

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

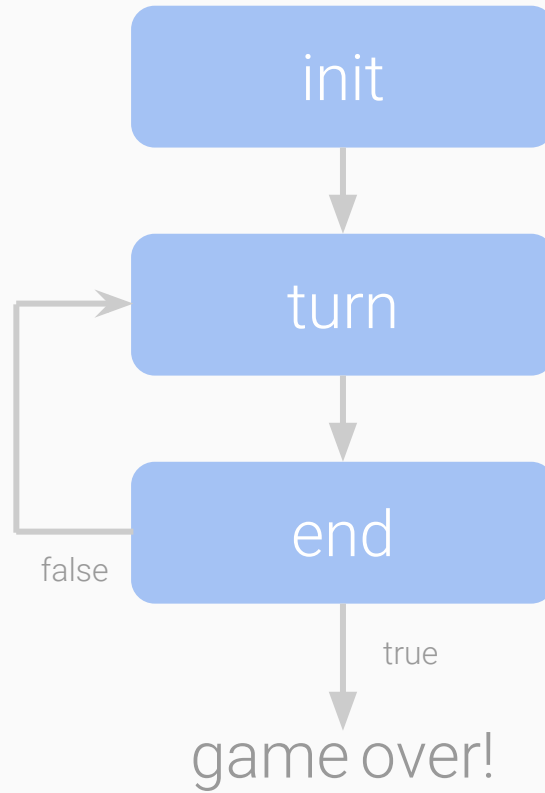
Blocks

Turn-handling

End conditions

Input collection

Game Loop



Syntax - Program Structure

```
#size 500 250           // set window width and height (optional)
#color 0 0 255          // set window background color (optional)
#title "Hello World"    // set window title (optional)

int x; int y;           // declare global variables
int add(...) {...}     // user-defined functions
class Piece {...}      // user-defined classes
init {                  // init block (required)
    tile(3, 3);         // initialize grid size to 3 by 3
    background("hello.bmp"); // initialize board image
}
turn {...}              // turn block - continuous looping of turn block
end {...}              // end block - returns boolean for game end
```

Syntax

Types

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

```
x = 5; y = 5;
```

```
f = 4.5;
```

```
b = true;
```

```
s = "hello";
```

```
c = [x, y];
```

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

Operators

```
=
```

```
+ - * / %
```

```
== !=
```

```
&& ||
```

```
> < >= <=
```

```
! -
```

Keywords

```
gridh;
```

```
gridw;
```

```
init {...}
```

```
turn {...}
```

```
end {...}
```

Syntax - Functions

Built-in Functions

```
tile(3, 3);  
background("hello.bmp");  
  
iprint(0); fprint(4.0);  
sprint("Hello World!");  
  
capture();
```

Function Definition

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

Control Flow

```
if (condition) { ... } else { ... }  
  
if (condition) { ... }  
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

```
<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

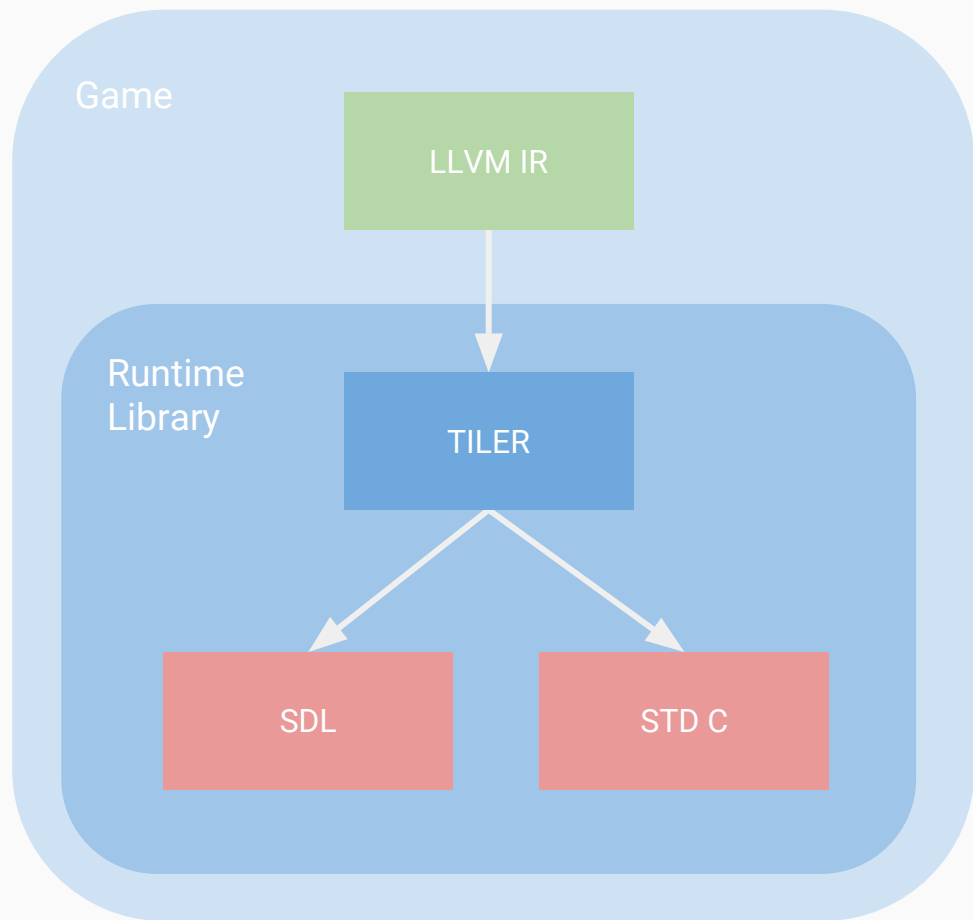
```
grid[x, y] = p;      // grid assignment and moving object p to new location on grid
grid[x, y] = NULL;  // removes object on grid location [x, y]

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

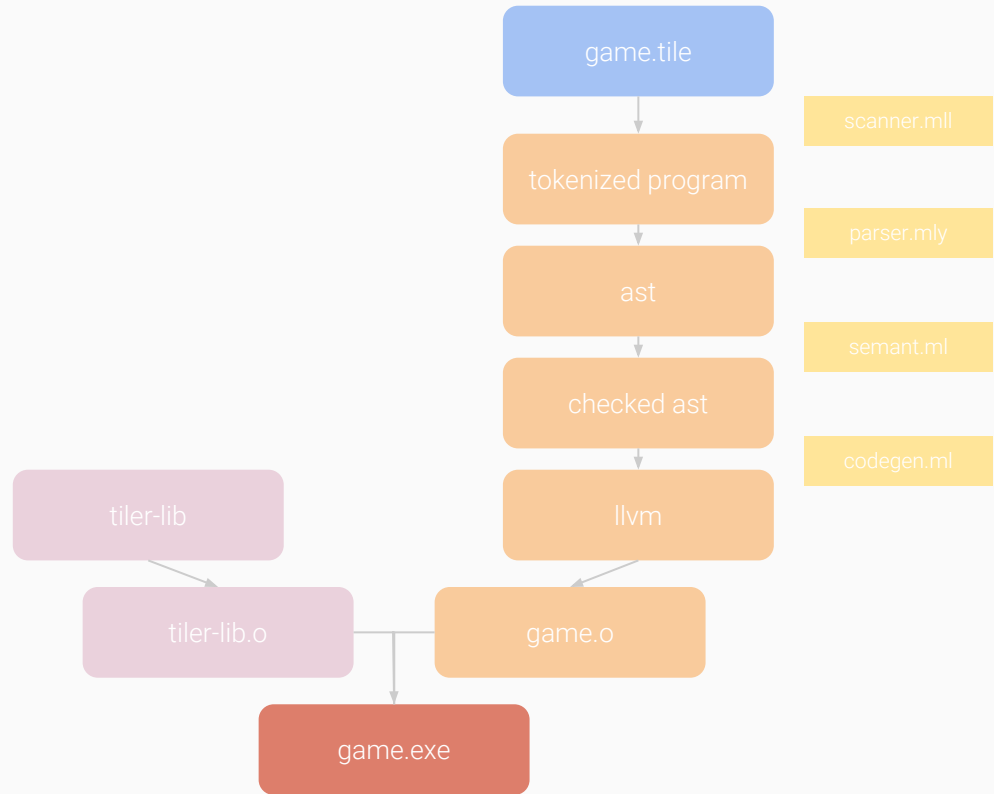
isNull(p);          // returns 1 if object is null
type(p);            // returns class of object

p.player;           // access explicit attributes of object
p.x; p.y;           // access object's location by its implicit attributes, x and y
```

Language Architecture



Compilation Pipeline



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!

```
1  init {  
2      tile(3, 3);  
3      background("hello.bmp");  
4  }
```



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

Testing: Results

test-arith1.tile...OK

test-float-compare1.tile...OK

test-if1.tile...OK

test-turn1.tile...OK

test-arith2.tile...OK

test-float-compare2.tile...OK

test-obj-access.tile...OK

test-while1.tile...OK

test-assign1.tile...OK

test-func-rec.tile...OK

test-obj-assign.tile...OK

test-dowhile1.tile...OK

test-func1.tile...OK

test-print-bool.tile...OK

test-end1.tile...OK

test-global1.tile...OK

test-print-expr.tile...OK

test-float-arith1.tile...OK

test-global2.tile...OK

test-print-float.tile...OK

test-float-arith2.tile...OK

test-global3.tile...OK

test-print-int.tile...OK

test-float-assign.tile...OK

test-helloworld.tile...OK

test-print-string.tile...OK

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 !!!! :-)