottMorelli <gmn2123@barnard.edu>
Date: Thu Apr 22 15:56:29 2021 -0400

finished built in up down left right player controls

commit fd7ec65ccf9bd780e8b003cda304a0bb577b6bf
Merge: 703fd35 b631c6a
Author: ElliottMorelli <gmn2123@barnard.edu>
Date: Thu Apr 22 14:34:56 2021 -0400

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 703fd353f88203adeceadc87556a2a7234a3a82
Merge: c515eb3 291cda
Author: ElliottMorelli <gmn2123@barnard.edu>
Date: Thu Apr 22 14:32:12 2021 -0400
player update needs to be merged with allowing negative input Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 86d1c6a20d788c150f9cc0f77f7a8dfb8cad304
Merge: c464117 291cdae
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Thu Apr 22 14:32:08 2021 -0400

merged master.

commit c515eb313fc17259f3b81ce8fa200d452b82f445
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Thu Apr 22 14:31:32 2021 -0400

added player rendering

commit c4641174267e330430f4ec14c3d228ae1bd3670
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Thu Apr 22 14:30:41 2021 -0400

add entity semant and tests.

commit 291cdae12553566907b9b96255db43e23a8b06ce
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Thu Apr 22 11:49:25 2021 -0400

added failed tests for player/entity objects and object function calls.

commit 8607490bd98db1c1f0b5c77b11af0d9fc66f9b34
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Wed Apr 21 20:40:28 2021 -0400

updated entity zeroing to match new arguments

commit 59a171c73571baff05bc11413fa0b6fcd11fcb8
Merge: 152de98 07527fc
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Wed Apr 21 20:36:36 2021 -0400

zeroed entities to prevent segfault Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 152de98c0da55ed8054674e97c52820419664281
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Wed Apr 21 20:36:04 2021 -0400

zeroed uninitialized entities in initsdl()

commit 07527fc8925d51953dd4c4f059ced63d48190
Merge: 0f7262c 7aabd74
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Apr 21 18:03:25 2021 -0400

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 0f7262ca47d822898602e4cc1e6bcdf7451da50b
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Apr 21 18:03:13 2021 -0400

modified entity and player ast type and add rules in semant.ml for player.

commit 7aabd7422094db09ac99dc350c6c5a932fc809bc2
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Wed Apr 21 15:46:13 2021 -0400

testing undeclared entities
commit 805af65a10ecce65d53903b555a95139210e138
Author: Michael Fagan <mef2224@columbia.edu>
Date: Wed Apr 21 12:05:20 2021 -0400

Added modulo operator with test

commit 6f6205bcbcb535e363703e1a5c976239fe9e1391
Author: ElliottMorelli <gnn2123@barnard.edu>
Date: Tue Apr 20 22:09:39 2021 -0400

added test program arrow keys bat

commit e1ce4a97a507a28dd4aa53383204f6ea342869d66
Author: ElliottMorelli <gnn2123@barnard.edu>
Date: Tue Apr 20 22:03:27 2021 -0400

fixed white square rendering problem by adding init function

commit f0be4ac1c845819f6464f1e08d4d269bef1e4c7f
Merge: 5dd5b66 aa64ccc6
Author: ElliottMorelli <gnn2123@barnard.edu>
Date: Tue Apr 20 17:02:02 2021 -0400

adding tests Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 5dd5b664b05ff501ce025042e723464ae510d2
Author: ElliottMorelli <gnn2123@barnard.edu>
Date: Tue Apr 20 17:13:32 2021 -0400

testing sdl init and rendering

commit aa64ccc685cb44c83f38e06145231ff2b6ebd6
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Apr 20 15:15:41 2021 -0400

merged master.

commit 66712a1a95f1fa62e0f05bd95b54fad0
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Apr 20 15:06:27 2021 -0400

added user defined struct feature.

commit 1bafe34a19a52070666c3294c70c82e6fff8c
Author: ElliottMorelli <gnn2123@barnard.edu>
Date: Mon Apr 19 15:45:15 2021 -0400

integrated sdl rendering and keyboard input

commit cc76085816fe23b861e73140145e60b1c1216d
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 17 14:52:24 2021 -0400

add arrow key SCANCODEs and tests.

commit c69417a304575fe9bdaa851e776956c1c69
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 17 14:35:04 2021 -0400

fixed player control array allocation; added addPlayerControl()

commit b0978d5349e44777404701d54ac24150d1801236
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Apr 14 10:28:33 2021 -0400

set control size to 2.
commit 1b11ec9cae7eed4f3532bc18da3a9c42a3c1c839
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Apr 14 10:14:50 2021 -0400

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 376c120bc9df17f7773a8225bc2986d2f9e4ae
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Apr 14 10:14:46 2021 -0400

change control array size to 5.

commit 94de0fda2542562be1fe00a9fad59da26fb104e
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Wed Apr 14 09:46:16 2021 -0400

setup rungame() to take entities

commit c47f29b7ab865a2467088a4317d8d763587488e
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Tue Apr 13 10:34:04 2021 -0400

testing new fns Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 3bbf11c60b5c80fa75a5648c406f6881930c6901
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Apr 13 09:48:04 2021 -0400

added Player object and controls array.

commit 087c15587783d3f4e5089e4673f1b4e7e8ac31
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 10 23:19:26 2021 -0400

Added entity functions addHitBox() and changeXY(). Added tests.

commit 6d55fe60db327ad2266c63b197cbb5455272c71a
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 10 22:13:29 2021 -0400

needed to get code to test getX() Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit e0b59365eb6d078a4c9f66e4cedf103ef69dc88ec
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 10 22:10:18 2021 -0400

committing broken codegen rungame()

commit 9b64fae4c1cf4b01453062a7bf134aa1b8b035e
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 10 21:13:24 2021 -0400

passed test-entity without seg fault.

commit 39a790052bb35a3a947ff6310c411c9f274ef7d1
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sat Apr 10 16:41:19 2021 -0400

added entity object and tests.

commit 9b7a333a4803d4a49ccf873c0a52985bad699e4
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Fri Apr 9 23:20:38 2021 -0400
added getX() test.

commit e1e942137455294c8acff14e8714ea3c1c8c04c7
Merge: 5c7b183 786e421
Date: Fri Apr 9 23:19:34 2021 -0400
Author: Cherry Chu <ccc2207@columbia.edu>

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 5c7b1832ea6e7147ac97d101481cf07c74bd3582
Date: Fri Apr 9 23:19:24 2021 -0400
Author: Cherry Chu <ccc2207@columbia.edu>

added object member call getX().

commit 786e42128d8c466bcb5f2c13e0b4cb0bd08ddede
Merge: 12cc59c Sa3f789
Date: Fri Apr 9 16:30:12 2021 -0400
Author: ElliottMorelli <gnm2123@barnard.edu>

Merge branch 'master' of https://github.com/ElliottMorelli/Parser

commit 12cc59c9ad7b783e0d0a9b0253344d91ef3a93a
Date: Fri Apr 9 16:29:52 2021 -0400
Author: ElliottMorelli <gnm2123@barnard.edu>
created object map

commit 5a3f7891e74d09341f8aadd5098da2d01b7b1d97
Date: Fri Apr 9 16:27:18 2021 -0400
Author: Cherry Chu <ccc2207@columbia.edu>
moved tests.

commit f3fdbc9539a6dae4f53a3f3dfed31af965d5f891
Date: Fri Apr 9 16:13:54 2021 -0400
Author: Cherry Chu <ccc2207@columbia.edu>
added print variable and object construction tests.

commit 925ba38fcb83c14777b8c16d4f0784c2f7c810
Merge: 72f98dd a29b3dd
Date: Fri Apr 9 15:25:39 2021 -0400
Author: ElliottMorelli <gnm2123@barnard.edu>
resolved microcparse.mly conflict

commit 72f98ddf1429df41a5aa284c07e6e0be743261c
Date: Fri Apr 9 15:15:19 2021 -0400
Author: ElliottMorelli <gnm2123@barnard.edu>
working on runGame() in progress

commit bc4de7a5b3f39024bf5ae84b158bf61fd82a94f3
Date: Fri Apr 9 15:14:09 2021 -0400
Author: ElliottMorelli <gnm2123@barnard.edu>
added runGame() in progress

commit a29b3dda69d62ca6a75daf6a812d946dbf5e8fa
Date: Fri Apr 9 15:10:41 2021 -0400
Author: Cherry Chu <ccc2207@columbia.edu>
added getX function.

commit 8129df65a1a3d15c75fce580ef834c34148ee3

43
added player object type
commit 8051637a9a5a600434eescf8c864143a6931e16c6
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Thu Apr 8 16:24:37 2021 -0400
add Player object type.
commit 4f969662f307af3763be7baa1eacb2633218f65b5
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Apr 3 13:17:23 2021 -0400
integrated sdl to compiler and linked in testall
commit be75e3a710b05a3208f474d2623392df41e18f16
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Sat Mar 27 15:43:05 2021 -0400
created sdl install script and make sdl command
commit 1b18eecc630cf05d202e9eb6e5dcd35364626c7f3
Author: ElliottMorelli <gnm2123@barnard.edu>
Date: Wed Mar 24 16:32:40 2021 -0400
added four additional tests
commit 8c6ebbdd747ababab99ca55df77cb943f20da5
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Thu Mar 25 00:18:05 2021 +0800
cell main to return int in the background.
commit 493345675f6dfaf7bc467c88e2e2cc662f965d7781
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Mar 24 23:48:16 2021 +0800
Add string type and change void type to return 0
commit 88627ab2b60d3088be8b4f39f9a40c5a9143bed
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Mar 24 11:41:27 2021 +0800
added string type.
commit 3d8fc6bce85e6e5d227f962976b7cf40b9a770
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Mar 24 05:00:45 2021 +0800
change main func to return void.
commit bc3c664154399a70e419e9f7f9f9b28bd46dc5807
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Thu Mar 18 23:24:40 2021 +0800
added hello world implementation.
commit a11ce08ba874f57bfebda96bad30f08c4355f16
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Mon Mar 15 00:11:44 2021 +0800
modified if function in ast.ml.
commit 845740fb5c2d3ab72ea9002b6c75a149e2017d2
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sun Mar 14 23:43:03 2021 +0800
added elif to ast.ml

commit 38addb7e25c6407a335da50027daee88ab61c5db2
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Sun Mar 14 19:09:21 2021 +0800

moved type class_def to resolve unbound error.

commit d2cb66df8fba74828412231ccf48877b84026851
Author: ElliottMorelli <gmn2123@barnard.edu>
Date: Sat Mar 13 11:32:32 2021 -0500

adding classes to ast

commit 873625334d31dd1e12b2f955e7bc1202a86ea16
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 21:31:32 2021 -0500

Array Instantiation and Obj instantiation

commit 5736fba58964e79a9e177aad0d4fda5ed3c3
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 20:10:56 2021 -0500

Added mod to parser

commit 00f1c7e5dcd02bb8a6b95e01c41462052a60519c
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 20:00:37 2021 -0500

Removed Shift reduce from assignments

commit 8204a3db6783b11097be3b8b4c10c0d0d4c2bee6
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 19:37:50 2021 -0500

Reworked a lot to support assignments better

commit 9bc7bc961eedefe67e8a6b26f19548c8caf92d6
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 15:51:09 2021 -0500

Changed how variable declaration and assignment were related

commit af8f680860a5ea26b245b19f7b6db508e9674
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 15:35:00 2021 -0500

Removed cnstr shift reduct errors

commit d896a273492097fe3360bb2b358f8fa2f13ed135
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 15:24:55 2021 -0500

Added some shift reduce errors for fun

commit 87adce5b12bb690e04466f1d76ce7a1b502fc5d8
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 14:52:17 2021 -0500

Added comments and fixed error in instantiation

commit 94c5f0044128db13f61f7af275c4a252ed2e2a0e
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 14:51:51 2021 -0500

Added comments and fixed error in instantiation
commit deb15af5f86b47f7b58426424c0f466431422d3
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 14:45:49 2021 -0500

Fixed shift reduce errors from object instantiation

commit 307c6b5f54dd95778c36707ef3d6993ca196a96f
Merge: e0b5530 4de9e95
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 13:27:12 2021 -0500

Fixed merge conflicts again

commit e0b5530edf799dd87086ad3ec75b11efa848e5b0
Merge: d4c7408 317bb71
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Wed Feb 24 13:23:22 2021 -0500

Fixed the merge conflicts

commit 4de9e95f96330d45f3535e710059024c4f61b8
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Wed Feb 24 11:19:48 2021 +0800

Added rungame grammar.

commit d4c740858436d3907b3874464ef99a85c810c1 (master/michael-edits)
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Tue Feb 23 20:09:51 2021 -0500

Fixed object instantiation

commit aad1635b773dd0d624f658e1abbe60cf051ad5f1d
Author: Michael Fagan <Erin@Erins-MacBook-Air.local>
Date: Tue Feb 23 19:56:45 2021 -0500

Added Object instantiation and started method calls

commit 317bb71b4986b8422a6e564a824ca3d5373b479
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Feb 23 18:57:45 2021 +0800

Added object member access.

commit 85003a50cd6e0519720851804d33a5b5cfbc43e
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Feb 23 17:37:00 2021 +0800

Add object instantiation grammar with keyword NEW.

commit 8c8c39989fe6e11962ec6a33fd04e2888740f5
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Feb 23 15:08:46 2021 +0800

Added Array type.

commit 6f7f6e6d1c1878fa0605b7bafc994704be4f3c9ee
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Feb 23 14:32:00 2021 +0800

Added array access.

commit 6e16918b4164c613a595041e28a72fa3113ba28e
Author: Cherry Chu <ccc2207@columbia.edu>
Date: Tue Feb 23 14:30:48 2021 +0800

reformatted.
5 Architectural Design

5.1 Block Diagram

5.2 Compiler

The Lucifer Compiler operates in 5 stages: scanning, parsing, semantic checking, code generation, and linking.
5.2.1 Scanner
The Lucifer scanner takes the input code and creates a stream of objects of type token. The Scanner also converts the alphanumeric, SPACE, and directional SDL-scancodes directly into their integer literal counterparts.

5.2.2 Parser and AST
The Lucifer Parser takes the stream from the scanner uses it to construct the Abstract Syntax Tree (AST) using the unambiguous grammar we wrote. It accomplishes this using ocamlyacc to construct and execute an SLR parsing table from our provided grammar. Additionally the parser is responsible for throwing an error if an object instantiation is made inside a Run game loop.

5.2.3 Semantic Checking
The Semantic checker for Lucifer takes the AST from the parser and recursively generates a Semantically-checked AST (SAST). In semantic checking, the compiler must verify the validity of the Abstract Syntax Tree by checking that expressions on either side of binary operators evaluate to the same type, ensuring that variables have been declared if they are used, ensuring that variables are in function scope (using lists for local function variables), and that their types are correct. Additionally, semantic checking checks that all built-in object instantiations are type-correct, and checks against a hashtable to ensure that all built in objects are instantiated after they are declared. This is to ensure proper rendering when linking with SDL.

5.2.4 Code Generation
The code generation for Lucifer is done via LLVM IR. It does so by traversing the SAST and matching the Lucifer types to LLVM types. It also replaces the Lucifer operators with LLVM operators. SCall makes build calls to C functions at runtime, by checking that the function names match up and linking to their respective functions.

5.2.5 Linking with C (Runtime) and integrating SDL functionality
Citation: When integrating SDL, we use some C source code from https://www.parallelrealities.co.uk/ such as their logic for maintaining an "app" struct, maintaining a keyboard array, and their prepareScene(), presentScene(), and blit() functions which we use in our own C source to allow for texture rendering. These files can be found in the /src/ folder in our source code.

When .luc files are compiled by Lucifer.native, any global functions that link with SDL are accessed explicitly through function calls (whose names are stored in a StringMap in semant.ml), or implicitly called by the rungame() loop, whose implementation is in codegen.ml. Codegen uses LLVM build_call to access C
functions by name. These C functions use SDL at runtime. Linking the Lucifer compiler to the C code that integrates SDL is done in the test script.

5.3 Architecture Design Responsibility

Elliott worked on linking with C and integrating SDL functionality, code generation for the `runGame()` loop, and semantic checking for object instantiation. They also helped work on code generation for Players and Entities. Cherry worked on built-in object functionality, feature testing, code generation and semantic checking for objects. Michael worked on writing the context-free grammar and the scanner, as well as some semantic checking and code generation.

6 Testing Plan

6.1 Source To Target

Source program 1

```c
fun main void () {
    Player p;
    int i;
    p = new Player(0, 0, "image.png", 10);
    i = 0;
    while (i < 10) {
        p.changePlayerX(2);
        i = i + 1;
        print(p.getPlayerX());
    }
}
```

Target program 1

```c
; ModuleID = 'Lucifer'
source_filename = "Lucifer"
%entity_t = type { i32, i32, i8*, i8*, i32, i32 }
%player_t = type { i32, i32, i8*, i32, i32, i32*, i8* }
@fmt = private unnamed_addr constant [4 x i8] c "%d\0A\00"
@fmt.1 = private unnamed_addr constant [4 x i8] c "%s\0A\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c "%g\0A\00"
@str = private unnamed_addr constant [10 x i8] c "image.png\00"
@str.3 = private unnamed_addr constant [10 x i8] c "image.png\00"
declare i32 @printf(i8*, ...)
declare i32 @initsdl(i32, ...)
declare i32 @initEntity(%entity_t*, ...)
declare i32 @initPlayer(%player_t*, ...)
declare i32 @prepareScene(i32, ...)
declare i32 @presentScene(i32, ...)
declare i1 @isKeyPressed(i32, ...)
declare i32 @updateE(%entity_t*, ...)
```
declare i32 @updateP(%player_t*, ...)  
declare i32 @printbig(i32)  
declare i32 @getPlayerX(%player_t*, ...)  
declare i32 @getPlayerY(%player_t*, ...)  
declare i32 @setPlayerX(%player_t*, i32, ...)  
declare i32 @setPlayerY(%player_t*, i32, ...)  
declare i32 @getPlayerControl(%player_t*, i32, ...)  
declare i32 @zeroPlayerControls(%player_t*, ...)  
declare i32 * @getPlayerText(%player_t*, ...)  
declare i32 @getPlayerHx(%player_t*, ...)  
declare i32 @getPlayerHy(%player_t*, ...)  
declare i32 @addPlayerHitBox(%player_t*, i32, i32, ...)  
declare i32 @changePlayerX(%player_t*, i32, ...)  
declare i32 @changePlayerY(%player_t*, i32, ...)  
declare i32 @addPlayerControl(%player_t*, i32, i32, ...)  
declare i32 @controlPlayer(%player_t*, i32, ...)  
declare i32 @getEntityX(%entity_t*, ...)  
declare i32 @getEntityY(%entity_t*, ...)  
declare i32 * @getEntityText(%entity_t*, ...)  
declare i32 @getEntityHx(%entity_t*, ...)  
declare i32 @getEntityHy(%entity_t*, ...)  
define i32 @main() {  
  entry:  
    %p = alloca %player_t*  
    %l = alloca i32  
    %player_tap = alloca %player_t  
    %player_ptr = alloca %player_t*  
    %x = getelementptr inbounds %player_t, %player_t* %player_tap, i32 0, i32 0  
    store i32 0, i32 * %x  
    %y = getelementptr inbounds %player_t, %player_t* %player_tap, i32 0, i32 1  
    store i32 0, i32 * %y  
    %texture = getelementptr inbounds %player_t, %player_t* %player_tap, i32 0, i32 2  
    store i8* getelementptr inbounds ([10 x i8], [10 x i8]* @str, i32 0, i32 0) , i8** %texture  
    %hx = getelementptr inbounds %player_t, %player_t* %player_tap, i32 0, i32 50
store i32 0, i32 * %hx
%hy = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 4
store i32 0, i32 * %hy
%n = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 5
%malloccall = tail call i8* @malloc (i32 add (i32 mul (i32 ptrtoint (i32 getelementptr (i32, i32* null, i32 1) to i32), i32 10), i32 1), i32 1)
%tmp = bitcast i8* %malloccall to i32*
store i32* %tmp, i32** %n
store %player_t** %player_tmp, %player_t** %player_ptr
%zeroPlayerControls = call i32 (%player_t*, ...) @zeroPlayerControls(%player_t* %player_tmp)
%text = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 6
store i8* getelementptr inbounds ([10 x i8], [10 x i8]* @str.3, i32 0, i32 0), i8** %text
%0 = load %player_t*, %player_t** %player_ptr
store %player_t** %0, %player_t** %p
store i32 0, i32* %i
br label %while
while:
; preds = %while_body, %entry
%15 = load i32, i32* %i
%tmp6 = icmp slt i32 %15, 10
br i1 %tmp6, label %while_body, label %merge
while_body:
; preds = %while
%p1 = load %player_t*, %player_t** %p
%changePlayerX = call i32 (%player_t*, i32, ...) @changePlayerX(%player_t* %p, i32 2)
%i2 = load i32, i32* %i
%tmp3 = add i32 %i2, 1
store i32 %tmp3, i32* %i
%p4 = load %player_t*, %player_t** %p
%getPlayerX = call i32 (%player_t*, ...) @getPlayerX(%player_t* %p4)
%printf = call i32 (i8*, ...) @printf (i8* getelementptr inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %getPlayerX)
br label %while
merge:
; preds = %while
ret i32 0
}
declare noalias i8* @malloc(i32)

We chose the test this program because it shows the the basic usage of built-in object declaration and instantiation. It also shows how functions are being called alone and on objects.

Source program 2

fun checkCollision bool (Entity e, Player p){
  int pRightX;
  int pDownY;
  int eRightX;
  int eDownY;

  pRightX = p.getPlayerX() + p.getPlayerHx();
  pDownY = p.getPlayerY() + p.getPlayerHy();
  eRightX = e.getEntityX() + e.getEntityHx();

}
eDownY = e.getEntityY() + e.getEntityHy();

if(p.getPlayerX() == pRightX || p.getPlayerY() == pDownY || e.getEntityX() == eRightX || e.getEntityY() == eDownY)
    return false;
}
if(p.getPlayerX() >= eRightX || e.getEntityX() >= pRightX)
    return false;
if(p.getPlayerY() >= eDownY || e.getEntityY() >= pDownY)
    return false;
return true;
}

fun main void () {

    Player knight;
    Entity e;
    Entity e2;
    Entity e3;
    Entity winscreen;

    int lives;
    bool down;
    bool collide;
    bool running;

    collide = false;
    down = true;
    running = true;

    initsdl();
    lives = 3;

    knight = new Player(0, 200, "gfx/knight.png", 4);
    knight.addPlayerControl(0, SDL_SCANCODE_UP);
    knight.addPlayerControl(1,81);
    knight.addPlayerControl(2,80);
    knight.addPlayerControl(3,79);

    e = new Entity(300, 0, "gfx/bat.png");
    e2 = new Entity(550, 0, "gfx/bat.png");
    e3 = new Entity(800, 0, "gfx/bat.png");

    winscreen = new Entity(-500, -500, "gfx/youwin.png");

    runGame(running; 60) {

        knight.addPlayerHitBox(80, 80);
        e.addEntityHitBox(100, 100);
        e2.addEntityHitBox(120, 120);
        e3.addEntityHitBox(120, 120);

        collide = (checkCollision(e, knight) || checkCollision(e2, knight) ||
                   checkCollision(e3, knight));

        knight.controlPlayer(15);

        if(e.getEntityY() >= 700){

        }
down = false;
}

if(e.getEntityY() <= 5) {
    down = true;
}

if(down == true){
    e2.changeEntityY(15);
    e.changeEntityY(25);
    e3.changeEntityY(35);
}

if(down == false) {
    e2.changeEntityY(-15);
    e.changeEntityY(-25);
    e3.changeEntityY(-35);
}

/*@ends the knight back to the beginning @*/
if(collide){
    lives = lives - 1;
    knight.setPlayerX(0);
    knight.setPlayerY(200);
}

/*@ends game if out of lives */
if(lives <= 0){
    print("Game Over");
    running = false;
}

/*@shows win screen*/
if(knight.getPlayerX()+knight.getPlayerHx() >= 1200){
    winscreen.setEntityX(200);
    winscreen.setEntityY(100);
}

}

}
@fmt .14 = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@fmt .15 = private unnamed_addr constant [4 x i8] c"%g\0A\00"

declare i32 @printf (i8*, ...)
declare i32 @initsdl(i32, ...)
declare i32 @initEntity(%entity_t*, ...)
declare i32 @initPlayer(%player_t*, ...)
declare i32 @prepareScene(i32, ...)
declare i32 @presentScene(i32, ...)
declare i1 @isKeyPressed(i32, ...)
declare i32 @updateE(%entity_t*, ...)
declare i32 @updateP(%player_t*, ...)
declare i32 @getPlayerX(%player_t*, ...)
declare i32 @getPlayerY(%player_t*, ...)
declare i32 @setPlayerX(%player_t*, i32, ...)
declare i32 @setPlayerY(%player_t*, i32, ...)
declare i32 @getPlayerControl(%player_t*, i32, ...)
declare i32 @zeroPlayerControls(%player_t*, ...)
declare i32 @getPlayerText(%player_t*, ...)
declare i32 @getPlayerHx(%player_t*, ...)
declare i32 @getPlayerHy(%player_t*, ...)
declare i32 @addPlayerHitBox(%player_t*, i32, i32, ...)
declare i32 @changePlayerX(%player_t*, i32, ...)
declare i32 @changePlayerY(%player_t*, i32, ...)
declare i32 @addPlayerControl(%player_t*, i32, i32, ...)
declare i32 @controlPlayer(%player_t*, i32, ...)
declare i32 @getEntityX(%entity_t*, ...)
declare i32 @getEntityY(%entity_t*, ...)
declare i32 @getEntityText(%entity_t*, ...)
declare i32 @getEntityHx(%entity_t*, ...)
declare i32 @getEntityHy(%entity_t*, ...)
declare i32 @setEntityX(%entity_t*, i32, ...)
declare i32 @setEntityY(%entity_t*, i32, ...)
declare i32 @addEntityHitBox(%entity_t*, i32, i32, ...)
declare i32 @changeEntityX(%entity_t*, i32, ...)
declare i32 @changeEntityY(%entity_t*, i32, ...)
declare i32 @changeEntityX(%entity_t*, i32, ...)
declare i32 @changeEntityY(%entity_t*, i32, ...)

54
define i32 @main() {  
entry:  
  %knight = alloca %player_t*  
  %e = alloca %entity_t*  
  %e2 = alloca %entity_t*  
  %e3 = alloca %entity_t*  
  %winscreen = alloca %entity_t*  
  %lives = alloca i32  
  %down = alloca i1  
  %collide = alloca i1  
  %running = alloca i1  
  store i1 false, i1* %collide  
  store i1 true, i1* %down  
  store i1 true, i1* %running  
  %initSDL = call i32 (i32, ...) @initSDL(i32 0)  
  store i32 3, i32* %lives  
  %player_tmp = alloca %player_t  
  %player_ptr = alloca %player_t*  
  %x = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 0  
  store i32 0, i32* %x  
  %y = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 1  
  store i32 200, i32* %y  
  %texture = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 2  
  store i8* getelementptr inbounds ([15 x i8], [15 x i8]* @str, i32 0, i32 0), i8** %texture  
  %hx = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 3  
  store i32 0, i32* %hx  
  %hy = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 4  
  store i32 0, i32* %hy  
  %n = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 5  
  %malloccall = tail call i8* @malloc(i32 add(i32 mul(i32 %0 to i32), i32 4), i32 1), i32 1) to i32*  
  store i32* %malloccall to i32*  
  store %player_t* %player_ptr  
  %zeroPlayerControls = call i32 (%player_t*, ...) @zeroPlayerControls(%player_t* %player_ptr)  
  %text = getelementptr inbounds %player_t, %player_t* %player_tmp, i32 0, i32 6  
  store i8* getelementptr inbounds ([15 x i8], [15 x i8]* @str.3, i32 0, i32 0), i8** %text  
  %0 = load %player_t**, %player_t** %player_ptr  
  store %player_t* %0, %player_t** %knight  
  %knight1 = load %player_t**, %player_t** %knight  
  %addPlayerControl = call i32 (%player_t*, %knight1, i32 2, i32 81)  
  %knight2 = load %player_t*, %player_t** %knight  
  %addPlayerControl3 = call i32 (%player_t*, %knight2, i32 1, i32 82)  
  %knight3 = load %player_t*, %player_t** %knight  
  %addPlayerControl5 = call i32 (%player_t*, %knight3, i32 2, i32 80)  
  %knight4 = load %player_t*, %player_t** %knight  
  %addPlayerControl7 = call i32 (%player_t*, %knight4, i32 3, i32 79)  
  %entity_tmp = alloca %entity_t*  
  %entity_ptr = alloca %entity_t*  
  %x8 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp, i32 0, i32 0  
  store i32 300, i32* %x8  
  %y9 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp, i32 0, i32 1  
  store i32 0, i32* %y9
%entity_ptr31 = alloca %entity_t *
% x32 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp30, i32 0, i32 0
store i32 -500, i32* % x32
% y33 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp30, i32 0, i32 1
store i32 -500, i32* % y33
% texture34 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp30, i32 0, i32 2
store i8* getelementptr inbounds ([15 x i8], [15 x i8]* @str.10, i32 0, i32 0), i8** % texture34
% text35 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp30, i32 0, i32 3
store i8* getelementptr inbounds ([15 x i8], [15 x i8]* @str.11, i32 0, i32 0), i8** % text35
%hx36 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp30, i32 0, i32 4
store i32 0, i32* %hx36
%hy37 = getelementptr inbounds %entity_t, %entity_t* %entity_tmp30, i32 0, i32 5
store i32 0, i32* %hy37
store %entity_t* %entity_tmp30, %entity_t** %entity_ptr31
%4 = load %entity_t*, %entity_t** %entity_ptr31
store %entity_t* %4, %entity_t** % winscreen
% initEntity = call i32 (%entity_t*, ...) @initEntity (%entity_t* % winscreen38)
% e339 = load %entity_t*, %entity_t** %e3
% initEntity40 = call i32 (%entity_t*, ...) @initEntity (%entity_t* % e339)
% e241 = load %entity_t*, %entity_t** %e2
% initEntity42 = call i32 (%entity_t*, ...) @initEntity (%entity_t* % e241)
% e43 = load %entity_t*, %entity_t** %e
% initEntity44 = call i32 (%entity_t*, ...) @initEntity (%entity_t* % e43)
% knight45 = load %player_t*, %player_t** %knight
% initPlayer = call i32 (%player_t*, ...) @initPlayer (%player_t* %knight45)
% winscreen46 = load %entity_t*, %entity_t** % winscreen
% initEntity47 = call i32 (%entity_t*, ...) @initEntity (%entity_t* % winscreen46)
br label %while
while:
  ; preds = %merge121, %entry
  %Xrunning126 = load i1, i1* %Xrunning
  br i1 %Xrunning126, label %while_body, label %merge127
while_body:
  ; preds = %while
  % prepareScene = call i32 (i32, ...) @prepareScene(i32 0)
  % winscreen48 = load %entity_t*, %entity_t** % winscreen
  % updateE = call i32 (%entity_t*, ..., ...) @updateE(%entity_t* % winscreen48)
  % e349 = load %entity_t*, %entity_t** %e3
  % updateE52 = call i32 (%entity_t*, ..., ...) @updateE(%entity_t* % e349)
  % e53 = load %entity_t*, %entity_t** %e
  % updateE54 = call i32 (%entity_t*, ..., ...) @updateE(%entity_t* % e53)
  % knight55 = load %player_t*, %player_t** %knight
  % updateP = call i32 (%player_t*, ..., ...) @updateP(%player_t* %knight55)
  % winscreen56 = load %entity_t*, %entity_t** % winscreen
  % addPlayerHitBox = call i32 (%player_t*, i32, i32, ...) @addPlayerHitBox(%player_t* % winscreen56, i32 80, i32 80)
  % e69 = load %entity_t*, %entity_t** %e69
  % addEntityHitBox = call i32 (%entity_t*, i32, i32, ...) @addEntityHitBox(%entity_t* % e69, i32 100, i32 100)
  %e260 = load %entity_t*, %entity_t** %e260
  % addEntityHitBox61 = call i32 (%entity_t*, i32, i32, ...) @addEntityHitBox(%entity_t* % e260, i32 120, i32 120)
  %e362 = load %entity_t*, %entity_t** %e362
  br label %while
%addEntityHitBox63 = call i32 (%entity_t*, i32, i32, ...) @addEntityHitBox
( %entity_t* %e362, i32 120, i32 120)
%knight64 = load %player_t*, %player_t** %knight
%e65 = load %entity_t*, %entity_t** %e
%checkCollision_result = call i1 @checkCollision(%entity_t* %e65, %player_t* %knight64)
%e267 = load %entity_t*, %entity_t** %e2
%checkCollision_result68 = call i1 @checkCollision(%entity_t* %e267, %player_t* %knight66)
%tmp69 = or i1 %checkCollision_result, %checkCollision_result68
%knight70 = load %player_t*, %player_t** %knight
%e371 = load %entity_t*, %entity_t** %e3
%checkCollision_result72 = call i1 @checkCollision(%entity_t* %e371, %player_t* %knight70)
%tmp73 = or i1 %tmp69, %checkCollision_result72
store i1 %tmp73, i1* %collide
%knight74 = load %player_t*, %player_t** %knight
%controlPlayer = call i32 (%player_t*, i32, ...) @controlPlayer(%player_t* %knight74, i32 15)
%e75 = load %entity_t*, %entity_t** %e
%getEntityY = call i32 (%entity_t*, ...) @getEntityY(%entity_t* %e75)
%tmp76 = icmp sge i32 %getEntityY, 700
br i1 %tmp76, label %then, label %else
merge:
%e77 = load %entity_t*, %entity_t** %e
%getEntityY78 = call i32 (%entity_t*, ...) @getEntityY(%entity_t* %e77)
%tmp79 = icmp sle i32 %getEntityY78, 5
br i1 %tmp79, label %then81, label %else82
then:
store i1 false, i1* %down
br label %merge
else:
store i1 true, i1* %down
br label %merge
merge80:
%down83 = load i1, i1* %down
%tmp84 = icmp eq i1 %down83, true
br i1 %tmp84, label %then86, label %else92
then81:
store i1 true, i1* %down
br label %merge80
else82:
br label %merge80
merge85:
%down93 = load i1, i1* %down
%tmp94 = icmp eq i1 %down93, false
br i1 %tmp94, label %then96, label %else103
then86:
%e287 = load %entity_t*, %entity_t** %e2
%changeEntityY = call i32 (%entity_t*, i32, ...) @changeEntityY(%entity_t* %e287, i32 15)
%e88 = load %entity_t*, %entity_t** %e
%changeEntityY89 = call i32 (%entity_t*, i32, ...) @changeEntityY(%entity_t* %e88, i32 25)
%e390 = load %entity_t*, %entity_t** %e3
%changeEntityY91 = call i32 (%entity_t*, i32, ...) @changeEntityY(%entity_t* %e390, i32 35)
br label %merge85
else92:

br label %merge80

58
merge95: ; preds = %else103, %then96
%collide104 = load i1, i1* %collide
br i1 %collide104, label %then106, label %else111

then96: ; preds = %merge85
%e297 = load %entity_t*, %entity_t** %e2
%changeEntityY98 = call i32 (%entity_t*, i32, ...) @changeEntityY(%)entity_t* %e297, i32 -15)
%e99 = load %entity_t*, %entity_t** %e
%changeEntityY100 = call i32 (%entity_t*, i32, ...) @changeEntityY(%entity_t* %e99, i32 -25)
%e3101 = load %entity_t*, %entity_t** %e3
%changeEntityY102 = call i32 (%entity_t*, i32, ...) @changeEntityY(%entity_t* %e3101, i32 -35)
br label %merge95

else103: ; preds = %merge85
br label %merge95

merge105: ; preds = %else111, %then106
%lives112 = load i32, i32* %lives
%tmp113 = icmp sle i32 %lives112, 0
br i1 %tmp113, label %then115, label %else116

then106: ; preds = %merge95
%lives107 = load i32, i32* %lives
%tmp108 = sub i32 %lives107, 1
store i32 %tmp108, i32* %lives
%knight109 = load %player_t*, %player_t** %knight
%setPlayerX = call i32 (%player_t*, i32, ...) @setPlayerX(%player_t* %knight109, i32 0)
%knight110 = load %player_t*, %player_t** %knight
%setPlayerY = call i32 (%player_t*, i32, ...) @setPlayerY(%player_t* %knight110, i32 200)
br label %merge105

else111: ; preds = %merge95
br label %merge105

merge114: ; preds = %else116, %then115
%knight117 = load %player_t*, %player_t** %knight
%getPlayerX = call i32 (%player_t*, ..., ...) @getPlayerX(%player_t* %knight117)
%knight118 = load %player_t*, %player_t** %knight
%getHx = call i32 (%player_t*, ..., ...) @getHx(%player_t* %knight118)
%tmp119 = add i32 %getHx, %getHx
%tmp120 = icmp sge i32 %tmp119, 1200
br i1 %tmp120, label %then122, label %else125

then115: ; preds = %merge105
%printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8],
[4 x i8]* @fmt.1, i32 0, i32 0), i8* getelementptr inbounds ([10 x i8]
[10 x i8]* @str.12, i32 0, i32 0))
store i1 false, i1* %running
br label %merge114

else116: ; preds = %merge105
br label %merge114

merge121: ; preds = %else125, %then122
%presentScene = call i32 (i32, ...) @presentScene(i32 60)
br label %while
define i1 @checkCollision (%entity_t* %e, %player_t* %p) {
  entry:
  %e1 = alloca %entity_t *
  store %entity_t* %e, %entity_t** %e1
  %p2 = alloca %player_t *
  store %player_t* %p, %player_t** %p2
  %pRightX = alloca i32
  %pDownY = alloca i32
  %eRightX = alloca i32
  %eDownY = alloca i32
  %p3 = load %player_t*, %player_t** %p2
  %getPlayerX = call i32 (%player_t*, ...) @getPlayerX (%player_t* %p3)
  %p4 = load %player_t*, %player_t** %p2
  %getPlayerHx = call i32 (%player_t*, ...) @getPlayerHx (%player_t* %p4)
  %tmp = add i32 %getPlayerX, %getPlayerHx
  store i32 %tmp, i32* %pRightX
  %p5 = load %player_t*, %player_t** %p2
  %getPlayerY = call i32 (%player_t*, ...) @getPlayerY (%player_t* %p5)
  %p6 = load %player_t*, %player_t** %p2
  %getPlayerHy = call i32 (%player_t*, ...) @getPlayerHy (%player_t* %p6)
  %tmp7 = add i32 %getPlayerY, %getPlayerHy
  store i32 %tmp7, i32* %pDownY
  %e8 = load %entity_t*, %entity_t** %e1
  %getEntityX = call i32 (%entity_t*, ...) @getEntityX (%entity_t* %e8)
  %e9 = load %entity_t*, %entity_t** %e1
  %getEntityHx = call i32 (%entity_t*, ...) @getEntityHx (%entity_t* %e9)
  %tmp10 = add i32 %getEntityX, %getEntityHx
  store i32 %tmp10, i32* %eRightX
  %e11 = load %entity_t*, %entity_t** %e1
  %getEntityY = call i32 (%entity_t*, ...) @getEntityY (%entity_t* %e11)
  %e12 = load %entity_t*, %entity_t** %e1
  %getEntityHy = call i32 (%entity_t*, ...) @getEntityHy (%entity_t* %e12)
  %tmp13 = add i32 %getEntityY, %getEntityHy
  store i32 %tmp13, i32* %eDownY
  %p14 = load %player_t*, %player_t** %p2
  %getPlayerX15 = call i32 (%player_t*, ...) @getPlayerX (%player_t* %p14)
  %p16 = load %player_t*, %player_t** %p2
  %getPlayerY16 = load %player_t*, %player_t** %p16
  %tmp17 = icmp eq i32 %getPlayerX15, %p16
  %p18 = load %player_t*, %player_t** %p2
  %getPlayerY18 = call i32 (%player_t*, ...) @getPlayerY (%player_t* %p18)
  %pDownY20 = load %player_t*, %player_t** %p2
  %tmp21 = icmp eq i32 %getPlayerY18, %pDownY20
  %tmp22 = or i1 %tmp17, %tmp21
  %p23 = load %player_t*, %player_t** %p2
  %getEntityX24 = call i32 (%entity_t*, ...) @getEntityX (%entity_t* %p23)
  %eRightX26 = load %entity_t*, %entity_t** %e26
  %tmp26 = icmp eq i32 %getEntityX24, %eRightX26
  %tmp27 = or i1 %tmp22, %tmp26
  %e28 = load %entity_t*, %entity_t** %e28
  %getEntityY29 = call i32 (%entity_t*, ...) @getEntityY (%entity_t* %e28)
  %eDownY30 = load %entity_t*, %entity_t** %e28
  %tmp31 = icmp eq i32 %getEntityY29, %eDownY30

  ret i32 0
}
We chose to test the above program because it tests function declaration and definition using Player and Entity Objects, instantiation and rendering of multiple Entities, and tests their behavior inside the runGame() loop. The checkCollision() function works as expected in the runGame() loop and sends the player back to the left-hand side of the screen upon collision. Interaction between players and entities is crucial for building a game, so this test was important.

6.2 Test Suite

Starting from the hello world milestone, we have been writing tests for each feature as we are developing. Each feature has at least one success test case and
at least one fail test case. For features with numerical boundary, we carefully
tested values just within and just outside of the boundary. For function testing,
we tested each argument with different data types.

The test suite contains tests on every feature we implemented. We started
with the small building blocks such as operators, arithmetic and function calls
to more complex blocks such as built-in object, object member functions and
runGame logic.

Much of the unit tests in our language were chosen using MicroC as a reference,
since Lucifer shares a lot of its basic functionality with it. We then built on top
of this base, and wrote pass and fail cases for each additional feature that we
implemented.

Also, visual tests are written so that we can examine the behavior of the pro-
gram when it responds to user inputs.

Here is the test output for our compiler:

```
./testall.sh
test-addEntityHitBox...OK
  test-addPlayerControl...OK
  test-addPlayerControl2...OK
  test-addPlayerHitBox...OK
  test-arith1...OK
  test-arith2...OK
  test-arith3...OK
  test-changeEntityXY...OK
  test-changePlayerXY...OK
  test-comment...OK
  test-entity...OK
  test-entityExprArg...OK
  test-fib...OK
  test-for1...OK
  test-for2...OK
  test-fun1...OK
  test-fun2...OK
  test-fun3...OK
  test-fun4...OK
  test-fun5...OK
  test-fun6...OK
  test-fun7...OK
  test-fun8...OK
  test-fun9...OK
  test-gcd...OK
  test-hello-world-nums...OK
  test-hello-world-variable...OK
  test-hello-world...OK
  test-if1...OK
  test-if2...OK
  test-if3...OK
  test-if4...OK
  test-if5...OK
  test-if6...OK
  test-objectFunction1...OK
  test-objectFunction2...OK
  test-ops1...OK
  test-ops2...OK
  test-player...OK
```
test-playerExprArg...OK
test-playerExtend...OK
test-singleComment...OK
test-while1...OK
test-while2...OK
fail-after-return...OK
fail-arith1...OK
fail-arith2...OK
fail-arith3...OK
fail-assign1...OK
fail-assign2...OK
fail-assign3...OK
fail-comment...OK
fail-controlPlayer-rungame...OK
fail-declEntity-changeXY...OK
fail-entity-uninitialized...OK
fail-entityInvalidName1...OK
fail-entityInvalidName2...OK
fail-entityInvalidTexture...OK
fail-entityInvalidX...OK
fail-entityInvalidY...OK
fail-expr1...OK
fail-for1...OK
fail-for2...OK
fail-fun1...OK
fail-fun10...OK
fail-fun2...OK
fail-fun3...OK
fail-fun4...OK
fail-fun5...OK
fail-fun6...OK
fail-fun7...OK
fail-fun8...OK
fail-fun9...OK
fail-hello-world-nofun...OK
fail-hello-world-nosemi...OK
fail-hello-world-openstring...OK
fail-hello-world1...OK
fail-hello-world2...OK
fail-if1...OK
fail-if2...OK
fail-if3...OK
fail-no-main...OK
fail-objectFunctionArgOrder...OK
fail-playerInvalidSize1...OK
fail-playerInvalidSize2...OK
fail-playerInvalidSize3...OK
fail-playerInvalidTexture...OK
fail-playerInvalidX...OK
fail-playerInvalidY...OK
fail-rungame-entity-decl1...OK
fail-rungame-init...OK
fail-rungame-undecl-entities...OK
fail-sdl-init...OK
fail-singleComment...OK
fail-structNotCap...OK
fail-while1...OK
fail-while2...OK
visual-big-program...OK
visual-controlPlayer-rungame...OK
visual-iskeypressed-noeventity...OK
visual-iskeypressed...OK
visual-moving-entities...OK
visual-render-player...OK
visual-rungame-first...OK
visual-rungame-multiple-entities...OK
visual-rungame-render-origin...OK
visual-rungame-render...OK
The code for each test in included in Appendix 8.8

6.3 Test Automation

As the number of tests grows in our test suite, we start to run our tests automatically. We utilize a shell script named `testall.sh` to run our tests. The terminal will show if each of the test is passed (which is shown as "OK"), if not, a failed test case will be marked as "FAILED" and intermediate output files are kept. The test output is stored in a separate file named `testall.log`, which contained details of each test. Visual tests are handled differently than pass and fail tests in the script—they have a different prefix: "visual" and are simply compiled into .exes so that they can be run locally with the Xserver.

The complete code of `testall.sh` is in the Appendix.

6.4 Testing Responsibility

Elliott was responsible for visual testing, and wrote the pass-tests to verify proper rendering and behaviour of built-in objects. They also were responsible for pass/fail tests for object instantiation and SDL function calls.

Cherry was responsible for built-in objects and functions testing. She wrote pass/fail tests for object instantiation, and object function calls, and comment features. Also, she modified the test automation shell script to produce executables for visual tests.

Michael was responsible for testing the rungame loop parsings and some of the arithmetic tests. As well as helping with the `testall.sh` script.

Robert was responsible for the thorough testing of the rest of the non-SDL features that Lucifer is built with. He wrote pass/fail tests for user defined functions, assignments, loops, and conditionals, as well as other standard algorithms.

7 Lessons Learned

7.1 Elliott Morelli

I learned that in compilers, if you change one thing, it might change a lot of things. Broadly I think I learned how computers turn words into numbers, which is awesome. I also learned that trying to make user-defined structs is hard, and a future goal for this project. I think I learned perseverance.. and a lot about LLVM types. Being conscious of how you’re using your memory is good, both in codegen and in real life. For future groups, my advice would be to start smaller than we did, and if the course is still virtual meet or text very frequently. And, for better or worse, I finally learned how to end a Zoom call.
7.2 Cherry Chu

Communication and coordination is extremely important in this project. Since each member on the team focuses on a different feature, we are constantly modifying all the files at the same time. Therefore, we need to constantly communicate with each other and push frequently to avoid code conflicts.

Also, due to the remote nature of this semester, we found it difficult to collaborate at first. However, we were able to utilize Zoom to set up a meeting space which member can choose to pop in and out throughout the day to discuss and collaborate on things which are too complicated to be expressed through chats. By not having to schedule a meeting all the time, we were able to collaborate more freely and efficiently.

7.3 Robert Becker

Since this is one of the biggest group projects I’ve worked on, this was an interesting experience for me. This project helped me to get much more comfortable using git to track my work alongside others. The remote setting, while posing a challenge, was a good exercise in collaborating with people you can’t meet directly. We set up times to meet over zoom after our weekly meetings with our TA, and I feel this was very helpful in keeping everyone up to date.

Writing tests for Lucifer helped me become very familiar with common mistakes that would need to be checked, while also looking carefully for cases that may pass in an unexpected way. Doing this involved consistently checking back in with our semant file to see what errors should be thrown, and exactly how our statements can be formed.

7.4 Michael Fagan

Compilers has really solidified the connection between the theoretical and practical sides of computer science for me. Seeing how a context free grammar and deterministic automata can be used to implement a compiler has helped to bridge the gap and give motivation to the study of these abstract objects. I also learned how to work on a programming team and how to coordinate while working remote. Our weekly meetings both together, and with the TA, were invaluable, and my advice for future groups would be to not be afraid of asking for help, you are a group after all.

8 Appendix

8.1 Testall.sh

```bash
#!/bin/sh
```
# Regression testing script for MicroC
# Step through a list of files
# Compile, run, and check the output of each expected-to-work test
# Compile and check the error of each expected-to-fail test

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

# Path to the LLVM compiler
LLD = "llc"

# Path to the C compiler
CC = "cc"

# Path to the microc compiler. Usually ".microc.native"
# Try ".build/microc.native" if ocamlbuild was unable to create a symbolic
# link.
MICROC = "./lucifer.native"
#MICROC = "./build/microc.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] [.luc 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" > "$3"
    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 $* > $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
}

CheckVisual() {
  basename="echo $1 | sed 's/\./\///'
  reffile="echo $1 | sed 's/\./\$\//'
  basedir="echo $1 | sed 's/\/\/[\"\\]*/\//'."
  echo -n "$basename..."
  echo 1 >&2
  echo "#### Testing $basename" 1 >&2
  generatedfiles=""
  generatedfiles="$generatedfiles ${basename}.ll ${basename}.s" &&
  Run "MICROC "$1" >> "$({basename}).ll" &&
  Run "LLC" "-relocation-model=pic" "${{basename}}.ll" " >" "${{basename}}.s" &&
  Run "CC" "-o" "${{basename}}.exe" "${{basename}}.s" "initbig.o" "entity.o" "bin/draw.o" "bin/init.o" "bin/input.o" "'sdl2-config --libs" "-lSDL2_mixer" "-lSDL2_image" "-lSDL2_ttf" "-lm" &&
  rm -rf $generatedfiles
  echo "OK"
  echo "#### SUCCESS" 1 >&2
}

Check() {
  error=0
  basename="echo $1 | sed 's/\./\///'
  reffile="echo $1 | sed 's/\./\$\//'
  basedir="echo $1 | sed 's/\/\/[\"\\]*/\//'."
  echo -n "$basename..."
  echo 1 >&2
  echo "#### Testing $basename" 1 >&2
  generatedfiles=""
  generatedfiles="$generatedfiles ${basename}.ll ${basename}.s" &&
  Run "MICROC "$1" >> "$({basename}).ll" &&
  Run "LLC" "-relocation-model=pic" "${{basename}}.ll" " >" "${{basename}}.s" &&
  Run "CC" "-o" "${{basename}}.exe" "${{basename}}.s" "initbig.o" "entity.o" "bin/draw.o" "bin/init.o" "bin/input.o" "'sdl2-config --libs" "-lSDL2_mixer" "-lSDL2_image" "-lSDL2_ttf" "-lm" &&
  Run "./${{basename}}.exe" > "${{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 $generatedfiles
    fi
  fi
}
fi
echo "OK"
else
  echo "######## FAILED" 1>&2
globalerror=$error
fi
}
CheckFail() {
  error=0
  basename=`echo $1 | sed 's/.\///
  s/.luc//'
  reffile=`echo $1 | sed 's/.luc$//'
  basedir=`echo $1 | sed "s/\([^-]*/\)/\}"/`
  echo -n "$basename..."
  echo 1 >&2
  echo " Testing $basename" 1 >&2
  generatedfiles=""
  generatedfiles="$generatedfiles $(basename).err $(basename).diff " &&
  RunFail "$MICROD" "$1" "$(basename).err" "$(basename).err $(globallog)"
  Compare $(basename).err $(reffile).err $(basename).diff
  # Report the status and clean up the generated files
  if [ $error = 0 ] ; then
    if [ $keep = 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
      Help
      ;;
  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 [ ! -f initsd1.o ]
then
  echo "Could not find initsd1.o"
  echo "Try \"make initsd1.o\""s
  exit 1
}
if 

if [ ! -f player.o ]
then
  echo "Could not find player.o"
  echo "Try \"make player.o\"
  exit 1
fi

if [ ! -f entity.o ]
then
  echo "Could not find entity.o"
  echo "Try \"make entity.o\"
  exit 1
fi

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

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

exit $globalerror

8.2 scanner.mll

(*Scanner for Lucifer
Authors: Cherry Chu, Michael Fagan
Citation: Microc scanner.mll*)

{ open Luciferparse }

let digit = ['0' - '9']
let digits = digit +

rule token = parse
  [' ' '	' '' '
'] { token lexbuf } (* Whitespace *)
| /*" { comment lexbuf } (* Comments *)
| "/" { single_comment lexbuf } (* Single line comments *)
| '(' { LPAREN }
| ')' { RPAREN }
| '{' { LBRACE }
| '}' { RBRACE }
| ',' { SEMI }
| ',' { COMMA }
8.3 ast.ml

(* Abstract Syntax Tree and functions for printing it
Authors: Elliott Morelli, Cherry Chu, Michael Fagan
Citation: Microc ast.ml *)

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

type uop = Neg | Not

type typ = Int | Bool | Float | String | Void | Player | Entity

type bind = typ * string

type expr =
  | Literal of int
  | Fliteral of string
  | BoolLit of bool
  | Clit of string
  | Id of string
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of string * expr
  | Call of string * expr list
  | Noexpr
  | NewPlayer of expr * expr * expr * expr
  | NewEntity of expr * expr * expr
  | MemberCall of string * string * expr list

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
  | RunGame of expr * expr * stmt

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

type program = bind list * func_decl list

(* Pretty-printing functions *)

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

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

let rec string_of_expr = function
  Literal(l) -> string_of_int l
  Fliteral(l) -> l
  BoolLit(true) -> " true "
  BoolLit(false) -> " false "
  Id(s) -> s
  CLit(l) -> l
  Binop (e1 , o, e2) ->
    " 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 -> " *
  NewPlayer(e1 , e2 , e3 , e4) ->
    " new Player ( " ^ ( string_of_expr e1) ^ " , " ^ ( string_of_expr e2) ^ ",
    " ^ ( string_of_expr e3) ^ ", " ^ ( string_of_expr e4) ^ " )"
  NewEntity(e1 , e2 , e3) -> " new Entity ( " ^ ( string_of_expr e1) ^ " , " ^ ( string_of_expr e2) ^ ",
    " ^ ( string_of_expr e3) ^ " )"

let rec string_of_stmt = function
  Block ( stmts ) ->
    " { " ^ String . concat " ( List . map string_of_stmt stmts) " } ">
  Expr ( expr ) -> string_of_expr expr ^ " ; ">
  Return ( expr ) -> " return " ^ string_of_expr expr ^ " ; ">
  If ( e, s ) -> " if ( " ^ string_of_expr e ^ " ) " ^ string_of_stmt s
  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
  RunGame ( e, s ) -> " runGame ( " ^ string_of_expr e ^ " ) " ^ string_of_stmt s

let string_of_typ = function

72
let string_of_vdecl (t, id) = string_of_typ t ^ " ^ id ^ ";

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

let string_of_program (vars, funcs) =
  String.concat " (List.map string_of_vdecl vars) ^ " ^
  String.concat " (List.map string_of_fdecl funcs)
8.4 Luciferparse.mly

```ml
/* Ocamlyacc parser for Lucifer
Authors: Elliott Morelli, Cherry Chu, Michael Fagan
Citation: Microcparse.mly */

%(open Ast %)
%
%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA PLUS MINUS TIMES DIVIDE ASSIGN MOD
%token NOT EQ NEQ LT LEQ GT GEQ AND OR
%token RETURN IF ELSE FOR WHILE RUNGAME INT BOOL FLOAT STRING VOID FUN
%token <int> LITERAL
%token <bool> BLIT
%token <string> ID FLIT CLIT SID
%token EOF

%start program
%type <Ast.program> program

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE MOD
%right NOT
%
program : decls EOF { $1 }
decs : /* nothing */ { ([] , []) } |
decs vdecl { (((2 :: fst $1), snd $1)) } |
decs fdecl { ( fst $1 , (2 :: snd $1)) } fdecl : FUN ID typ LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE |
formals_opt : /* nothing */ { [] } |
formal_list { $1 } typ : INT { Int } |
BOOL { Bool } |
FLOAT { Float } |
STRING { String } |
VOID { Void }

74
```
vdecl_list:
  /* nothing */ { [] }

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

stmt_list:
  /* nothing */ { [] }

b_stmt_list:
  /* nothing */ { [] }

stmt:
  expr SEMI { Expr $1 }

b_stmt:
  niexpr SEMI { Expr $1 }

expr_opt:
  /* nothing */ { Noexpr }

niexpr_opt:
  /* nothing */ { Noexpr }

expr:
  LITERAL { Literal($1) }

niexpr:
  { $1 }

bStmt:
  niexpr SEMI { Expr $1 }

expr MOD expr { Binop($1, Mod, $3) }
MINUS expr %prec NOT { Unop(Not, $2) }
NOT expr { Unop(Not, $2) }
ID ASSIGN expr { Assign($1, $3) }
ID LPAREN args_opt RPAREN { Call($1, $3) }
LPAREN expr RPAREN { $2 }
NEW PLAYER LPAREN expr COMMA expr COMMA expr COMMA expr RPAREN
{ NewPlayer($4, $6, $8, $10) }
NEW ENTITY LPAREN expr COMMA expr COMMA expr RPAREN
{ NewEntity($4, $6, $8) }
ID DOT ID LPAREN args_opt RPAREN
{ Call($3, List.rev(List.rev $5 @ [Id($1)])) }

niexpr:
LITERAL { Literal($1) }
| FLIT { Fliteral($1) }
| BLIT { BoolLit($1) }
| CLIT { CLit($1) }
| ID { Id($1) }
| niexpr PLUS niexpr { Binop($1, Add, $3) }
| niexpr MINUS niexpr { Binop($1, Sub, $3) }
| niexpr TIMES niexpr { Binop($1, Mult, $3) }
| niexpr DIVIDE niexpr { Binop($1, Div, $3) }
| niexpr EQ niexpr { Binop($1, Equal, $3) }
| niexpr NEQ niexpr { Binop($1, Neq, $3) }
| niexpr LT niexpr { Binop($1, Less, $3) }
| niexpr LEQ niexpr { Binop($1, Leq, $3) }
| niexpr GT niexpr { Binop($1, Greater, $3) }
| niexpr GEQ niexpr { Binop($1, Geq, $3) }
| niexpr AND niexpr { Binop($1, And, $3) }
| niexpr OR niexpr { Binop($1, Or, $3) }
| niexpr MOD niexpr { Binop($1, Mod, $3) }
MINUS niexpr %prec NOT { Unop(Not, $2) }
| NOT niexpr { Unop(Not, $2) }
| ID ASSIGN niexpr { Assign($1, $3) }
| ID LPAREN nargs_opt RPAREN { Call($1, $3) }
LPAREN niexpr RPAREN { $2 }
ID DOT ID LPAREN nargs_opt RPAREN
{ Call($3, List.rev(List.rev $5 @ [Id($1)])) }

args_opt:
/* nothing */ { [] }
| args_list { List.rev $1 }

args_list:
expr { [$1] }
| args_list COMMA expr { $3 :: $1 }
nargs_opt:
/* nothing */ { [] }
| nargs_list { List.rev $1 }
nargs_list:
niexpr { [$1] }
| nargs_list COMMA niexpr { $3 :: $1 }
open Ast

module type sexpr = typ * sx

and sx =
| SLiteral of int
| SFliteral of string
| SBoolLit of bool
| SCLit of string
| SId of string
| SBinop of sexpr * op * sexpr
| SUnop of uop * sexpr
| SAssign of string * sexpr
| SCall of string * sexpr list
| SKexpr
| SNewPlayer of sexpr * sexpr * sexpr * sexpr
| SNewEntity of sexpr * sexpr * sexpr

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

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

module type sprogram = bind list * sfunc_decl list

(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
| "(" ^ string_of_typ t ^ " : " ^ ( match e with
| SLiteral(l) -> string_of_int l
| SFliteral(l) -> l
| SCLit(l) -> l
| SId(s) -> s
| SBinop(e1, o, e2) ->
  string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
| SUnop(o, e) -> string_of_expr e ^ " " ^ string_of_expr e
| SAssign(v, e) -> v ^ " = " ^ string_of_expr e
| SCall(f, el) ->
  f ^ "(" ^ (List.map string_of_expr el) ^ "")"
| SKexpr -> ""
| SNewPlayer(e1, e2, e3, e4) ->
  "new Player (" ^ (string_of_expr e1) ^ ", " ^ (string_of_expr e2) ^ ", " ^ (string_of_expr e3) ^ " , " ^ (string_of_expr e4) ^ ")"
| SNewEntity(e1, e2, e3) ->
  "new Entity (" ^ (string_of_expr e1) ^ ", " ^ (string_of_expr e2) ^ " , " ^ (string_of_expr e3) ^ ")"
let rec string_of_sstmt = function
| SBlock (stmts) -> "\n{\n" ^ String.concat "" (List.map string_of_sstmt stmts) "\n}\n"
| SExpr (expr) -> string_of_sexpr expr ";\n"
| SReturn (expr) -> "return " ^ string_of_sexpr expr ";\n"
| SIf (e, s, SBlock ([])) -> "if (" ^ string_of_sexpr e "\n" ^ string_of_sstmt s
| SIf (e, s1, s2) -> "if (" ^ string_of_sexpr e "\n" ^ string_of_sstmt s1 " else\n" ^ string_of_sstmt s2
| SFor (e1, e2, e3, s) -> "for (" ^ string_of_sexpr e1 " ; " ^ string_of_sexpr e2 " ; " ^ string_of_sexpr e3 "\n" ^ string_of_sstmt s
| SWhile (e, s) -> "while (" ^ string_of_sexpr e "\n" ^ string_of_sstmt s
| SKunGame(e, e2, s) -> "runGame (" ^ string_of_sexpr e " ; " ^ string_of_sexpr e2 "\n" ^ string_of_sstmt s

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

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

8.6 semant.ml

(* Semantic checking for the Lucifer compiler
Authors: Elliott Morelli, Cherry Chu, Michael Fagan 
Citation: MicroC semant.ml *)

open Ast
open Sast

module StringMap = Map.Make(String)

(* Semantic checking of the AST. Returns an SAST if successful, 
throws an exception if something is wrong. 
Check each global variable, then check each function *)

let check (globals, functions) =

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

(**** Check global variables ****)
check_binds "global" globals;
(**** Check functions ****)

78
let built_in_decls =
StringMap.add "print" { typ = Void; fname = "print"; locals = []; body = [] }
(StringMap.add "printf" { typ = Void; fname = "printf"; locals = []; body = [] }
(StringMap.add "printb" { typ = Void; fname = "printb"; locals = []; body = [] }
(StringMap.add "prints" { typ = Void; fname = "prints"; locals = []; body = [] }
(StringMap.add "initsdl" { typ = Void; fname = "initsdl"; locals = []; body = [] }
(StringMap.add "Player" { typ = Void; fname = "Player"; locals = []; body = [] }
(StringMap.add "Entity" { typ = Void; fname = "Entity"; locals = []; body = [] }
(StringMap.add "isKeyPressed" { typ = Bool; fname = "isKeyPressed"; locals = []; body = [] }
(StringMap.add "getPlayerX" { typ = Int; fname = "getPlayerX"; locals = []; body = [] }
(StringMap.add "getPlayerY" { typ = Int; fname = "getPlayerY"; locals = []; body = [] }
(StringMap.add "setPlayerX" { typ = Void; fname = "setPlayerX"; locals = []; body = [] }
(StringMap.add "setPlayerY" { typ = Void; fname = "setPlayerY"; locals = []; body = [] }
(StringMap.add "getPlayerControl" { typ = Int; fname = "getPlayerControl"; locals = []; body = [] }
(StringMap.add "zeroPlayerControls" { typ = Void; fname = "zeroPlayerControls"; locals = []; body = [] }
(StringMap.add "getPlayerText" { typ = String; fname = "getPlayerText"; locals = []; body = [] }
(StringMap.add "getPlayerHx" { typ = Int; fname = "getPlayerHx"; locals = []; body = [] }
(StringMap.add "getPlayerHy" { typ = Int; fname = "getPlayerHy"; locals = []; body = [] }
(StringMap.add "addPlayerHitBox" { typ = Void; fname = "addPlayerHitBox"; locals = []; body = [] }
(StringMap.add "changePlayerX" { typ = Void; fname = "changePlayerX"; locals = []; body = [] }
(StringMap.add "changePlayerY" { typ = Void; fname = "changePlayerY"; locals = []; body = [] }
(StringMap.add "controlPlayer" { typ = Void; fname = "controlPlayer"; locals = []; body = [] }
(StringMap.add "addPlayerControl" { typ = Void; fname = "addPlayerControl"; locals = []; body = [] }
(StringMap.add "getEntityX" { typ = Int; fname = "getEntityX"; locals = []; body = [] }
(StringMap.add "getEntityY" { typ = Int; fname = "getEntityY"; locals = []; body = [] }
(StringMap.add "getEntityText" { typ = String; fname = "getEntityText"; locals = []; body = [] }
(StringMap.add "getEntityHx" { typ = Int; fname = "getEntityHx"; locals = []; body = [] }
(StringMap.add "setEntityX" { typ = Void; fname = "setEntityX"; locals = []; body = [] }
(StringMap.add "setEntityY" { typ = Void; fname = "setEntityY"; locals = []; body = [] })
(StringMap.add "addEntityHitBox" { typ = Void; fname = "addEntityHitBox";
    formals = [(Entity, "e"); (Int, "x"); (Int, "y")] ; locals = []; body = [] })
(StringMap.add "changeEntityX" { typ = Void; fname = "changeEntityX";
    formals = [(Entity, "e"); (Int, "x")] ; locals = []; body = [] })
(StringMap.singleton "changeEntityY" { typ = Void; fname = "changeEntityY";
    formals = [(Entity, "e"); (Int, "y")] ; locals = []; body = [] })

in

(* Add function name to symbol table *)
let add_func map fd =
  let built_in_err = " function " ^ fd.fname ^ " may not be defined"
  and dup_err = " duplicate function " ^ fd.fname
  and make_err er = raise (Failure er)
  and n = fd.fname (* Name of the function *)
  in
  match fd with (* No duplicate functions or redefinitions of built-ins *)
          _ when StringMap.mem n built_in_decls -> make_err built_in_err
          | _ when StringMap.mem n map -> make_err dup_err
          | _ -> StringMap.add n fd map
in
(* Collect all function names into one symbol table *)
let function_decls = List.fold_left add_func built_in_decls functions
in
(* Return a function from our symbol table *)
let find_func s =
  try StringMap.find s function_decls
  with Not_found -> raise (Failure (" unrecognized function " ^ s))
in
let _ = find_func " main " in (* Ensure " main " is defined *)

let check_function func =
  (* Make sure no formals or locals are void or duplicates *)
  check_binds " formal " func.formals;
  check_binds " local " func.locals;
  (* Raise an exception if the given rvalue type cannot be assigned to
     the given lvalue type *)
  let check_assign lvaluet rvaluet err =
    if lvaluet = rvaluet then lvaluet else raise (Failure err)
in
(* Build local symbol table of variables for this function *)
let symbols = List.fold_left (fun m (ty, name) -> StringMap.add name ty m )
              StringMap.empty (globals @ func.formals @ func.locals)
in
(*Build hash table of global and function local Entity and Player
   declarations*)
let obj_hash = Hashtbl.create 123456 in

let obj_decls = List.iter (fun (ty,name) -> match ty with
          Player -> Hashtbl.add obj_hash name ty
        | Entity -> Hashtbl.add obj_hash name ty
        _ -> ()) (globals @ func.locals); in

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

(* Return a semantically-checked expression, i.e., with a type *)
let rec expr = function
  Literal l -> (Int , SLiteral l)
  | Fliteral l -> ( Float , SFliteral l)
  | BoolLit l -> (Bool , SBoolLit l)
  | CLit l -> (String , SCLit l)
  | Noexpr -> (Void , SNoexpr)
  | Id s -> (type_of_identifier s , SId s)
  | Assign (var , e) as ex ->
    let lt = type_of_identifier var
    and (rt , e') = expr e in
    (* remove e from obj_hash as it is instantiated to check obj
     instantiation **)
    let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
    string_of_typ rt ^ " in " ^ string_of_expr ex
    in ignore (obj_decls = obj_instantiated var); (check_assign lt rt
    err , SAssign (var , (rt , e')))
  | Unop (op , e) as ex ->
    let (t , e') = expr e in
    let ty = match op with
      Neg when t = Int || t = Float -> t
    | Not when t = Bool -> Bool
    | _ -> raise (Failure ("illegal unary operator " ^
        string_of_uop op ^ string_of_typ t ^
        " in " ^ string_of_expr ex))
    in (ty , SUnop (op , (t , e')))
  | Binop (e1 , op , e2) as e ->
    let (t1 , e1 ') = expr e1
    and (t2 , e2 ') = expr e2 in
    (* All binary operators require operands of the same type *)
    let same = t1 = t2 in
    (* Determine expression type based on operator and operand types *)
    let ty = match op with
      Add | Sub | Mult | Div | Mod when same && t1 = Int -> Int
    | Add | Sub | Mod | Div when same && t1 = Float -> Float
    | Equal | Neq when same -> Bool
    | _ when same && (t1 = Int || t1 = Float) -> Bool
    | _ when same && t1 = Bool -> Bool
    | _ -> raise (Failure ("illegal binary operator " ^
        string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
        string_of_typ t2 ^ " in " ^ string_of_expr e))
    in (ty , SBinop ((t1 , e1 '), op , (t2 , e2 ')))
  | Call (fname , args) as call ->
    let fd = find_func fname in
    if List.length args != param_length then
      raise (Failure ("expecting " ^ string_of_int param_length ^
        " arguments in " ^ string_of_expr call))
    else let check_call (ft , _) e =
      let (et , e') = expr e in
      let err = "illegal argument found " ^ string_of_typ et ^
        " expected " ^ string_of_typ ft ^ " in " ^ string_of_expr e
      in (check_assign ft et err , e')
    in let args' = List.map2 check_call fd.formals args
        in (fd.typ , SCall (fname , args'))
  | NewPlayer (e1 , e2 , e3 , e4) ->
    let (t1 , e1 ') = expr e1
    and (t2 , e2 ') = expr e2
    and (t3 , e3 ') = expr e3
    and (t4 , e4 ') = expr e4
    in (check_assign t1 t2 err , SNewPlayer ((t1 , e1 '),
        (t2 , e2 '), (t3 , e3 '), (t4 , e4 ')))

(*Function to remove a declaration from the hash once it is instantiated*
*)

let (t2, e2') = expr e2 in
let (t3, e3') = expr e3 in
let (t4, e4') = expr e4 in
if t1 != Int then raise (Failure ("expecting Int type for x position argument"))
else if t2 != Int then raise (Failure ("expecting Int type for y position argument"))
else if t3 != String then raise (Failure ("expecting String type for texture argument"))
else if t4 != Int then raise (Failure ("expecting Int type for control array size argument"))
else (match e4 with
    | Literal n -> if n < 4 then raise (Failure ("expecting a minimum size 4(up,down,left,right) for control array"))
    else (Player, SNewPlayer((t1, e1'), (t2, e2'),
        (t3, e3'), (t4, e4')))
    | _ -> raise (Failure ("expecting a minimum size 4(up,down,left,right) for control array"))
  )
| NewEntity(e1, e2, e3) ->
  let (t1, e1') = expr e1 in
  let (t2, e2') = expr e2 in
  let (t3, e3') = expr e3 in
  if t1 != Int then raise (Failure ("expecting Int type for x position argument"))
  else if t2 != Int then raise (Failure ("expecting Int type for y position argument"))
  else if t3 != String then raise (Failure ("expecting String type for texture argument"))
  else (Entity, SNewEntity((t1, e1'), (t2, e2'), (t3, e3')))
in
let check_bool_expr e =
  let (t', e') = expr e
  and err = "expected Boolean expression in " ^ string_of_expr e
  in if t' != Bool then raise (Failure err) else (t', e')
in
(* checking that objs are instantiated as they all would be removed *)
let check_decls h = match Hashtbl.length h with
    0 -> ()
    |_ -> raise (Failure ("all entities and players must be instantiated"))
in
(* Return a semantically-checked statement i.e. containing sexprs *)
let rec check_stmt = function
    | Expr e -> SExpr (expr e)
    | If(p, b1, b2) -> SIf (check_bool_expr p, check_stmt b1, check_stmt b2)
    | For(e1, e2, e3, st) -> SFor (expr e1, check_bool_expr e2, expr e3, check_stmt st)
    | While(p, s) -> SWhile (check_bool_expr p, check_stmt s)
    | RunGame(e1, e2, s) -> SCRanGame (check_bool_expr e1, expr e2, check_stmt st)
    | Return e -> let (t, e') = expr e in
        if t = func.typ then SReturn (t, e')
        else raise (Failure ("return gives " ^ string_of_typ t ^ " expected " ^ string_of_typ func.typ " in " ^ string_of_expr e))
    else raise (Failure ("nothing may follow a return")
(* A block is correct if each statement is correct and nothing follows any Return statement. Nested blocks are flattened. *)
let Block sl ->
    let rec check_stmt_list = function
        [ Return _ as s] -> [check_stmt s]
        | Return _ :: _ -> raise (Failure ("nothing may follow a return"))
8.7 codegen.ml

(* Code generation for Lucifer:
Authors: Elliott Morelli, Cherry Chu, Michael Fagan
Citation: MicroC codegen.ml *)

module L = Llvm
module A = Ast
open Sast
open StringMap

(* translate : Sast.program -> Llvm.module *)
let translate (globals, functions) =
  let context = L.global_context () in
  (* Create the LLVM compilation module into which
  we will generate code *)
  let the_module = L.create_module context "Lucifer" in
  (* Get types from the context *)
  let i32_t = L.i32_type context
  and i8_t = L.i8_type context
  and i1_t = L.i1_type context
  and float_t = L.double_type context
  and void_t = L.void_type context
  in
  let player_t = L.named_struct_type context "player_t" in
  L.struct_set_body player_t [i32_t, i32_t, L.pointer_type i8_t, i32_t, i1_t, L.pointer_type i32_t, L.pointer_type i8_t] false;

  let entity_t = L.named_struct_type context "entity_t" in
  L.struct_set_body entity_t [i32_t, i32_t, L.pointer_type i8_t, L.pointer_type i8_t, i32_t, i32_t] false;

  (* Return the LLVM type for a MicroC type *)
  let ltype_of_typ = function
    A.Int -> i32_t
    | A.Bool -> i1_t
    | A.Float -> float_t
    | A.String -> L.pointer_type i8_t
    | A.Void -> void_t
    | A.Player -> L.pointer_type player_t
let global_vars : L.llvalue StringMap.t =
  let global_var m (t, n) =
    let init = match t with
      | A.Float -> L.const_float (ltype_of_typ t) 0.0
      | A.Entity -> print_string("in global") ; L.const_int (ltype_of_typ t) 99
      | _ -> L.const_int (ltype_of_typ t) 0
    in StringMap.add n (L.define_global n init the_module) m in
  List.fold_left global_var StringMap.empty globals in

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

let initsdl_t : L.lltype =
  L.var_arg_function_type i32_t [i32_t []] in
let initsdl_func : L.llvalue =
  L.declare_function "initsdl" initsdl_t the_module in

let initentity_t : L.lltype =
  L.var_arg_function_type i32_t [L.pointer_type entity_t []] in
let initentity_func : L.llvalue =
  L.declare_function "initEntity" initentity_t the_module in

let initplayer_t : L.lltype =
  L.var_arg_function_type i32_t [L.pointer_type player_t []] in
let initplayer_func : L.llvalue =
  L.declare_function "initPlayer" initplayer_t the_module in

let preparescene_t : L.lltype =
  L.var_arg_function_type i32_t [i32_t []] in
let preparescene_func : L.llvalue =
  L.declare_function "prepareScene" preparescene_t the_module in

let presentscene_t : L.lltype =
  L.var_arg_function_type i32_t [i32_t []] in
let presentscene_func : L.llvalue =
  L.declare_function "presentScene" presentscene_t the_module in

let iskeypressed_t : L.lltype =
  L.var_arg_function_type i1_t [i32_t []] in
let iskeypressed_func : L.llvalue =
  L.declare_function "isKeyPressed" iskeypressed_t the_module in

let update_e_t : L.lltype =
  L.var_arg_function_type i32_t [L.pointer_type entity_t []] in
let update_e_func : L.llvalue =
  L.declare_function "updateE" update_e_t the_module in

let update_p_t : L.lltype =
  L.var_arg_function_type i32_t [L.pointer_type player_t []] in
let update_p_func : L.llvalue =
  L.declare_function "updateP" update_p_t the_module in

let getplayerx_t : L.lltype =
  L.var_arg_function_type i32_t [L.pointer_type player_t []] in
let getplayerx_func : L.llvalue =
  L.declare_function "getPlayerX" getplayerx_t the_module in

let global_vars : L.llvalue StringMap.t =
  let global_var m (t, n) =
    let init = match t with
      | A.Float -> L.const_float (ltype_of_typ t) 0.0
      | A.Entity -> print_string("in global") ; L.const_int (ltype_of_typ t) 99
      | _ -> 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
L. declare_function "getPlayerX" getplayerx_t the_module in
let getplayery_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t | ] in
let getplayery_func : L. llvalue =
  L. declare_function "getPlayerY" getplayery_t the_module in

let setplayerx_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t ; i32_t | ] in
let setplayerx_func : L. llvalue =
  L. declare_function "setPlayerX" setplayerx_t the_module in

let setplayery_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t ; i32_t | ] in
let setplayery_func : L. llvalue =
  L. declare_function "setPlayerY" setplayery_t the_module in

let getplayercontrol_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t ; i32_t | ] in
let getplayercontrol_func : L. llvalue =
  L. declare_function "getPlayerControl" getplayercontrol_t the_module in

let zeroplayercontrols_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t | ] in
let zeroplayercontrols_func : L. llvalue =
  L. declare_function "zeroPlayerControls" zeroplayercontrols_t the_module in

let getplayertext_t : L. lltype =
  L. var_arg_function_type ( L. pointer_type i32_t ) [ | L. pointer_type player_t | ] in
let getplayertext_func : L. llvalue =
  L. declare_function "getPlayerText" getplayertext_t the_module in

let getplayerhx_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t | ] in
let getplayerhx_func : L. llvalue =
  L. declare_function "getPlayerHx" getplayerhx_t the_module in

let getplayerhy_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t | ] in
let getplayerhy_func : L. llvalue =
  L. declare_function "getPlayerHy" getplayerhy_t the_module in

let addplayerhitbox_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t ; i32_t ; i32_t | ] in
let addplayerhitbox_func : L. llvalue =
  L. declare_function "addPlayerHitBox" addplayerhitbox_t the_module in

let changeplayerx_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t ; i32_t | ] in
let changeplayerx_func : L. llvalue =
  L. declare_function "changePlayerX" changeplayerx_t the_module in

let changeplayery_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t ; i32_t | ] in
let changeplayery_func : L. llvalue =
  L. declare_function "changePlayerY" changeplayery_t the_module in

let addplayercontrol_t : L. lltype =
  L. var_arg_function_type i32_t [ | L. pointer_type player_t ; i32_t ; i32_t | ] in
let addplayercontrol_func : L. llvalue =
  L. declare_function "addPlayerControl" addplayercontrol_t the_module in
let controlplayer_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type player_t ; i32_t | ] in
let controlplayer_func : L.llvalue = 
    L.declare_function "controlPlayer" controlplayer_t the_module in

let getentityx_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t | ] in
let getentityx_func : L.llvalue = 
    L.declare_function "getEntityX" getentityx_t the_module in

let getentity_y_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t | ] in
let getentity_y_func : L.llvalue = 
    L.declare_function "getEntityY" getentity_y_t the_module in

let getentitytext_t : L.lltype = 
    L.var_arg_function_type (L.pointer_type i32_t ) [ | L.pointer_type 
entity_t | ] in
let getentitytext_func : L.llvalue = 
    L.declare_function "getEntityText" getentitytext_t the_module in

let getentityhx_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t | ] in
let getentityhx_func : L.llvalue = 
    L.declare_function "getEntityHx" getentityhx_t the_module in

let getentityhy_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t | ] in
let getentityhy_func : L.llvalue = 
    L.declare_function "getEntityHy" getentityhy_t the_module in

let setentityx_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t ; i32_t | ] in
let setentityx_func : L.llvalue = 
    L.declare_function "setEntityX" setentityx_t the_module in

let setentity_y_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t ; i32_t | ] in
let setentity_y_func : L.llvalue = 
    L.declare_function "setEntityY" setentity_y_t the_module in

let addentityhitbox_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t ; i32_t ; i32_t 
| ] in
let addentityhitbox_func : L.llvalue = 
    L.declare_function "addEntityHitBox" addentityhitbox_t the_module in

let change_entityx_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t ; i32_t | ] in
let change_entityx_func : L.llvalue = 
    L.declare_function "changeEntityX" change_entityx_t the_module in

let change_entity_y_t : L.lltype = 
    L.var_arg_function_type i32_t [ | L.pointer_type entity_t ; i32_t | ] in
let change_entity_y_func : L.llvalue = 
    L.declare_function "changeEntityY" change_entity_y_t the_module in

(* Define each function (arguments and return type) so we can 
call it even before we’ve created its body *)
let function_decls : (L.llvalue * sfunc_decl) StringMap.t =
    let function_decl m fdecl =
        let name = fdecl.sfname
        and formal_types =
            Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) fdecl.sformals)
        in
            let ftype = L.function_type (match fdecl.sfname with
                | "controlPlayer" -> controlplayer_t
                | "getEntityX" -> getentityx_t
                | "getEntityY" -> getentity_y_t
                | "getEntityText" -> getentitytext_t
                | "getEntityHx" -> getentityhx_t
                | "getEntityHy" -> getentityhy_t
                | "setEntityX" -> setentityx_t
                | "setEntityY" -> setentity_y_t
                | "addEntityHitBox" -> addentityhitbox_t
                | "changeEntityX" -> change_entityx_t
                | "changeEntityY" -> change_entity_y_t
                else L.function_type i32_t [ | ]
        in
            (m, { sfunc_name = name; sfunc_type = ftype; sformals = formal_types })
    in
        let fdecls =
            List.map (fun (name, decl) -> function_decl decl)
            function_decls
        in
            StringMap.of_list fdecls
    in
        function_decls
}
"main" -> i32_t
| _ -> ltype_of_typ fdecl.styp)
formal_types
in
StringMap.add name (L.define_function name ftype the_module, fdecl) n
in
List.fold_left function_decl StringMap.empty functions in

(* Fill in the body of the given function *)
let build_function_body fdecl =
let (the_function, _) = StringMap.find fdecl.sfname function_decls in
let builder = L.builder_at_end context (L.entry_block the_function) in
let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder
and str_format_str = L.build_global_stringptr "%s\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 local_vars =
let add_formal m (t, n) p =
  L.set_value_name n p;
let local = L.build_alloca (ltype_of_typ t) n builder in
ignore (L.build_store p local builder);
StringMap.add n local m
(* Allocate space for any locally declared variables and add the resulting registers to our map *)
and add_local m (t, n) =
let local_var = L.build_alloca (ltype_of_typ t) n builder
in
StringMap.add n local_var m
in
let formals = List.fold_left2 add_formal StringMap.empty fdecl.sformals (Array.to_list (L.params the_function)) in
List.fold_left add_local formals fdecl.slocals
in

(* creates list of entity and player declarations as expressions *)
let object_vars =
let add_local_obj l (t, n) =
  match t with
  | A.Entity -> (t, (SId n)) :: l
  | A.Player -> (t, (SId n)) :: l
  | _ -> l
in
List.fold_left add_local_obj [] fdecl.slocals
in

(* Return the value for a variable or formal argument. Check local names first, then global names *)
let lookup n = try StringMap.find n local_vars
  with Not_found -> try StringMap.find n global_vars
  with Not_found -> raise (Failure "internal error: variable may not be initialized")
in

(* Construct code for an expression; return its value *)
let rec expr builder ((_, e) : sexpr) = match e with
  SLiteral i -> L.const_int i32_t i
  | SBoolLit b -> L.const_int i1_t (if b then 1 else 0)
  | SFliteral l -> L.const_float_of_string float_t l
  | SCLit s -> L.build_global_stringptr s "str" builder
  | SNoexpr -> L.const_int i32_t 0
  | SId s -> L.build_load (lookup s) s builder
```haskell
| SAssign (s, e) -> let e' = expr builder e in
| ignore (L.build_store e' (lookup s) builder); e'
| SBinop ((A.Float, _) as e1, op, e2) ->
| let e1' = expr builder e1
| and e2' = expr builder e2 in
| (match op with
| A.Add -> L.build_fadd
| A.Sub -> L.build_fsub
| A.Mult -> L.build_fmul
| A.Div -> L.build_fdiv
| A.Mod ->
| raise (Failure "internal error: semant should have rejected mod
| on float")
| A.Equal -> L.build_fcmp L.Fcmp.Oeq
| A.Neq -> L.build_fcmp L.Fcmp.One
| A.Less -> L.build_fcmp L.Fcmp.Olt
| A.Greater -> L.build_fcmp L.Fcmp.Obge
| A.And | A.Or ->
| raise (Failure "internal error: semant should have rejected and
| or on float")
| )
| e1' e2' "tmp" builder
| SBinop (e1, op, e2) ->
| let e1' = expr builder e1
| and e2' = expr builder e2 in
| (match op with
| A.Add -> L.build_add
| A.Sub -> L.build_sub
| A.Mult -> L.build_mul
| A.Div -> L.build_sdiv
| A.Mod -> L.build_srem
| A.And -> L.build_and
| A.Or -> L.build_or
| A.Equal -> L.build_icmp L.Icmp.Eq
| A.Neq -> L.build_icmp L.Icmp.Ne
| A.Less -> L.build_icmp L.Icmp.Slt
| A.Greater -> L.build_icmp L.Icmp.Stge
| A.And | A.Or ->
| raise (Failure "internal error: semant should have rejected and
| or on float")
| )
| e1' e2' "tmp" builder
| SUnop (op, ((t, _) as e)) ->
| let e' = expr builder e in
| (match op with
| A.Neg when t = A.Float -> L.build_fneg
| A.Neg -> L.build_neg
| A.Not -> L.build_not) e' "tmp" builder
| SCall ("printf", [e]) | SCall ("printfb", [e]) ->
| L.build_call printf_func [ int_format_str ; (expr builder e) ]
| "printf" builder
| SCall ("initsdl", []) ->
| L.build_call initsdl_func [ ((L.const_int i32_t 0)) ] "initsdl" builder
| SCall ("printf", [e]) ->
| L.build_call printf_func [ float_format_str ; (expr builder e) ]
| "printf" builder
| SCall ("printfb", [e]) ->
| L.build_call printfb_func [ str_format_str ; (expr builder e) ] "
| printf" builder
| SCall ("getPlayeX", [e]) ->
| L.build_call getplayeX_func [ (expr builder e) ] "getPlayeX"
| builder
| SCall ("getPlayerY", [e]) ->
| L.build_call getplayeY_func [ (expr builder e) ] "getPlayerY"
| builder
| SCall ("setPlayerX", [e1;e2]) ->
| L.build_call setplayerx_func [ (expr builder e1); (expr builder e2
| ) ] "setPlayerX" builder
```

SCall ("setPlayerY", [e1; e2]) ->
L.build_call setplayery_func [\[(expr builder e1); (expr builder e2)\]] "setPlayerY" builder

SCall ("getPlayerControl", [e1; e2]) ->
L.build_call getplayercontrol_func [\[(expr builder e1); (expr builder e2)\]] "getPlayerControl" builder

SCall ("getPlayerText", [e]) ->
L.build_call getplayertext_func [\[(expr builder e)\]] "getPlayerText" builder

SCall ("getPlayerHx", [e]) ->
L.build_call getplayerhx_func [\[(expr builder e)\]] "getPlayerHx" builder

SCall ("getPlayerHy", [e]) ->
L.build_call getplayerhy_func [\[(expr builder e)\]] "getPlayerHy" builder

SCall ("addPlayerHitBox", [e1; e2; e3]) ->
L.build_call addplayerhitbox_func [\[(expr builder e1); (expr builder e2); (expr builder e3)\]] "addPlayerHitBox" builder

SCall ("changePlayerX", [e1; e2]) ->
L.build_call changerxx_func [\[(expr builder e1); (expr builder e2)\]] "changePlayerX" builder

SCall ("changePlayerY", [e1; e2]) ->
L.build_call changeplayery_func [\[(expr builder e1); (expr builder e2)\]] "changePlayerY" builder

SCall ("addPlayerControl", [e1; e2; e3]) ->
L.build_call addplayercontrol_func [\[(expr builder e1); (expr builder e2); (expr builder e3)\]] "addPlayerControl" builder

SCall ("controlPlayer", [e1; e2]) ->
L.build_call controlplayer_func [\[(expr builder e1); (expr builder e2)\]] "controlPlayer" builder

SCall ("getEntityX", [e]) ->
L.build_call getentityx_func [\[(expr builder e)\]] "getEntityX" builder

SCall ("getEntityY", [e]) ->
L.build_call getentity_y_func [\[(expr builder e)\]] "getEntityY" builder

SCall ("getEntityText", [e]) ->
L.build_call getentitytext_func [\[(expr builder e)\]] "getEntityText" builder

SCall ("getEntityHx", [e]) ->
L.build_call getentityhx_func [\[(expr builder e)\]] "getEntityHx" builder

SCall ("getEntityHy", [e]) ->
L.build_call getentityhy_func [\[(expr builder e)\]] "getEntityHy" builder

SCall ("setEntityX", [e1; e2]) ->
L.build_call setentityx_func [\[(expr builder e1); (expr builder e2)\]] "setEntityX" builder

SCall ("setEntityY", [e1; e2]) ->
L.build_call setentity_y_func [\[(expr builder e1); (expr builder e2)\]] "setEntityY" builder

SCall ("addEntityHitBox", [e1; e2; e3]) ->
L.build_call addentityhitbox_func [\[(expr builder e1); (expr builder e2); (expr builder e3)\]] "addEntityHitBox" builder

SCall ("changeEntityX", [e1; e2]) ->
L.build_call change_entityx_func [\[(expr builder e1); (expr builder e2)\]] "changeEntityX" builder

SCall ("changeEntityY", [e1; e2]) ->
L.build_call change_entityy_func [\[(expr builder e1); (expr builder e2)\]] "changeEntityY" builder

(* this function is called in runGame() to render entities and players at updated positions *)
let update_entity (t,n) = match t with
  A.Player -> L.build_call update_p_func [\[(expr builder (t,n))\]] "updateP" builder
  _ -> L.build_call update_e_func [\[(expr builder (t,n))\]] "updateE" builder
in
let update_entities l =

89
ignore(List.map update_entity l); update_entity (List.hd l)
in
update_entities object_vars
| SCall ("initEntities", []) ->
(* this function is called before runGame() iterates to load entity and
player textures*)
let init_entity (t,n) = match t with
  A.Player -> L.build_call initplayer_func [||expr builder (t
  ,n)||] "initPlayer" builder
|_ -> L.build_call initentity_func [||expr builder (t,n)||]
  "initEntity" builder
in
let init_entities l =
  ignore(List.map init_entity l); init_entity (List.hd l)
in
init_entities object_vars
| SCall ("prepareScene", []) ->
  L.build_call preparescene_func [||L.const_int i32_t 0||] "prepareScene" builder
| SCall ("presentScene", [e]) ->
  L.build_call presentscene_func [||expr builder e||] "presentScene" builder
| SCall ("isKeyPressed", [e]) ->
  L.build_call iskeypressed_func [||expr builder e||] "isKeyPressed" builder
| SCall ("changeEntityY", [e1; e2]) ->
  L.build_call change_entity_y_func [||expr builder e1||; ||expr
  builder e2||] "changeEntityY" builder
| SCall (f, args) ->
  let (fdef, fdecl) = StringMap.find f function_decls in
  let llargs = List.rev (List.map (expr builder) (List.rev args)) in
  let result = (match fdecl.styp with
    A.Void -> ""
    |_ -> f "" "result") in
  L.build_call fdef (Array.of_list llargs) result builder
| SNewPlayer (x, y, texture, n) ->
  let ptmp = L.build_alloca player_t "player_tmp" builder in
  let pptr = L.build_alloca (L.pointer_type player_t) "player_ptr"
    builder in
  let e1 = expr builder x
  and e2 = expr builder y
  and e3 = expr builder texture
  and e4 = expr builder (A.Int, SLiteral (0))
  and e5 = expr builder (A.Int, SLiteral (0))
  and e6 = expr builder n
  and e7 = expr builder texture in
  let xtmp = L.build_struct_gep ptmp 0 "x" builder in
  ignore (L.build_store e1 xtmp builder);
  let ytmp = L.build_struct_gep ptmp 1 "y" builder in
  ignore (L.build_store e2 ytmp builder);
  let texturetmp = L.build_struct_gep ptmp 2 "texture" builder in
  ignore (L.build_store e3 texturetmp builder);
  let hxtmp = L.build_struct_gep ptmp 3 "hx" builder in
  ignore (L.build_store e4 hxtmp builder);
  let hytmp = L.build_struct_gep ptmp 4 "hy" builder in
  ignore (L.build_store e5 hytmp builder);
  let ntmp = L.build_struct_gep ptmp 5 "n" builder in
  let ltype = ltype_of_typ A.Int in
  let size_t = L.build_intcast (L.size_of ltype) i32_t "tmp" builder
  in
  let total_size = L.build_mul size_t e6 "tmp" builder in
  let total_size = L.build_add total_size (L.const_int i32_t 1) "tmp"
    builder in

90
let arr_malloc = L. build_array_malloc ltype total_size "tmp" builder in
let arr = L. build_pointercast arr_malloc (L. pointer_type ltype) "tmp" builder in
ignore (L. build_store arr ntmp builder);
ignore (L. build_store ptmp pptr builder);
ignore (L. build_call zeroplayercontrols_func [ | ptmp | ] "zeroPlayerControls" builder);
let texture2tmp = L. build_struct_gep ptmp 6 "text" builder in
ignore (L. build_store e7 texture2tmp builder);
L. build_load pptr "" builder

| SNewEntity (x, y, texture) ->
let etmp = L. buildalloca entity_t "entity_tmp" builder in
let etmp = L. buildalloca (L. pointer_type entity_t) "entity_ptr" builder in
let e1 = expr builder x
and e2 = expr builder y
and e3 = expr builder texture
and e4 = expr builder texture
and e5 = expr builder (A. Int, SLiteral (0))
and e6 = expr builder (A. Int, SLiteral (0)) in
let xtmp = L. build_struct_gep etmp 0 "x" builder in
ignore (L. build_store e1 xtmp builder);
let ytmp = L. build_struct_gep etmp 1 "y" builder in
ignore (L. build_store e2 ytmp builder);
let texturetmp = L. build_struct_gep etmp 2 "texture" builder in
ignore (L. build_store e3 texturetmp builder);
let texture2tmp = L. build_struct_gep etmp 3 "text" builder in
ignore (L. build_store e4 texture2tmp builder);
let hxtmp = L. build_struct_gep etmp 4 "hx" builder in
ignore (L. build_store e5 hxtmp builder);
let hytmp = L. build_struct_gep etmp 5 "hy" builder in
ignore (L. build_store e6 hytmp builder);
ignore (L. build_store etmp eptr builder);
L. build_load eptr "" builder

in

(* LLVM insists each basic block end with exactly one "terminator" instruction that transfers control. This function runs "instr builder "
if the current block does not already have a terminator. Used,
e.g., to handle the "fall off the end of the function" case. *)
let add_terminator builder instr =
match L. block_terminator (L. insertion_block builder) with
  Some _ -> ()
| None -> ignore (instr builder)
in

(* Build the code for the given statement; return the builder for
the statement's successor (i.e., the next instruction will be built
after the one generated by this call) *)
let rec stmt builder = function
  SBlock sl -> L. fold_left stmt builder sl
| SExpr e -> ignore (expr builder e); builder
| SReturn e -> ignore (match fdecl.styp with
  (* Special "return nothing" instr *)
  A. Void -> (match fdecl.sfname with
    "main" -> L. build_ret (L. const_int i32_t 0) builder
    | _ -> L. build_ret_void builder
  )
  (* Build return statement *)
)
| _  -> L. build_ret (expr builder e) builder |
| SIf (predicate, then_stmt, else_stmt) -> |
| let bool_val = expr builder predicate in |
| let build_bb = L. append_block context "merge" the_function in |
| let build_br_merge = L. build_br merge_bb in (* partial function *) |
| let then_bb = L. append_block context "then" the_function in |
| add_terminal (stmt (L.builder_at_end context then_bb) then_stmt) |
| build_br_merge |
| let else_bb = L. append_block context "else" the_function in |
| add_terminal (stmt (L.builder_at_end context else_bb) else_stmt) |
| build_br_merge |
| ignore(L. build_cond_br bool_val then_bb else_bb builder); |
| L.builder_at_end context merge_bb |
| SWhile (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 |
| (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 |
| (* Implement for loops as while loops *) |
| SFor (e1, e2, e3, body) -> stmt builder |
| (SBlock [SExpr e1; SWhile (e2, SBlock [body; SExpr e3])]) |
| (*Implement runGame() as a while loop with calls to initialize textures |
| and render objs*) |
| SRunGame (e1, e2, body) -> stmt builder |
| (SBlock [SExpr (A.Void, SCall("initEntities",[])); |
| SWhile (e1, SBlock [SExpr (A.Void, SCall("prepareScene",[])); |
| SExpr (A.Void, SCall("updateEntities",[])); |
| body; SExpr (A.Void, SCall("presentScene",[e2])))]) |
| in |
| (* Build the code for each statement in the function *) |
| let builder = stmt builder (SBlock fdecl.sbody) in |
| (* Add a return if the last block falls off the end *) |
| add_terminal builder (match fdecl.styp with |
| A.Void -> (match fdecl.sfname with |
| "main" -> L.build_ret (L.const_int i32_t 0) |
| _  -> L.build_ret_void |
| ) |
| A.Float -> L.build_ret (L.const_float float_t 0.0) |
| t -> L.build_ret (L.const_int (ltype_of_typ t) 0)) |
| in |
| List.iter build_function_body functions; |
| the_module |
8.8 lucifer.ml

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

type action = Ast | Sast | LLVM_IR | Compile

let () =
  let action = ref Compile in
  let set_action a () = action := a in
  let speclist = [  
    ("-a", Arg.Unit (set_action Ast), "Print the AST"),  
    ("-s", Arg.Unit (set_action Sast), "Print the SAST"),  
    ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR"),  
    ("-c", Arg.Unit (set_action Compile),  
      "Check and print the generated LLVM IR (default)"),
  ] in
  let usage_msg = "usage: ./lucifer.native [-a|-s|-l|-c] [file.luc]" in
  let channel = ref stdin in
  Arg.parse speclist (fun filename -> channel := open_in filename) usage_msg ;

  let lexbuf = Lexing.from_channel !channel in
  let ast = Luciferparse.program Scanner.token lexbuf in
  match !action with
    Ast -> print_string (Ast.string_of_program ast)
  | _ -> let sast = Semant.check ast in
    match !action with
      Ast -> ()
    | Sast -> print_string (Sast.string_of_sprogram sast)
    | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate sast))
    | Compile -> let m = Codegen.translate sast in
      Llvm_analysis.assert_valid_module m;
      print_string (Llvm.string_of_llmodule m)
8.9 C files

8.9.1 entity.c

```c
#include <stdio.h>
#include <string.h>
#include "./src/entity.h"

typedef struct Entity {
    int x;
    int y;
    char *texture;
    SDL_Texture *text;
    int hx;
    int hy;
} Entity;

int getEntityX(Entity *e) {
    return e->x;
}

int getEntityY(Entity *e) {
    return e->y;
}

char *getEntityText(Entity *e) {
    return e->texture;
}

int getEntityHx(Entity *e) {
    return e->hx;
}

int getEntityHy(Entity *e) {
    return e->hy;
}

void setEntityX(Entity *e, int n) {
    e->x = n;
}

void setEntityY(Entity *e, int n) {
    e->y = n;
}

void addEntityHitBox(Entity *e, int g, int h) {
    e->hx = g;
    e->hy = h;
}

void changeEntityX(Entity *e, int dx) {
    e->x = e->x + dx;
}
```

94
void changeEntityY (Entity *e, int dy)
{
    e->y = e->y + dy;
}

void initEntity (Entity *e) {
    e->text = loadTexture(e->texture);
}

int updateE (Entity *e) {
    doInput();
    blit(e->text, e->x, e->y);
    return 0;
}

#ifdef BUILD_TEST
int main () {
    Entity e;
    e.x = 1;
    e.y = 2;
    e.texture = "image.png";
    e.hx = 3;
    e.hy = 4;
    getEntityX(e);
}
#endif

8.9.2 player.c

/*
 * Player struct
 * Authors: Elliott Morelli, Cherry Chu
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include "./src/player.h"

typedef struct Player {
    int x;
    int y;
    char *texture;
    int hx;
    int hy;
    int *controls;
    SDL_Texture *text;
} Player;

int getPlayerX (Player *p) {
    return p->x;
}

int getPlayerY (Player *p) {
    return p->y;
}
void setPlayerX (Player *p, int x) {
    p->x = x;
}
void setPlayerY (Player *p, int x) {
    p->y = x;
}
int getPlayerControl (Player *p, int index) {
    return p->controls[index];
}
void zeroPlayerControls (Player *p) {
    for (int i = 0; i < 10; i++) {
        p->controls[i] = 0;
    }
}
char *getPlayerText (Player *p) {
    return p->texture;
}
int getPlayerHx (Player *p) {
    return p->hx;
}
int getPlayerHy (Player *p) {
    return p->hy;
}
void addPlayerHitBox (Player *p, int x, int y) {
    p->hx = x;
    p->hy = y;
}
void changePlayerX (Player *p, int dx) {
    p->x = p->x + dx;
}
void changePlayerY (Player *p, int dy) {
    p->y = p->y + dy;
}
bool isKeyPressed (int k) {
    if (app.keyboard[k]) {
        return true;
    } else {
        return false;
    }
}
void controlPlayer (Player *p, int speed) {
    if (isKeyPressed(p->controls[0])) {
        changePlayerY(p, -speed);
    }
}
if(isKeyPressed(p->controls[1])){
    changePlayerY(p, speed);
}
if(isKeyPressed(p->controls[2])){
    changePlayerX(p, -speed);
}
if(isKeyPressed(p->controls[3])){
    changePlayerX(p, speed);
}
}

void initPlayer(Player *p){
    p->text = loadTexture(p->texture);
}

int updateP(Player *p){
    doInput();
    blit(p->text, p->x, p->y);
    return 0;
}

void addPlayerControl(Player *p, int index, int code) {
    p->controls[index] = code;
}

#define BUILD_TEST
int main()
{
    Player p;
    p.x = 1;
    p.y = 2;
    p.texture = "image.png";
    zeroPlayerControl(&p, 0);
    getPlayerX(&p);
}
#endif
8.10 Test file bodies & outputs

8.10.1 fail-after-return.luc

```luc
fun main void () {
    int i;
    i = 1;
    print(i);
    return 0;
}
```

8.10.2 fail-after-return.err

Fatal error: exception Failure("return gives int expected void in 0")

8.10.3 fail-arith1.luc

```luc
fun main void () {
    print(1+2+);
}
```

8.10.4 fail-arith1.err

Fatal error: exception Parsing.Parse_error

8.10.5 fail-arith2.luc

```luc
fun main void () {
    print(1 + 2 3 + 4)
}
```

8.10.6 fail-arith2.err

Fatal error: exception Parsing.Parse_error

8.10.7 fail-arith3.luc

```luc
fun main void () {
    x = x + 42;
    print(x);
}
```

8.10.8 fail-arith3.err

Fatal error: exception Failure("undeclared identifier x")
8.10.9  fail-assign1.luc

```luc
fun main void () {
  int i;
  bool b;
  i = 42;
  i = 5;
  b = true;
  b = false;
  i = false; /* Fail: assigning a bool to an integer */
}
```

8.10.10  fail-assign1.err

Fatal error: exception Failure("illegal assignment int = bool in i = false")

8.10.11  fail-assign2.luc

```luc
fun main void () {
  int i;
  bool b;
  b = 42; /* illegal assignment of int to a bool */
}
```

8.10.12  fail-assign2.err

Fatal error: exception Failure("illegal assignment bool = int in b = 42")

8.10.13  fail-assign3.luc

```luc
fun the_void void () {
  return;
}
fun main void () {
  int i;
  i = the_void(); /* fails: assignment of void to an integer */
}
```

8.10.14  fail-assign3.err

Fatal error: exception Failure("illegal assignment int = void in i = the_void ()")
8.10.15 fail-comment.luc

```luc
/*
 * main function of program
 * fun main void () {
   return;
}
```

8.10.16 fail-comment.err

Fatal error: exception Failure("lexing: empty token")

8.10.17 fail-controlPlayer-rungame.luc

```luc
fun main void () {
    int x;
    Player p;
    x = 200;
    initsdl(2);
    p = new Player(50, 20, "gfx/bat.png", 10);
    p.addPlayerControl(0, 24); /* this is the 'u' key */
    p.addPlayerControl(1, 81);
    p.addPlayerControl(2, 80);
    p.addPlayerControl(3, 79);
    runGame(true; 1){
        controlPlayer(10);
    }
}
```

8.10.18 fail-controlPlayer-rungame.err

Fatal error: exception Failure("expecting 2 arguments in controlPlayer(10)")

8.10.19 fail-declEntity-changeXY.luc

```luc
fun main void() {
    Entity e;
    initsdl();
    print(e.getEntityX());
    print(e.getEntityY());
    e.changeEntityX(10);
    e.changeEntityY(-20);
    print(e.getEntityX());
    print(e.getEntityY());
}
```
8.10.20 fail-declEntity-changeXY.err

Fatal error: exception Failure("all entities and players must be instantiated")

8.10.21 fail-entity-uninitialized.luc

```luc
fun main void () {
    int x;
    Entity e;
    initsdl();
    x = 2;
    if (x > 0) {
        x = x - 1;
    }
}
```

8.10.22 fail-entity-uninitialized.err

Fatal error: exception Failure("all entities and players must be instantiated")

8.10.23 fail-entityInvalidName1.luc

```luc
fun main void () {
    Entity E1;
    E1 = new Entity(0, 0, "texture.png");
}
```

8.10.24 fail-entityInvalidName1.err

Fatal error: exception Parsing.Parse_error

8.10.25 fail-entityInvalidName2.luc

```luc
fun main void () {
    Entity 1e;
    1e = new Entity(0, 0, "image.png");
}
```

8.10.26 fail-entityInvalidName2.err

Fatal error: exception Parsing.Parse_error
fun main void () {
    Entity e;
    e = new Entity(1, 3, 10);
}

Fatal error: exception Failure("expecting String type for texture argument")

fun main void () {
    Entity e;
    e = new Entity(1.0, 3, "image.png");
}

Fatal error: exception Failure("expecting Int type for x position argument")

fun main void () {
    Entity e;
    e = new Entity(1, "texture.png", "image.png");
}

Fatal error: exception Failure("expecting Int type for y position argument")

fun main void () {
    bool a;
    int b;
    a + b;
}

Fatal error: exception Failure("illegal binary operator bool + int in a + b")
8.10.35 fail-for1.luc

```plaintext
fun main void () {
  int i;
  for (j = 0; i < 10; i = i + 1) {} /* j undefined */
}
```

8.10.36 fail-for1.err

```
Fatal error: exception Failure("undeclared identifier j")
```

8.10.37 fail-for2.luc

```plaintext
fun main void () {
  int i;
  for (i = 0; j < 10; i = i + 1) {} /* j undefined */
}
```

8.10.38 fail-for2.err

```
Fatal error: exception Failure("undeclared identifier j")
```

8.10.39 fail-fun1.luc

```plaintext
fun foo int () {}
fun bar int () {}
fun baz int () {}
fun bar void () {}
fun main void () {
}
```

8.10.40 fail-fun1.err

```
Fatal error: exception Failure("duplicate function bar")
```

8.10.41 fail-fun10.luc

```plaintext
fun foo void (int a) {
  print(a * 2);
  return;
  print(a - 2);
}
```
fun main void () {
foo(40);
}

8.10.42 fail-fun10.err
Fatal error: exception Failure("nothing may follow a return")

8.10.43 fail-fun2.luc
fun foo int (int a, bool b, int c) {}
fun bar void (int a, bool b, int a) {} 
fun main void () {
}

8.10.44 fail-fun2.err
Fatal error: exception Failure("duplicate formal a")

8.10.45 fail-fun3.luc
fun foo int (int a, bool b, int c) {}
fun bar void (int a, void b, int c) {}
fun main void () {
}

8.10.46 fail-fun3.err
Fatal error: exception Failure("illegal void formal b")

8.10.47 fail-fun4.luc
fun print int () {}
fun main void () {
}

8.10.48 fail-fun4.err
Fatal error: exception Failure("function print may not be defined")
8.10.49 fail-fun5.luc

```luc
fun foo int () {}
fun bar int () {
  int a;
  void b;
  bool c;
  return 0;
}
fun main void () {
}
```

8.10.50 fail-fun5.err

Fatal error: exception Failure("illegal void local b")

8.10.51 fail-fun6.luc

```luc
fun foo void ( int a, bool b) {}
fun main void () {
  foo(42 , true);
  foo(42);
}
```

8.10.52 fail-fun6.err

Fatal error: exception Failure(" expecting 2 arguments in foo(42)")

8.10.53 fail-fun7.luc

```luc
fun foo void ( int a, bool b) {}
fun main void () {
  foo(42 , true);
  foo(42 , true, false);
}
```

8.10.54 fail-fun7.err

Fatal error: exception Failure(" expecting 2 arguments in foo(42, true, false)"
fun foo void (int a, bool b) {} 
fun bar void () {} 
fun main void () { 
  foo(42, true); 
  foo(42, bar()); 
}

Fun: fatal error: exception Failure("illegal argument found void expected bool in bar()")

fun foo void (int a, bool b) {} 
fun main void () { 
  foo(42, true); 
  foo(42, 42); 
}

Fun: fatal error: exception Failure("illegal argument found int expected bool in 42")

main int () { 
  prints(); 
  return 0; 
}

Fun: fatal error: exception Parsing.Parse_error

fun main int () { 
  prints("hello world") 
  return 0; 
}
8.10.62  fail-hello-world-nosemi.err

Fatal error: exception Parsing.Parse_error

8.10.63  fail-hello-world-openstring.luc

```luc
fun main int () {
  prints("hello world");
  return 0;
}
```

8.10.64  fail-hello-world-openstring.err

Fatal error: exception Failure("illegal character ")

8.10.65  fail-hello-world1.luc

```luc
fun main int () {
  prints();
  return 0;
}
```

8.10.66  fail-hello-world1.err

Fatal error: exception Failure("expecting 1 arguments in prints()")

8.10.67  fail-hello-world2.luc

```luc
fun main int () {
  prints(12);
  return 0;
}
```

8.10.68  fail-hello-world2.err

Fatal error: exception Failure("illegal argument found int expected string in 12")

8.10.69  fail-ifl.luc

```luc
fun main void () {
  if (true) {} 
  if (false) {} else {} 
  if (42) {} /* Failz: predicate doesn't evaluate to bool */
}
```
8.10.70  fail-if1.err

Fatal error: exception Failure("expected Boolean expression in 42")

8.10.71  fail-if2.luc

```java
fun main void () {
    if (true) {
        foo;  /* Fails: undeclared variable */
    }
}
```

8.10.72  fail-if2.err

Fatal error: exception Failure("undeclared identifier foo")

8.10.73  fail-if3.luc

```java
fun main void () {
    if (true) {
        42;
    } else {
        bar;  /* Error: undeclared variable */
    }
}
```

8.10.74  fail-if3.err

Fatal error: exception Failure("undeclared identifier bar")

8.10.75  fail-no-main.luc

8.10.76  fail-no-main.err

Fatal error: exception Failure("unrecognized function main")

8.10.77  fail-objectFunctionArgOrder.luc

```java
fun main void () {
    Entity e;
    e = new Entity(0, 0, "image.png");
    setEntityX(10, e);
    print(e.getEntityX());
}
```
8.10.78  fail-objectFunctionArgOrder.err

Fatal error: exception Failure("illegal argument found int expected Entity in 10")

8.10.79  fail-playerInvalidSize1.luc

fun main void () {
  Player p1;
  p1 = new Player(10, 10, "image.png", 3);
}

8.10.80  fail-playerInvalidSize1.err

Fatal error: exception Failure("expecting a minimum size 4(up,down,left,right ) for control array")

8.10.81  fail-playerInvalidSize2.luc

fun main void () {
  Player p1;
  p1 = new Player(10, 10, "image.png", -1);
}

8.10.82  fail-playerInvalidSize2.err

Fatal error: exception Failure("expecting a minimum size 4(up,down,left,right ) for control array")

8.10.83  fail-playerInvalidSize3.luc

fun main void () {
  Player p1;
  p1 = new Player(10, 10, "image.png", 1.3);
}

8.10.84  fail-playerInvalidSize3.err

Fatal error: exception Failure("expecting Int type for control array size argument")

8.10.85  fail-playerInvalidTexture.luc

fun main void () {
  Player p1;
  p1 = new Player(1, 2, 3, 4);
}
8.10.86 fail-playerInvalidTexture.err

Fatal error: exception Failure("expecting String type for texture argument")

8.10.87 fail-playerInvalidX.luc

```java
fun main void () {
    Player p;
    p = new Player (1.0, 2, "texture.png", 4);
}
```

8.10.88 fail-playerInvalidX.err

Fatal error: exception Failure("expecting Int type for x position argument")

8.10.89 fail-playerInvalidY.luc

```java
fun main void () {
    Player p1;
    p1 = new Player(1, true, "texture.png", 4);
}
```

8.10.90 fail-playerInvalidY.err

Fatal error: exception Failure("expecting Int type for y position argument")

8.10.91 fail-rungame-entity-decl.luc

```java
fun main void () {
    int x;
    Entity e;
    initsdl();
    x = 2;
    runGame(x > 0; 1){
        x = x - 1;
    }
}
```

8.10.92 fail-rungame-entity-decl.err

Fatal error: exception Failure("all entities and players must be instantiated")

110
8.10.93 fail-rungame-init.luc

```luc
fun main void () {
    Player p;
    runGame()
        p = new Player(10, 20, "texture.png", 10);
    }
}
```

8.10.94 fail-rungame-init.err

```
Fatal error: exception Parsing.Parse_error
```

8.10.95 fail-rungame-undecl-entities.luc

```luc
fun main void () {
    int x;
    x = 100;
    initsdl();
    e = new Entity(100, 50, "gfx/playerBullet.png");
    e2 = new Entity(100, 100, "gfx/bat.jpg");
    runGame(x > 0; 1){
        x = x - 1;
    }
}
```

8.10.96 fail-rungame-undecl-entities.err

```
Fatal error: exception Failure("undeclared identifier e2")
```

8.10.97 fail-sdl-init.luc

```luc
fun main void () {
    initsdl(2);
}
```

8.10.98 fail-sdl-init.err

```
Fatal error: exception Failure("expecting 0 arguments in initsdl(2)")
```

8.10.99 fail-singleComment.luc

```luc
fun main void () {
    int r;
    r = 3.7 / \text{radius of circle}
    return 0;
}
```
8.10.100  fail-singleComment.err

Fatal error: exception Parsing.Parse_error

8.10.101  fail-structNotCap.luc

```java
struct myStruct {
    int field1;
    float field2;
}
fun main void () {
}
```

8.10.102  fail-structNotCap.err

Fatal error: exception Parsing.Parse_error

8.10.103  fail-while1.luc

```java
fun main void () {
    int i;
    while (42) {
        /* must evaluate to boolean */
        i = i + 1;
    }
}
```

8.10.104  fail-while1.err

Fatal error: exception Failure("expected Boolean expression in 42")

8.10.105  fail-while2.luc

```java
fun main void () {
    int i;
    while (true) {
        foo();
        /* undefined function*/
    }
}
```

8.10.106  fail-while2.err

Fatal error: exception Failure("unrecognized function foo")
8.10.107 test-addEntityHitBox.luc

```luc
fun main void() {
    Entity e;
    e = new Entity(11, 22, "image.png");
    e.addEntityHitBox(40, 60);
    print(e.getEntityHx());
    print(e.getEntityHy());
}
```

8.10.108 test-addEntityHitBox.out

```
40
60
```

8.10.109 test-addPlayerControl.luc

```luc
fun main void () {
    Player p;
    p = new Player(10, 20, "texture.png", 10);
    p.addPlayerControl(7, 82);
    print(p.getPlayerControl(7));
}
```

8.10.110 test-addPlayerControl.out

```
82
```

8.10.111 test-addPlayerControl2.luc

```luc
fun main void () {
    Player p;
    p = new Player(10, 20, "texture.png", 10);
    p.addPlayerControl(7, SDL_SCANCODE_DOWN);
    print(p.getPlayerControl(7));
}
```

8.10.112 test-addPlayerControl2.out

```
81
```

8.10.113 test-addPlayerHitBox.luc

```luc
fun main void() {
    Player p;
    p = new Player(11, 22, "image.png", 10);
    p.addPlayerHitBox(40, 60);
    print(p.getPlayerHx());
    print(p.getPlayerHy());
}
```
fun main void () {
    print(1+2);
}

fun main void () {
    print(1 + 2 * 3 + 4);
}

int x;
x = 42;
x = x + 8;
print(x);
8.10.121  test-changeEntityXY.luc

```luc
fun main void() {
    Entity e;
    e = new Entity(11, 22, "image.png");
    print(e.getEntityX());
    print(e.getEntityY());
    e.changeEntityX(10);
    e.changeEntityY(-20);
    print(e.getEntityX());
    print(e.getEntityY());
}
```

8.10.122  test-changeEntityXY.out

```
11
22
21
2
```

8.10.123  test-changePlayerXY.luc

```luc
fun main void() {
    Player p;
    p = new Player(11, 22, "image.png", 10);
    print(p.getPlayerX());
    print(p.getPlayerY());
    p.changePlayerX(10);
    p.changePlayerY(-20);
    print(p.getPlayerX());
    print(p.getPlayerY());
}
```

8.10.124  test-changePlayerXY.out

```
11
22
21
2
```

8.10.125  test-comment.luc

```luc
/*
 * add(x, y) adds the two arguments
 * and returns the result
 */
fun add int (int x, int y) {
    return x + y;
}
fun main void () {
    print(add(2, 4));
}
```
fun main void() {
    Entity e;
    e = new Entity(0, 0, "image.png");
    print(e.getEntityX());
    print(e.getEntityY());
    print(e.getEntityText());
}

0
0
image.png

fun main void() {
    Entity e;
    e = new Entity(0, 13 * 2, "image.png");
    print(e.getEntityX());
    print(e.getEntityY());
    print(e.getEntityText());
}

0
26
image.png

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

fun main void () {
    print(fib(0));
    print(fib(1));
    print(fib(2));
    print(fib(3));
    print(fib(4));
    print(fib(5));
}
8.10.132 test-fib.out

1
1
2
3
5
8

8.10.133 test-for1.luc

```luc
fun main void () {
    int i;
    for(i = 0; i < 5; i = i + 1) {
        print(i);
    }
    print(42);
}
```

8.10.134 test-for1.out

0
1
2
3
4
42

8.10.135 test-for2.luc

```luc
fun main void () {
    int i;
    i = 0;
    for ( ; i < 5; ) {
        print(i);
        i = i + 1;
    }
    print(42);
}
```

8.10.136 test-for2.out

0
1
2
3
4
42
fun add int (int a, int b) {
    return a + b;
}

fun main void () {
    int a;
    a = add(32, 10);
    print(a);
}

fun printer void (int a, int b, int c, int d) {
    print(a);
    print(b);
    print(c);
    print(d);
}

fun main void () {
    printer(42, 8, 4, 2);
}

fun add int (int a, int b) {
    int c;
    c = a + b;
    return c;
}

fun main void () {
    int d;
    d = add(42, 10);
    print(d);
}

fun add int (int a, int b) {
    int c;
    c = a + b;
    return c;
}

fun main void () {
    int d;
    d = add(42, 10);
    print(d);
}

42
4
8
2

52
fun foo int (int a) {
    return a;
}
fun main void () {
}

fun foo void () {}
fun bar int (int a, bool b, int c) {
    return a + c;
}
fun main int () {
    print(bar(17, false, 25));
}

int a;
fun foo void (int c) {
    a = c + 42;
}
fun main void () {
    foo(10);
    print(a);
}
fun foo void (int a) {
    print(a + 3);
}
fun main void() {
    foo(40);
}

fun foo void (int a) {
    print(a + 2);
    return;
}
fun main void () {
    foo(40);
}

fun foo int (int x, int y) {
    return 0;
}
fun main void () {
    int i;
    i = 1;
    foo(i = 4, i = i+1);
    print(i);
}
fun gcd int (int a, int b) {
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}

fun main void () {
    print(gcd(2, 14));
    print(gcd(3, 15));
    print(gcd(14, 21));
    print(gcd(8, 36));
    print(gcd(99, 121));
}

fun main void () {
    print(" hello world");
}

fun main void () {
    string s;
    string t;
    t = "heck";
    s = "hello world";
    print(s);
}

fun main void () {
    string s;
    string t;
    t = "heck";
    s = "hello world";
    print(s);
}

fun main void () {
    string s;
    string t;
    t = "heck";
    s = "hello world";
    print(s);
}
8.10.161 test-hello-world.luc

```luc
fun main void () {
    prints("hello world");
}
```

8.10.162 test-hello-world.out

```
hello world
```

8.10.163 test-if1.luc

```luc
fun main void () {
    if (true) print(42);
    print(17);
}
```

8.10.164 test-if1.out

```
42
17
```

8.10.165 test-if2.luc

```luc
fun main void () {
    if (true) print(42); else print(8);
    print(17);
}
```

8.10.166 test-if2.out

```
42
17
```

8.10.167 test-if3.luc

```luc
fun main void () {
    if (false) print(42);
    print(17);
}
```

8.10.168 test-if3.out

```
17
```
fun main void () {
    if (false) print(42); else print(8);
    print(17);
}

fun cond int (bool b) {
    int x;
    if (b)
        x = 42;
    else
        x = 17;
    return x;
}

fun main void () {
    print(cond(true));
    print(cond(false));
}

fun cond int (bool b) {
    int x;
    x = 10;
    if (b)
        if (x == 10)
            x = 42;
    else
        x = 17;
    return x;
}

fun main void () {
    print(cond(true));
    print(cond(false));
}
test-if6.out

```
42
10
```

test-modulo.luc

```
fun main void()
    print(1%2);
}
```

test-modulo.out

```
1
```

test-objectFunction1.luc

```
fun main void () {
    Entity e;
    e = new Entity(0, 0, "image.png");
    e.setEntityX(10);
    print(e.getEntityX());
}
```

test-objectFunction1.out

```
10
```

test-objectFunction2.luc

```
fun main void () {
    Entity e;
    e = new Entity(0, 0, "image.png");
    setEntityX(e, 10);
    print(e.getEntityX());
}
```

test-objectFunction2.out

```
10
```
8.10.181 test-ops1.luc

```luc
fun main void () {
    print(1 + 2);
    print(1 - 2);
    print(1 * 2);
    print(100 / 2);
    print(99);
    printb(1 == 2);
    printb(1 == 1);
    print(99);
    printb(1 != 2);
    printb(1 != 1);
    print(99);
    printb(1 < 2);
    printb(2 < 1);
    print(99);
    printb(1 <= 2);
    printb(1 <= 1);
    print(99);
    printb(1 > 2);
    printb(2 > 1);
    print(99);
    printb(1 >= 2);
    printb(1 >= 1);
    print(99)
}
```

8.10.182 test-ops1.out

```
3
-1
2
50
99
0
1
99
1
0
99
1
0
99
1
1
0
99
0
1
99
0
1
1
```

8.10.183 test-ops2.luc

```luc
fun main void () {
    printb(true);
    printb(false);
}
```
printb(true && true);
printb(true && false);
printb(false && true);
printb(false && false);
printb(true || true);
printb(true || false);
printb(false || true);
printb(false || false);
printb(!false);
printb(!true);
print(-10);
print(-42);

8.10.184  test-ops2.out

1
0
1
0
0
0
1
1
1
0
1
0
0
-10
42

8.10.185  test-player.luc

fun main void() {
    Player p;
p = new Player(0, 0, "texture.png", 4);
print(p.getPlayerX());
print(p.getPlayerY());
print(p.getPlayerControl(0));
print(p.getPlayerControl(1));
print(p.getPlayerControl(2));
print(p.getPlayerControl(3));
}

8.10.186  test-player.out

0
0
0
0
0
0

8.10.187  test-playerExprArg.luc
fun main void () {
    Player p;
    p = new Player(5 + 5, 0, "texture.png", 4);
    print(p.getPlayerX());
    print(p.getPlayerY());
}

8.10.188 test-playerExprArg.out
10
0

8.10.189 test-playerExtend.luc
fun main void () {
    Player p;
    int i;
    p = new Player(0, 0, "image.png", 10);
    i = 0;
    while (i < 10) {
        p.changePlayerX(2);
        i = i + 1;
        print(p.getPlayerX());
    }
}

8.10.190 test-playerExtend.out
2
4
6
8
10
12
14
16
18
20

8.10.191 test-singleComment.luc
fun main void () {
    float r;
    r = 1.2; // r is radius of circle
    printf(r);
}

8.10.192 test-singleComment.out
1.2
8.10.193  test-while1.luc

```luc
fun main void () {
  int i;
  i = 5;
  while (i > 0) {
    print(i);
    i = i - 1;
  }
  print(42);
}
```

8.10.194  test-while1.out

```
5
4
3
2
1
42
```

8.10.195  test-while2.luc

```luc
fun foo int (int a) {
  int j;
  j = 0;
  while (a > 0) {
    j = j + 2;
    a = a - 1;
  }
  return j;
}
fun main void () {
  print(foo(7));
}
```

8.10.196  test-while2.out

```
14
```

8.10.197  visual-big-program.luc

```luc
fun checkCollision bool (Entity e, Player p) {
  int pRightX;
  int pDownY;
  int eRightX;
  int eDownY;
  pRightX = p.getPlayerX() + p.getPlayerHx();
  pDownY = p.getPlayerY() + p.getPlayerHy();
```
eRightX = e.getEntityX() + e.getEntityHx;
eDownY = e.getEntityY() + e.getEntityHy;

if(p.getPlayerX() == pRightX || p.getPlayerY() == pDownY || e.getEntityX() == eRightX || e.getEntityY() == eDownY){
    return false;
}

if(p.getPlayerX() >= eRightX || e.getEntityX() >= pRightX){
    return false;
}

if(p.getPlayerY() >= eDownY || e.getEntityY() >= pDownY){
    return false;
}

return true;

fun main void () {
    Player knight;
    Entity e;
    Entity e2;
    Entity e3;
    Entity winscreen;
    int lives;
    bool down;
    bool collide;
    bool running;

collide = false;
down = true;
running = true;
initsdl();
lives = 3;
knight = new Player(0, 200, "gfx/knight.png", 4);
knight.addPlayerControl(0, SDL_SCANCODE_UP);
knight.addPlayerControl(1, 81);
knight.addPlayerControl(2, 80);
knight.addPlayerControl(3, 79);

e = new Entity(300, 0, "gfx/bat.png");
e2 = new Entity(550, 0, "gfx/bat.png");
e3 = new Entity(800, 0, "gfx/bat.png");
winscreen = new Entity(-500, -500, "gfx/youwin.png");

runGame(running; 60){
    knight.addPlayerHitBox(80, 80);
e.addEntityHitBox(100, 100);
e2.addEntityHitBox(120, 120);
e3.addEntityHitBox(120, 120);

collide = (checkCollision(e, knight) || checkCollision(e2, knight) ||
    checkCollision(e3, knight));
knight.controlPlayer(15);
if(e.getEntityY() >= 700)
    down = false;

if(e.getEntityY() <= 5) {
    down = true;
}

if(down == true)
    e2.changeEntityY(15);
    e.changeEntityY(25);
    e3.changeEntityY(35);

if(down == false) {
    e2.changeEntityY(-15);
    e.changeEntityY(-25);
    e3.changeEntityY(-35);
}

/* sends the knight back to the beginning */
if(collide){
    lives = lives - 1;
    knight.setPlayerX(0);
    knight.setPlayerY(200);
}

/* ends game if out of lives */
if(lives <= 0){
    prints("Game Over");
    running = false;
}

/* shows win screen*/
if(knight.getPlayerX() + knight.getPlayerHx() >= 1200){
    winscreen.setEntityX(200);
    winscreen.setEntityY(100);
}

}
8.10.199 visual-iskeypressed-moveentity.luc

```luc
fun main void () {
    int x;
    Entity e;
    Entity e2;
    x = 200;
    initsdl();
    e = new Entity(100, 50, "gfx/playerBullet.png");
    e2 = new Entity(100, 100, "gfx/bat.jpg");
    runGame(x > 0; 1){
        x = x - 1;
        if(isKeyPressed(4)){
            e.changeEntityX(10);
        }
    }
}
```

8.10.200 visual-iskeypressed.luc

```luc
fun main void () {
    int x;
    Entity e;
    Entity e2;
    x = 200;
    initsdl();
    e = new Entity(100, 50, "gfx/playerBullet.png");
    e2 = new Entity(100, 100, "gfx/bat.jpg");
    runGame(x > 0; 1){
        x = x - 1;
        if(isKeyPressed(4)){
            prints("a pressed");
        }
    }
}
```

8.10.201 visual-moving-entities.luc

```luc
fun main void () {
    int x;
    Entity e;
    Entity e2;
    ```
x = 600;
initsdl();
e = new Entity(100, 50, "gfx/playerBullet.png");
e2 = new Entity(100, 100, "gfx/bat.png");

runGame(x > 0; 1){
    x = x - 1;
    if(isKeyPressed(4)){
        e.changeEntityX(-10);
    }
    if(isKeyPressed(7)){
        e.changeEntityX(10);
    }
    if(isKeyPressed(82)){
        e2.changeEntityY(-10);
    }
    if(isKeyPressed(81)){
        e2.changeEntityY(10);
    }
    if(isKeyPressed(80)){
        e2.changeEntityX(-10);
    }
    if(isKeyPressed(79)){
        e2.changeEntityX(10);
    }
}

8.10.202 visual-render-player.luc

fun main void () {
    int x;
    Player p;

    x = 200;
initsdl();
p = new Player(10, 20, "gfx/bat.png", 10);

    runGame(x > 0; 60){
        x = x - 1;
    }
}

8.10.203 visual-rungame-first.luc

fun main void () {
    int x;
    Entity e;

    x = 2;
e = new Entity(11, 22, "image.png");

    runGame(x > 0; 1){

fun main void () {
    int x;
    Entity e;
    Entity e2;
    x = 100;
    initSDL();
    e = new Entity(200, 50, "gfx/playerBullet.png");
    e2 = new Entity(100, 100, "gfx/bat.jpg");
    runGame(x > 0; 1){
        x = x - 1;
    }
}

8.10.205 visual-rungame-render-origin.luc

fun main void () {
    int x;
    Entity e;
    initSDL();
    x = 100;
    e = new Entity(0, 0, "gfx/bat.jpg");
    runGame(x > 0; 1){
        x = x - 1;
    }
}

8.10.206 visual-rungame-render.luc

fun main void () {
    int x;
    Entity e;
    x = 2;
    e = new Entity(11, 22, "gfx/bat.png");
    initSDL();
    runGame(x > 0; 1){
        x = x - 1;
    }
}
fun main void () {
    initsdl();
}