tiler.

A 2D turn-based gaming language

Tester: Jacky Cheung
Manager: Jason Lei
System Architects: Jiayin Tang, Evan Ziebart
Language Guru: Monica Ting
Why tiler?

- Intuitive structure for programming games
- Java-like syntax
- Simple interface for handling user input and graphics
Language Features

Grid
Object classes
Blocks
Turn-handling
End conditions
Input collection
Game Loop

- init
- turn
- end

- game over!
- true
- false
#size 500 250
#color 0 0 255
#title "Hello World"

int x; int y;
int add(...) {...}
class Piece {...}
init {
    tile(3, 3);
    background("hello.bmp");
}

turn {...}
end {...}
Syntax

Types

```c
int x; int y; int z; float f;
bool b; string s; coord c;
```

```c
x = 5; y = 5;
f = 4.5;
b = true;
s = "hello";
c = [x, y];
```

```c
z = c[x];  // coord access
```

Operators

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

Keywords

```c
gridh;  
gridw;  
init {...}  
turn {...}  
end {...}
```
Syntax - Functions

**Built-in Functions**

tile(3, 3);
background(“hello.bmp”);

`i`print(0); fprint(4.0);
sprint(“Hello World!”);
capture();

**Function Definition**

```c
int add(int x, int y)
{
    int z = 100;
    return x + y + z;
}
```

**Control Flow**

```c
if (condition) { ... } else { ... }
else if (condition) { ... }
while (condition) { ... }
do { actions } while (condition);
for (i = 0; i < end; i=i+1) { ... }
```
Syntax - Classes

Classes

class Piece {
    attr: string player;
}

class Obstacle {
    attr: int size;
}

Example Object Declaration

<br>

`<Piece> p;`  // declare p with class

`p = new Piece(“Edwards”);`  // create p with attr values

`setSprite(p, “edwards.bmp”);`  // set sprite for p

`grid[x, y] = p;`  // set location on grid for p
Syntax - Objects & Grid

Other Object & Grid Operations

grid[x, y] = p;  // grid assignment and moving object p to new location on grid
grid[x, y] = NULL;  // removes object on grid location [x, y]

p = grid[x, y];  // grid access - getting object at location [x, y] on grid

isNull(p);  // returns 1 if object is null
type(p);  // returns class of object

p.player;  // access explicit attributes of object
p.x; p.y;  // access object’s location by its implicit attributes, x and y
Language Architecture

Game

LLVM IR

Runtime Library

TILER

SDL

STD C
Compilation Pipeline

tiler-lib
	| tiler-lib.o
	| game.o

tokenized program
	| ast
	| checked ast
	| llvm

game.tile
	| scanner.mly
	| parser.mly
	| semant.ml
	| codegen.ml

game.exe
tiler-lib

- Uses SDL in C
- Displays the window
- Runs the “game loop”
- Manages the grid
- Renders background and objects
- Handles events
- Memory mgmt of class objects (AGC-ish)
Hello, World!

```
init {
  tile(3, 3);
  background("hello.bmp");
}

Hello World
```
Testing: Challenges

Automation:

- When a window is open, an infinite loop occurs until window is closed
- An close function was designed to avoid manual closing of test windows

Significance of Tests:

- Tests can only check program logic and operations, still need to check actual game behavior manually
<table>
<thead>
<tr>
<th>Test Name</th>
<th>Status</th>
<th>Test Name</th>
<th>Status</th>
<th>Test Name</th>
<th>Status</th>
<th>Test Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-arith1.tile...OK</td>
<td>OK</td>
<td>test-float-compare1.tile</td>
<td>OK</td>
<td>test-if1.tile...OK</td>
<td>OK</td>
<td>test-turn1.tile...OK</td>
<td>OK</td>
</tr>
<tr>
<td>test-arith2.tile...OK</td>
<td>OK</td>
<td>test-float-compare2.tile</td>
<td>OK</td>
<td>test-obj-access.tile...OK</td>
<td>OK</td>
<td>test-while1.tile...OK</td>
<td>OK</td>
</tr>
<tr>
<td>test-assign1.tile...OK</td>
<td>OK</td>
<td>test-func-rec.tile...OK</td>
<td>OK</td>
<td>test-obj-assign.tile...OK</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>test-dowhile1.tile...OK</td>
<td>OK</td>
<td>test-func1.tile...OK</td>
<td>OK</td>
<td>test-print-bool.tile...OK</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>test-end1.tile...OK</td>
<td>OK</td>
<td>test-global1.tile...OK</td>
<td>OK</td>
<td>test-print-expr.tile...OK</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>test-float-arith1.tile</td>
<td>OK</td>
<td>test-global2.tile...OK</td>
<td>OK</td>
<td>test-print-float.tile...OK</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>test-float-arith2.tile</td>
<td>OK</td>
<td>test-global3.tile...OK</td>
<td>OK</td>
<td>test-print-int.tile...OK</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>test-float-assign.tile</td>
<td>OK</td>
<td>test-helloworld.tile...OK</td>
<td>OK</td>
<td>test-print-string.tile...OK</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Demo
Future Work

Rules, Enhanced for-loops, Random function...
Lessons Learned?
...Start early.

“How” is more important than “what.” Get something working soon.

Learn to read the code... Learn to read the manuals. It really helps!
Time 2 nap !!!! :-)