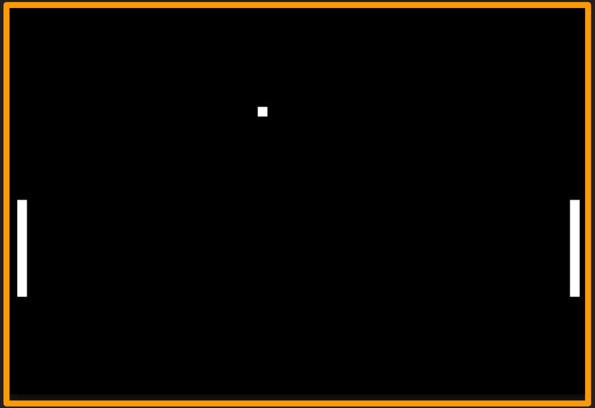
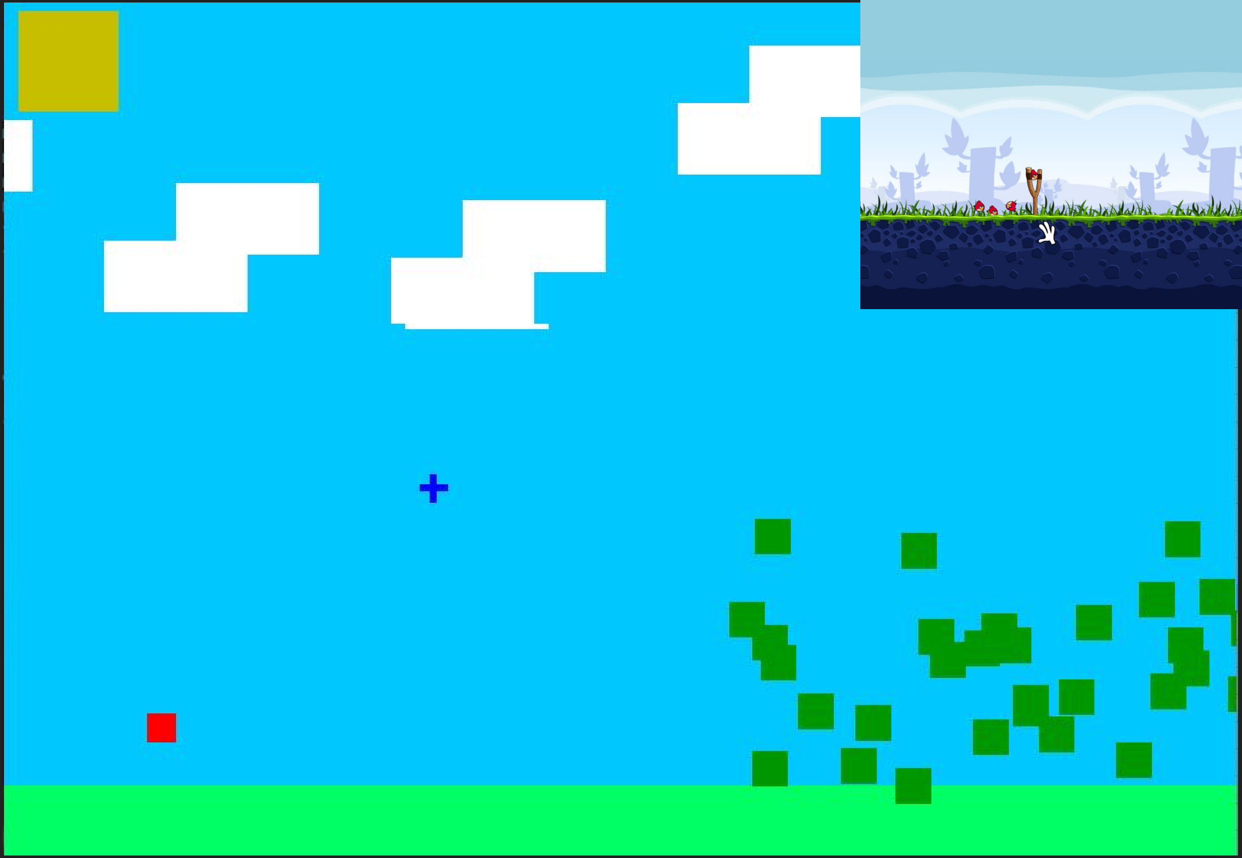
Ballr: A 2D Game Generator

Players Gonna Play



Freddy Kellison-Linn, Rochelle Jackson, Jessie Vandebon, Noah Zweben

A taste of what's to come...



Overview and Motivation

- Build simple 2D games with user-defined environment, rules and player control
- Moving or stationary rectangular “entities” throughout a bounded space
  - Displayed in a graphical user interface
- Each entity has size, color, and behavior
  - Behavior consists of associated events which define their response to user inputs
    - Keypresses
    - Clicks
    - Collisions
    - Per-frame
- Ballr abstracts away the event loop
- Programmer is free to focus on events that may occur
  - Allows the structure of the game to be clearer

Team Member Roles

Manager - Rochelle

Language Guru - Noah

System Architect - Freddy

Tester - Jessie

Crash Course in Ballr

Types

```
int x = 10;
float y = 6.27;
bool not_false = true;
color red = (255,0,0);
vec meaningOfLife = (42,42);
```

Events

```
key_LEFT -> {..do something ..}
init -> {..do something ..}
self >< other -> {..do
something..}
click -> {..do something ..}
frame -> {..do something ..}
```

Entities

```
entity player {
    clr = (0,0,255);
    size = (50,50);
    key_LEFT -> {
        self.pos[0]=self.pos[0]-10
;
    }
}
```

Gameboard

```
gameboard main {
    clr = (255,255,255);
    size = (500,500);
    init -> {
        vec start = (100,100);
        add(player,start);
    }
}
```

Crash Course in Ballr

Operators

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

Functions

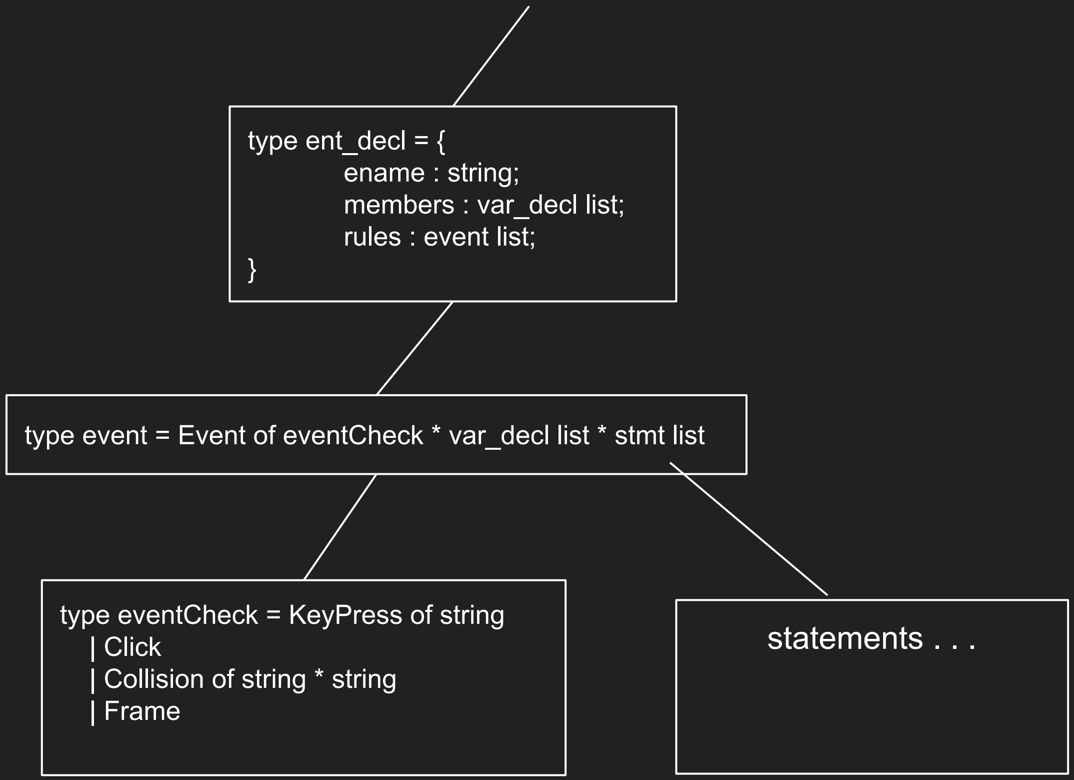
```
func vec foo(int x, int y){
    return (x,y);
}
add(entity,vec);
remove();
restart();
print(int)
```

Overall Program

```
int moveAmount = 10;

func vec moveLeft(vec p1, int amt){
      return (p1[0]-amt,p1[1]);
}
entity wall {
      clr = (0,0,0);
      size = (50,100);
}

entity player {
      clr = (0,0,255);
      size = (50,50);
      key_LEFT -> {
            self.pos = moveLeft(self.pos,moveAmount);
      }
      self >< wall -> {restart();}
}

gameboard main {
      clr = (255,255,255);
      size = (500,500);
      init -> {
            vec start = (100,100);
            add(player,start);
            add(wall,(200,100));
      }
}
```

AST For Entities + Game Logic

```
type program = var_decl list * func_decl list * ent_decl list * game_decl
```



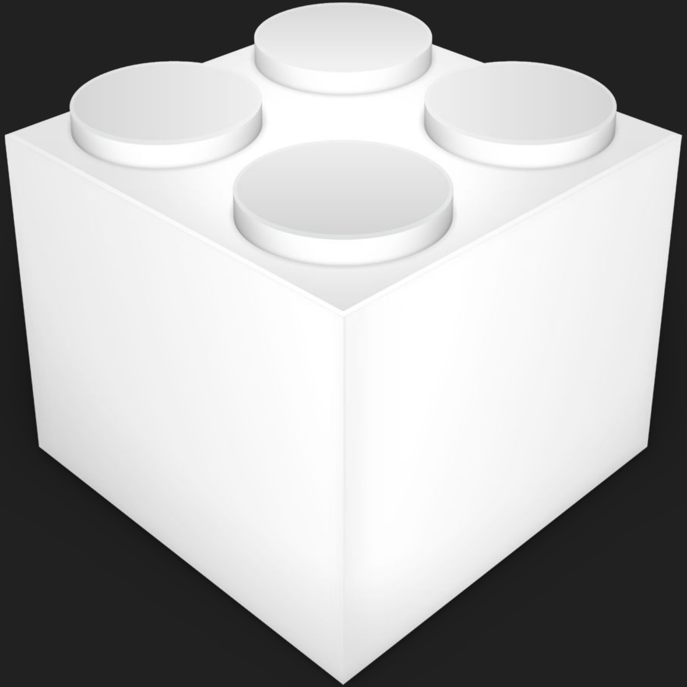
```
type ent_decl = {
        ename : string;
        members : var_decl list;
        rules : event list;
}
```

```
type event = Event of eventCheck * var_decl list * stmt list
```

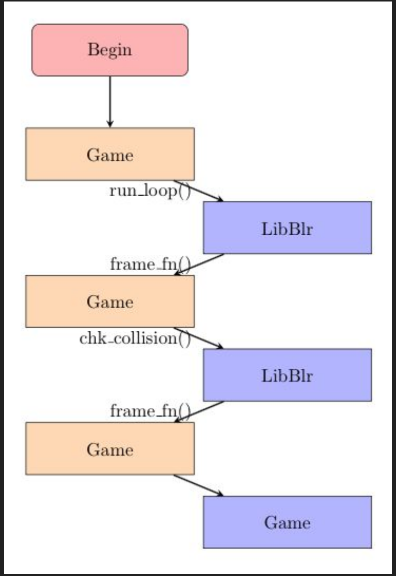
```
type eventCheck = KeyPress of string
    | Click
    | Collision of string * string
    | Frame
```

```
statements . . .
```

The Ballr Runtime



The Ballr Runtime



The Ballr Runtime

register_gb()

run_loop()

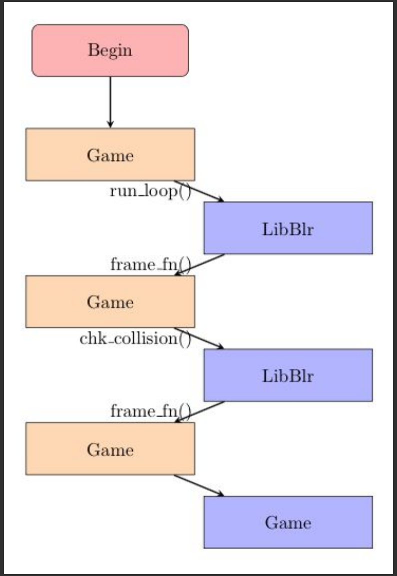
ent_add()

ent_remove()

chk_collision()

chk_click()

chk_keypress()



Testing Ballr

- Manual Testing
  - GUI Visual Verification
  - User Input Events
- Automated Test Suite
  - Declarations
  - Statements
  - Expressions
  - Functions
  - Semantic Correctness

Testing Ballr

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);
     }
}
```

test-arith.out

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

fail-entNoClr.blr

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

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

fail-entNoClr.err

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

DEMO