raft

2D Gaming Language
## The Team

<table>
<thead>
<tr>
<th>Member</th>
<th>Main Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Martin Fagerhus</td>
<td>Code Generation</td>
</tr>
<tr>
<td>Roy Prigat</td>
<td>Compiler Front End</td>
</tr>
<tr>
<td>Abhijeet Mehrotra</td>
<td>C – Implementation, SDL</td>
</tr>
<tr>
<td>Daniel Tal</td>
<td>Semantic Checking</td>
</tr>
</tbody>
</table>
Motivation and Goals

- Why is it so difficult to create games in languages such as C or Java?
  - Worry about game loop
  - Difficulty in defining binding events to specific elements.
  - Tedious
Why Us?

- We allow a user to build a 2D game with ease of just worrying about adding user defined elements, events, and a world.
- This is all done with much less code and makes it more straightforward for the programmer to develop a game.
- We include in-built language components to make game building easy to do/understand.
Program Structure

- Designed for ease of use and straightforward semantics.
- A world is the only required component.
- If an element is defined, it is required to add its properties as well (color, size).

<global variables>
<global functions>
<event definitions>
  <Event>
    <condition>
    <action>
<element definitions>
  <element>
    <properties> - required
<world definition> - required
  <world>
    <properties> - required
    <local variables>
    <statements block>
Types

- Color and size are properties of world and element components.
- Color is a string literal which corresponds to hex codes.
- Size is a pair type which defines the pixel size of an element/world.

```cpp
int a = 3;
bool b = true;
float f = 3.4;
string s = "hello";
pair p = (50, 70);
```
Events

- Define game “rules”
- Condition defines the expression that triggers the event when true.
- Action defines how the event reacts with the element it binds to.

```java
event move_down(player) {
    condition = key_press("DOWN");
    action {
        Player1.pos.x = 400;
    }
}
```
Elements

- Properties
  - Size of type Pair
  - Color (a hex string)
  - Direction (Integer) - optional
  - Speed (Integer) - optional

```java
element player {
  size = (50,50);
  color = "f44141";
  direction = 90;
  speed = 1;
}
```
World

- Properties – required
  - Size of type Pair
  - Color (a hex string)
- Adding new elements to the game environment
- Adding event to the event loop, where each event is bound to a selected element.

```plaintext
world {
    properties {
        size = (500,200);
        color = "42f4eb";
    }
    element player = new player(20,20);
    add_event(move_up);
}
```
Control Flow

| If/else        | If (true) {
|               |   x = 5;
|               | } else {
|               |   x = 3;
|               | }

| While loops   | int x = 0;
|               | while (x == 0) {
|               |   add_event(move_up);
|               |   x = x + 1;
|               | }

| For loops     | int x = 0;
|               | for (x, x <= 10, x++) {
|               |   add_event(move_down);
|               | }
Sample Program

- A game with one player which can move up on pressing “UP” arrow key

```python
int global_x = 300;

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

event move_up(player) {
    condition = key_press("UP");
    action {
        player.pos.x = player.pos.x + add(3,4);
    }
}

element player {
    size = (50,50);
    color = "ff0000";
}

world {
    properties {
        size = (500,500);
        color = "42f4eb";
    }
    while (x < 301) {
        element player = new player(x,x);
        add_event(move_up);
        x = x + 100;
    }
}
Runtime

- Based on SDL
- Infinite loop
- Has functions to:
  - Render elements
  - Help determine collisions
  - Trigger callback functions
Architecture
Testing

- An automated testing script runs over all test files and produces a testall.log.
- The log file includes the output of all tests.
- Fail tests output exceptions as defined in the semantics checker, these are printed out in the log file.
- Success tests simply produce an executable program which is later manually tested.

Automated Tests

- Declarations
- Statements
- Functions
- Expressions
- Semantics

Manual Tests

- Colors and size
- Elements
- Event actions
Demo