What’s in a name?

Recursive Acronyms

- GNU
  “GNU’s Not Unix!”
- PHP
  “PHP: Hypertext Preprocessor”
- PIP
  “PIP installs packages”

GAME

GAME
Analyzes
Metrics
Easily
Problem Domain:
Manipulating statistics from sports and athletic events

Existing Solutions:
- Microsoft Excel
- R
- MATLAB
In Andy Register’s A Guide to MATLAB Object-Oriented Programming, he states:

“Object-oriented techniques also require an expert’s knowledge of both standard and obscure MATLAB functions. Object-oriented programming is an advanced topic and the examples and idioms assume a certain level of MATLAB-language expertise.” [emphasis added]
GAME is the Answer!

Who is it for?

- Coaches and Players
- Team Managers
- Experienced programmers
- New programmers
Purpose

- Find correlation between different factors and success
- Determine best strategies
- Organize and view sports data
serves.game:

class tPlayer {
    text Player
    num height
    num aces
    num matches
    num avg_aces
    num win_perc_clay
    num win_perc_grass
    num win_perc_hard
    num ranking
}

function main() {
    list(tPlayer) players
    load players from "tennis_2011.json"

    list(num) win_clay
    list(num) win_hard
    list(num) win_grass
    list(num) av_aces
    ...
}
GAME Features
## Primitive Types

### Java
- byte
- short
- int
- long
- float
- double
- boolean
- char
- array

### C
- char
- short
- int
- long
- float
- double
- struct

### GAME
- num
- text
- bool
- list
Look Ma, No Semicolons!

GAME syntax draws from Python and Java

- Newline sensitive
- Curly braces, Not Indentation
### Loop: straightforward

<table>
<thead>
<tr>
<th>loop</th>
<th>start</th>
<th>while</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>if</td>
<td>else</td>
</tr>
<tr>
<td>function</td>
<td>return</td>
<td>null</td>
</tr>
<tr>
<td>new</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>class</td>
<td>include</td>
<td>load</td>
</tr>
<tr>
<td>from</td>
<td>num</td>
<td>text</td>
</tr>
<tr>
<td>list</td>
<td>geteach</td>
<td>in</td>
</tr>
<tr>
<td>where</td>
<td>export</td>
<td>to</td>
</tr>
<tr>
<td>break</td>
<td>continue</td>
<td></td>
</tr>
</tbody>
</table>
Loop: straightforward

while loop in C:

```c
int x = 0;

while ( x < 10 ) {
    printf("%d ", x);
    x++;
}
```

for loop in C:

```c
int x;

for (x = 0; x < 10; x++) {
    printf("%d ", x);
}
```

loop in GAME:

```c
loop (start num i = 1, while i <= 10, set i = i + 1) {
    print(i + " ")
}
```
loop.game:

#GAME program, exemplifying loop capabilities
function main() {
    num sum = 0
    loop (start num i = 0, set i = i + 1, while i <= 10, start num j = 0, while j <= 20, set j = j + 2) {
        sum = sum + i + j
    }
    print("Total: " + sum)
}

Output:
0 3 6 9 12 15 18 21 24 27 30

GAME Demo Program #2 (loop.game)
function main() {
    list(num) points_scored = { 15, 9, 27 }

    foreach (num i in points_scored) {
        print(i)
    }

    points_scored.rem(9)
    points_scored.add(22)

    foreach (num i in points_scored) {
        print(i)
    }
}
geteach

```javascript
function main() {
    list(num) points_scored = { 15, 9, 27, 22, 13 }
    list(num) good_games = geteach (num i in points_scored where i > 20)

    foreach (num i in good_games) {
        prints out "27 22"
        print(num_form("#", i) + " ")
    }
}
```

*same as:*

```javascript
function main() {
    list(num) points_scored = { 15, 9, 27, 22, 13 }
    list(num) good_games = {}
    foreach (num i in points_scored) {
        if (i > 20) {
            good_games.add(i)
        }
    }
    foreach (num i in good_games) {
        prints out "27 22"
        print(num_form("#", i) + " ")
    }
}
```
Classes

class MyClass {
    num field1
    text field2

    ... function myFunction() {
        ...
    }

    ...
}

- Game initializes all primitives automatically at declaration
- Makes life easier for user
JSON

class MyClass {
...
}

function main() {
  list(MyClass) mylist
  load mylist from "data.json"
  ...
  export mylist to "output.json"
}

- Easily convert data from JSON file to object-oriented representation
- Easily output list of objects to JSON file
Library Structuring and Inclusion

- Write your own library files
- Use include to include their functions and classes from libraries in your programs
- Handles recursive include conditions and proper code placement insertion (e.g. basketball → math)

```cpp
#include "stdlib/basketball.game"
#include "stdlib/math.game"

function main() {
    list(BasketballPerformance) season
    BasketballPerformance p1
    BasketballPerformance p2
    ...
    BasketballPlayer iSykes

    p1.points = 21
    p1.turnovers = 3
    ...
}
include "stdlib/math.game"

class tPlayer {
    function main() {
        list(tPlayer) players
        load players from "tennis_2011.json"
        list(tPlayer) goodPlayers = geteach(tPlayer i in players where i.avg_aces > 10)

        foreach (tPlayer x in goodPlayers) {
            print(x.Player + '"'s average aces per match: " + x.avg_aces + '" height: " + num_form("#", x.height) + " cm"
        }

        list(num) heights
        foreach (tPlayer x in goodPlayers) {
            heights.add(x.height)
        }
        print("Average height of players with > 10 aces/match:
            " + num_form("#", mean(heights)) + " cm")
        list(tPlayer) fewAces = geteach(tPlayer i in players where i.avg_aces < 5)
        print("Players with fewer than 5 aces per match:
        
        foreach (tPlayer x in fewAces) {
            print(x.Player + '"'s average aces per match: " + x.avg_aces + " height: " + num_form("#", x.height) + " cm"
        }
        heights = { }
        foreach (tPlayer x in fewAces) {
            heights.add(x.height)
        }
        print("Average height of players with < 5 aces/match: " + num_form("#", mean(heights)) + " cm")
        export goodPlayers to "loads.json"
Compiler Architecture

Source GAME File

Pre-Processor

Resolved GAME File

Scanner

Syntactically Valid GAME File, Function Definitions, Class Definitions

Lexer

nextToken() token

Parser

Python Code
Compiler Architecture

- Provides interface for compiler
- Resolves include statements
- Prevents recursive includes
- Saves output as a temporary file
Compiler Architecture

- Checks the file against the grammar
- Collects definitions to allow out of order function/class use
- Pipes definitions to compiler

Resolved GAME File

Scanner

Syntactically Valid GAME File, Function Definitions, Class Definitions
Compiler Architecture

- Lexer creates a stream of tokens
- Parser identifies which rule to apply
- Keeps a symbol stack to perform semantic checking
- Returns python code up the tree
Testing Framework

To Run:
- run make in the testing directory

Example Run:
source/BasketballTest.game
correct/BasketballTest.game
1. Compiles the source/BasketballTest.game file in the source directory
2. Moves the compiled source/BasketballTest.game.py file to the target directory
3. Runs the target/BasketballTest.game.py files in the target directory and pipes the output into output/BasketballTest.game.txt
4. Uses diff to compare output/BasketballTest.game.txt file to correct/BasketballTest.game.txt and stores result in diffs/BasketballTest.game.txt.diff
5. If the diffs/BasketballTest.game.txt.diff file is empty, then that test case has passed, otherwise, show the user the expected vs. actual and mark test case as failed
Mid-range Shots in the NBA?

- LaMarcus Aldridge (Portland Trailblazers): the most prolific mid-range shooter in the NBA
- Houston Rockets: deemphasizes mid-range
- Should teams rely on the mid-range shot?
class NBATeam {
    text Team
    num points
    num in_paint
    num from_threes
}

function main() {
    list(NBATeam) teams
    load teams from "nba_teams.json"

    list(num) perc_from_mid
    list(num) total_points
    foreach(NBATeam i in teams){
        num points_from_mid = i.points - i.in_paint - i.from_threes
        perc_from_mid.add(points_from_mid / i.points)
        total_points.add(i.points)
    }
}
GAME Development

March 1st 2014 - May 10th 2014
Contributions to master, excluding merge commits

Contribution type: Commits
Lessons Learned

Will: Convention is O.K.
Theo: Harness Unique Strengths
Dylan: You Are Never Done
James: Turn Down for What
Sean: Eat Together!
got game?