Ballr: A 2D Game Generator

Players gonna play

Noah Zweben (njz2104)
Jessica Vandebon (jav2162)
Rochelle Jackson (rsj2115)
Frederick Kellison-Linn (fjk2119)

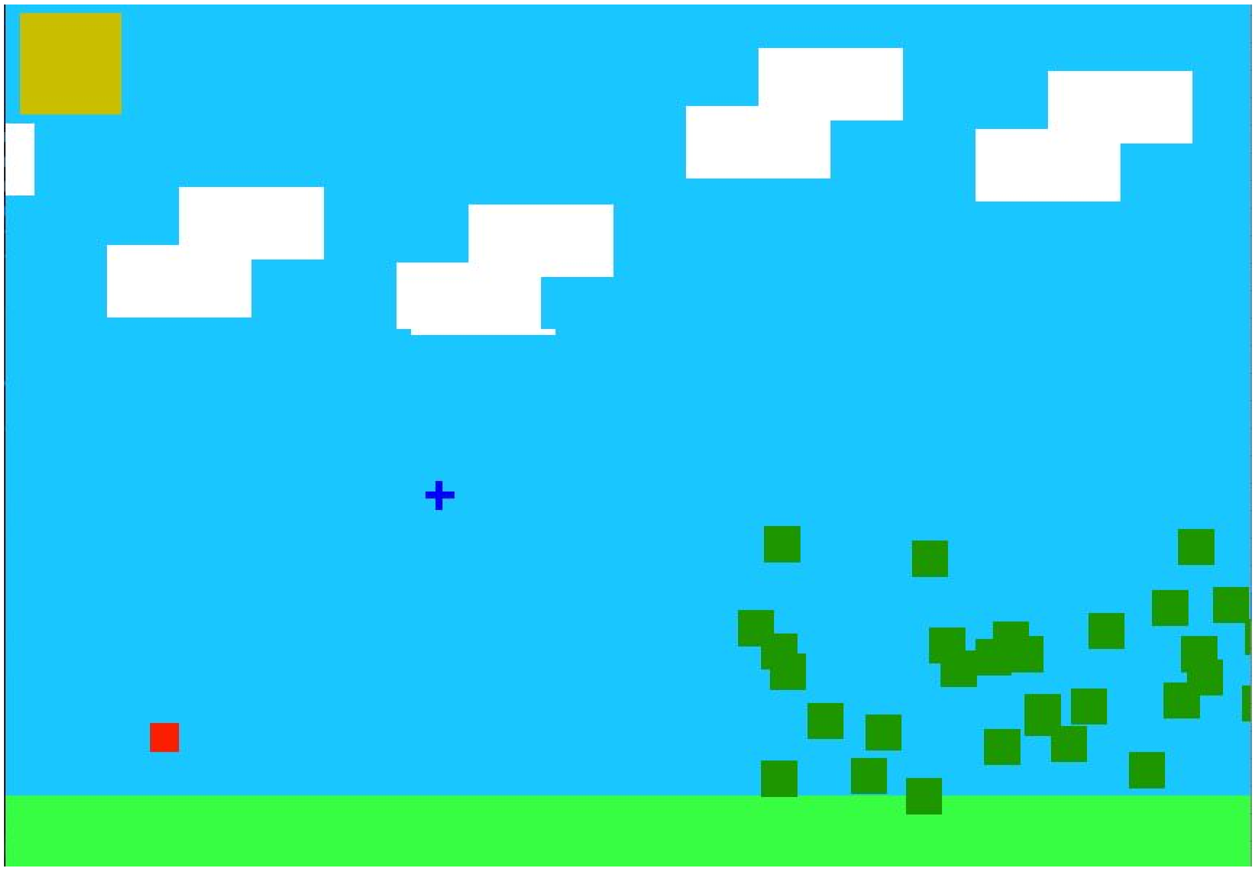


Table of Contents

1 Introduction

Ballr is a programming language that enables users to build simple 2D games with user-defined environment, rules and player control. Users are able to place moving or stationary rectangular "entities" throughout a bounded space that will be displayed in a graphical user interface. They are able to define entity size, color, and behavior. Entities have associated events which define their response to user inputs (keypresses or mouse-clicks), their frame-to-frame changes, and the actions associated with entity collisions. Ballr can also be used for creating simple drawing or animation applications.

Ballr abstracts away the event loop that is necessary when writing a game in, for example, C++. Instead, the programmer is free to focus on events that may occur during the course of the game: collisions, keypresses, and lifetime events being primary examples. This allows the structure of the game to be clear from a simple glance at the code.

2 Tutorial

2.1 Setup

Before beginning, make sure you have a working installation of Ballr. To do this, download all the project files, and run the following commands:

```
$ cd Ballr
$ make
$ cd runtime
$ make
$ sudo make install
```

This will build the compiler, the runtime (see **6.5 Runtime System**), and copy the runtime library to /usr/local/lib so that the compiler can link your games with it. If all of these steps run successfully, then you should have a working Ballr installation. Now let's make some games!

2.2 Hello, Ballr!

The basic structure of a game in Ballr consists of two things: entities, and a gameboard. Entities are the objects in a game that interact on screen, such as bullets, characters, coins, enemies and walls. The gameboard is the space in which all of the entities interact. Let's begin by making a simple game that just brings up a window on screen.

helloworld.blr

```
gameboard helloworld {
    clr = (255, 0, 0);
    size = (300, 300);
    init -> {}
}
```

Save this file, and then at the command line, run

```
$ ./ballr -c helloworld.blr
```

This should produce a binary called 'helloworld' which you can then run:

```
$ ./helloworld
```

You should now see a window that looks like this:

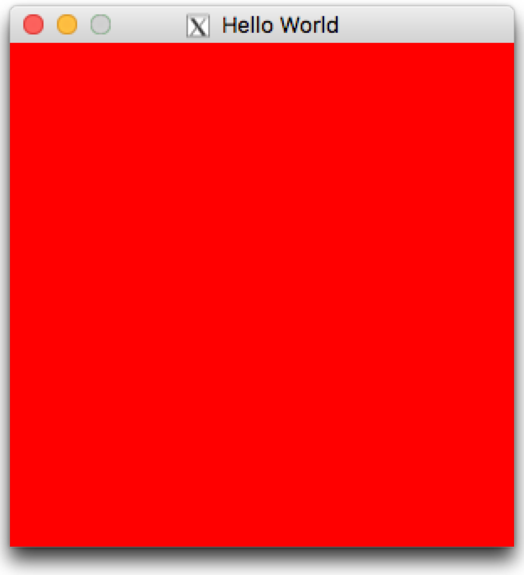


Fig 2.1 What a fun game!

Congratulations, you just wrote your first Ballr program! Let's break it down line by line.

2.3 Gameboard

```
gameboard helloworld {
```

The gameboard declaration in a Ballr game defines the space in which the game will take place. A gameboard declaration begins with the `gameboard` keyword followed by the name of the board; in this case, the name is `helloworld`.

```
    clr = (255, 0, 0);
    size = (300, 300);
```

Every gameboard has three components. The first two are color and size, which must be defined at the beginning of the declaration. Color is defined as a tuple of red, green, and blue (0-255), and represents the background color of the board. Size is a tuple of width and height, and is the on-screen size of the board in pixels. Thus, the `clr` and `size` defined above indicate a game board that has a red background and is 300 by 300 pixels, just as we see on the screen.

```
    init -> {}
```

This is what is known as a behavior block, which we will talk about in more detail when we discuss events. In short, behavior blocks describe the action that should be taken when a specific event occurs in the game. Here, the `init` behavior is executed at the beginning of the game (i.e. on startup, and whenever the `restart()` function is called). Generally, the `init` behavior is used to set up the state of the gameboard as it should be at the start of the game, whether that means adding entities or setting global variables to the appropriate values.

Always make sure to match up your braces!

2.3 Entities

Now, we will see a game that is a little more interesting. The gameboard defines a static space where the game takes place, but entities are what make the game "go". Let's look at the definition of a simple entity:

helloworld.blr

```
entity player {
    clr = (0, 255, 0);
    size = (20, 20);
    key_UP -> {
        self.pos[1] = self.pos[1] - 5;
    }
}
```

A lot of this syntax is reminiscent of that from the gameboard. An entity declaration begins with the keyword `entity`, followed by the name you want to give to the entity. We then have the same two required members for a gameboard, `clr` and `size`, which describe how the entity should be drawn on screen (in this example, a 20 by 20 pixel square of green).

Next, we have a behavior block. This is the first example we have of an event, which will be talked about extensively in the next section. For now, just know that events are things that can happen throughout the course of the game, and that `key_UP` is the event that is triggered when the up arrow key is pressed.

One thing that is different from the gameboard is that this time, we actually have some behavior in the behavior block. To anyone that is already familiar with programming, the line in the braces is probably not too hard to understand, but we will still go over statements in detail in **2.5 Statements**. In addition to the `size` and `clr` members, all entities also have a `pos` member representing the position of their upper left hand corner on the screen. Changing this member will move the entity around the screen. Note that in Ballr, the position (0, 0) is the upper left hand corner of the screen, with y values increasing downward and x values increasing to the right. Thus, this statement says to move up by five pixels.

If you compile and run your game now, you might be surprised to see that nothing shows up. If we just defined an entity, where is it? The answer is that defining an entity simply defines a blueprint for how that type of entity should look and behave—because we can have multiple of one entity on the screen at a time, we also have to declare that we want this entity to appear on our gameboard. To do this, modify the `init` behavior in our gameboard from the last section so that it looks like this:

```
    init -> {
        add(player, (140, 140));
    }
```

The psuedo-function `add()` takes the name of an entity and a position, and creates a new entity on the gameboard at that position. Now, when you compile and run your game, you should see a green square appear in the middle of the screen. Pressing the up key should make it move up.

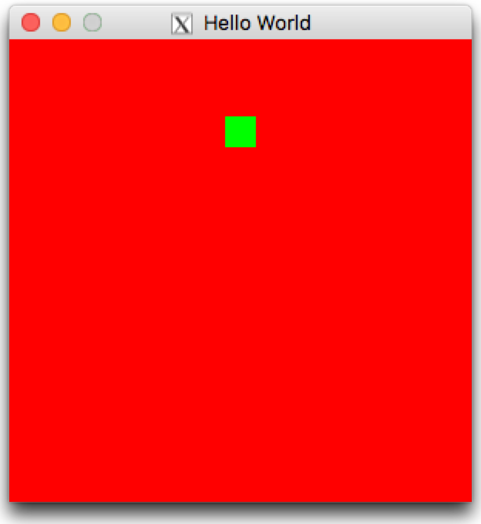


Fig 2.2 Hooray! A game that actually does something!

Entities can also define custom members in addition to `clr`, `size`, and `pos`. The syntax for these declarations is the same as for local variables, but they occur at the top level of the entity rather than in a behavior block. When using these members you must prepend '`self.`' to the name in order to differentiate it from a local variable.

2.4 Events

Events in Ballr encompass pretty much everything that can happen over the course of a game, so we will discuss in detail the different kind of events that can be defined. One thing to note is that in addition to local variables, all events have access to a special variable called `self` which represents the entity that the event was triggered on. This is analogous to `this` in Java methods, and is what allows you to do things like move the player entity when the up arrow key is pressed.

Another important thing to note about events is that the order in which events are called is implementation defined. This means that just because entity A is defined before entity B, or added before entity B, does not mean that the events of A will be executed before those of B. Thus, game code should not rely on the order in which any events are executed.

2.4.1 Keypress

As we have already seen, one of the types of events that can occur during a game is a keypress event. This means that the user is currently pressing a key. In addition to key_UP, there are keypress events for all the arrow keys, the spacebar, and the WASD keys. For arrow keys, the event name is "key_" followed by the direction name in all caps. For the space bar, the event name is key_SPACE. For all letter keys, the event name is "key_" followed by the capital letter. Let's give our player a bit more freedom. In helloworld.blr, add the following events to the player entity:

```
    key_DOWN -> {
        self.pos[1] = self.pos[1] + 5;
    }
    key_LEFT -> {
        self.pos[0] = self.pos[0] - 5;
    }
    key_RIGHT -> {
        self.pos[0] = self.pos[0] + 5;
    }
```

This will let us move our character all over the screen.

2.4.2 Collisions

One of the most important things in a game is defining the interactions between different entities on the screen. In particular, you will frequently want to do something when entities collide with one another, be it a player with a wall, an enemy with a bullet, or any other number of combinations. Before we can define collisions, though, we need another entity.[1] Define the entity:

```
entity wall {
    clr = (0, 0, 0);
    size = (20, 20);
}
```

and add it to the room somewhere in the gameboard's init behavior. Now, in the player entity, let's add a new event:

```
    self >< wall -> {
        restart();
    }
```

[1] Actually, it is possible to define a collision with an entity of the same type, but you will usually only have one 'player' on the screen, so we will use a different entity.

The structure of a collision event is not too difficult to understand. The above will execute whenever a `player` entity collides with a `wall` entity. Specifically, this event will restart the gameboard whenever the player runs into a wall. The `restart()` function, like `add()`, is part of the language.

2.4.3 Click

The next event is `click`, which executes whenever there is a click within the bounds of the gameboard. In addition to the special variable `self`, the `click` event has another special variable `clickpos` that stores the location of the click event that occurred. This variable is of type `vec` and can essentially be thought of as an argument to the event.

2.4.4 Frame

Although the above events encompass many of the possible behaviors that would be desired in a game, it is always possible that there are other things that a game programmer might want to do. In these cases, there is a sort of catchall event called `frame` which is triggered on every frame, where the entity is free to do anything that it wants. This could involve checking for things like whether the entity is out of bounds, updating a timer, etc.

2.5 Functions

Top level functions can be declared as follows:

```
func int timesTwo(int x) {
    return x * 2;
}
```

The func keyword is immediately followed by the return type of the function (see **3.2 Types**), the name, and the arguments, in order. Then comes the behavior block that will execute when the function is run. The one special thing about function behaviors is that they can return values via the `return` statement. This is not supported from event behaviors or the gameboard's `init` behavior. Next, we will discuss what exactly goes on inside a behavior block.

2.6 Behavior Blocks

Here, we will talk in depth about the various language elements supported by behaviors. This is the part of Ballr that will feel most familiar to veteran programmers. It is the actual code that runs while your game is executing—the other constructs basically tell *when* to run each behavior. We will begin by discussing expressions. Let's jump in!

2.6.1 Literals

The literal expressions are the simplest of the expressions. They are constants which can be stored in variables of types `int`, `float`, or `bool`. Some examples of literals are:

```
100  /* int literal */
6.0  /* float literal*/
true /* bool literal */
```

These literals (as expected) evaluate to their literal values.

2.6.2 Identifiers

Identifiers are just the names of variables (local or global) which have been previously declared. They evaluate to their declared type, and to the value which they currently store. See (2.6.x Variable Declarations) for more info.

2.6.3 Unary Operators

Next we have the simplest operators, the unary operators. The two unary operators are the not operator (`!`) and the negative (–) operator. To use these, you simply place them before an expression of the proper type (`bool` for `!` and `int` or `float` for –). The former performs the logical negation of the operand, while the latter performs the numerical negation. Example usage is as follows:

```
!true /* evaluates to false */
-3    /* unary negative and integer literal,
         not literal -3! */
```

The unary operators are both right associative, and have the highest precedence of any operator.

2.6.4 Binary Operators

The binary operators are probably the most familiar, and can be broken down into two types: logical and numerical. The logical operators are ==, `!=`, <, >, <=, >=, `&&`, and `||`, which in English are the equality, non-equality, less than, greater than, less than or equal, greater than or equal, and, and or operators, respectively. These must all appear with two operands on either side, and are less associative. We will briefly discuss each logical operator and its behavior, grouped by precedence level.

The equality operator (==) can be called with expressions of any primitive type (`int`, `float`, or `bool`) as its operands, provided they are both numerical or both boolean expressions. The result of this operation has boolean type, and is `true` if the two operands are equal, and `false` if the operands are not equal. The non-equality operator (`!=`) is identical, except that it evaluates to the logical negation of the equality operator. These operators have higher precedence than the and operator, and lower precedence than the less than, greater than, less than or equal to, and greater than or equal to operators.

The less than operator (<) must have two numerically-typed expressions (`int` or `float`) as its operands, and evaluates to a boolean type value which is `true` if the left hand side is less than the right hand side and `false` otherwise. The greater than operator (>) is identical, but with the left hand side and right hand side swapped. The greater than or equal to operator (>=) is identical to the less than operator, except that the former evaluates to the logical negation of the latter. The less than or equal to operator (<=) is identical to the greater than operator, except that the former evaluates to the logical negation of the latter. These operators have lower precedence than the plus and minus operators which we will discuss shortly.

Lastly we have the logical and (`&&`) and logical or (`||`) operators. These expressions take boolean-typed arguments and evaluate to a boolean-typed value, which evaluate as their names suggest. The and operator has lower precedence than the equality and non-equality operators, and the or operator has lower precedence than the and operator, giving it the lowest precedence of all binary operators.

The numerical operators are `+`, `-`, `*`, `/`, and `%`, called the plus, minus, multiplication, division, and modulus operators. All of these take numerically-typed expressions as their operands, except for modulus which can only take `int`-typed expressions. If the types of the operands are `int` and `float`, these operators evaluate to `float`-typed values, and otherwise evaluate to the type of their operands. All the numerical operators are left associative.

The plus (+) and minus (−) operators compute the sum and difference of their operands, respectively. Their precedence is just above that of all the logical operators.

The multiplication (`*`) and (`/`) division operators compute the product and quotient of the operands. If the result is `float`-typed, then the quotient will be accurate based on IEEE floating point standards. If the operands are `int`-typed, then the quotient will be the result of integer division (i.e. the fractional part will be truncated). These operators have higher precedence than the plus and minus operators.

Finally the modulus operator computes the integer remainder that would result from the division of two `int`s. In other words, the expressions below are equivalent:

```
a % b
a - (a / b) * b
```

The modulus operator has precedence equal to the division and multiplication operators.

Here are some example uses of binary operators:

```
4 + 3 /* evaluates to 7 (with type int) */
4 > 3 /* evaluates to true */
true && false /*evaluates to false */
5 != 10 /*evaluates to true */
```

2.6.5 Function Calls

A function call, as in most languages, executes the specified function, and evaluates to the value that the function returns. For instance, with the `timesTwo` example defined previously, a call to this function would look like:

```
timesTwo(3) /* evaluates to 6 */
```

2.6.6 Color and Vector Expressions

The two non-primitive types, `color` and `vec`, can be created with the following syntax:

```
(<expr>, <expr>, <expr>) /* a color expression */
(<expr>, <expr>) /* a vector expression */
```

With any expressions that are `int`-typed. These can be assigned to variables of type `color` and `vector`, respectively.

2.6.7 Member Accesses

The primary form of member access is the bracketed access, which applies to variables of type color and vec, and is used to access and assign to the individual members. For example, if `a` is a variable with type `color`, then `a[0]` refers to the red component of that color. Between the brackets can be any expression with type `int`. Accessing an index greater than `2` for a color, or `1` for a vec is undefined behavior.

The other form of member access is the dot access, and is only used in one case: `self`. Event behaviors can use `self.memName` to refer to the member of the entity named `memName`. This is treated just like an identifier semantically, except that its value will persist for the lifetime of the entity. This can be combined with bracketed access, so that, for instance, `self.clr[1]` refers to the green component of the entity's color.

2.6.8 Assignment

Assignment is the only right associative binary expression in Ballr, and also the operator with the lowest precedence. Denoted with =, its semantics are relatively simple. The right hand side is evaluated and stored in the location on the left hand side. This means that the left hand side must either be an identifier, a bracketed access, or a dot access expression. Assignment works as follows:

```
a = 30; /* stores 30 in the variable a and evaluates to 30 */
a + 3 /* evaluates to 33 */
```

2.6.9 Statements

Expressions are the building blocks of statements, which are the building blocks of behaviors. There are only a few types of statements.

The expression statement is the simplest: any expression on its own line, followed by a semicolon, is a valid statement. Ballr will evaluate the expression, and then move to the next line.

```
3;
timesTwo(3);
a = 3;
```

The `if` statement is used as follows:

```
if (<bool-expr>) {
    <stmts1>
} else {
    <stmts2>
}
```

This code evaluates the expression `<bool-expr>` (which must be `bool`-typed) and executes `<stmts1>` in the first block if the result is `true`, and `<stmts2>` otherwise. The else clause is optional, so this is valid as well:

```
if (<bool-expr>) {
    <stmts1>
}
```

If `<stmts1>` or `<stmts2>` consist of just a single statement, the brackets are optional.

The `while` statement allows Ballr code to run loops:

```
while (<condition>) {
    <stmts>
}
```

The above code will evaluate `<condition>`, and if it evaluates to true, execute `<stmts>`. Execution then returns to the top of the statement, evaluates `<condition>`, and so on. This is the same as the behavior as in many languages, such as C.

Lastly we have the `return` statement, which is only valid in function context. As discussed previously, the `return` statement is used as:

```
return <expr>;
```

This will evaluate `<expr>`, end execution of the function, and return control to the calling location.

2.7 Go Make Some Games!

You now know everything you need to know to start coding up some Ballr games. If you want a more in depth look at the language, keep reading, or check out **Appendix A.3** to see some examples of the kinds of games you can make with Ballr. Happy balling!

3 Reference Manual

3.1 Lexical Conventions

3.1.1 Identifiers

Identifiers are for naming entities, gameboards, colors, and other data types. They are defined as at least one lowercase letter followed by any combination of letters, numbers, and underscores.

3.1.2 Reserved Keywords & Symbols

The following case sensitive keywords are reserved for specific purposes and cannot be used as identifiers:

`entity`	`func`	`restart`	`gameboard`
`click`	`key_`	`add`	`remove`
`float`	`int`	`bool`	`><`
`clr`	`pos`	`if`	`else`
`while`	`self`	`color`	`vec`
`clickpos`	`init`	`->`	`size`

3.1.3 Literals

1. **Integer literals**
   Sequences of one or more digits (ex. 253)

2. **Float Literals**

Sequences of one or more digits containing a ‘.’ with at least one digit before the ‘.’ and at least one digit after the ‘.’ (ex. 1.23)

3. **Boolean Literals**
   Boolean literals are denoted by ‘true’ or ‘false’

3.1.4 Comments

Only multi-line comments are supported and are opened with **/*** and closed with ***/**

3.2 Types

3.2.1 Primitive Data Types:

Integers: Integers are 32 bit numbers defined as follows:
int x = 42;
Floats: Floats are floating point numbers defined as follows:
float x = 42.42;
Booleans: Boolean values can take either *true* or *false* and are declared as follows:
bool x = true;

3.2.2 Non-Primitive Data Types:

Vector
a `vec` is a built in datatype composed of an x and a y value. The values in the `vec` may be constructed with floats or ints, but everything will be implicitly rounded to int. The values of x and y are enclosed in () and separated by a comma. A vector’s x and y component can be accessed or set using [index] notation. Vectors are used to store the position as (x,y) and size as (width,height);

Color
A `color` is a built in datatype composed of a red (r), green (g), and blue (b) value. The values in the `color` may be constructed with floats or ints, but everything will be implicitly rounded to int. The values of the red, green, and blue components are integers between 0 and 255 and are enclosed in () and separated by commas. Color’s r,g,b components can be accessed or set using [] notation.

3.3 Operators

The following operators are supported by Ballr (in order of increasing precedence)

Operator	Name	Supported Types	Association
=	Assignment	All	Right-to-Left
&&	And	Booleans	Left-to-Right
||	Or	Booleans	Left-to-Right
!=	Not Equal	Ints, Floats, Bool	Left-to-Right
==	Equals	Ints, Floats, Bool	Left-to-Right
>	Greater than	Ints and Floats	Left-to-Right
>=	Greater than or Equal	Ints and Floats	Left-to-Right
<	Less than	Ints and Floats	Left-to-Right
<=	Less than or Equal	Ints and Floats	Left-to-Right
+	Addition	Ints and Floats	Left-to-Right
-	Subtraction	Ints and Floats	Left-to-Right
*	Multiplication	Ints and Floats	Left-to-Right
/	Division	Ints and Floats	Left-to-Right
%	Modulu	Ints	Left-to-Right
!	Unary Logical Negation	Booleans	Right-to-Left
-	Unary Negation	Ints and Floats	Right-to-Left

Precedence (increasing)

```
=
&&, ||
!=, ==
>, >=, <, <=
+, -,
*, /, %
!,- (unary)
```

3.4 Delimiters

1. **Parentheses**
   Parentheses are used to enclose arguments in function calls as well as to force precedence orders in expression evaluations.

2. **Commas**
   Commas are used to separate arguments in function calls and to separate values in color and vector data types.

3. **Semicolons**
   Used to terminate a statement.

4. **Curly braces**
   Used to enclose a series of statements in conditional blocks, loops and entity, gameboard and function definition blocks

3.5 Expressions

The following are valid expressions

Literals: Literals as covered in section 3.1.3 are valid expressions

Binary Operations: The binary operators can be used (as defined in the operator table) to perform arithmetic operations and comparisons.

Unary Operations: ! and - can be used as unary operators to negate booleans, ints, and floats

Assignments: Assignments are right associative and return the value of the assigned object.

Function Calls: Function calls are valid expressions and covered more in depth below.

Variables: Variables are valid expressions and can be variables in the current scope, i.e.

```
int x = 10;
x /* valid expression */
```

or access member variables of entities (more on this later) using the syntax `self.clr`

Color/Vector Access: Accessing (for set & get) the interior values of color and vector types are supported expressions and the notation is color[index] (defined from 0-2 inclusive) and vector[index] (defined from 0-1 inclusive)

```
For example:
      color myColor = (255,0,55);
      myColor[1] = 10;
/* myColor now equals (255,10,55) */
```

3.6 Statements

Single expressions

```
10; x; (255,0,55);
```

Return Statements:

Return a value from a function

```
return 10;
```

Conditional Statements:

```
if (<bool>) {
        <statements>
}  else {
        <statements>
}
```

While loops:

```
while (<bool>) {
        <statements>
}
```

3.7 Scoping

There are three levels of scoping, which is statically defined in Ballr.

Global

Variables and functions can be defined globally and these are placed at the top of the program. These are accessible throughout the program

Entities

Entities have associated variables (built in such as pos) and user-defined. These are accessed through self.<variable>

Behavior Block

Variables can be defined at the top-level of any behavior block. Any Variable definitions **must** be placed at the very top of these blocks. For example at the top of a function, or the top of an event behavior block. If variables local to these blocks have the same name as global variables, they will hide the global value.

3.8 Entities

An `entity` is a rectangular component within a gameboard. An entity is composed of a color and a size. Users may also define additional variables to be associated with the entity. Event driven behavior can be added to entities. Events are described in the following section. In addition, users may define movement for an entity to set up automatic motion.

Entity.size (required)
All entities must possess a size variable of type `vec`. The size encodes the dimensions of the entity as width and height. It can be both read and set using self.

Entity.clr (required)
All entities must posses a clr variable of type color. This variable sets the color of the entity when it appears on the gameboard. It can be both read and set using self.

Entity.pos
Once entities are placed on a gameboard they contain a variable called `pos` of type `vector` encoding the x and y position of the entity center. `self.pos` can be set or read from.

Additional Members
The user may define additional member variables which can be useful for concepts such as lives or score .These are set like normal variable declarations
<type> <name> = <value>
and can be accessed in events using self.<name>

`When defining an entity all members must be defined using integer, boolean, and float literals. Otherwise behaviour is undefined.`

Static Entity

Syntax	Example
`entity <name> {`	`entity player {`
`//required`	`size = (10,10);`
`size = (width,height);`	`clr = (255,255,0);`
`clr = (R,G,B);`	`//adding non-required variables`
`//adding non-required variables`	`int score = 10;`
`<type> <name> = <init. val>;`	`}`

}	

3.9 Events

Events in Ballr encompass pretty much everything that can happen over the course of a game, so we will discuss in detail the different kind of events that can be defined. These events are included within entity definition to create entity-specific behaviors.

One thing to note is that in addition to local variables, all events have access to a special variable called `self` which represents the entity that the event was triggered on. This is analogous to `this` in Java methods, and is what allows you to do things like move the player entity when the up arrow key is pressed.

Another important thing to note about events is that the order in which events are called is not user-defined. This means that just because entity A is defined before entity B, or added before entity B, does not mean that the events of A will be executed before those of B. Thus, game code should not rely on the order in which any events are executed.

The syntax for defining events is as followed:
`event trigger -> { behavior }`

The event trigger is the specific runtime event that occurs during game-time (such as a keypress or collision), and the -> {behavior} denotes the associated behavior to be triggered by the game-time occurrence.

Here are the following event triggers:

3.9.1 Keypress

This event gets triggered when a user presses a key. The following keypress triggers are defined:
key_UP, key_DOWN, key_LEFT, key_RIGHT, key_W, key_A, key_S, key_D, key_SPACE

Example:

```
    key_DOWN -> {
        self.pos[1] = self.pos[1] + 5;
    }
    key_LEFT -> {
        self.pos[0] = self.pos[0] - 5;
    }
    key_RIGHT -> {
        self.pos[0] = self.pos[0] + 5;
    }
```

3.9.2 Collisions

One of the most important things in a game is defining the interactions between different entities on the screen. In particular, you will frequently want to do something when entities collide with one another, be it a player with a wall, an enemy with a bullet, or any other number of combinations.

The syntax for defining collision triggers is

`self >< other_entity -> { behavior }`

now when the entity you have defined the collision event collides with “other_entity” it will trigger the associated behavior.

```
    self >< enemy -> {
        self.damage = self.damage + 10;
    }
```

3.9.3 Click

The event `click` executes whenever there is a click within the bounds of the gameboard. In addition to the special variable `self`, the `click` event has another special variable `clickpos` that stores the location of the click event that occurred. This variable is of type `vec` and can essentially be thought of as an argument to the event. The syntax is

`click -> { behavior }`

```
Example
click -> {self.pos = clickpos;}
```

3.9.4 Frame

Although the above events encompass many of the possible behaviors that would be desired in a game, it is always possible that there are other things that a game programmer might want to do. In these cases, there is a sort of catchall event called `frame` which is triggered on every frame, where the entity is free to do anything that it wants. This could involve checking for things like whether the entity is out of bounds, updating a timer, etc.

Syntax:

`frame -> {behavior}`

```
frame -> {
     self.clr[0] = (self.clr[0]+10)% 255;
}
```

3.10 Gameboard

Gameboard

`gameboard` is the data type that represents the current window that a user is playing in and interacting with. Only one gameboard is allowed per file.

gameboard.size

The size represents the size of the window the user will interact with and is of type `vector`. The vector encodes information about the playable window's width and height.

gameboard.init

The **init** event is a reserved event keyword that is triggered at the very beginning of the game (see section ? for description of events). Behavior needed to set up the game such as adding players is contained in the gameboard's init event.

Syntax	Example

`gameboard <name> {`
`     size = (width,height);`
`     clr = (r,g,b);`
`     init -> { setup code }`
`}`
	`gameboard <name> {`
	`     size = (200,100);`
	`     clr = (255,255,255);`
	`     init -> {add(player,(100,50));}`
	`}`

3.11 Functions

1. **Built-in Functions**
   add(entity, vec) - adds an entity at specified position
   remove() - removes entity from gameboard. Only used within entities, so automatically removes the entity it is in
   restart() - restarts gameboard from init. Global variables are not reset, but all entities are removed from the gameboard and init is rerun

2. **User-Defined Functions**
   Defining a named function:
   ```
   func <return_type> function_name(type formal_name,...)
   ```
   To call a named function:
   ```
   function_name(args)
   ```

3.12 Overall Program Structure

To create a correct .blr program, your file must be organized in the following way precisely following this order:

```
<optional: global variable definitions>
<optional: global function definitions>
<optional: entity definitions>
     <entity structure>
          <entity variable definitions>
          <entity events>

<mandatory: gameboard definition>
     <gameboard size and color definitions>
     <gameboard init event block>
```

3.13 Context Free Grammar

```
program → var_decl_list func_decl_list ent_decl_list gboard EOF
prim_type →  INT | FLOAT  | BOOL | COLOR | VECTOR
var_decl_list → ε   | var_decl_list var_decl
var_decl → prim_type ID ASSIGN expr SEMI
func_decl_list → ε   | func_decl_list func_decl
func_decl → FUNC prim_type ID LPAREN formal_list_opt RPAREN
     LBRACE var_decl_list stmt_list RBRACE
ent_decl_list → ε   | ent_decl_list ent_decl
ent_decl → ENT ID LBRACE property_list event_list RBRACE
property_list → ε  | property_list property
property → var_decl | CLR ASSIGN expr SEMI | SIZE ASSIGN expr
     SEMI
event_list → ε | event_list event
event → eventCheck DO LBRACE var_decl_list stmt_list RBRACE
eventCheck → KEYPRESS | CLICK  | ID COLLIDE ID | FRAME

gboard → GBOARD ID LBRACE property_list INIT DO LBRACE
     var_decl_list stmt_list RBRACE RBRACE
```

```
stmt_list → ε   | stmt_list stmt
stmt → expr SEMI | RETURN expr SEMI | LBRACE stmt_list RBRACE |
     IF LPAREN expr RPAREN stmt  %prec NOELSE  | IF LPAREN expr
     RPAREN stmt ELSE stmt | WHILE LPAREN expr RPAREN stmt
expr → INT_LITERAL | FLOAT_LITERAL | TRUE | FALSE | member |
     tmember | expr PLUS expr   | expr MINUS expr  | expr TIMES
     expr  | expr DIVIDE expr  | expr ISEQ 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
     | NOT expr | MINUS expr %prec NEG  | LPAREN expr RPAREN  |
     ID LPAREN actuals_opt RPAREN  | member ASSIGN expr  |
     tmember ASSIGN expr    | LPAREN expr COMMA expr RPAREN   |
     LPAREN expr COMMA expr COMMA expr RPAREN
member → ID  | CLR | SIZE | ID PERIOD ID | ID PERIOD CLR  | ID
     PERIOD SIZE
tmember →  member LSQUARE expr RSQUARE
actuals_opt → ε | actuals_list
actuals_list → expr | actuals_list COMMA expr
formal_list_opt → ε | formal_list
formal_list →  prim_type ID  | formal_list COMMA prim_type ID
```

4 Project Plan

4.1 Team Member Roles and Responsibilities

At the start of the project, team roles were assigned as follows:

Role	**Team Member**
Manager	Rochelle
Language Guru	Noah
System Architect	Freddy

Tester	Jessica

Although each team member has maintained responsibility for their originally assigned role, these roles have been fluid at best. Each team member has contributed to the development and testing of multiple aspects of the overarching compiler architecture as well as decisions regarding the language and system design. Rather than dividing implementation work by role, it was generally split up by feature.

4.2 Planning Process

Weekly planning and development meetings were held throughout the course of the project. After devising a language idea and putting together the initial Ballr reference manual, the first few meetings were development-heavy sessions as we built the initial stages of the compiler architecture. As the project progressed, the meetings were used more for planning and dividing tasks rather than development. A general projected timeline was devised after the HelloWorld milestone with weekly goals, and this was revisited and adapted each week (see the Project Timeline section below). Fortunately, there were very few unforeseen pitfalls and we were mainly able to keep up with the original projected schedule, allowing for time during the last weeks to develop interesting programs with an almost-entirely-done compiler system.

4.3 Development Process

In the early stages of the project, until we passed the HelloWorld milestone, the bulk of development was done collaboratively at weekly project meetings. After we had a somewhat-functional system with a working Hello World program, we decided to take a bottom up approach to the development of Ballr. We finalized language details early on so that we could build the entire front-end of the compiler (scanner, parser and AST) and focus only on semantic checking, code generation and our runtime library to add features. The scanner, parser and AST rarely had to be revised as we continued development. Features were assigned to team members each week, and progress was tracked using git commits. We made an effort to be constantly in communication in order to keep all team members up to date with any new developments to the project.

4.4 Testing Process

Due to the event driven and interactive behavior or Ballr programs, much of the language's functionality is hard to test with an automated regression test suite (see the Testing Plan section). Because of this, we agreed early on that some feature testing would have to be done manually - through visual GUI verification and checking user input behavior. Each team member took responsibility for ensuring that their assigned features were thoroughly tested and refined before they were pushed to the shared master repository. After features were pushed, all bugs found (if any) were quickly reported and the team member in charge of the feature would fix them.

All language features that could be tested with the automated regression test suite (mathematical operations, syntax, required definitions, semantical correctness etc.) had unit tests written and their functionality verified using the test script (see Appendix A.4 for a full test listing).

4.5 Project Timeline

Week	**Goals**	**Milestones**
Feb 6 - Feb 12	- Develop language idea: Ballr	**Proposal (Feb 8th)**
Feb 13 - Feb 19	- Brainstorm Ballr features	
Feb 20 - Feb 26	- Decide on Ballr syntax and semantics	**LRM (Feb 22nd)**
Feb 27 - Mar 5	- Set up development environments locally - Look into how SDL is used with C	
Mar 6 - Mar 12	- Brainstorm design for system architecture and runtime library	
Mar 13 - Mar 19	< SPRING BREAK >	
Mar 20 - Mar 26	- Implement barebones scanner - Implement barebones parser - Implement barebones AST - Implement codegen for HelloWorld	

	- Implement runtime for HelloWorld	
Mar 27 - Apr 2	- Finalize language features	**Hello World (Mar 27)**
Apr 3 - Apr 9	- Build barebones test suite - Implement barebones semantic checking - Implement basic entities and add()	
Apr 10 - Apr 16	- Implement all statements and expressions - Implement functions - Implement entity members - Implement global variables - Implement frame event for entities	
Apr 17 - Apr 23	- Implement keypress events - Implement remove(), restart()	
Apr 24 - Apr 30	- Implement click events and clickpos - Handle automatic float rounding for vectors	
May 1 - May 7	- Extensive testing - Build cool games	
May 8 - May 14	- Finalize compiler and put together report	**Presentation (May 10)**

5 Language Evolution

The original concept of Ballr was to have it closely model instructions for a game. To achieve this, we had the same concepts of entities and gameboards as above, but instead of including event driven behavior within entities, we instead organized them into a separate instructions block (to closely follow how a game's instructions would be written). For example:

```
entity e1{
     size = (200,200);
     clr = (0,0,0);
}

entity e2{
     size = (100,100);
     clr = (255,0,0);

}
```

```
gameboard g1{
     clr = (255,255,255);
     size = (1000,1000);
     init -> {add(e1,(0,0)); add(e2,(500,500));}
}

instructions {
     e1 >< e2 -> {remove(e2) /* for example */

}
```

This paradigm quickly brought up several conceptual difficulties. First and foremost was how to differentiate between all the instances of a given entity and the entity itself. For example, if were were to have the event

```
 e1 >< e2 -> {e2.pos[0] = e2.pos[0]+10;}
```

It wasn't immediately conceptually clear whether or not we were speaking about all entities or just the e2 that triggered the collision. And especially for frame by frame events, we found it significantly reduced clarity:

```
frame -> {
     remove(e2);
     if (e1.pos[0] > 10) remove(e1);
}
```

In this example, it is unclear exactly what e2 is referring to -- a specific instance of e2 or all e2's. Likewise, e1.pos is not clear as e1 is the name for an entity, not a specific instance. What does it mean to access the pos of the entity e1, does this automatically get turned into a structure which loops through every instance of e1 and then remove it if that specific e1 instance is too far to the right? To avoid some of the ambiguity associated with separating entities from their events, we restructured our language to be more object oriented. Now, events are contained *within* entity blocks and have local access to their specific entity. Using the concept of object-oriented coding and *self*, every single **instance** of an entity has its own events, making it very clear what is happening when you access position or call remove from within an event.

6 Translator Architecture

The basic translator architecture is very simple, and will be quite familiar to anyone already familiar with the basics of compiler design. In **Fig 6.1**, the high level design of the system can be seen. The original Ballr code is passed, in sequence, to the lexer, the parser, the semantic analyzer, and the code generator. These are all the components of the Ballr compiler, which produces LLVM IR as its target. Then, the compiler takes this LLVM IR and passes it to the Clang compiler. This final step also links in the runtime library LibBlr,

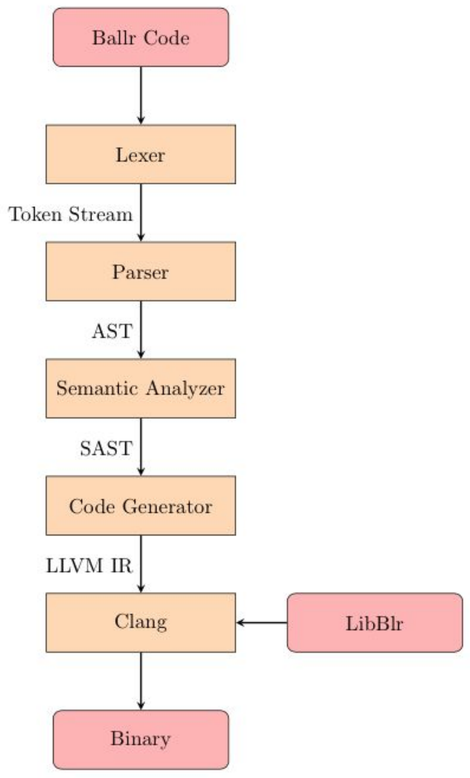


Fig 6.1 A high-level view of the compilation system.

We will discuss each of these steps briefly, spending the most time on the code generation and runtime sections. A full listing of the code can be found in **Appendix A.1**.

6.1 Lexer

The lexer (also known as the scanner) constitutes the first step in the compiler, so it sees the raw Ballr code as typed in the '.blr' file provided as input. The job of the lexer is to take the character stream that makes up the file, and translate it into a token stream that is ready for parsing. For example, a section of the file that the lexer sees might looks something like:

example.blr
```
entity e1 {
    size = (20, 20);
    clr = (255, 0, 0);
}
```

The lexer turns this into:

```
ENT ID(e1) LBRACE ID(size) ASSIGN LPAREN INT_LITERAL(20) COMMA INT_LITERAL(20)
RPAREN SEMI ID(clr) ASSIGN LPAREN INT_LITERAL(255) COMMA INT_LITERAL(0) COMMA
INT_LITERAL(0) RPAREN SEMI RBRACE
```

While this may appear harder to read to a human, breaking down the program into the language units makes it much easier for the compiler to parse our language. See 'scanner.mll' for a full definition of the lexer. This file is written to be compiled by a program called ocamllex, which will generate the actual OCaml code. The file as written simply defines the structure of the tokens as seen in **3.1 Lexical Conventions**.

6.2 Parser

Once the lexer has given us a stream of tokens, we want to figure out what the program is actually supposed to be *doing*. This is the job of the parser. It analyzes the token stream and constructs an abstract syntax tree (AST) which describes the components of the program in a hierarchical structure rather than a linear one. For a Ballr program such as the one above, the AST might look something like **Fig 6.2**. The definition of the parser can be seen in the file 'parser.mly', which, like the lexer, is written for a different tool to generate OCaml code. In this case, ocamlyacc will take the grammar and generate the code that produces an AST as defined in 'ast.ml'.

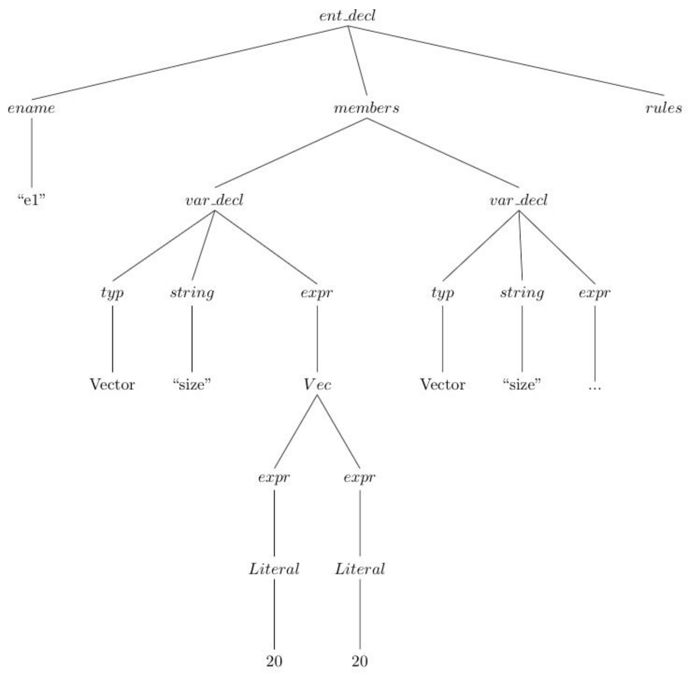


Fig 6.2 Part of the AST of a possible Ballr program.

6.3 Semantic Analyzer

The role of the semantic analyzer is to make sure that the input program is actually meaningful—i.e. that it defines a correct program. Examples of incorrect constructs that cannot be discarded syntactically are programs which use identifiers before they are defined, or, for a Ballr-specific example, entities which attempt to collide with undefined entities. See 'semant.ml' for a full definition of the semantic analyzer.

6.4 Code Generator

Once the semantic analyzer has verified that the program is correct, we can actually generate code for it, without having to worry if we are generating nonsense. In Ballr, the code generator does a lot of work, because the program as typed in the .blr file doesn't have a clear linear structure as many programming languages do. Instead, the code to execute is spread out over the various behavior blocks—when each of these gets called is dependent on the runtime. The code generator produces LLVM IR, the intermediate language for the LLVM compiler project. A full definition of this can be found at llvm.org.

For each entity, there are two pieces of code that get generated: the create function and the frame function. The create function is called whenever a new entity is added to the gameboard. This function allocates space for the entity and ensures that all the members are initialized to the proper values. This function is called only once in the lifetime of an entity. On the other hand, the frame function is called for each entity during each frame of the game. In this function, the code for each of the event behaviors is compiled along with the proper checks to make sure that the code only executes at the correct time. To do this, the frame function makes heavy use of the runtime system, which we will discuss in the next section.

For gameboards, the situation is similar: there is a create function which allocates and initializes the gameboard, and an init function which the runtime calls at the appropriate times. Instead of every frame, the init function is called once at the beginning of the game, and then once for every call to `restart()`.

Finally, free standing functions are compiled just as global functions in LLVM.

6.5 Runtime System

The functionality of Ballr is supported by an extensive runtime library. This system, LibBlr, is responsible for handling all of the low level aspects of running the game. The runtime is based heavily on SDL, which is used for drawing to the screen and detecting events like keypresses and mouse-clicks. Operations that are primitives in Ballr are implemented by LibBlr. This includes the functions add, remove, and restart, as well as all the code for ensuring that events are executed at the proper time. The following is a brief overview of the main components of the runtime system.[2]

In the runtime, all entities are represented by the `entity_t` struct, which provides a common interface to all Ballr entities regardless of type. This struct stores the name of the Ballr entity, as well as the three required members for every entity (`size`, `pos`, and `color`). An entity's `frame_fn` member is a pointer to a function that will be called on each frame of the game so that the entity can perform its event checks appropriately. Each Ballr entity declaration results in exactly one frame function being generated. Thus, runtime entities which have the same `name` member will have the same `frame_fn` parameter. Because Ballr entities can also have their own custom members, it is not possible for the runtime's `entity_t` to fully describe every possible Ballr entity. Thus, an additional opaque pointer is needed which points to a struct containing the custom members of the specific Ballr entity. This struct is created and assigned when the Ballr entity's code is generated (see **6.x Entities**). Lastly, the `next` and `hh` members are present

[2] It will sometimes be necessary to make a distinction between concepts as they are used within Ballr code, and as they are implemented in the runtime. To do this, we will refer to 'Ballr ___' and 'runtime ___' in order to distinguish.

for the uthash library. As we will see later, runtime entities are managed by being stored in linked lists and hash tables, and these members allow the library functions to work with the `entity_t` struct.

For runtime gameboards the `name`, `size` and `color` members are analagous to those in the runtime entity struct, corresponding to the declared name and the `size` and `clr` members of the Ballr gameboard, respectively. Strangely, even though the `ents` member is of type `entity_t *`, it is actually a hash table of the entities currently present in the room, keyed by `name`. The details of how this works is outside the scope of this document, and can be found in the documentation of the uthash library. As you can probably guess, `init_fn` is a pointer to the function that should be called when the board is initialized, either when the game first begins or when `restart()` is called. Finally, for a better understanding of the `hh` member, see the documentation of the uthash library.

6.5.1 The Run Loop

The runtime exposes a function called `run_loop()`, which handles all the heavy lifting of actually running the game. Every Ballr program, when compiled, first does some minor setup work, and then calls this routine. There is a lot of SDL boilerplate that we will mostly omit in this discussion in favor of focusing on the Ballr specific parts of the code.

runtime/window.c

```
blr_color_t c = current_board->color;
Uint32 bg_color_sdl = SDL_MapRGB(screenSurface->format, c.r, c.g, c.b);
SDL_FillRect(screenSurface, NULL, bg_color_sdl);
```

Each frame, the runtime begins by filling in the window with the color of `current_board`. This is a global variable which always points to the gameboard currently being run. Doing this first ensures that this color will be in the background and not overlap any of the entities.

runtime/window.c

```
kb_state = SDL_GetKeyboardState(NULL);
```

This line updates the global `kb_state` variable which stores an array of integers indicating whether each key of the keyboard is currently pressed or not. This is used, when checking for keypress events.

runtime/window.c

```
entity_t *elist;
for (elist = current_board->ents; elist != NULL; elist = elist->hh.next) {
    entity_t *tmp, *elt;
    LL_FOREACH_SAFE(elist->next, elt, tmp) {
        if (elt->frame_fn != NULL) {
            elt->frame_fn(elt);
        }
        draw_entity(elt);
    }
}
```

This is the 'meat' of the run loop. Here, we loop through the hash table (whose entries are lists of entities) and then loop through each list. Then, we ensure that the frame function is non-null, and if so, call it. Finally, we draw the entity, which is done in a similar manner to the drawing of the background seen previously, and continue on to the next iteration. The simplicity of this function is made possible by abstracting away the event checking code for each type of event into separate functions.

runtime/window.c

```
if (should_restart) {
     reset_board();
     current_board->init_fn(current_board);
     should_restart = 0;
}
```

Major game events can only happen at frame boundaries, so (for example) a call to `restart()` by an entity simply sets the `should_restart` flag and moves on. Here, once we have finished iterating through all the entities

runtime/window.c

```
entity_t *tmp, *elt;
LL_FOREACH_SAFE(to_delete, elt, tmp) {
    LL_DELETE(to_delete, elt);
    free(elt->members);
    free(elt);
}
```

Finally, the run loop finishes by removing and freeing all the elements slated for removal by calls to `remove()`.

7 Testing

7.1 Manual Testing

Ballr programs generate games displayed in GUIs with event-based behavior mostly triggered by user input (keypresses, clicks) - for these features, there is no trivial way to generate output that can be automatically verified. Because of this, we decided to do testing for the high-level Ballr features manually. Each team member took responsibility for thoroughly testing the features they implemented. All appropriate tests were done and features verified before any code was pushed to the shared repository. This included visually verifying anything displayed in the GUI (ensuring entities showed up where they were expected and looked as they were expected) and checking user input behavior (correct changes due to keypresses or mouse clicks).

Most features built on pre-existing features, and thus as development progressed, all team members made an effort to find bugs in these existing foundational features. Any bugs found were quickly reported and fixed. Because of this development process, bugs emerged naturally as new features were added - they were easy to spot and fix (and fortunately for us, quite rare).

7.2 Automated Test Suite

Although some of the high level functionality of Ballr could not be automatically tested, all basic language features such as arithmetic operations, boolean expressions and general semantic correctness could be tested using unit tests that produce some expected output. Each of these features had at least one fail test as well as an expected-to-work test written to verify their functionality. Our test script is based on MicroC's testall.sh, modified to work with .blr input files. A `print()` function identical to that in MicroC was implemented in Ballr for the purpose of being able to produce output for unit test verification.

A full listing of test files is included in Appendix A.4.

8 Conclusions & Lessons Learned

8.1 Noah

I think making a language that was exciting and fun to use was a huge advantage in staying motivated throughout the course. I think one of my biggest takeaways is to start testing really early, because I think there were some bugs that came up later on in the process that we fixed with "band-aid" code, which does work, but we could have come up with far more elegant solutions had we started developing an understanding of problem areas sooner. Having a schedule and starting early was incredibly helpful for us. In addition, clearly defining your language at the very beginning allows you to finish your AST, scanner, and parser within the first few weeks -- then all you have to do is make minor changes, and can spend the majority of the time semantically checking and code generating which are more difficult!

8.2 Jessie

Make a structured plan from the beginning and set goals with deadlines - even if this has to be adjusted as development progresses, it's super helpful to know that you're on track to finish on time (or not) as the semester progresses, and to have an idea of what being on track should look like. And definitely try to design a language you would want to use yourself—after getting over the hump of learning OCaml, Ballr was genuinely so much fun to implement. This made the entire process painless and enjoyable for our whole team.

8.3 Freddy

Designing a language while implementing it is a difficult task. Furthermore, designing a language in a few short weeks is a bad idea. I felt that we had to compromise on some decisions that made our language aesthetically less pleasing in favor of being able to implement it in the allotted time. Part of this was due to familiarity with the tools we were using: if I were going to design and implement a language professionally, I would want to have a lot more experience with OCaml, or work in a language that I am already comfortable in. Also, frontloading the design work is a really good idea. If you build in a bad design early on just to get something working, it will become harder and harder to go back and change as you build features on top of it.

My favorite part of this project as System Architect was designing the runtime system. A lot of the code generation was relatively formulaic and could be copied from MicroC, but the interactions between entities and gameboards and deciding how to represent them in memory was a really interesting design problem.

I also found working in OCaml to be an interesting experience. Having not had any experience building a large project in a functional language before, I definitely learned a lot both about OCaml specifically and functional programming in general. However, at other times I felt as though I was hacking something together in a way that felt I was wrestling the compiler to my will rather than writing 'idiomatic' code. This is something that would only be resolved with more experience with OCaml, which I definitely will continue to use.

8.4 Rochelle

I learned that before trying to play catch up, make sure you understand code written by others, it will save you a lot of time. Also, make sure you have a solid idea of what you want your compiler to do, it will make figuring out your grammar and semantics much easier. Moreover, learning OCaml, parser and AST might seem intimidating at first but when you look back you will realize that various parts your work link to classes before. Try to fix broken tools need for coding as quickly as possible and take this as a great learning experience, especially if you team has great ideas and motivating people.

Appendix A: Code Listing

A.1 Compiler

scanner.mll

```
{
open Parser
open String
}

rule token = parse
       [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
       | "/*"     { comment lexbuf }
       | '{'   { LBRACE }
       | '}'   { RBRACE }
       | '['   { LSQUARE }
       | ']'   { RSQUARE }
       | ','   { COMMA }
       | '('   { LPAREN }
       | ')'   { RPAREN }
       | ';'   { SEMI }
       | '+'   { PLUS }
       | '-'   { MINUS }
       | '*'   { TIMES }
       | '/'   { DIVIDE }
```

```
        | '%'  { MOD }
        | '='  { ASSIGN }
        | '>'  { GT }
        | '<'  { LT }
        | "==" { ISEQ }
        | "!=" { NEQ }
        | "<=" { LEQ }
        | ">=" { GEQ }
        | "&&" { AND }
        | "||" { OR }
        | "!"  { NOT }
        | "."  { PERIOD }
        | "true" { TRUE }
        | "false" { FALSE }
        | "if"   { IF }
        | "else" { ELSE }
        | "while" { WHILE }
        | "int"   { INT }
        | "bool"  { BOOL }
        | "float" { FLOAT }
        | "color"  { COLOR }
        | "vec"       { VECTOR}
        | "gameboard" { GBOARD }
        | "entity"      { ENT }
        | "func"  { FUNC }
        | "return" { RETURN }
        | "init"      { INIT }
        | "clr"       { CLR }
        | "size"      { SIZE }
        | "->"    { DO }
        | "><"    { COLLIDE }
        | "func"  { FUNC }
        | "frame" { FRAME }
        | "click" { CLICK }
        | "key_"(['a'-'z' 'A'-'Z' '0'-'9'] | "UP" | "DOWN" | "LEFT" | "RIGHT" | "SPACE") as lxm
        {KEYPRESS(lxm) }
        | ['0'-'9']*'.'['0'-'9']+ | ['0'-'9']+'.'['0'-'9']* as lxm {
FLOAT_LITERAL(float_of_string lxm)}
        | ['0'-'9']+ as lxm { INT_LITERAL(int_of_string lxm) }
        | ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
        | eof  { EOF }
        | _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }


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

parser.mly

```
%{ open Ast %}

%token LBRACE RBRACE COMMA LPAREN RPAREN PLUS MINUS TIMES DIVIDE MOD
%token ASSIGN GT LT ISEQ NEQ LEQ GEQ AND OR NOT PERIOD TRUE FALSE IF ELSE
%token WHILE INT BOOL FLOAT COLOR VECTOR GBOARD ENT RULES FUNCTION RETURN
%token DO COLLIDE EOF SEMI CLR MOV SIZE INIT FUNC LSQUARE RSQUARE
%token FRAME CLICK SELF

%token <float> FLOAT_LITERAL
%token <int> INT_LITERAL
%token <string> ID
%token <string> KEYPRESS
```

```
%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left ISEQ NEQ
%left LT GT LEQ GEQ
%left COLLIDE
%left PLUS MINUS
%left TIMES DIVIDE MOD
%right NOT NEG
%left ACCESS

%start program
%type < Ast.program> program

%%

program:
              var_decl_list func_decl_list ent_decl_list gboard EOF
        { (List.rev $1, List.rev $2, List.rev $3, $4) }

prim_type:
        INT           { Int }
        | FLOAT       { Float }
        | BOOL               { Bool }
        | COLOR              { Color}
        | VECTOR      { Vector }


var_decl_list:        /* nothing */                { [] }
                      | var_decl_list var_decl            { $2 :: $1 }

var_decl: prim_type ID ASSIGN expr SEMI            { VarInit($1, $2, $4) }

func_decl_list:
        /* nothing */                              { [] }
        | func_decl_list func_decl                        { $2 :: $1 }

func_decl:
        FUNC prim_type ID LPAREN formal_list_opt RPAREN LBRACE var_decl_list stmt_list RBRACE
        { {
                typ = $2;
                             fname = $3;
                             formals = $5;
                             locals = List.rev $8;
                             body = List.rev $9;
        } }
ent_decl_list:
              /* nothing */                               { [] }
        | ent_decl_list ent_decl                          { $2 :: $1 }

ent_decl:
              ENT ID LBRACE property_list event_list RBRACE
              { {
                     ename = $2;
```

```
                      members = List.rev $4;
                      rules = List.rev $5;
               } }

property_list:
           /* nothing */                           { [] }
        | property_list property                          { $2 :: $1 }

property:
              var_decl                                                    { $1 }
              | CLR ASSIGN expr SEMI                              { VarInit(Color,
"clr", $3) } /*****************/
              | SIZE ASSIGN expr SEMI                             { VarInit(Vector,
"size", $3) } /*****************/

event_list:
           /* nothing */                           { [] }
        | event_list event                                        { $2 :: $1 }

event:
              eventCheck DO LBRACE var_decl_list stmt_list RBRACE              { Event($1,
List.rev $4, List.rev $5) }

eventCheck:
              KEYPRESS                                                    {
KeyPress($1)}
              | CLICK                                                            { Click
}
              | ID COLLIDE ID                                     { Collision($1,$3) }
              | FRAME                                                     { Frame }

gboard:
              GBOARD ID LBRACE property_list INIT DO LBRACE var_decl_list stmt_list RBRACE
RBRACE
              {{
                      gname = $2;
                      gmembers = List.rev $4;
                      init_mem = List.rev $8;
                      init_body = List.rev $9;
              }}


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

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


expr:
       INT_LITERAL                                 { Literal($1) }
```

```
        | FLOAT_LITERAL                             { FLiteral($1) }
        | TRUE                                             { BoolLit(true) }
        | FALSE                                            { BoolLit(false) }
        | member                                           { $1 }
        | tmember                                          { $1 }
        | expr PLUS expr                { Binop($1, Add, $3) }
    | expr MINUS expr               { Binop($1, Sub, $3) }
    | expr TIMES expr                          { Binop($1, Mult, $3) }
    | expr DIVIDE expr                         { Binop($1, Div, $3) }
    | expr ISEQ expr                           { Binop($1, Equal, $3) }
    | expr NEQ expr                    { Binop($1, Neq, $3) }
    | expr LT expr                     { Binop($1, Less, $3) }
    | expr LEQ expr                    { Binop($1, Leq, $3) }
    | expr GT expr                             { Binop($1, Greater, $3) }
    | expr GEQ expr                    { Binop($1, Geq, $3) }
    | expr AND expr                    { Binop($1, And, $3) }
    | expr OR expr                             { Binop($1, Or, $3) }
    | expr MOD expr                    { Binop($1, Mod, $3) }
    | NOT expr                                 { Unop(Not, $2) }
    | MINUS expr %prec NEG             { Unop(Neg, $2) }
    | LPAREN expr RPAREN               { $2 }
    | ID LPAREN actuals_opt RPAREN   { Call($1, $3) }
    | member ASSIGN expr                       { Assign($1, $3)}
    | tmember ASSIGN expr              { Assign($1,$3)}
    | LPAREN expr COMMA expr RPAREN            { Vec($2, $4) }
    | LPAREN expr COMMA expr COMMA expr RPAREN  { Clr($2, $4, $6) }


member:
       ID                           { Id($1) }
       | CLR                 { Id("clr") }
       | SIZE           { Id("size") }
       | ID PERIOD ID  { Access($1, $3) }
       | ID PERIOD CLR  { Access($1,"clr") }
       | ID PERIOD SIZE { Access($1, "size") }

tmember:
       | member LSQUARE expr RSQUARE       {ArrayAccess($1,$3)}



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

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

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

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

semant.ml

```
open Ast

module StringMap = Map.Make(String)

let check (vardecls, funcdecls, entdecls, gboard) =

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

  (* check if given type is an int or float *)
  let isNumType t = if (t = Int || t = Float) then true else false in

  (* extract variable name from varinit *)
  let varDeclName = function VarInit(_, n, _) -> n in

  (*** CHECK VAR DECLS & BUILD GLOBALS MAP ***)

  (* only allow constants or negative unop for global declarations *)
  let rec globalExpr = function
      Literal _ -> Int
      | FLiteral _ -> Float
      | BoolLit _ -> Bool
      | Unop(op, e) as ex -> let t = globalExpr e in
      (match op with
        Neg when t = Int -> Int
      | Neg when t = Float -> Float
      | _ -> raise (Failure ("Illegal global declaration: " ^ string_of_expr ex))
      )

      | Clr(r,g,b)  -> let t1 = globalExpr r and t2 = globalExpr g and t3 = globalExpr b in
        if (isNumType(t1) && isNumType(t2) && isNumType(t3)) then Color
        else raise (Failure ("Expected numeric input for type color"))

      | Vec(x,y)  -> let t1 = globalExpr x and t2 = globalExpr y in
        if (isNumType(t1) && isNumType(t2)) then Vector
        else raise (Failure ("Expected numeric input for type vector"))

      | _ -> raise (Failure ("Illegal global declaration"))
  in

  let checkGlobalVarInit m = function
    VarInit(t,n,e) -> let e_typ = globalExpr e in
      if t != e_typ
        then raise (Failure ("Inconsistent types in global declaration " ^ n ^ " " ^
string_of_expr e))
      else ()
  in

  (* build map of global names & types *)
  let globals =
    let global_var m (VarInit(t, n, e)) = StringMap.add n t m
    in List.fold_left global_var StringMap.empty vardecls
  in

  let globals =
    StringMap.add "clickpos" Vector globals
  in

  (* check assignment types *)
  List.iter (checkGlobalVarInit globals) vardecls;

  (* check for duplicates *)
  reportDuplicate (fun n -> "Duplicate global variable " ^ n)
      (List.map varDeclName vardecls);

  (*** DONE CHECKING VARDECLS ***)

  let builtInDecls =  StringMap.add "print" (*** ADD BUILT IN FUNCTIONS ***)
     { typ = Int; fname = "print"; formals = [(Int, "x")];
       locals = []; body = [] }
       (StringMap.singleton "add"
     { typ = Int; fname = "add"; formals = [(Bool, "e") ; (Vector, "pos")]; (* NEED AN ENTITY
TYPE ? *)
       locals = []; body = [] })
  in

  let builtInDecls = StringMap.add "remove"  { typ = Int; fname = "remove"; formals = []; (*
NEED AN ENTITY TYPE ? *)
       locals = []; body = [] } builtInDecls
   in

  let builtInDecls = StringMap.add "restart"  { typ = Int; fname = "remove"; formals = []; (*
NEED AN ENTITY TYPE ? *)
       locals = []; body = [] } builtInDecls
   in

  (* map of all callable functions, don't allow duplicates *)
  let functionDecls = List.fold_left
    (fun m fd ->

      try
        StringMap.find fd.fname m; raise (Failure ("Duplicate function " ^ fd.fname));
      with
        Not_found -> StringMap.add fd.fname fd m
    )
    builtInDecls funcdecls
  in

  (* find a function given its name *)
  let functionDecl s = try StringMap.find s functionDecls
       with Not_found -> raise (Failure ("Unrecognized function " ^ s))
  in

  (* return the type of an ID (check given symbols map and globals) *)
  let type_of_identifier m s =
      try StringMap.find s m
      with Not_found ->
        try StringMap.find s globals
        with Not_found -> raise (Failure ("Undeclared identifier " ^ s))
    in

(* builds a map of each entity and its corresponding available members *)

(* build a map given a list of members *)
let memTypes memz =
  let map = List.fold_left (fun m (VarInit(t, n, e)) -> StringMap.add n t m) StringMap.empty
memz in
  StringMap.add "pos" Vector map
in

let allEntMembers =
    let entMems m ent = StringMap.add ent.ename (memTypes ent.members) m in
    List.fold_left entMems StringMap.empty entdecls
in

  (* check expressions *)
  let rec expr m ent = function
              Literal _ -> Int
          | FLiteral _ -> Float
     | BoolLit _ -> Bool
     | Id s -> type_of_identifier m s
     | Noexpr -> Bool (* THIS *)

     | Binop(e1, op, e2) as e -> let t1 = expr m ent e1 and t2 = expr m ent e2 in
               (match op with
       Add | Mod | Sub | Mult | Div when t1 = Int && t2 = Int -> Int
                |  Add | Sub | Mult | Div when t1 = Int && t2 = Float -> Float
                |  Add | Sub | Mult | Div when t1 = Float && t2 = Int -> Float
         |  Add | Sub | Mult | Div when t1 = Float && t2 = Float -> Float
                | Equal | Neq when t1 = t2 -> Bool
            | Less | Leq | Greater | Geq when t1 = Int && t2 = Int -> Bool
            | Less | Leq | Greater | Geq when t1 = Float && t2 = Float -> Bool
      | Less | Leq | Greater | Geq when t1 = Float && t2 = Int -> Bool
      | Less | Leq | Greater | Geq when t1 = Int && t2 = Float -> Bool

                | And | Or when t1 = Bool && t2 = Bool -> Bool
      | _ -> raise (Failure ("illegal binary operator " ^
         string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
         string_of_typ t2 ^ " in " ^ string_of_expr e))
     )

     | Unop(op, e) as ex -> let t = expr m ent e in
          (match op with
               Neq when t = Int -> Int
          | Neg when t = Float -> Float
               | Not when t = Bool -> Bool
       | _ -> raise (Failure ("illegal unary operator " ^ string_of_uop op ^
                      string_of_typ t ^ " in " ^ string_of_expr ex))
     )

     | Clr(r,g,b)  -> let t1 = expr m ent r and t2 = expr m ent g and t3 = expr m ent b in
       if (isNumType(t1) && isNumType(t2) && isNumType(t3)) then Color
       else raise (Failure ("expected numeric input for type color"))
     | Vec(x,y)  -> let t1 = expr m ent x and t2 = expr m ent y in
       if (isNumType(t1) && isNumType(t2)) then Vector
       else raise (Failure ("expected numeric input for type vector"))

     | Call(fname, actuals) as call -> let fd = functionDecl fname in
        if List.length actuals != List.length fd.formals then
          raise (Failure ("Expecting " ^ string_of_int
            (List.length fd.formals) ^ " arguments in " ^ string_of_expr call))
        else
         if (fname="add") then (
           let entName = string_of_expr (List.hd actuals) in
           if not (StringMap.mem entName allEntMembers) then
             raise (Failure(entName ^ " is not a defined entity"))
             else Int
         )
         else
          let checkSameT t1 t2 = if t1 != t2 then raise (Failure ("Incorrect actual argument
type in " ^ string_of_expr call)) in
           List.iter2 (fun (ft,_) e -> let et = expr m ent e in checkSameT ft et ) fd.formals
actuals;
          fd.typ
     | ArrayAccess(e1, e2) -> let e_type = expr m ent e1 and e_num = expr m ent e2 in
       if (e_type != Color && e_type != Vector)
         then raise (Failure ("Can only access Color and Vector types, not " ^ string_of_typ
e_type))
       else
         if (e_num != Int) then raise (Failure ("Expecting Integer for access index, got " ^
string_of_typ e_num))
         else Int

     | Assign (e1,e2) as ex-> let t1 = expr m ent e1 and t2 = expr m ent e2 in
       if t1 == t2 then t1 else raise (Failure ("illegal assignment " ^ string_of_typ t1 ^
              " = " ^ string_of_typ t2 ^ " in " ^
              string_of_expr ex))

     | Access(name,prop) ->
       if name = "self"
         then
           let availProps = StringMap.find ent allEntMembers in
           type_of_identifier availProps prop;
       else
           if (StringMap.mem name m) then raise (Failure ("Cannot access " ^ name))
           else raise (Failure ("Undeclared identifier " ^ name))
    in

  (* check if types in a varinit statement match *)
  let checkVarInit m ent  = function
     VarInit(t,n,e) -> let e_typ = expr m ent e in
       if t != e_typ
       then raise (Failure ("expected type " ^ string_of_typ t ^ ", not " ^ string_of_expr e ^
" of type " ^ string_of_typ e_typ))
        else ()
  in

  (* check if given expression is of type boolean *)
  let checkBoolExpr e m ent  = if expr m ent  e != Bool
```

```
     then raise (Failure ("expected Boolean expression in " ^ string_of_expr e))
     else ()
in

(* check statements *)
let rec stmt m ent = function
  Block sl -> let rec checkBlock = function
       [Return _ as s] -> stmt m ent s
      | Return _ :: _ -> raise (Failure "nothing may follow a return")
      | Block sl :: ss -> checkBlock (sl @ ss)
      | s :: ss -> stmt m ent s ; checkBlock ss
      | [] -> ()
     in checkBlock sl
  | Expr e -> ignore (expr m ent e)
  | If(p, b1, b2) -> checkBoolExpr p m ent ; stmt m ent  b1; stmt m ent  b2
  | While(p, s) -> checkBoolExpr p m ent ; stmt m ent  s
  | Return e -> ()

in

(* add variable to a map *)
let var m (VarInit(t, n, e)) = StringMap.add n t m in

(*** CHECK FUNC DECLS ***)

let checkReturnStmt func m =  function
  Return e -> let t = (expr m "" e) in if t = func.typ then () else
   raise (Failure ("Return gives " ^ string_of_typ t ^ " expected " ^
                   string_of_typ func.typ ^ " in " ^ string_of_expr e))
  | _ -> raise (Failure ("Function must end with return statement"))
in

let checkFunc func =

  (* check for duplicate formals *)
  reportDuplicate (fun n -> "Duplicate formal " ^ n ^ " in " ^ func.fname)
    (List.map snd func.formals);

  (* check for duplicate locals *)
  reportDuplicate (fun n -> "Duplicate local " ^ n ^ " in " ^ func.fname)
    (List.map varDeclName func.locals);

  (* build map of in-scope variables *)
  let formal_var m (t, n) = StringMap.add n t m in
  let symbols = List.fold_left var StringMap.empty func.locals in
  let symbols = List.fold_left formal_var symbols func.formals in

  (* check function locals *)
  List.iter (checkVarInit symbols "" ) func.locals;

  (* check function statements *)
  stmt symbols "" (Block func.body);

  (* make sure the last statement is a return and that it is the correct type *)
  let rev_statements = List.rev func.body in
  let getReturnStmt = function x::_ -> x | _ -> raise (Failure "Empty function") in
  let return_stmt = getReturnStmt rev_statements in
```

```
    checkReturnStmt func symbols return_stmt;

  in

  List.iter checkFunc funcdecls;

  (* check if a given member type exists *)
  let checkMemExists s t m =
    try
      let myT = StringMap.find s m
      in
      if myT != t then raise (Failure ("Inconsistent types"))
     with Not_found -> raise (Failure ("You haven't defined " ^ s))
  in

  (* check the statements within events *)
  let checkOccurence = function
    KeyPress (code) -> if (not (code = "key_UP" || code = "key_DOWN" || code = "key_LEFT" ||
code = "key_RIGHT" || code = "key_W" || code = "key_A" || code = "key_S" || code = "key_D" ||
code = "key_SPACE")) then raise (Failure (code ^ " is an undefined keypress type"))
  | Collision (slf, othr) ->
      if (slf = "self" ) then () else raise (Failure ("must define collisions against self,
not " ^ slf));
      if not (StringMap.mem othr allEntMembers) then
            raise (Failure(othr ^ " is not a defined entity"));
  | Frame -> ()
  | Click -> ()
  | _ -> raise (Failure ("Not a valid entity event"))

  in

  let checkEvent ent = function
    Event (occurence, _, bhvr) ->
       checkOccurence occurence;
       stmt StringMap.empty ent (Block bhvr)
  in

  (*** CHECK ENT DECLS ***)
  let getEntName e = e.ename in
   reportDuplicate (fun n -> "Duplicate entity " ^ n)
      (List.map getEntName entdecls);

  (* check for required members, check member types, check event statements *)
  let checkEntDecl e =
    let myMems = memTypes e.members in
      checkMemExists "clr" Color myMems;
      checkMemExists "size" Vector myMems;

    (* check for duplicate members *)
    reportDuplicate (fun n -> "Duplicate member " ^ n ^ " in " ^ e.ename)
      (List.map varDeclName e.members);

    (* build a symbols map - variables within scope *)
    let symbols =  List.fold_left var StringMap.empty e.members in

    (* run checks *)
    List.iter (checkVarInit symbols e.ename) e.members;
```

```
       List.iter (checkEvent e.ename) e.rules ;
  in
  List.iter checkEntDecl entdecls;

  (*** CHECK GAMEBOARD DECL ***)

  let checkGbDecl gb =

    (* check required members clr and size *)
    let myMems = memTypes gb.gmembers in
      checkMemExists "clr" Color myMems;
      checkMemExists "size" Vector myMems;

    (* check for duplicate members *)
    reportDuplicate (fun n -> "Duplicate member " ^ n ^ " in " ^ gb.gname)
      (List.map varDeclName gb.gmembers);

    (* build a symbols map - variables within scope *)
    let symbols =  List.fold_left var StringMap.empty gb.gmembers in

    (* check members *)
    List.iter (checkVarInit symbols "" ) gb.gmembers;

    (* check for duplicate init members *)
    reportDuplicate (fun n -> "Duplicate init function member " ^ n ^ " in " ^ gb.gname)
      (List.map varDeclName gb.init_mem);

    (* check init members and add to symbols list *)
    let symbols =  List.fold_left var symbols gb.init_mem in
    List.iter (checkVarInit symbols "") gb.init_mem;

    (* check init body *)
    stmt symbols "" (Block gb.init_body);
  in
  checkGbDecl gboard;
```

codegen.ml

```
module L = Llvm
module A = Ast

module StringMap = Map.Make(String)

exception Blr_err of string;;

let print_map m =
  print_string ("Map:\n");
  let print_key k v =
    print_string (k ^ "\n")
  in
  StringMap.iter print_key m;;

let translate (vardecls, fdecls, ents, gboard) =
  let context = L.global_context () in
  let the_module = L.create_module context "Ballr" in
  let i64_t  = L.i64_type  context in
  let i32_t  = L.i32_type  context in
  let i8_t   = L.i8_type   context in
  let i1_t   = L.i1_type   context in
  let flt_t = L.float_type context in
  let ut_hash_handle_t = L.named_struct_type context "UT_hash_handle" in
  let ut_hash_table_t = L.named_struct_type context "UT_hash_table" in
  let ut_hash_bucket_t = L.named_struct_type context "UT_hash_bucket" in
    L.struct_set_body ut_hash_handle_t [|L.pointer_type ut_hash_table_t; L.pointer_type i8_t;
L.pointer_type i8_t; L.pointer_type ut_hash_handle_t; L.pointer_type ut_hash_handle_t;
L.pointer_type i8_t; i32_t; i32_t|] false;
    L.struct_set_body ut_hash_table_t  [|L.pointer_type ut_hash_bucket_t; i32_t; i32_t; i32_t;
L.pointer_type ut_hash_handle_t; i64_t; i32_t; i32_t; i32_t; i32_t; i32_t|] false;
    L.struct_set_body ut_hash_bucket_t [|L.pointer_type ut_hash_handle_t; i32_t; i32_t|]
false;
  let clr_t = L.named_struct_type context "blr_color_t" in
    L.struct_set_body clr_t [|i32_t; i32_t; i32_t|] false;
  let vec_t = L.named_struct_type context "blr_size_t" in
    L.struct_set_body vec_t [|i32_t; i32_t|] false;
  let ent_t = L.named_struct_type context "blr_entity_t" in
    L.struct_set_body ent_t [|  L.pointer_type i8_t; vec_t; vec_t; clr_t; L.pointer_type
(L.function_type (L.void_type context) [| L.pointer_type ent_t |]); L.pointer_type i8_t;
L.pointer_type ent_t; ut_hash_handle_t |] false;
  let gb_t = L.named_struct_type context "blr_gameboard_t" in
    L.struct_set_body gb_t [| L.pointer_type i8_t; vec_t; clr_t; L.pointer_type ent_t;
L.pointer_type (L.function_type (L.void_type context) [| L.pointer_type gb_t |]);
ut_hash_handle_t |] false;

  let ltype_of_typ = function
      A.Int -> i32_t
    | A.Bool -> i1_t
    | A.Float -> flt_t
    | A.Color -> clr_t
    | A.Vector -> vec_t
  in

  let keycode_of_keyname name =
    if name = "key_UP" then 82 else
    if name = "key_DOWN" then 81 else
    if name = "key_LEFT" then 80 else
    if name = "key_RIGHT" then 79 else
    if name = "key_SPACE" then 44 else
    if name = "key_A" then 4 else
    if name = "key_D" then 7 else
    if name = "key_S" then 22 else
    if name = "key_W" then 26 else
    0
  in

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

  let run_loop_t = L.function_type i32_t [| |] in
  let run_loop_func = L.declare_function "run_loop" run_loop_t the_module in

  let register_gb_t = L.function_type (L.void_type context) [| (L.pointer_type gb_t) |] in
  let register_gb_func = L.declare_function "register_gb" register_gb_t the_module in
  let add_fn_t = L.function_type (L.void_type context) [| (L.pointer_type ent_t) |] in
  let add_fn = L.declare_function "ent_add" add_fn_t the_module in

  let remove_fn_t = L.function_type (L.void_type context) [| (L.pointer_type ent_t) |] in
  let remove_fn = L.declare_function "ent_remove" remove_fn_t the_module in

  let restart_fn_t = L.function_type (L.void_type context) [| |] in
  let restart_fn = L.declare_function "restart" restart_fn_t the_module in

  let chk_kp_t = L.function_type i32_t [| i32_t |] in
  let chk_kp_fn = L.declare_function "chk_keypress" chk_kp_t the_module in

  let coll_callback_t = L.function_type (L.void_type context) [| L.pointer_type ent_t |] in
  let chk_coll_t = L.function_type (L.void_type context) [| L.pointer_type ent_t;
L.pointer_type i8_t; L.pointer_type coll_callback_t |] in
  let chk_coll_fn = L.declare_function "chk_collision" chk_coll_t the_module in

  let clk_callback_t = L.function_type (L.void_type context) [| L.pointer_type ent_t;
L.pointer_type vec_t |] in
  let chk_clk_t = L.function_type (L.void_type context) [| L.pointer_type ent_t;
L.pointer_type clk_callback_t |] in
  let chk_clk_fn = L.declare_function "chk_click" chk_clk_t the_module in

  let get_decl name decls =
    match List.filter (fun (A.VarInit (t, s, e)) -> s = name) decls with
      | A.VarInit (t, s, e) :: tl -> e
      | [] -> A.Noexpr in

  let clr_lit = function
    | A.Clr (A.Literal r, A.Literal g, A.Literal b) -> Some([|L.const_int i32_t r; L.const_int
i32_t g; L.const_int i32_t b|])
    | _ -> None in

  let vec_lit = function
    | A.Vec (A.Literal v1, A.Literal v2) -> Some([|L.const_int i32_t v1; L.const_int i32_t
v2|])
    | _ -> None in

  let int_format_str builder = L.build_global_stringptr "%d\n" "fmt" builder in (* format
string for printf calls *)

  let add_terminal builder f =
    match L.block_terminator (L.insertion_block builder) with
      Some _ -> ()
      | None -> ignore (f builder) in

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

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

  (* BUILD EXPRESSIONS *)
  let rec expr builder m mem_m ent = function
      A.Literal i -> L.const_int i32_t i
    | A.FLiteral f -> L.const_float flt_t f
    | A.BoolLit b -> L.const_int i1_t (if b then 1 else 0)
    | A.Noexpr  -> L.const_int i32_t 0
    | A.Id s    -> L.build_load (StringMap.find s m) s builder

    | A.Call ("print", [e]) | A.Call ("printb", [e]) ->
      L.build_call printf_func
        [| int_format_str builder ; (expr builder m mem_m ent e) |]
        "printf" builder

    | A.Call ("add", [e1; e2]) ->
      let e2' = expr builder m mem_m ent e2 in
      (match e1 with
        A.Id (name) ->
          let creat_fn = StringMap.find (name ^ "_create") m in
          let ent_ptr = L.build_call creat_fn [| |] name builder in
          let pos_ptr = L.build_struct_gep ent_ptr 2 "pos" builder in
          ignore (L.build_store e2' pos_ptr builder);
          ignore (L.build_call add_fn [| ent_ptr |] "" builder);
          L.const_int i32_t 0
        | _ -> raise (Blr_err "expected identifier"))

    | A.Call ("remove", []) ->
          let (ent_ptr, _, _) = StringMap.find ent mem_m in
          L.build_call remove_fn [| ent_ptr |] "" builder
    | A.Call ("restart", []) ->
          L.build_call restart_fn [||] "" builder
    | A.Call (name, args) ->
      let func = StringMap.find name m in
      let arg_arr = Array.of_list (List.map (expr builder m mem_m ent) args) in
      L.build_call func arg_arr name builder

    | A.Binop (e1, op, e2) ->
      let e1' = expr builder m mem_m ent e1
      and e2' = expr builder m mem_m ent e2 in
      if (L.type_of e1' = flt_t || L.type_of e2' = flt_t) then
        (match op with
          A.Add     -> L.build_fadd
        | A.Sub     -> L.build_fsub
        | A.Mult    -> L.build_fmul
        | A.Mod     -> L.build_frem
        | A.Div     -> L.build_fdiv
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
        | A.Neq     -> L.build_fcmp L.Fcmp.One
        | A.Less    -> L.build_fcmp L.Fcmp.Olt
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge
        | _         -> raise (Blr_err "invalid operands for operator")
        ) (ensureFloat e1') (ensureFloat e2') "tmp" builder
      else
      (match op with
        A.Add     -> L.build_add
      | A.Sub     -> L.build_sub
      | A.Mult    -> L.build_mul
      | A.Mod     -> L.build_srem
      | A.Div     -> L.build_sdiv
      | A.And     -> L.build_and
      | A.Or      -> L.build_or
```

```
      | A.Equal   -> L.build_icmp L.Icmp.Eq
      | A.Neq     -> L.build_icmp L.Icmp.Ne
      | A.Less    -> L.build_icmp L.Icmp.Slt
      | A.Leq     -> L.build_icmp L.Icmp.Sle
      | A.Greater -> L.build_icmp L.Icmp.Sgt
      | A.Geq     -> L.build_icmp L.Icmp.Sge
      ) e1' e2' "tmp" builder

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

    | A.Clr (e1, e2, e3) as clr ->
      (match (clr_lit clr) with
        Some(vals) -> L.const_named_struct clr_t vals
        | None ->
         let e1' = ensureInt (expr builder m mem_m ent e1)
         and e2' = ensureInt (expr builder m mem_m ent e2)
    and e3' = ensureInt (expr builder m mem_m ent e3) in
          let clr_ptr = L.build_alloca clr_t "tmp" builder in
         let r_ptr = L.build_struct_gep clr_ptr 0 "r" builder in
         ignore (L.build_store e1' r_ptr builder);
          let g_ptr = L.build_struct_gep clr_ptr 1 "g" builder in
         ignore (L.build_store e2' g_ptr builder);
          let b_ptr = L.build_struct_gep clr_ptr 2 "b" builder in
          ignore (L.build_store e3' b_ptr builder);
         L.build_load clr_ptr "c" builder)

    | A.Vec (e1, e2) as vec ->
      (match (vec_lit vec) with
        Some(vals) -> L.const_named_struct vec_t vals
        | None ->
          let e1' = ensureInt (expr builder m mem_m ent e1)
          and e2' = ensureInt (expr builder m mem_m ent e2) in
          let vec_ptr = L.build_alloca vec_t "tmp" builder in
          let x_ptr = L.build_struct_gep vec_ptr 0 "x" builder in
          ignore (L.build_store e1' x_ptr builder);
          let y_ptr = L.build_struct_gep vec_ptr 1 "y" builder in
          ignore (L.build_store e2' y_ptr builder);
          L.build_load vec_ptr "v" builder)

    | A.Assign (e1, e2) ->
      let e' = expr builder m mem_m ent e2 in
      (match e1 with
        | A.Id (s) -> L.build_store e' (StringMap.find s m) builder
        | A.Access (s1, s2) ->
          (match s1 with
            | "self" ->
              (match s2 with
                | "size" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                            let size_ptr = L.build_struct_gep ent_ptr 1 ("ent_size_ptr")
builder in
                            L.build_store e' size_ptr builder;
                | "clr" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
```

```
                            let clr_ptr = L.build_struct_gep ent_ptr 3 ("ent_clr_ptr") builder
in
                            L.build_store e' clr_ptr builder;
                | "pos" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                            let pos_ptr = L.build_struct_gep ent_ptr 2 ("ent_pos_ptr") builder
in
                            L.build_store e' pos_ptr builder;
                |_ -> let (ent_mem_ptr, index, m_t) = StringMap.find s2 mem_m in
                        let mem_struct_ptr = L.build_load ent_mem_ptr "mem_struct_ptr" builder
in
                        let mem_struct_ptr = L.build_pointercast mem_struct_ptr
(L.pointer_type m_t) "cast_ptr" builder in
                        let mem_ptr = L.build_struct_gep mem_struct_ptr index (s2 ^ "_ptr")
builder in
                        L.build_store e' mem_ptr builder;
              )
            | _ -> L.const_int i32_t 0
          )

        | A.ArrayAccess (e1', e2') ->
          let arr_ptr =
          (match e1' with
            | A.Access (s1, s2) ->
              (match s1 with
                | "self" ->
                  ( match s2 with
                    | "size" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                                L.build_struct_gep ent_ptr 1 ("ent_size_ptr") builder
                    | "clr" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                                L.build_struct_gep ent_ptr 3 ("ent_clr_ptr") builder
                    | "pos" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                                L.build_struct_gep ent_ptr 2 ("ent_pos_ptr") builder
                    |_ -> let (ent_mem_ptr, index, m_t) = StringMap.find s2 mem_m in
                          let mem_struct_ptr = L.build_load ent_mem_ptr "mem_struct_ptr"
builder in
                          let mem_struct_ptr = L.build_pointercast mem_struct_ptr
(L.pointer_type m_t) "cast_ptr" builder in
                          L.build_struct_gep mem_struct_ptr index (s2 ^ "_ptr") builder
                  )

                | _ -> L.const_int i32_t 0
              )
            | A.Id (s) -> L.build_gep (StringMap.find s m) [|L.const_int i32_t 0|] s builder
            | _ -> L.const_int i32_t 0
          )

          in

          let index =
            ( match e2' with
            | A.Literal i -> i
            | _ -> 0
            )
          in

          let el_accessed = L.build_struct_gep arr_ptr index ("el_"^string_of_int index)
builder in
```

```
          L.build_store e' el_accessed builder;

        | _ -> L.const_int i32_t 0
      )

    | A.Access (s1, s2) ->
    (match s1 with
      | "self" ->
        (match s2 with
          | "size" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                      let size_ptr = L.build_struct_gep ent_ptr 1 ("ent_size_ptr") builder in
                      L.build_load size_ptr s2 builder;
          | "clr" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                      let clr_ptr = L.build_struct_gep ent_ptr 3 ("ent_clr_ptr") builder in
                      L.build_load clr_ptr s2 builder;
          | "pos" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                      let pos_ptr = L.build_struct_gep ent_ptr 2 ("ent_pos_ptr") builder in
                      L.build_load pos_ptr s2 builder;
          |_ -> let (ent_mem_ptr, index, m_t) = StringMap.find s2 mem_m in
                  let mem_struct_ptr = L.build_load ent_mem_ptr "mem_struct_ptr" builder in
                  let mem_struct_ptr = L.build_pointercast mem_struct_ptr (L.pointer_type m_t)
"cast_ptr" builder in
                  let mem_ptr = L.build_struct_gep mem_struct_ptr index (s2 ^ "_ptr") builder
in
                  L.build_load mem_ptr s2 builder;
        )
      | _ -> L.const_int i32_t 0
    )

    | A.ArrayAccess (e1, e2) ->

      let arr_ptr =
        (match e1 with
          | A.Access (s1, s2) ->
            (match s1 with
              | "self" ->
                (match s2 with
                  | "size" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                              L.build_struct_gep ent_ptr 1 ("ent_size_ptr") builder
                  | "clr" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                              L.build_struct_gep ent_ptr 3 ("ent_clr_ptr") builder
                  | "pos" -> let (ent_ptr, _, _) = StringMap.find ent mem_m in
                              L.build_struct_gep ent_ptr 2 ("ent_pos_ptr") builder
                  |_ -> let (ent_mem_ptr, index, m_t) = StringMap.find s2 mem_m in
                          let mem_struct_ptr = L.build_load ent_mem_ptr "mem_struct_ptr"
builder in
                          let mem_struct_ptr = L.build_pointercast mem_struct_ptr
(L.pointer_type m_t) "cast_ptr" builder in
                          L.build_struct_gep mem_struct_ptr index (s2 ^ "_ptr") builder
                )

              | _ -> L.const_int i32_t 0
            )
          | A.Id (s) -> L.build_gep (StringMap.find s m) [|L.const_int i32_t 0|] s builder
          | _ -> L.const_int i32_t 0
        )
      in

      let index =
        ( match e2 with
        | A.Literal i -> i
        | _ -> 0
        )
      in

      let el_accessed = L.build_struct_gep arr_ptr index ("el_"^string_of_int index) builder
in
      L.build_load el_accessed "ptr" builder;
  in

  let int_of_bool b = if b then 1 else 0 in

  (* BUILD STATEMENTS *)
  let rec stmt builder func m mem_m ent = function
      A.Block sl -> List.fold_left (fun b s -> stmt b func m mem_m ent s) builder sl
    | A.Expr e -> ignore (expr builder m mem_m ent e); builder
    | A.Return e -> ignore (L.build_ret (expr builder m mem_m ent e) builder); builder

    | A.If (predicate, then_stmt, else_stmt) ->
      let bool_val = expr builder m mem_m ent predicate in
      let merge_bb = L.append_block context "merge" func in
        let then_bb = L.append_block context "then" func in
        add_terminal
          (stmt (L.builder_at_end context then_bb) func m mem_m ent then_stmt)
          (L.build_br merge_bb);

        let else_bb = L.append_block context "else" func in
        add_terminal
          (stmt (L.builder_at_end context else_bb) func m mem_m ent else_stmt)
          (L.build_br merge_bb);

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

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

      let body_bb = L.append_block context "while_body" func in
      add_terminal (stmt (L.builder_at_end context body_bb) func m mem_m ent body)
          (L.build_br pred_bb);

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

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

  in

  let add_locals m mem_m ent decls builder =
    let add_local m (A.VarInit(t, n, e)) =
      let e' = expr builder m mem_m ent e in
      let local_var = L.build_alloca (ltype_of_typ t) n builder in
      ignore (L.build_store e' local_var builder);
      StringMap.add n local_var m
    in
   List.fold_left add_local m decls
  in

  let rec get_init_val = function
    A.Literal i -> L.const_int i32_t i
    | A.FLiteral f -> L.const_float flt_t f
    | A.BoolLit b -> L.const_int i1_t (int_of_bool b)
    | A.Noexpr    -> L.const_int i32_t 0
    | A.Clr (e1, e2, e3) as clr ->
      (match (clr_lit clr) with
        Some(vals) -> L.const_named_struct clr_t vals
        | None -> L.const_named_struct clr_t [|L.const_int i32_t 0; L.const_int i32_t 0;
L.const_int i32_t 0|](* shouldnt get here if passed semant *)
      )
    | A.Vec (e1, e2) as vec ->
      (match (vec_lit vec) with
        Some(vals) -> L.const_named_struct vec_t vals
        | None -> L.const_named_struct vec_t [|L.const_int i32_t 0; L.const_int i32_t 0|]
      )
    | A.Unop(op, e) ->
      let e' = get_init_val e in
      ( match op with
          A.Neg     -> (match e with
                          A.Literal i -> L.const_neg e'
                        | A.FLiteral f -> L.const_fneg e'
                        | _ -> L.const_int i32_t 0 )
        | _ -> L.const_int i32_t 0
      )
    | _ -> L.const_int i32_t 0 (* semant shouldn't let it get here *)
  in

  (* GLOBAL VAR DECLARATIONS *)
  let global_vars =
    let global_var m (A.VarInit(t, n, e)) =
      let initval = get_init_val e in
      StringMap.add n (L.define_global n initval the_module) m
    in
    List.fold_left global_var StringMap.empty vardecls
  in

  (* FUNCTION DECLARATIONS *)

  let func_create f m =
    let name = f.A.fname
    and formal_types = Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) f.A.formals) in
    let ftype = L.function_type (ltype_of_typ f.A.typ) formal_types in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  (* Fill in the body of the given function *)
  let fill_func_body f m =
    let (map, func) = func_create f m in
    let builder = L.builder_at_end context (L.entry_block func) in

    let map =
      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
      in List.fold_left2 add_formal map f.A.formals (Array.to_list (L.params func))
    in
    let map = add_locals map StringMap.empty "" f.A.locals builder in
    ignore (stmt builder func map StringMap.empty "" (A.Block f.A.body)); map
  in

  (* ENTITY DECLARATIONS *)

  let decl_type = function A.VarInit(t,s,e) -> ltype_of_typ(t) in
  let decl_name = function A.VarInit(t,s,e) -> s in
  let decl_expr = function A.VarInit(t,s,e) -> e in

  (* make struct <ename>_mems_t for each entity *)
  let define_emembers_type e m =
    let add_mem_type a mem =
      let name = decl_name mem in
      if name = "clr" || name = "size" then a
      else Array.append a [|(decl_type mem)|]
    in

    (* array of member types (not including clr, size) *)
    let mem_types = List.fold_left add_mem_type [||] e.A.members in
    (* define named struct *)
    let mems_t = L.named_struct_type context (e.A.ename ^ "_mems_t") in
    L.struct_set_body mems_t mem_types false;
    (* add to map *)
    StringMap.add e.A.ename mems_t m
  in

  (** make struct name_mems_t structs for each entity and put in a map *)
  let ent_mem_types = List.fold_left (fun m e -> define_emembers_type e m) StringMap.empty
ents in

  (* build a map of accessible member names to corresponding struct gep *)
    let remove_first l = match l with
      | [] -> []
      | hd::rest -> rest
    in

  let ent_collision e oname m =
    let name = e.A.ename ^ "." ^ oname ^ ".collision" in
    let ftype = L.function_type (L.void_type context) [| L.pointer_type ent_t |] in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  let ent_keypress e k m =
    let name = e.A.ename ^ ".keypress" ^ k in
    let ftype = L.function_type (L.void_type context) [| L.pointer_type ent_t |] in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  let ent_onclick e m =
    let name = e.A.ename ^ ".onclick" in
    let ftype = L.function_type (L.void_type context) [| L.pointer_type ent_t; L.pointer_type
vec_t |] in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  let ent_frame e m =
    let name = e.A.ename ^ "_frame" in
    let ftype = L.function_type (L.void_type context) [| L.pointer_type ent_t |] in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  let build_mem_map e sp ep =
    let mems_t = StringMap.find e.A.ename ent_mem_types in

    let rec index_of x l = match l with
      | [] -> 0
      | hd :: tl -> if x = hd then 0 else 1 + index_of x tl
    in

    let add_mem map mem =
      let index = (index_of mem e.A.members) - 2 in
      StringMap.add (decl_name mem) (sp, index, mems_t) map;
    in

    (* members list without clr and pos *)
    let members = remove_first e.A.members in
    let members = remove_first members in
    let map = List.fold_left add_mem StringMap.empty members in
    StringMap.add e.A.ename (ep, 0, mems_t) map

  in

  let fill_ent_collision_func e (A.Event(A.Collision(name1, name2), v, s))  m =
    let (map, func) = ent_collision e name2 m in
    let builder = L.builder_at_end context (L.entry_block func) in
    let self_ptr = L.param func 0 in
    let mem_struct_ptr = L.build_struct_gep self_ptr 5 ("members_ptr") builder in

    let mem_map = build_mem_map e mem_struct_ptr self_ptr in

    let map = add_locals map mem_map e.A.ename v builder in
    ignore (stmt builder func map mem_map e.A.ename (A.Block s));
    ignore (L.build_ret_void builder);
    func
  in

  let fill_ent_keypress_func e (A.Event(A.KeyPress(k), v, s)) m =

    let (map, func) = ent_keypress e k m in
    let builder = L.builder_at_end context (L.entry_block func) in
    let self_ptr = L.param func 0 in
    let mem_struct_ptr = L.build_struct_gep self_ptr 5 ("members_ptr") builder in

    let mem_map = build_mem_map e mem_struct_ptr self_ptr in

    let map = add_locals map mem_map e.A.ename v builder in
    ignore (stmt builder func map mem_map e.A.ename (A.Block s));
    ignore (L.build_ret_void builder);
    func
  in

  let fill_ent_clk_func e (A.Event(A.Click, v, s)) m =
    let (map, func) = ent_onclick e m in
    let builder = L.builder_at_end context (L.entry_block func) in
    let self_ptr = L.param func 0 in
    let mem_struct_ptr = L.build_struct_gep self_ptr 5 ("members_ptr") builder in

    let mem_map = build_mem_map e mem_struct_ptr self_ptr in

    let clkpos_ptr = L.param func 1 in
    let local_clkpos = L.build_alloca vec_t "clickpos" builder in
    let clkpos_val = L.build_load clkpos_ptr "clkpos_val" builder in
    ignore (L.build_store clkpos_val local_clkpos builder);
    let map = StringMap.add "clickpos" local_clkpos map in

    let map = add_locals map mem_map e.A.ename v builder in
    ignore (stmt builder func map mem_map e.A.ename (A.Block s));
    ignore (L.build_ret_void builder);
    func
  in

  let build_event e ep (A.Event(ec,v,s) as ev) m mem_map f builder = match ec with
    | A.Collision(_, s2) ->
      let func = fill_ent_collision_func e ev m in
      let str_ptr = L.build_global_stringptr s2 "namestr" builder in
      ignore (L.build_call chk_coll_fn [| ep;  str_ptr; func |] "" builder);
      builder
    | A.KeyPress(k) ->
      let code = keycode_of_keyname k in
      let pressed = L.build_call chk_kp_fn [| L.const_int i32_t code |] "pressed" builder in

      let true_bb = L.append_block context ("true_"^k) f in
      let false_bb = L.append_block context ("false_"^k) f in

      let func = fill_ent_keypress_func e ev m in
      let true_builder = L.builder_at_end context true_bb in
      ignore (L.build_call func [| ep |] "" true_builder);
      ignore (L.build_br false_bb true_builder);

      let pressed = L.build_icmp L.Icmp.Ne pressed (L.const_int i32_t 0) ("pressed_"^k)
builder in
      ignore(L.build_cond_br pressed true_bb false_bb builder);

      L.builder_at_end context false_bb

    | A.Click ->
```

```
      let func = fill_ent_clk_func e ev m in
      ignore (L.build_call chk_clk_fn [| ep; func |] "" builder);
      builder

    | A.Frame ->
      let map = add_locals m mem_map e.A.ename v builder in
      stmt builder f map mem_map e.A.ename (A.Block s)

  in

  let fill_ent_frame_function e m =

    let (map, func) = ent_frame e m in
    let builder = L.builder_at_end context (L.entry_block func) in
    let ent_ptr = L.param func 0 in
    let mem_struct_ptr = L.build_struct_gep ent_ptr 5 ("members_ptr") builder in

    let mem_map = build_mem_map e mem_struct_ptr ent_ptr in

    let builder = List.fold_left (fun b ev -> build_event e ent_ptr ev map mem_map func b)
builder e.A.rules in

    ignore (L.build_ret_void builder);
    map
  in

  let ent_create e m =
    let name = e.A.ename ^ "_create" in
    let ftype = L.function_type (L.pointer_type ent_t) [| |] in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  let fill_ent_mem_struct e builder =

    let mems_t = StringMap.find e.A.ename ent_mem_types in
    let mems_struct_ptr = L.build_malloc mems_t ("ent_mems_t_ptr") builder in

    let build_store_mem index mem =
        let ptr = L.build_struct_gep mems_struct_ptr index ("ptr_" ^ (string_of_int index))
builder in
        ignore (L.build_store (get_init_val (decl_expr mem))  ptr builder);
    in

    (* members list without clr and pos *)
    let members = remove_first e.A.members in let members = remove_first members in
    List.iteri build_store_mem members;

    let mems_struct_ptr = L.build_pointercast mems_struct_ptr (L.pointer_type i8_t) "new"
builder in

    mems_struct_ptr
  in

  let fill_ent_create_function m e =
    let (map, func) = ent_create e m in
    let builder = L.builder_at_end context (L.entry_block func) in
    let ent_ptr = L.build_malloc ent_t ("ent_ptr") builder in

    let name_str_ptr = L.build_global_stringptr e.A.ename (e.A.ename ^ "_name_str_ptr")
builder in
    let name_ptr = L.build_struct_gep ent_ptr 0 ("name_ptr") builder in
    ignore (L.build_store name_str_ptr name_ptr builder);

    let ent_size_ptr = L.build_struct_gep ent_ptr 1 ("size_ptr") builder in
    let size_decl_expr = get_decl "size" e.A.members in
    ignore (L.build_store (expr builder map StringMap.empty e.A.ename size_decl_expr)
ent_size_ptr builder);

    let ent_pos_ptr = L.build_struct_gep ent_ptr 2 ("pos_ptr") builder in
    let pos_zero = [|L.const_int i32_t 0; L.const_int i32_t 0|] in
    ignore (L.build_store (L.const_named_struct vec_t pos_zero) ent_pos_ptr builder);

    let ent_color_ptr = L.build_struct_gep ent_ptr 3 ("color_ptr") builder in
    let color_decl_expr = get_decl "clr" e.A.members in
    ignore (L.build_store (expr builder map StringMap.empty e.A.ename color_decl_expr)
ent_color_ptr builder);

    let ent_mems_t_ptr = L.build_struct_gep ent_ptr 5 ("members_ptr") builder in
    let mems_struct_ptr = fill_ent_mem_struct e builder in
    ignore (L.build_store mems_struct_ptr ent_mems_t_ptr builder);

    let ent_frame_fn_ptr = L.build_struct_gep ent_ptr 4 ("frame_fn_ptr") builder in
    let map = fill_ent_frame_function e map in
    let frame_fn = StringMap.find (e.A.ename ^ "_frame") map in
    ignore (L.build_store frame_fn ent_frame_fn_ptr builder);

    ignore (L.build_ret ent_ptr builder); map
  in

  (* GAMEBOARD DECLARATION *)

  let gb_init gb m =
    let name = gb.A.gname ^ "_init" in
    let ftype = L.function_type (L.void_type context) [| L.pointer_type gb_t |] in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  let fill_init_function gb m =
    let (map, func) = (gb_init gb m) in
    let builder = L.builder_at_end context (L.entry_block func) in

    let map = add_locals map StringMap.empty "" gb.A.init_mem builder in
    let builder = stmt builder func map StringMap.empty "" (A.Block gb.A.init_body) in
    ignore (L.build_ret_void builder);
    map
  in

  let gb_create gb m =
    let name = gb.A.gname ^ "_create" in
    let ftype = L.function_type (L.pointer_type gb_t) [| |] in
    let func = L.define_function name ftype the_module in
    (StringMap.add name func m, func)
  in

  let fill_gb_create_function gb m =
    let map = fill_init_function gb m in
    let (map, func) =  (gb_create gb map) in
    let builder = L.builder_at_end context (L.entry_block func) in
    let gb_ptr = L.build_malloc gb_t ("board_ptr") builder in

    let name_str_ptr = L.build_global_stringptr gb.A.gname (gb.A.gname ^ "_name_str_ptr")
builder in
    let name_ptr = L.build_struct_gep gb_ptr 0 ("name_ptr") builder in
    ignore (L.build_store name_str_ptr name_ptr builder);

    let gb_size_ptr = L.build_struct_gep gb_ptr 1 ("size_ptr") builder in
    let size_decl_expr = get_decl "size" gb.A.gmembers in
    ignore (L.build_store (expr builder map StringMap.empty "" size_decl_expr) gb_size_ptr
builder);

    let gb_color_ptr = L.build_struct_gep gb_ptr 2 ("color_ptr") builder in
    let color_decl_expr = get_decl "clr" gb.A.gmembers in
    ignore (L.build_store (expr builder map StringMap.empty "" color_decl_expr) gb_color_ptr
builder);

    let gb_init_fn_ptr = L.build_struct_gep gb_ptr 4 ("init_fn_ptr") builder in
    let init_fn = StringMap.find (gb.A.gname ^ "_init") map in
    ignore (L.build_store init_fn gb_init_fn_ptr builder);

    ignore (L.build_call register_gb_func [| gb_ptr |] "" builder);

    ignore (L.build_ret gb_ptr builder);
    map
  in

  (* ADD FUNCTION AND ENTITY DECLARATIONS *)
  let fmap = List.fold_left (fun m f -> fill_func_body f m) global_vars fdecls in
  let fmap = List.fold_left (fun m e -> fill_ent_create_function m e) fmap ents in

  (* CREATE GAMEBOARD *)
  let fmap = fill_gb_create_function gboard fmap in

  (* DEFINE MAIN FUNCTION: calls gb_create, run_loop, return 0 *)
  let main_ftype = L.function_type i32_t [||] in
  let main_function = L.define_function "main" main_ftype the_module in
  let main_builder = L.builder_at_end context (L.entry_block main_function) in
  ignore (L.build_call (StringMap.find (gboard.A.gname ^ "_create") fmap) [||] ""
main_builder);
  ignore (L.build_call run_loop_func [| |] "" main_builder);
  ignore (L.build_ret (L.const_int i32_t 0) main_builder);

  the_module
```

ballr.ml

```
open Printf

type action = Ast | LLVM_IR | Compile
```

```
let _ =
  let action =
  if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [ ("-a", Ast);  (* Print the AST only *)
                                              ("-l", LLVM_IR);  (* Generate LLVM, don't check
*)
                                              ("-c", Compile) ] (* Generate, check LLVM IR *)
  else
    Compile
  in
  let (ic, oc, exec_name) =
  if Array.length Sys.argv > 2 then
    let in_f = Sys.argv.(2) in
    let i = (try String.rindex in_f '.' with Not_found -> String.length in_f) in
    let exec_name = String.sub in_f 0 i in
    let out_f = exec_name ^ ".ll" in
    (open_in in_f, open_out out_f, exec_name)
  else
    (stdin, stdout, "")
  in
  let lexbuf = Lexing.from_channel ic in
  let ast = Parser.program Scanner.token lexbuf in
  Semant.check ast;
    match action with
    Ast -> print_string (Ast.string_of_program ast)
  | LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate ast))
  | Compile -> let m = Codegen.translate ast in
    Llvm_analysis.assert_valid_module m;
    fprintf oc "%s\n" (Llvm.string_of_llmodule m);
    close_out oc;
    if exec_name != "" then
      let cmd = "clang -c " ^ exec_name ^ ".ll && clang -g " ^ exec_name ^ ".o -o " ^
exec_name ^ " -L/usr/local/lib -Wl,-rpath,/usr/local/lib -Wl,--enable-new-dtags -lSDL2
-lblr_rt" in
      print_string (cmd ^ "\n");
      ignore (Sys.command cmd)
    else ()
```

A.2 Runtime

ballr.h

```
#ifndef BALLR_H
#define BALLR_H

#include "uthash/include/uthash.h"
#include "uthash/include/utlist.h"

typedef struct blr_color {
    int r;
    int g;
    int b;
} blr_color_t;

typedef struct blr_pos {
    int x;
    int y;
} blr_pos_t;

typedef struct blr_size {
    int w;
    int h;
} blr_size_t;


typedef struct entity {
    const char *name;
    blr_size_t size;
    blr_pos_t pos;
    blr_color_t color;
    void (*frame_fn)(struct entity *);
    void *members;
    struct entity *next;
    UT_hash_handle hh;
} entity_t;

typedef struct gameboard {
    const char *name;
    blr_size_t size;
    blr_color_t color;
    entity_t *ents;
    void (*init_fn)(struct gameboard *);
    UT_hash_handle hh;
} gameboard_t;

void ent_add(entity_t *e);

void ent_remove(entity_t *e);

void chk_collision(entity_t *e, const char *other_name, void (*callback)(entity_t *));

void chk_click(entity_t *e, void (*callback)(entity_t *, blr_pos_t *));

int chk_keypress(int keycode);

#endif
```

ballr.c

```
#include "window.h"
#include "ballr.h"

extern gameboard_t *current_board;
extern entity_t *to_delete;
extern blr_pos_t click_pos;
```

```
void ent_add(entity_t *e) {
    entity_t *elist;
    HASH_FIND_STR(current_board->ents, e->name, elist);
    if (elist == NULL) {
        elist = malloc(sizeof(entity_t));
        elist->name = e->name;
        elist->next = NULL;
        HASH_ADD_STR(current_board->ents, name, elist);
    }
    LL_APPEND(elist, e);
}

void ent_remove(entity_t *e) {
    entity_t *elist;
    HASH_FIND_STR(current_board->ents, e->name, elist);
    if (elist != NULL) {
        LL_DELETE(elist, e);
        LL_APPEND(to_delete, e);
    }
}

int rect_intersect(int x1, int y1, int w1, int h1, int x2, int y2, int w2, int h2) {
    if (x1 > x2 + w2 ||
        x2 > x1 + w1 ||
        y1 > y2 + h2 ||
        y2 > y1 + h1) {
        return 0;
    } else {
        return 1;
    }
}

int ents_touching(entity_t *e1, entity_t *e2) {
    return rect_intersect(e1->pos.x, e1->pos.y, e1->size.w, e1->size.h,
        e2->pos.x, e2->pos.y, e2->size.w, e2->size.h);
}

void chk_collision(entity_t *e, const char *other_name, void (*callback)(entity_t *)) {
    entity_t *elist;
    HASH_FIND_STR(current_board->ents, other_name, elist);

    if (elist == NULL) {
        return;
    }

    entity_t *tmp, *elt;
    LL_FOREACH_SAFE(elist->next, elt, tmp) {
        if (e == elt) {
            continue;
        } else if (ents_touching(e, elt)) {
            callback(e);
        }
    }
}

void chk_click(entity_t *e, void (*callback)(entity_t *, blr_pos_t *)) {
    if (click_pos.x != -1) {
        callback(e, &click_pos);
    }
}

int chk_keypress(int keycode) {
    return is_key_down(keycode);
}
```

window.h

```
#ifndef WINDOW_H
#define WINDOW_H

#include "ballr.h"

void register_gb(gameboard_t *board);

int run_loop();

int is_key_down(int scancode);

#endif
```

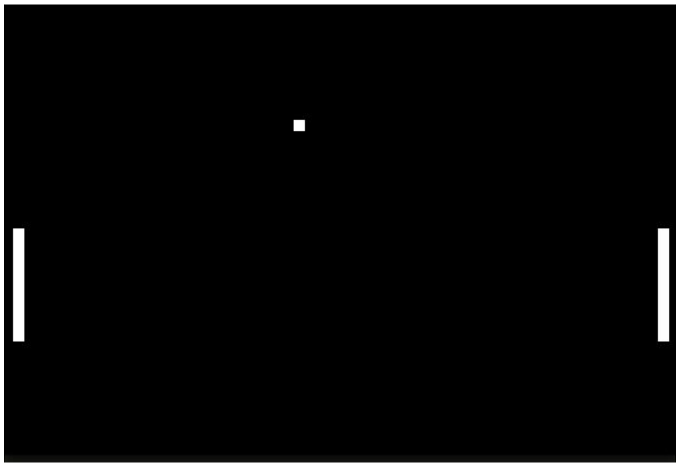
window.c

```
#include <stdio.h>
#include <stdlib.h>
#include "SDL2/SDL.h"
#include "window.h"

#define MS_PER_FRAME 8

const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;

SDL_Window *window = NULL;
SDL_Surface *screenSurface = NULL;

const Uint8 *kb_state = NULL;

gameboard_t *boards = NULL;
gameboard_t *current_board = NULL;

blr_pos_t click_pos;

entity_t *to_delete = NULL;

int should_restart = 0;

void restart() {
    should_restart = 1;
}

void reset_board() {
    entity_t *elist;
    for (elist = current_board->ents; elist != NULL; elist = elist->hh.next) {
        entity_t *tmp, *elt;
        LL_FOREACH_SAFE(elist->next, elt, tmp) {
            LL_DELETE(elist, elt);
        }
    }
}

void register_gb(gameboard_t *board) {
    if (HASH_COUNT(boards) == 0) {
        current_board = board;
        if (board->init_fn != NULL) {
            board->init_fn(board);
        }
    }

    gameboard_t *p;
    HASH_FIND_STR(boards, board->name, p);
    if (p == NULL) {
        HASH_ADD_STR(boards, name, board);
    }
}

int is_key_down(int scancode) {
    return (int)kb_state[scancode];
}

int initWindow()
{

       if ( SDL_Init ( SDL_INIT_VIDEO ) < 0 ){
              printf( "SDL could not initialize! SDL Error: %s\n", SDL_GetError() );
              return 0;
       } else {

              int board_w = (current_board->size).w;
        int board_h = (current_board->size).h;
        window = SDL_CreateWindow( "Hello World", SDL_WINDOWPOS_UNDEFINED,
SDL_WINDOWPOS_UNDEFINED, board_w, board_h, SDL_WINDOW_SHOWN );

        if ( window == NULL ) {
                     printf( "Window could not be created. SDL Error: %s\n", SDL_GetError()
);

                     return 0;
              } else {
                     screenSurface = SDL_GetWindowSurface( window );
                     if ( screenSurface == NULL ) {
                            printf( "Surface could not be created! SDL Error: %s\n",
SDL_GetError() );
                            return 0;
                     }
              }
       }

       return 1;
}
```

```
void closeWindow()
{
       SDL_DestroyWindow(window);
       window = NULL;
       SDL_Quit();
}

int inEndZone( SDL_Rect block, SDL_Rect endZone )
{

    int leftBlock = block.x;
    int rightBlock = block.x + block.w;
    int bottomBlock = block.y;
    int topBlock = block.y + block.h;

    int leftEZ = endZone.x;
    int rightEZ = endZone.x + endZone.w;
    int bottomEZ = endZone.y;
    int topEZ  = endZone.y + endZone.h;

    if ( bottomBlock > bottomEZ && topBlock < topEZ && leftBlock > leftEZ && rightBlock <
rightEZ ) {
       return 1;
    }

    return 0;
}

void draw_entity(entity_t *e) {
    SDL_Rect r;
    r.x = e->pos.x;
    r.y = e->pos.y;
    r.w = e->size.w;
    r.h = e->size.h;

    int red = e->color.r;
    int green = e->color.g;
    int blue = e->color.b;

    SDL_FillRect(screenSurface, &r, SDL_MapRGB(screenSurface->format, red, green, blue));
}

int run_loop()
{
    if ( !initWindow() ){
        printf( "Window initialization failed.");
        return 0;
    }

    int quit = 0;
    SDL_Event e;

    SDL_FillRect( screenSurface, NULL, SDL_MapRGB( screenSurface->format, 0xFF, 0xFF, 0xFF )
);
    SDL_UpdateWindowSurface( window );

    while( !quit ) {
        click_pos.x = -1;
        while ( SDL_PollEvent ( &e ) != 0 ) {
            if (e.type == SDL_MOUSEBUTTONDOWN) {
                SDL_MouseButtonEvent mouse_event = e.button;
                click_pos.x = mouse_event.x;
                click_pos.y = mouse_event.y;
            }
            if (e.type == SDL_QUIT) {
                quit = 1;
            }
        }

        blr_color_t c = current_board->color;
        Uint32 bg_color_sdl = SDL_MapRGB(screenSurface->format, c.r, c.g, c.b);
        SDL_FillRect(screenSurface, NULL, bg_color_sdl);
        kb_state = SDL_GetKeyboardState(NULL);

        entity_t *elist;
        for (elist = current_board->ents; elist != NULL; elist = elist->hh.next) {
            entity_t *tmp, *elt;
            LL_FOREACH_SAFE(elist->next, elt, tmp) {
                if (elt->frame_fn != NULL) {
                    elt->frame_fn(elt);
                }
                draw_entity(elt);
            }
        }

        if (should_restart) {
             reset_board();
             current_board->init_fn(current_board);
             should_restart = 0;
        }

        entity_t *tmp, *elt;
        LL_FOREACH_SAFE(to_delete, elt, tmp) {
            LL_DELETE(to_delete, elt);
            free(elt->members);
            free(elt);
        }

        SDL_UpdateWindowSurface( window );
        SDL_Delay(MS_PER_FRAME);
    }

    closeWindow();

        return 0;
}
```

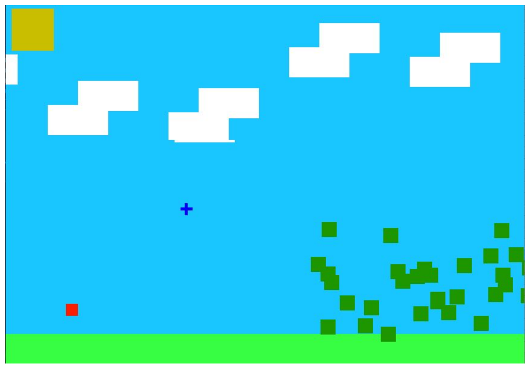
A.3 Examples

pong.blr



```
entity paddle1 {
    clr = (255, 255, 255);
    size = (10, 100);

    key_UP -> {
        self.pos[1] = self.pos[1] - 10;
    }

    key_DOWN -> {
        self.pos[1] = self.pos[1] + 10;
    }
}

entity paddle2 {
    clr = (255, 255, 255);
    size = (10, 100);

    key_W -> {
        self.pos[1] = self.pos[1] - 10;
    }

    key_S -> {
        self.pos[1] = self.pos[1] + 10;
    }

}

entity ball {
    clr = (255, 255, 255);
    size = (10, 10);
    int xspeed = 7;
    int yspeed = 7;

    self >< paddle1 -> {
        self.xspeed = -self.xspeed;
    }
    self >< paddle2 -> {
        self.xspeed = -self.xspeed;
    }
    frame -> {
        self.pos[0] = self.pos[0] + self.xspeed;
        if (self.pos[0] > 600 - self.size[1] || self.pos[0] < 0 ) {
            restart();
        }
        self.pos[1] = self.pos[1] + self.yspeed;
        if (self.pos[1] > 400 - self.size[1] || self.pos[1] < 0) {
            self.yspeed = -self.yspeed;
        }
    }
}

gameboard board {
    clr = (0, 0, 0);
    size = (600, 400);

    init -> {
        add(paddle2, (10, 200));
        add(paddle1, (580, 200));
        add(ball, (300, 300));
    }
}
```

Angrybird.blr



```
int rseed = 26854266;
vec birdPos = (100,500);
int maxPigs = 30;
vec cursorPos = (0,0);

func int rand(int min, int max) {
    rseed = rseed * 3627 + min;
    if (rseed < 0) {
        rseed = -rseed;
    }
    return (rseed % (max - min)) + min;
}


func int flicker(int val) {
       int newVal = val+5;
       if (newVal > 240) {newVal = 175;}
       return newVal;
}
```

```
entity bird {
       size = (20,20);
       clr = (255,0,0);
       bool fly = false;
       int yspeed = 0;
       int xspeed = 0;

       key_SPACE -> {
              self.fly = true;
              self.yspeed = -1*(cursorPos[1]-self.pos[1])/5;
              self.xspeed = (cursorPos[0] - self.pos[0])/5;
       }

       self >< ground -> {remove(); add(bird,birdPos);}

       frame -> {
              if (self.fly){
                     self.yspeed = self.yspeed - 5;
              self.pos = (self.pos[0]+self.xspeed, self.pos[1] - self.yspeed);

           if (self.pos[0]> 900) {remove();add(bird,birdPos);}
              }
       }
}

entity pig {
       size = (25,25);
       clr = (0,150,0);
       self >< bird -> {remove();}
}

entity cloud {
       size = (100,50);
       clr = (255,255,255);
       frame -> {
              self.pos[0] = self.pos[0] -10;
              if (self.pos[0] < -100) remove();
       }

}

entity sun {
       size = (70,70);
       clr = (200,255,0);
       frame -> {
              self.clr[1] = flicker(self.clr[1]);
       }
}

entity ground {
       size = (860,50);
       clr = (0,255,100);
}
```

```
entity graphicController {
       size = (0,0);
       clr = (0,0,0);
       int timer = 20;
       int cloud_y = 0;
       frame -> {
              self.timer = self.timer - 1;
              if (self.timer == 0) {
                     self.cloud_y = rand(0,200);
            add(cloud, (900,self.cloud_y));
            add(cloud,(900+50,self.cloud_y-40));
            self.timer = 20;
        }
       }
}


entity aim1 {
       size = (20,5);
       clr = (0,0,255);
       key_UP -> {self.pos[1]= self.pos[1]-10;}
       key_DOWN -> {self.pos[1]= self.pos[1]+10;}
       key_LEFT -> {self.pos[0]= self.pos[0]-10;}
       key_RIGHT -> {self.pos[0]= self.pos[0]+10;}
       frame -> {cursorPos = self.pos;}
}
entity aim2 {
       size = (5,20);
       clr = (0,0,255);
       key_UP -> {self.pos[1]= self.pos[1]-10;}
       key_DOWN -> {self.pos[1]= self.pos[1]+10;}
       key_LEFT -> {self.pos[0]= self.pos[0]-10;}
       key_RIGHT -> {self.pos[0]= self.pos[0]+10;}
}



gameboard g1{
       size = (860,600);
       clr = (0,200,255);
       init -> {
              int pigs = 0;
              add(graphicController,(-100,-100));
              add(ground,(0,550));
              add (sun,(10,10));
              add(bird,birdPos);
              add (aim1, birdPos);
              add (aim2,(birdPos[0]+7,birdPos[1]-7));
              while (pigs < maxPigs) {
                     add(pig,(rand(500,860),rand(350,550)));
                     pigs = pigs + 1;
              }

       }
}
```

A.4 Test Files

fail-assign.blr

```
gameboard g1 {
       clr = (251,142,74);
       size = (680,420);
       init -> {
              int x = 10;
              bool c = true;
              c = 43;
       }
}

Fatal error: exception Failure("illegal assignment bool = int in c = 43")
make: *** [fail-assign.test] Error 2
```

fail-assign2.blr

```
gameboard g1 {
       clr = (251,142,74);
       size = (680,420);
       init -> {
              int x = 10;
              float y = 3.44;
              x = y;
       }
}

Fatal error: exception Failure("illegal assignment int = float in x = y")
make: *** [fail-assign2.test] Error 2
```

fail-assign3.blr

```
gameboard g1 {
       clr = (251,142,74);
       size = (680,420);
       init -> {
              color my_clr = (2, 3, 4);
              vec my_vec = (5, 4);
              my_vec = my_clr;
       }

}

Fatal error: exception Failure("illegal assignment vec = color in my_vec = my_clr")
make: *** [fail-assign3.test] Error 2
```

fail-clrBool.blr

```
gameboard g1{
       clr = (251,142,true);
       size = (680,420);
       init -> {
        }
}

Fatal error: exception Failure("expected numeric input for type color")
make: *** [fail-clrBool.test] Error 2
```

fail-compareClrs.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {

                color my_clr2 = (1,2);
                color my_clr2 = (3,4);
                if ( my_clr1 <= my_clr2 ){}
        }
}

Fatal error: exception Failure("Duplicate init function member my_clr2 in g1")
make: *** [fail-compareClrs.test] Error 2
```

fail-compareTypes.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                if (6 != false){}
        }
}

Fatal error: exception Failure("illegal binary operator int != bool in 6 != false")
make: *** [fail-compareTypes.test] Error 2
```

fail-compareVecs.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {

                vec my_vec1 = (1,2);
                vec my_vec2 = (3,4);
                if ( my_vec1 > my_vec2 ){}
        }
}

Fatal error: exception Failure("illegal binary operator vec > vec in my_vec1 > my_vec2")
make: *** [fail-compareVecs.test] Error 2
```

fail-defAdd.blr

```
func int add(int x, int y){
        return 1;
}
```

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		add(5, 4)
	}
}

Fatal error: exception Failure("Duplicate function add")
make: *** [fail-defAdd.test] Error 2
```

fail-defPrint.blr

```
func int print(int x){
	/* can't define second print */
	return 1;
}

gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {

		print(5);

	}
}

Fatal error: exception Failure("Duplicate function print")
make: *** [fail-defPrint.test] Error 2
```

fail-defRemove.blr

```
func int remove(){
	return 1;
}

gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {

		remove();

	}
}

Fatal error: exception Failure("Duplicate function remove")
make: *** [fail-defRemove.test] Error 2
```

fail-defRestart.blr

```
func int restart(){
	return 1;
}

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {

                restart();

        }
}

Fatal error: exception Failure("Duplicate function restart")
make: *** [fail-defRestart.test] Error 2
```

fail-dupEnts.blr

```
entity e1{
        clr = (1, 2, 3);
        size = (1,2);
}

entity e1{
        clr = (4, 5, 6);
        size = (1,2);
}

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {}
}

Fatal error: exception Failure("Duplicate entity e1")
make: *** [fail-dupEnts.test] Error 2
```

fail-dupFormals.blr

```
func bool my_func(int x, bool y, int x){
        return y;
}

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                my_func(5, true, 10);
        }
}

Fatal error: exception Failure("Duplicate formal x in my_func")
make: *** [fail-dupFormals.test] Error 2
```

fail-dupFuncs.blr

```
func vec my_vec_func(){
        return (5, 10);
}

func vec my_vec_func(){
        return (10, 15);
}

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                vec my_vec = my_vec_func();
        }
}

Fatal error: exception Failure("Duplicate function my_vec_func")
make: *** [fail-dupFuncs.test] Error 2
```

fail-dupGlobals.blr

```
bool y = 2;
int x = 4;
bool y = 5;

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                print(x);
        }
}

Fatal error: exception Failure("Inconsistent types in global declaration y 2")
make: *** [fail-dupGlobals.test] Error 2
```

fail-dupLocals.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                int x = 2;
                int x = 4;

                print(x);
        }
}

Fatal error: exception Failure("Duplicate init function member x in g1")
make: *** [fail-dupLocals.test] Error 2
```

fail-entNoClr.blr

```
entity e1{
        clr = (1, 2, 3);
        size = (1,2);
}

entity e2{
        size = (1,2);
}
```

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {

	}
}

Fatal error: exception Failure("You haven't defined clr")
make: *** [fail-entNoClr.test] Error 2
```

fail-entNoSize.blr

```
entity e1{
	clr = (1, 2, 3);
}

gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {

	}
}

Fatal error: exception Failure("You haven't defined size")
make: *** [fail-entNoSize.test] Error 2
```

fail-exprTypes2.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		int a = 3;
		vec my_vec = (10, 5);
		my_vec - a;
	}
}

Fatal error: exception Failure("illegal binary operator vec - int in my_vec - a")
make: *** [fail-exprTypes2.test] Error 2
```

fail-exprTypes3.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		int a = 3;
		color my_clr = (10, 5, 15);
		my_clr * a;
	}
}

Fatal error: exception Failure("illegal binary operator color * int in my_clr * a")
```

```
make: *** [fail-exprTypes3.test] Error 2
```

fail-exprTypes4.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		color my_clr = (15, 20, 25);
		vec my_vec = (10, 5);
		my_vec + my_clr;
	}
}

Fatal error: exception Failure("illegal binary operator vec + color in my_vec + my_clr")
make: *** [fail-exprTypes4.test] Error 2
```

fail-exprTypes5.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		vec my_vec1 = (15, 20);
		vec my_vec2 = (10, 5);
		my_vec1 + my_vec2;
	}
}

Fatal error: exception Failure("illegal binary operator vec + vec in my_vec1 + my_vec2")
make: *** [fail-exprTypes5.test] Error 2
```

fail-exprTypes6.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		color my_clr1 = (15, 20, 25);
		color my_clr2 = (10, 5, 0);
		my_clr1 - my_clr2;
	}
}

Fatal error: exception Failure("illegal binary operator color - color in my_clr1 - my_clr2")
make: *** [fail-exprTypes6.test] Error 2
```

fail-exprTypes.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		int a = 3;
		bool x = true;
		x + a;
```

```
        }
}

Fatal error: exception Failure("illegal binary operator bool + int in x + a")
make: *** [fail-exprTypes.test] Error 2
```

ail-floatMod.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                float x = 5.3;
                float y = 3.2;
                x % y;
        }
}

Fatal error: exception Failure("illegal binary operator float % float in x % y")
make: *** [fail-floatMod.test] Error 2
```

fail-ifPredicate.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                if (5){}
        }
}

Fatal error: exception Failure("expected Boolean expression in 5")
make: *** [fail-ifPredicate.test] Error 2
```

fail-invalidCollision.blr

```
entity e {
        clr = (2,3,4);
        size = (1,2);
    self >< bad_ent -> {}
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {
        add(e, (5, 10));
        add(e2, (20, 30));
    }
}

Fatal error: exception Failure("bad_ent is not a defined entity")
make: *** [fail-invalidCollision.test] Error 2
```

fail-noGbClr.blr

```
gameboard g1{
        size = (200,100);
        init -> {
                if(5>4){
                        print(1);
                }
        }
}

Fatal error: exception Failure("You haven't defined clr")
make: *** [fail-noGbClr.test] Error 2
```

fail-noGbSize.blr

```
gameboard g1{
        clr = (251,142, 300);
        init -> {

        }
}

Fatal error: exception Failure("You haven't defined size")
make: *** [fail-noGbSize.test] Error 2
```

fail-noReturn.blr

```
func int sum(int x, int y)
{
  int result = x+y;
}

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                sum(5, 4);
        }
}

Fatal error: exception Failure("Empty function")
make: *** [fail-noReturn.test] Error 2
```

fail-notEnoughArgs.blr

```
func int my_func(bool x, int y){
        return 1;
}

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {

                my_func(true);

        }
}
```

```
Fatal error: exception Failure("Expecting 2 arguments in my_func(true)")
make: *** [fail-notEnoughArgs.test] Error 2
```

fail-returnType.blr

```
func int my_func(int x, int y)
{
  return true;
}


gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {
              my_func(5,4);
       }
}

Fatal error: exception Failure("Return gives bool expected int in true")
make: *** [fail-returnType.test] Error 2
```

fail-tooManyArgs.blr

```
func int my_func(int x, int y){
       return 1;
}

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              my_func(5, 4, 7);

       }
}

Fatal error: exception Failure("Expecting 2 arguments in my_func(5, 4, 7)")
make: *** [fail-tooManyArgs.test] Error 2
```

fail-undefFunc.blr

```
gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              foo(4, true);

       }
}

Fatal error: exception Failure("Unrecognized function foo")
make: *** [fail-undefFunc.test] Error 2
```

fail-undefined.blr

```
gameboard g1{
       size = (400, 500);
       clr = (251,142,74);
       init -> {
              int x = 5;
              x = y;
       }
}

Fatal error: exception Failure("Undeclared identifier y")
make: *** [fail-undefined.test] Error 2
```

fail-vecBool.blr

```
gameboard g1{
       clr = (251,142,300);
       size = (680,false);
       init -> {

       }
}

Fatal error: exception Failure("expected numeric input for type vector")
make: *** [fail-vecBool.test] Error 2
```

fail-whilePredicate.blr

```
gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {
              while (5.3) { print(2); }
       }
}

Fatal error: exception Failure("expected Boolean expression in 5.3")
make: *** [fail-whilePredicate.test] Error 2
```

fail-wrongArgTypes.blr

```
func int my_func(bool x, int y){
       return 1;
}

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              my_func(5, true);

       }
}

Fatal error: exception Failure("Incorrect actual argument type in my_func(5, true)")
make: *** [fail-wrongArgTypes.test] Error 2
```

test-add1.blr

```
entity e {
       clr = (2,3,4);
       size = (1,2);
}

entity e2 {
       clr = (2,3,4);
       size = (1,2);
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {
       add(e, (5, 10));
       add(e2, (20, 30));
    }
}
```

```
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-add2.blr

```
entity e {
       clr = (2,3,4);
       size = (1,2);
}

entity e2 {
       clr = (2,3,4);
       size = (1,2);
    int i = 0;

    frame -> {
        self.i = self.i + 1;
        if(self.i % 10 == 0){
            add(e, (10*self.i, 20*self.i));
        }
    }
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {
       add(e2, (20, 30));
    }
}
```

```
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-addIntToFloat.blr

```
gameboard g1 {
        clr = (251,142,74);
        size = (680,420);
        init -> {

                float y = 3.5 + 10;
                if ( y > 3.5 ){
                        print(10);
                }

        }
}
```

```
10
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-andOr.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {

         bool t = true;
         bool f = false;

         if (t && f) {
             print(50);
         }
         if (t && t){
             print(40);
         }
         if (f && f) {
             print(30);
         }
         if (t || f) {
             print(20);
         }
         if (t || t) {
             print(10);
         }
         if (f || f){
             print(0);
         }

        }
}
```

```
40
20
10
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-arith.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {
		int val1 = 1 + 2 * 3 - 1;
		int val2 = -1 + 10/2 + 5;
		print(val1);
		print(val2);
	}
}

6
9
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-clickEvent.blr

```
entity e {

	clr = (2,3,4);
	size = (1,2);
	click -> { clickpos[0]; }

}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {

    }
}

SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-clr.blr

```
gameboard gb {
	clr = (200, 300, 100);
	size = (400,600);
	init -> {
		color my_clr = (255,30,100);
		print(my_clr[0]);
		print(my_clr[1]);
		print(my_clr[2]);
	}
}

255
30
100
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-clrFunc.blr

```
func color my_clr_func(){

       int r = 20;
       int g = 250;
       int b = 100;
       return (r, g, b);
}

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              color my_clr = my_clr_func();
              print(my_clr[0]);
              print(my_clr[1]);
              print(my_clr[2]);

       }
}
```

```
20
250
100
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-compareFloats.blr

```
gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              float x = 1.2;
              float y = 3.5;

              if (x > y) {
                     print(0);
              }

              if (x < y) {
                     print(1);
              }

              if (x >= x) {
                     print(1);
              }

              if (y <= x){
                     print(0);
              }

              if (y != x) {
                     print(1);
```

```
                }

                if (y == y){
                        print(1);
                }

        }
}

1
1
1
1
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-entAccess.blr

```
entity e {
    clr = (255, 0, 255);
    size = (20, 20);
    int speed = 0;
    bool flag = false;
    color new_clr = (1,2,3);
    vec new_vec = (5,10);

    frame -> {
        self.speed = 5;
        if (self.flag){

        }
    }

}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {

    }
}

SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-entArrayAccess.blr

```
entity e {
    clr = (255, 0, 255);
    size = (20, 20);
    int speed = 0;
    bool flag = false;
    color new_clr = (1,2,3);
    vec new_vec = (5,10);
    frame -> {
        int x = self.pos[0];
        int y = self.clr[2];
        int z = self.size[1];
    }

}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {

    }
}

SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-entCollisionEvent.blr

```
entity e {
       clr = (2,3,4);
       size = (1,2);
       self >< e2 -> { 1; }
}

entity e2 {
       clr = (2,3,4);
       size = (1,2);
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {

    }
}

SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-entKeyEvents.blr

```
entity e {
       size = (5,20);
       clr = (0,0,255);
       key_UP -> {self.pos[1]= self.pos[1]-10;}
       key_DOWN -> {self.pos[1]= self.pos[1]+10;}
       key_LEFT -> {self.pos[0]= self.pos[0]-10;}
       key_RIGHT -> {self.pos[0]= self.pos[0]+10;}
       key_W -> {}
       key_A -> {}
       key_S -> {}
       key_D -> {}
       key_SPACE -> {}
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {

    }
}

SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-entLocals.blr

```
entity e {
    clr = (255, 0, 255);
    size = (20, 20);
    int speed = 0;
    bool flag = false;
    color new_clr = (1,2,3);
    vec new_vec = (5,10);
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {

    }
}

SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-eqNeq.blr

```
gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

          if (4 == 5) {
               print(1);
          }

          if (4 != 5) {
               print(2);
          }

          if (4 == 4) {
               print(3);
          }

          if (4 != 4) {
               print(4);
          }
       }
}

2
3
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-fibFunc.blr

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

gameboard g1 {
       clr = (251,142,74);
       size = (680,420);
       init -> {
              print(fib(0));
              print(fib(1));
              print(fib(2));
              print(fib(3));
       print(fib(4));
       print(fib(5));
       }
}

1
1
2
3
5
8
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-floatOps.blr

```
gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              float x = 1.2;
              float y = 3.5;

              x + y;
              y - x;
              x * y;
              y / x;

       }
}

SDL could not initialize! SDL Error: No available video device
```

```
Window initialization failed.
```

test-gcdFunc.blr

```
func int gcd(int a, int b) {
  while (a != b) {
    if (a > b) a = a - b;
    else b = b - a;
  }
  return a;
}

gameboard g1 {
       clr = (251,142,74);
       size = (680,420);
       init -> {
                print(gcd(2,14));
                print(gcd(3,15));
                print(gcd(99,121));
       }
}
```

```
2
3
11
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-geqLeq.blr

```
gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

          if (4 <= 5) {
               print(1);
          }

          if (4 <= 4) {
               print(2);
          }

          if (4 <= 3) {
               print(3);
          }

          if (5 >= 4) {
               print(4);
          }

          if (5 >= 6) {
               print(5);
          }

          if (5 >= 5){
               print(6);
          }

       }
}

1
2
4
6
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-global1.blr

```
bool x = true;
int y = 10;
float z = 5.5;
color my_clr = (300, 100, 200);
vec my_vec = (40, 80);

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {
              if(x){
                     print(10);
              }
              print (y);
              print(my_clr[2]);
              print(my_vec[0]);
       }
}

10
10
200
40
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-global2.blr

```
int x = 10;
bool flag = false;

func int my_gbl_func() {

       int toReturn = x;

       if (flag) {
              toReturn = 0;
       }

       return toReturn;
}

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {
               print(my_gbl_func());
               flag = true;
               print(my_gbl_func());
       }
}
```

```
10
0
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-global3.blr

```
int x = 10;

func int my_overwrite_func(int x){

        return x;
}

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                print(my_overwrite_func(20));
        }
}
```

```
20
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-global4.blr

```
int x = 10;

gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                int x = 5;
                print(x);
        }
}
```

```
5
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-if.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {
                if (5 < 4) {
                        print(5);
                }
                if (4 < 5) {
                        print(4);
                }
        }
}

4
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-ifElse.blr

```
gameboard g1 {
        clr = (251,142,74);
        size = (680,420);
        init -> {
                if (!true) {
                        print(5);
                } else {
                        print(10);
                }
        }
}

10
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-operations.blr

```
gameboard g1{
        clr = (251,142,74);
        size = (680,420);
        init -> {

                print(1 + 2);
                print(1 - 2);
                print(1 * 2);
                print(100 / 2);
                print(50 % 7);
        }
}

3
-1
2
50
1
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-printFunc.blr

```
func bool my_print(int a, int b, bool c){
       print(a);
       print(b);
       return c;
}

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              bool d = my_print(5, 10, true);
              if (d) {
                     print(15);
              }

       }
}
```

```
5
10
15
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-remove.blr

```
entity e {
       clr = (2,3,4);
       size = (1,2);
}

entity e2 {
       clr = (2,3,4);
       size = (1,2);
    int i = 0;
    bool removed = false;

    frame -> {
        self.i = self.i + 1;
        if(self.i % 20 == 0 && !self.removed){
            remove();
            self.removed = true;
        }
    }
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {
       add(e2, (20, 30));
        add(e, (10, 20));
    }
}
```

```
SDL could not initialize! SDL Error: No available video device
```

```
Window initialization failed.
```

test-restart.blr

```
entity e {
    clr = (2,3,4);
    size = (1,2);
}

entity e2 {
    clr = (2,3,4);
    size = (1,2);
    int i = 0;

    frame -> {
        self.i = self.i + 1;
        if(self.i == 50){
            restart();
        }
    }
}

gameboard g1 {
    clr = (5,10,15);
    size = (4,5);
    init -> {
        add(e2, (20, 30));
    }
}
```

```
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-simpleFunc.blr

```
func int sum(int x, int y) {
  return x + y;
}

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {
              int result = sum(5,4);
              print(result);
       }
}
```

```
9
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-vars.blr

```
gameboard gb {
       clr = (200, 300, 100);
       size = (400,600);
       init -> {
              int a = 5;
          bool b = false;
          float f = 3.2;
          print(a);
    }
}

5
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-vec.blr

```
gameboard gb {
       clr = (200, 300, 100);
       size = (400,600);
       init -> {
              vec my_vec = (50,100);
              print(my_vec[0]);
              print(my_vec[1]);
       }
}

50
100
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-vecFunc.blr

```
func vec my_vec_func(){

       int x = 20;
       int y = 250;
       return (x, y);
}

gameboard g1{
       clr = (251,142,74);
       size = (680,420);
       init -> {

              vec my_vec = my_vec_func();
              print(my_vec[0]);
              print(my_vec[1]);

       }
}

20
250
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

test-while.blr

```
gameboard g1{
	clr = (251,142,74);
	size = (680,420);
	init -> {

		int i = 5;
		while (i > 0) {
		print(i);
		i = i - 1;
		}
		print(42);
  }
}

5
4
3
2
1
42
SDL could not initialize! SDL Error: No available video device
Window initialization failed.
```

Appendix B: Commit Log

```
commit fb71132a27a980db3ded96d450a7154b2720adcf
Merge: e799044 137494f
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed May 10 17:39:39 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 137494feb36a9fade693b0ab76515e6add65877c
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Tue May 9 23:20:57 2017 -0400

    Remove old helloworld folder

commit 0cadf76cbf615bb3cccac2ca8912272275f8ce06
Merge: 3798fa8 9edac2c
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Tue May 9 23:19:32 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 3798fa85dc064ae61f3d2f7fd9dfa4f580b1facb
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Tue May 9 23:19:31 2017 -0400
```

```
    Rename codegen_simple -> codegen, fix warnings, remove unneeded comments

commit 9edac2c332e636305b8639f45a8eb5c357bef5b9
Merge: c79bb56 c0f11ee
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 21:26:50 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit c79bb56faef411fab549038a4b24e6fe59999a19
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 21:26:48 2017 -0400

    added clickpos to semant and tests

commit c0f11ee5c8210e3d9a62c82489ad8e1eecc55d93
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Tue May 9 21:16:42 2017 -0400

    Remove binary files

commit 139d9d447d2f320df9f81c2d1f7bfe4abe67bddd
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Tue May 9 21:15:39 2017 -0400

    Remove deprecated Testing directory

commit 3271f2a003c6c21232fb4855a2180912ef3eb38e
Merge: 39d1326 cefc64d
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Tue May 9 21:13:51 2017 -0400

    Merge

commit 39d1326c68f5960820b2badf9e449c1794d10986
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Tue May 9 21:12:48 2017 -0400

    Remove unneeded files

commit cefc64dd7d916dfcaf10328712ec0e77c1751feb
Merge: 2ccd5d4 3b8926b
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 21:09:23 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 2ccd5d4144739a1dfdb0cb04188289ab46e15078
```

```
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 21:09:21 2017 -0400

    adding more tests

commit 3b8926ba13b7dea483a2acd85fbf22bbb81b7a3b
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Tue May 9 20:58:34 2017 -0400

    fixed angrybirds

commit cc47995bc08df33a140ad1ef52c9cd4c8cc25fb1
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 19:01:36 2017 -0400

    added semant check to avoid duplicate funcs and fixed all associated test
files

commit 39fa28e59402452ca3099052b19ecb5617e491d3
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 18:38:03 2017 -0400

    fixed call semant check bug FINALLY

commit d3bf249cfc01a083cf026227ef9b12fc9f18cfae
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 15:17:59 2017 -0400

    cleaning up codegen & semant, removing unnecessary comments

commit b8fcf431d9642f5caaee22fd463591a73d5b9fa1
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 14:35:06 2017 -0400

    more testing

commit 0ea8a60682374ae789f6939eed33574f79829f0c
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 01:36:13 2017 -0400

    cleaning up test files:

commit 0b13059310cfcc7d28261a4baedb501eaf2041fa
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue May 9 01:34:55 2017 -0400

    more tests
```

```
commit f82c0a2f43f018843c7e3bdaf28f9f3cbaf068d9
Author: jvandebon <jvandebon@gmail.com>
Date:   Mon May 8 23:58:39 2017 -0400

    more fail tests

commit 6630355e1ae21cabfb09e4534d1ea031e7aee963
Author: jvandebon <jvandebon@gmail.com>
Date:   Mon May 8 22:11:04 2017 -0400

    adding fail tests

commit fec2f40bbeccc743d0d0eaf4bc962bccbc3a8f61
Author: jvandebon <jvandebon@gmail.com>
Date:   Mon May 8 20:41:53 2017 -0400

    more testing

commit 17726c418a471a75a5846dafb9a2791200868c6b
Author: jvandebon <jvandebon@gmail.com>
Date:   Mon May 8 20:36:58 2017 -0400

    moved files around, whoops!

commit 086533f01721884596b766b362f2999d7ccc8af3
Author: jvandebon <jvandebon@gmail.com>
Date:   Mon May 8 20:32:46 2017 -0400

    added lots of 'good' tests, some issues with negativer numbers to be fixed

commit 0a4bfe8ca41d410750138b0c777c7306e37510cf
Author: jvandebon <jvandebon@gmail.com>
Date:   Mon May 8 18:29:55 2017 -0400

    fail tests

commit c0f266bf27c6b76854fd0e84341f7cb5278cc540
Author: jvandebon <jvandebon@gmail.com>
Date:   Mon May 8 18:23:28 2017 -0400

    working on test suite:

commit 63bf0a13280d62e5af4190e7e8a20846794815a9
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Mon May 8 12:24:04 2017 -0400

    Fix possible null pointer error when adding entitiy
```

```
commit b621cbd7ab633eba7418893eb566ef61648f4236
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sun May 7 23:18:53 2017 -0400

    Enable click events

commit b6b0cd806f41b47218bc6fbb4e0531464d8bf885
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sun May 7 23:11:27 2017 -0400

    Runtime support for onclick

commit ebf3b7ab5a94a9342350dbd6bf8f5cd88a335524
Merge: 23f0e03 b0ebb4e
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sun May 7 22:07:07 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 23f0e0309710149ab262518cd7e7322496e67279
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sun May 7 22:06:52 2017 -0400

    Only search in /usr/local/lib for runtime

commit b0ebb4ee97187ac389af45e2b3ead9b3a63a1ead
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sat May 6 18:13:02 2017 -0400

    gravity

commit ad25c22745a89e33be9e063fbc472d2e84b310bf
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sat May 6 18:02:36 2017 -0400

    angrybirds

commit ff1bcff3ac59e94a1cb488d31b8c425093e0e859
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sat May 6 16:43:23 2017 -0400

    float conversion stuff more semantic checks

commit 8d2db4bde4d1f56f77fb10271e4d33141232c9c8
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Fri May 5 20:00:05 2017 -0400

    Make pong game!
```

```
commit c751af9c440deb841cf5ee0a4f45e7cfd8b722b2
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Fri May 5 19:59:51 2017 -0400

    Add more keys (WASD, space) to keypress events

commit 1e6f7c4efd2c88a30e4d7368f7603130b48dd55e
Merge: 4413e32 fb22a52
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Fri May 5 19:53:14 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 4413e32d1200c66edb345c6835dfb95289cbd13d
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Fri May 5 19:50:10 2017 -0400

    Add install target to runtime makefile

commit 569f4cec5e490fe4200d6e0bc2231d9d50d8bd5a
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Fri May 5 19:44:39 2017 -0400

    Remove printing from flappy.blr

commit d2751e5c32b635a271dc62d489e7a92d27728b3c
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Fri May 5 19:40:39 2017 -0400

    Update ballr.ml so that running ballr also compiles using clang, instead of
just outputting LLVM IR

commit fb22a52efdc75c9dcc28beb6e2c9f203f7e9389e
Merge: 96cf5eb 09bebf7
Author: roch09 <rsj2115@columbia.edu>
Date:   Wed May 3 14:09:20 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 96cf5eb4624581e4997aa634170b5898d6e97438
Author: roch09 <rsj2115@columbia.edu>
Date:   Wed May 3 14:08:47 2017 -0400

    test outputs

commit 09bebf778ee1777128e4a77559933c005bf615c9
Author: Noah Zweben <noahzweben@gmail.com>
```

```
Date:   Wed May 3 13:19:56 2017 -0400

    automatic int for vec and clr

commit 814cfceaf7dd8b4c5dfc3056303ba8da324ec7bc
Author: roch09 <rsj2115@columbia.edu>
Date:   Mon May 1 21:26:34 2017 -0400

    added

commit 5ada43721cff55df5c3160484b006addd7b0a823
Merge: 6195053 eadfbe4
Author: roch09 <rsj2115@columbia.edu>
Date:   Mon May 1 21:23:33 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 6195053662349975dbd0b9bda0d72a88730a5f78
Author: roch09 <rsj2115@columbia.edu>
Date:   Mon May 1 21:19:58 2017 -0400

    added more

commit eadfbe48da81054c60392f31a77e60f08ae4784f
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sat Apr 29 18:36:02 2017 -0400

    Flappy...

commit 63e850560362becc9cd0e7edb27c498dccbf05e7
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sat Apr 29 18:35:48 2017 -0400

    Add restart() to semant

commit ee9e5c8dc69185ac8f20586e7a789cdbc7b555ec
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sat Apr 29 18:35:31 2017 -0400

    Update runtime for collisions, error-free removal

commit 6ea4de2bda12831da7c3610c0b68726356bdf9b0
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sat Apr 29 18:34:25 2017 -0400

    Add tests for collisions

commit 7ed1b5be1aa2f6c039764af85eb0a5ca2aabc793
```

```
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sat Apr 29 18:33:44 2017 -0400

    Finish implementing codegen for collisions

commit fafbb4bfd57cfb6753e09086555140f650ae14e3
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 13:32:59 2017 -0400

    added array access for ID (not just access), so my_clr[1] = 5 etc works now

commit 0bfbbd38b6a357d16010ffc8c67d9e12214aa05f
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 12:58:24 2017 -0400

    keypress working for up down left right A

commit 405553f3557317d433612860aa209868ff942922
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 11:57:29 2017 -0400

    things

commit 9d952ef39ddfd609bc6de3b0a72a6f7d1f0a8716
Merge: d577e44 57c1565
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 11:54:44 2017 -0400

    fixing merge conflicts

commit d577e4494ec5afa5c0563675a66732c8adad6c27
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 11:53:33 2017 -0400

    working on frame

commit 57c15659e5fd6b6f349b2dee70cd558e17610322
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Apr 28 11:12:00 2017 -0400

    remove works

commit ce45ff3544242eab84c34417f90d56ee43ab0263
Merge: b811fe4 24f3daa
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 10:31:46 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr
```

```
commit 24f3daae36d9d66d5318a3e449c34a5f2f1bd004
Merge: ad82344 91ddd2e
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Apr 28 10:30:51 2017 -0400

    dd
    XMerge branch 'master' of https://github.com/noahzweben/Ballr

commit ad8234489e73f4860af869a807e5f6a0151a573b
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Apr 28 10:30:42 2017 -0400

    ready

commit b811fe44f36b4067e808b9e155b2ec937308de8b
Merge: b6e1eb2 91ddd2e
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 10:28:50 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit b6e1eb253f6a278295108bd351cdab9d84a52492
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Apr 28 10:28:22 2017 -0400

    new test game

commit 91ddd2e24bf9fd638b53af0f257160945a25af7c
Merge: 6c695e6 a2d60e1
Author: roch09 <rsj2115@columbia.edu>
Date:   Fri Apr 28 03:21:24 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 6c695e686d399d7b51c380fa9b087063da36d491
Author: roch09 <rsj2115@columbia.edu>
Date:   Fri Apr 28 03:21:00 2017 -0400

    tests and modified Makefile

commit a2d60e15072716c0cbfb540fff92ec41d1b79bdf
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Apr 23 23:05:30 2017 -0400

    test

commit 32edd5cceeda4baed6f3c4d35458f477017e5e9b
```

```
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Apr 23 13:04:40 2017 -0400

    parabola and some fun stuff

commit 5c530bce9997f5854069b62e3d39835265c739d8
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Apr 23 12:30:50 2017 -0400

    removed foreach

commit c6a08213878b5abe23c3352e6e1e35ac49abd746
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 19 22:32:56 2017 -0400

    access for clr/pos/size done - not thoroughly tested

commit c4efcd35ac0875cc5b98f16893c20db3966ed32b
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 19 21:56:01 2017 -0400

    access for clr/size/pos started

commit d94e001714e628754123ea5bcd261f7b1116a57d
Merge: 1261de7 737cad7
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 19 19:01:22 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 1261de790cd37696a101fe5b2fc81ac213034819
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 19 19:01:10 2017 -0400

    Begin implementing generation of event functions

commit 737cad702fbcaeb79ba94633c8de230596f83313
Author: roch09 <rsj2115@columbia.edu>
Date:   Wed Apr 19 18:28:33 2017 -0400

    test

commit 2e9f725bf020c691b533630f2102540429f28e36
Author: roch09 <rsj2115@columbia.edu>
Date:   Wed Apr 19 18:23:04 2017 -0400

    adding tests
```

```
commit 10af8ba4b3b6500211b620dcf856c27f4e819a1d
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Tue Apr 18 09:34:38 2017 -0400

    more semant

commit f0ba099a234d5932e83d68dd40e7651fdcbdecfc
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Tue Apr 18 08:52:35 2017 -0400

    check 4 dup entities

commit fe0de4271897b1ea368adf70644cb4ab3e70ddb5
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Apr 17 21:36:39 2017 -0400

    foreach check works

commit dea1af3fc551a6710d9506dba356990b1b139257
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Apr 17 20:10:52 2017 -0400

    fixed size/clr issue

commit fafdb383c795c8be5e119ff20c4f94baea8f5816
Author: jvandebon <jvandebon@gmail.com>
Date:   Sun Apr 16 17:31:26 2017 -0400

    fixing hello world to work with entities until we get frame function
working

commit 88efaeb8b5fcc2da28257bd49b544d567c00a811
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 15 20:14:59 2017 -0400

    mostly done with array access

commit 5ff76d23d5c5c110263d4dd004eae3a0c89ef72e
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 15 18:33:21 2017 -0400

    access working, some scopign issues

commit 43e6bb9b91aa2febe39b91765cddf1701c3144f4
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 15 18:19:57 2017 -0400

    starting to be able to access entity members
```

```
commit 6a8861abcb879ecfab4b6ed5f2c14afcc4de2955
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 15 14:28:39 2017 -0400

    setting member values in mem struct works

commit 39ef941d8d567b97ea3d45876942f85bd745c22c
Merge: 8073a03 c270c1d
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 15 14:03:30 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 8073a038ba1256fc2327fea99064099840e12c38
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 15 14:03:28 2017 -0400

    adding entity member struct types

commit c270c1d5af2a5b8e089140b43d8c4ad84cff7011
Author: roch09 <rsj2115@columbia.edu>
Date:   Fri Apr 14 19:02:29 2017 -0400

    added tests

commit faee866f5654b3a5772fb863c5e6d8a669052981
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Thu Apr 13 12:04:41 2017 -0400

    notes

commit b6a394be663ecfb27575ea46bed336635835a89e
Merge: b19bb20 50b0d10
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Thu Apr 13 12:00:44 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

    idk what is happening

commit b19bb2060627a0abd9dfc6bfd3ae4fda33b079c0
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Thu Apr 13 11:59:38 2017 -0400

    access works

commit 50b0d10fb7567e9cef5078bbc2d479af762ce89f
```

```
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Apr 13 11:20:29 2017 -0400

    adding notes from meeting to runtime/entity_notes.txt

commit d48279c917dafdfb2f91c0efe8f3b37a4c1252c8
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Apr 10 11:49:23 2017 -0400

    fixed hw

commit 2316643cd68051c1b86f5dc3c6f8eec3580d1dd8
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Apr 10 11:46:32 2017 -0400

    added Access and assign semant checks

commit 4d226f72941f5019f2c9b05fe360f3f1560e8360
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 14:33:39 2017 -0400

    added tests for failures in testall

commit dde234d5d575734c73dd632fba2f9911676da398
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 13:40:32 2017 -0400

    fixing scope

commit 6bc70e270ef1fbd440e8b9c94ba948f31795b279
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 12:52:45 2017 -0400

    added gb checking to semant

commit 0bb74465523c6968c7655655b3568f71e1083d1f
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 12:39:19 2017 -0400

    getting rid of errors except pattern matching for things to add

commit c4a90d5c758106698638c73c05c2cd2d4df9b2f6
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 12:29:33 2017 -0400

    cleaning up/commenting semant

commit b21d73af34ae6a6365cb0a9f25b2c9265bf08a09
```

```
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 12:15:38 2017 -0400

    semant checking for functions mostly done

commit af4f369f053ef439fbfefdc8451e80f6892d89d2
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 10:37:10 2017 -0400

    updated semant for global var declarations

commit ae6f1a134a07a09561593da8684f20a1d3ff26cd
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 8 10:22:29 2017 -0400

    fixing globals in codegen, can only have constant (or unop negative) global
declarations, vec and color work too

commit 7edb546f82665fb64bd2913daeb7c0abcd80f195
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Apr 6 16:49:56 2017 -0400

    fixed functions in codegen, now callable

commit 7448c287432bea41db11a95af23dc500114c31c8
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Apr 6 16:41:40 2017 -0400

    function declarations in codegen, calling not working yet

commit 1526781f209e86166634b66a590572d0f589425d
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Apr 6 15:43:23 2017 -0400

    added id to codegen

commit 29d9c39f21598fce36a7755025c7cb9dd2fdf3c4
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Apr 6 15:30:47 2017 -0400

    adding id expression to semant

commit 3e0a2fd9610ee29d3540205d3bb37aa5eb86fccb
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Apr 6 15:05:05 2017 -0400

    starting to add global var declarations
```

```
commit 0c16ac0d619cecf1fa9e9e34db9fd7fcb35bb86e
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 22:55:10 2017 -0400

    starting to add function checks to semant

commit 6f6a9b7a5bbf719153f2f82cefcad9b8a4055e55
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 22:11:05 2017 -0400

    commenting/cleaning up semant

commit 12f6fb89a231ec55ebe8227b33eac81c16fefe11
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 22:05:24 2017 -0400

    added checks for global vars in semant

commit 2dce51e9a6c3bbbd35ed1609fe5a6def533c77e0
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 20:40:37 2017 -0400

    Make pattern matching exhaustive

commit cb4a0714397c5e160b154b89e3f2591c995ec1aa
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 20:34:39 2017 -0400

    Add calls to print_map

commit f6c9ecd29bbb960c53a3974ae57161f844070ab1
Merge: 3d99ab9 a910428
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 20:33:27 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 3d99ab911448961576e74f47939b57ef8875a200
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 20:33:14 2017 -0400

    Test multiple entities

commit 7833f6ae009f0e4fb8080e49bcfb5e66f3a7bb3f
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 20:29:59 2017 -0400

    Add support for calling the add function
```

```
commit a910428fe296d8ff24f9dd7cb5d4152b0bb75444
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Apr 5 19:49:36 2017 -0400

    some good semantic shit

commit ab10761d5fbcfc899feb0ecafbe9bf05b036bd6b
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Apr 5 19:22:12 2017 -0400

    smxx

commit 6f3ec7a30595e066fc9c6f7bf11581809e9cc58c
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Apr 5 19:18:44 2017 -0400

    more semantxxxx

commit 6405f96405101772df76e88be7627b9f3adbf9e1
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 18:48:03 2017 -0400

    some semantic checking working

commit 294beebac2b70d6d3065113481300170cb14e893
Merge: b7be7e9 1f312eb
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 18:38:35 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit b7be7e900667fbbb8907df82eeb0016d397ce280
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 18:38:33 2017 -0400

    Make codegen handle colors that arent just constant

commit 0d27cee78a05be8cc3010e3c312d302a7ffbec4e
Merge: c5cd862 1f312eb
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 18:37:55 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 1f312ebc4566574fdad4acb07a01516fe5ed7bd9
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Apr 5 18:37:50 2017 -0400
```

```
    fixed cg

commit c5cd8621e9cb5c7f1dd8299bac11636d7841095a
Merge: b6fbd26 21cbb27
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 18:35:50 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit b6fbd26fbfab2d1bcfa6c4ea2214c0c152ddd32e
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 18:35:46 2017 -0400

    working

commit 21cbb2740e0604a73dbc9e8bc32aee162dfe76f9
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Apr 5 18:34:36 2017 -0400

    fixed AST

commit 274290089a7de2f034e9893af82fa654f01712dd
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Apr 5 18:25:39 2017 -0400

    commenting out semant so make works

commit 6781f4aae9d2ce5670366959a8b886a0af76a4ca
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Apr 5 18:24:10 2017 -0400

    started semant stuff

commit d800f174fa6b9c07336ae297ea142514ba6c932e
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 15:14:37 2017 -0400

    Restructure code so that stmt and expr are global, and use them to generate
color/size initializations

commit 70f3a69474ed7214edf4f7467cf6b46088218a9b
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Wed Apr 5 12:51:24 2017 -0400

    Generate create function for entities, stub for frame function

commit 25863e2462d8e2a885d75e66f9dfaa3af5e23181
```

```
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 1 19:19:41 2017 -0400

    basic testall working, added if/while statements and some expressons to
codegen

commit 1c5c1ff695170f5e93b0d86b878fdefc37424a68
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Apr 1 17:05:09 2017 -0400

    added print to codegen and helloworld

commit ef0c9caa26101d0ff35336577c982eb223617021
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sat Apr 1 14:48:45 2017 -0400

    fixed grammer, added foreach etc

commit 1b49c02a69f82260bb5f1373dcaa803057520d68
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 26 16:40:19 2017 -0400

    a nice orange color

commit f42e267c06c23a288673f767f01085c9c2f1df05
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 26 13:57:06 2017 -0400

    semantic checker

commit 57bbfd39ac0c1509ba61d1a82b51666a56dc0146
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 26 13:54:10 2017 -0400

    fixed runtime to update size from gameboard

commit 319c8568c3f3fcbc04a1107c63a139bb8cb928dc
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 26 13:42:06 2017 -0400

    x forwarding note

commit 0814dcdc5c04c63846307240e3bf66b80cc8118e
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sat Mar 25 19:25:52 2017 -0400

    Add remove() funciton in runtime and implement chk_keypress()
```

```
commit 2ed5fc38a476134b8e049a495c28be41720421ff
Author: Frederick Kellison-Linn <fjk2119@columbia.edu>
Date:   Sat Mar 25 18:30:34 2017 -0400

    Generate init_fn stub and store it in the gamboard_t structure

commit 8181a267b253d962f51f129fd1e592521497827b
Merge: ab307e2 907db85
Author: Frederick <fjk2119@columbia.edu>
Date:   Sat Mar 25 14:04:52 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit ab307e2c21a551a60d0bac1112e060c0df7c70de
Author: Frederick <fjk2119@columbia.edu>
Date:   Sat Mar 25 14:04:40 2017 -0400

    Update the build system to link with a static runtime library

commit 907db85cd71eb75756161bf3feb932bce760e100
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Mar 25 11:24:37 2017 -0400

    added event polling to allow for X-ing out of the window

commit 1bedec5fc61e1f454ec796558da8538b426ec691
Author: Frederick <fjk2119@columbia.edu>
Date:   Fri Mar 24 23:13:39 2017 -0400

    Update codegen to fully encompass gameboard and entity C types, and get rid
of most warnings

commit 29701778edadcb875ab31d598a20926496a49734
Author: Frederick <fjk2119@columbia.edu>
Date:   Fri Mar 24 23:03:58 2017 -0400

    Updated gitignore

commit 4acb0c9ac250a506a4be4001ee3a34738d8854de
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Mar 24 18:48:57 2017 -0400

    hello world working!

commit 86e1c9a8b8df9bd68e72475cad14c9c731b2ccd0
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Mar 24 18:19:09 2017 -0400
```

```
    codegen maybe mostly done for hello world

commit 8927012be80edd4311bc396c1b7f85229b9b183a
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Mar 24 17:49:29 2017 -0400

    codegen mostly working for hello world

commit afa9fe4de46661aa8572beca27474fbb9798d49d
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Mar 24 16:25:09 2017 -0400

    multiple statements in function

commit 01504b400fbfbafcc42d4cf254d4992ff6f55c00
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Mar 24 16:09:27 2017 -0400

    working on codegen, add return statement, add add not working

commit 4a83041d17172266af69d2e15eea609d09d2e402
Merge: f042195 577d6b7
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Mar 24 15:11:50 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit f0421955d78f253c48f65fae18279f87425d89b5
Author: jvandebon <jvandebon@gmail.com>
Date:   Fri Mar 24 15:11:48 2017 -0400

    adding to codegen

commit 577d6b7634d7c8edcd77d38052412de50868b0d3
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Mar 24 15:11:40 2017 -0400

     modified Ballr

commit 7ee21a60e759febd16ab39fd99858162b1ff33b4
Merge: c55b451 75beac4
Author: roch09 <rsj2115@columbia.edu>
Date:   Fri Mar 24 15:01:54 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit c55b451f0ee01240e288954ff864db340de8aaa5
Author: roch09 <rsj2115@columbia.edu>
```

```
Date:   Fri Mar 24 15:01:44 2017 -0400

    code gen simple update

commit 75beac49b42b9316f92feba76e6d61d2e3ff2331
Author: Frederick <fjk2119@columbia.edu>
Date:   Fri Mar 24 14:40:19 2017 -0400

    Remove gb_init from ballr.(h/c)

commit 0c653b92f13a5af720cb49cc7b933240c57f2a97
Author: Frederick <fjk2119@columbia.edu>
Date:   Fri Mar 24 14:34:53 2017 -0400

    Removed unnecessary files

commit 902864a3a9da5d9ce2037b0452d1290d35048788
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Mar 24 14:24:49 2017 -0400

    simple cg

commit 4126537cda786dfc5a1e37a51bf76a0cc439e04a
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Mar 24 14:19:29 2017 -0400

    reverted ast

commit 94eb81788d7c03895768b6f1594475b5cd131248
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Mar 24 14:10:32 2017 -0400

    hellowrld.blr

commit e799044712c81cddc700c9521706b64bb29358f1
Merge: a5f8046 619e179
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Mar 23 21:05:47 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit a5f8046eaa8f05a1e0f7a424f1df68f3a35c3984
Author: jvandebon <jvandebon@gmail.com>
Date:   Thu Mar 23 21:05:46 2017 -0400

    cleaning up to pull

commit 619e179732ce8dbf58940b607a2b55f3603fa47d
```

```
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 17:03:08 2017 -0400

    Changed some 'string's in ast.ml to 'expr'

commit 3fc186ed007be6ec5af0efadbbcdbc26238d5718
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 14:56:04 2017 -0400

    Rework gamec.c to make codegen easier

commit 439ff7c0f1680ecc83b1b4c67e1489fab0cc53f5
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 14:43:03 2017 -0400

    Small roadmap for codegen, update types

commit 26e91bc4c4f74c6b6820691af578106fd5aa3625
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 12:47:18 2017 -0400

    Add uthash as regular files

commit f32c9e5cf1e01df1a2ee66c9d1a1ab521e05c0c3
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 12:46:58 2017 -0400

    Remove uthash as submodule

commit 59f7538680e49788b6148cfb94a27be37ecafde1
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 12:45:55 2017 -0400

    Bring in gamec.c changes for new gb_init func signature

commit bcf11f684702dc5f29ec93aab9ad035af7787c73
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 12:41:18 2017 -0400

    Change function signature to pass pointers in runtime lib

commit d82688ef077154d5e0e77c691ccaaaf891c46048
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 11:54:31 2017 -0400

    Remove dead code from window.c

commit 19b763bf41a5121fe27cf6b22c199713a58a6ce0
```

```
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 11:52:31 2017 -0400

    Use clang for compiling

commit caf257976ec2a1cdb931047d43ba1f58405d799e
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 11:51:36 2017 -0400

    Use -framework SDL2 in makefile to compile on multiple machines

commit 06b65c52b07fa3ab69f73d61d068d96e3ea31d94
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Thu Mar 23 11:50:58 2017 -0400

    Make SDL path more general

commit 25d290822e2c90497400328a94d2995a7fe25e58
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 23:23:21 2017 -0400

    setting up SDL and linking evrything in VM

commit 387470146afbc71161d7e261fd73993cb0073512
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 22:48:08 2017 -0400

    fixing include, whoops

commit b33fc420c164d40c5a5ebb035c40944e257306a4
Merge: 2422f72 fdd02c8
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 18:09:38 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit fdd02c83277b1dea6d0d7b6117c4b7a248fde459
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Mar 22 18:08:41 2017 -0400

    codegen stuff

commit 2422f727a9318c903ae7970cef0dd47d154b6e37
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 18:08:39 2017 -0400

    super simplified hello world
```

```
commit 6b36b11926234ecf1694bd1ebf9da89afd61380e
Merge: 6745dbc 0380b0c
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 17:11:42 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 6745dbc0060f5cb0a1f2669769da75bfedef1cb0
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 17:11:40 2017 -0400

    dumb make file to get things to work

commit 0380b0c3d391167856b9ff2cebca8b115edb9210
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Mar 22 17:11:01 2017 -0400

    gitignore

commit 3f03c85b21906f9549e14a94c74b79d1d3cae3f0
Merge: c0729d7 b0cc922
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 17:10:54 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit c0729d7b0a38ecf4f87df563146791dd65095fac
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 17:10:52 2017 -0400

    moved llvm around

commit b0cc9222080938fc25c3d7de74ca75dbf9f7045a
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Mar 22 17:09:09 2017 -0400

    cleanup

commit b1ac7d11664db8a78c658426e581733723c5ada8
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 17:08:52 2017 -0400

    renamed simplified llvm

commit 6d5ba9b5e5c9b2eae43b2d30a6c9cee23380d772
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Mar 22 17:08:02 2017 -0400
```

```
    testing folder

commit 88740dcec179227827fc8dd7193aa996a2994f57
Merge: d5f0684 9c8bc4e
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 16:37:19 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit d5f0684571dd9bd6416dea8df4b49ee4b2ffa985
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 16:37:15 2017 -0400

    fixing small errors

commit 9c8bc4e52bd973717ddfc7aa3452804b0552aafa
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Mar 22 16:31:54 2017 -0400

    start codegen

commit 0a9c659f4ea871eb7c3bd5ab5b3fce85a94af5df
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Wed Mar 22 16:23:35 2017 -0400

    Update README.md

commit e5658335cb7b3d1e0ef1795fc7e889e659185e93
Merge: cf9483c 355d7a4
Author: jvandebon <jvandebon@gmail.com>
Date:   Wed Mar 22 16:12:26 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit 355d7a46fa4fda1bc03d9ffa99924dd30e309ca0
Merge: 2273fc4 fc26f93
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Wed Mar 22 15:52:27 2017 -0400

    Merge branch 'master' of github.com:noahzweben/Ballr

commit 2273fc4eefc4776b3fdcdd5c3b73f2ac73e7a212
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Wed Mar 22 15:52:21 2017 -0400

    Started working on runtime system
```

```
commit cf9483cdbc7907f674f3cc5c1f982fb78ac61de6
Merge: fa0e9fd fc26f93
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue Mar 21 17:35:05 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit fc26f93235e4139babeca838151d1342c0705487
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Tue Mar 21 17:30:03 2017 -0400

    changed player

commit fa0e9fd526394b094705bc8ae86f4e187591ed66
Merge: 1f70780 cfe0d8e
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue Mar 21 16:22:24 2017 -0400

    Merge branch 'master' of https://github.com/noahzweben/Ballr

commit cfe0d8ee02d1c039b5887b9d35d41fa3fb507b3d
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Tue Mar 21 16:12:28 2017 -0400

    easy helloworld

commit 003b151f7cc6a56597579541d1f6d711eeebe3fc
Merge: f1a46d8 e1c48f2
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Tue Mar 21 16:12:00 2017 -0400

    Merge branch 'master' of github.com:noahzweben/Ballr

commit e1c48f2277651afe7f9d5a4e4eed42a6c35162fa
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Tue Mar 21 16:11:38 2017 -0400

    easy hellowordl

commit f1a46d84539ab3a6edb13581a21a60f75002978c
Merge: 8fd20a5 9d9f7b9
Author: Frederick Kellison-Linn <fred.kl@me.com>
Date:   Tue Mar 21 16:11:27 2017 -0400

    Merge branch 'master' of github.com:noahzweben/Ballr

commit 8fd20a5cea0bd99e79eab72a32801191acdbd53d
Author: Frederick Kellison-Linn <fred.kl@me.com>
```

```
Date:   Tue Mar 21 16:10:23 2017 -0400

    Fix main Makefile

commit 9d9f7b9e79674522c33959e938e9051114f08b0c
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Tue Mar 21 16:08:34 2017 -0400

    fixed makefile

commit 1f707801f3e0bc08f1d2c5b711059cb5f1e93786
Merge: 3837c58 d16d51a
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue Mar 21 12:08:15 2017 -0400

    fixing merge conflicts

commit 3837c581404593ace89fbcfb9e10411627f06657
Author: jvandebon <jvandebon@gmail.com>
Date:   Tue Mar 21 12:07:24 2017 -0400

    fixing things

commit d16d51adfa2a3e9eed17bfeb421783ee2ab475aa
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 20:04:50 2017 -0400

    changed

commit 667372e216d94adf7657c46221bb31914d83d274
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 19:01:34 2017 -0400

    finished, no errors

commit 0c464e2251bbe6110ea2674c2a083ffb626030fb
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 18:56:01 2017 -0400

    instructions

commit ee824c05786687c8a1b2a3ee791eb5a78bff85df
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 18:53:46 2017 -0400

    pretty print works

commit 307df398197b6a0fe1da29f8397fdc483c609e72
```

```
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 18:36:53 2017 -0400

    pretty print worksgit add .

commit e48ab4d7430c0fc936c1b6e912d60333f82929ed
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 18:30:06 2017 -0400

    fixed errors

commit 22ef2f448b403d83f692d093dff40f6a633db141
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 18:17:54 2017 -0400

    fixed errors

commit a26b92324ba1f88d675e63455d1979e78034950d
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 18:06:08 2017 -0400

    ocamlc

commit ec34282dba423414bfd1695e835b8de9a8e8c73e
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 17:21:23 2017 -0400

    updated ballr

commit 210cd26cc4188a2ae98ca0964ec0f8de8a829d58
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 17:15:40 2017 -0400

    trying to pprint

commit 07211b5fc8f419eaaa61947eee7f69bf25df1b5a
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 16:53:10 2017 -0400

    run script

commit fecfe155116ad2e587227cb2c2447ba2e6bdb52e
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 16:52:56 2017 -0400

    working on pretty print

commit ba87ded4304c41cb78f110a260629ccd94d321d0
```

```
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Mon Mar 20 15:09:40 2017 -0400

    fixed ID/member (hopefully)

commit 6d1a25f379ca7f74bdc02b22333fad8db229d53e
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 18:56:00 2017 -0400

    cleanup

commit 8189456b1dad909b11a0a1fd4b63bc67defeb0df
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 18:54:01 2017 -0400

    gitignore

commit 0b0903f80f8354dc7dde10edc1bf9f0112509e57
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 18:47:53 2017 -0400

    fixed tokens

commit 8bcea21e82e4c5e1e5a8c3ba24a483e1f79ef207
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 18:44:04 2017 -0400

    tested grammar with menhir

commit da155f8c4859710e3135adc5690a117917a06c68
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 18:28:31 2017 -0400

    creates tokens of program, can test in menhir

commit 37f1a5ad8c574bdd718f57db54065cd9e3f5eb00
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 18:02:32 2017 -0400

    token generator

commit ea5f59d892d3f5080901ac849c863ddc55667b25
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 17:24:31 2017 -0400

    organizing

commit b84e04063a9fc2a594ff3dab673828bcc66bcf49
```

```
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 17:17:17 2017 -0400

    helloworld ballr

commit 82a328ac1bfe3f5727d2101bebe18b4e39b706af
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 17:03:59 2017 -0400

    fixed token name

commit ff026e0cf50fd500150c1bcb25e3c126f3c8965d
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 16:41:35 2017 -0400

    fixed shift/reduce conflicts

commit d1b31f95ab37b3bb68513d5d25742c8f799d9e51
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sun Mar 19 15:50:53 2017 -0400

    fixed typo

commit a1f4f009256d859f0e169838851871b7d6976bc2
Author: jvandebon <jvandebon@gmail.com>
Date:   Sun Mar 19 13:33:42 2017 -0400

    adding endzone and collision detection

commit 0135710b877c88897f5169fb567b1b815ac0331f
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Mar 18 23:30:21 2017 -0400

    fixed readme

commit 31a93dc499781cd2602435e99a7f8e1e83d5129a
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Mar 18 23:28:30 2017 -0400

    hello world in C

commit c0c377edb02495ee9bf0aae2ea0df57fd7b5d164
Author: jvandebon <jvandebon@gmail.com>
Date:   Sat Mar 18 13:31:31 2017 -0400

    Adding things

commit 05da7b2859298f6c7ead45d23eb63f339f36dc02
```

```
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sat Mar 18 11:27:19 2017 -0400

    started parser

commit a2c503b1a0c6818e7152b6c0711b97bcc5e510e4
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sat Mar 18 11:10:53 2017 -0400

    ast and more scanner

commit 3183ace18d500541a95399ca674e3c02a48c89a2
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Sat Mar 18 10:01:45 2017 -0400

    started scanner

commit 0118995b9d00c41cbefba699c2607d42cb4cb57c
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Mar 17 18:10:52 2017 -0400

    added scanner

commit d7a6df82e9fa6a256ff35965ad86d7e7610856dd
Author: Noah Zweben <noahzweben@gmail.com>
Date:   Fri Mar 17 18:07:17 2017 -0400

    Create README.md
```