Fantasy League Object Oriented Development (FLOOD)

A Project Proposal for *COMS W4115: Programming Language & Translators*

Team 9
is

Stephanie Aligbe
sna2111@columbia.edu

Elliot Katz
epk2102@columbia.edu

Tam Le
tvl2102@columbia.edu

Dillen Roggensinger
der2127@columbia.edu

Anuj Sampathkumaran
as4046@columbia.edu

February 23, 2011

Abstract

We present **FLOOD**, an object-oriented programming language designed to facilitate the creation of fantasy league games and simulations.

1 Introduction & Motivation

As defined on Wikipedia: *fantasy sport (also known as **rotisserie**, **roto**, or **owner simulation**) is a game where participants act as owners to build a team that competes against other fantasy owners based on the statistics generated by the real individual players or teams of a professional sport.*

The popularity of fantasy sports has exploded in recent years. A 2007 study by the *Fantasy Sports Trade Association (FSTA)* estimated that nearly 30 million people in the U.S. and Canada, ranging in age from 12 and above, participated and played in organized

fantasy sports leagues. In comparison, an estimated 3 million people in North America played in the early 1990s, swelling to 15 million by the early 2000s. This impressive growth looks to continue since the FSTA study revealed teenagers in both the U.S. and Canada play fantasy sports at a higher rate than the national average, with 13 percent of teens playing in the U.S. and 14 percent playing in Canada.

The 2007 study also estimated the spending habits and overall economic impact of fantasy sports players. Consumers engaging in this ever growing hobby spent $800 million directly on fantasy sports products and an additional $3 billion worth of related media products (such as DirecTV's NFL Sunday Ticket and satellite radio's coverage of MLB).

The growing popularity of fantasy sports is not restricted to North America alone. For example, a more recent 2008 study by a European-based market research company estimated the number of fantasy sports players in Britain ranging between 5.5 and 7.5 million and varying in age between 16-64, of which 80 percent participated in fantasy soccer.

The purpose of **FLOOD** is to provide a programming language tailored to create, with relative minimal effort, fantasy leagues. The language should make it straightforward for a developer to create a league, define its type (and not necessarily confined to sports, as expanded upon in section **2.5**), establish the rules governing the league, enumerate the number of players, and set various other parameters. Our design challenge is to lower as much as possible the barrier entry point for programmers, making it simple enough for novices and beginners, but doing so without sacrificing the robustness and extensibility expected by more advanced developers.

The hobby of playing in fantasy leagues, whether it be in sports, finance or any other fields of personal interests, is enjoying explosive popularity. Naturally, as computer scientists, the members of this team view this burgeoning segment of social gaming and media as an invaluable opportunity to apply our skills and technical know-how towards addressing a real world demand.

2 Language Overview

A **FLOOD** program can:

- Create a League and define attributes such as Team Size.
- Establish Rules for play using the existing libraries or overriding them.
- Rules can be shared with different programmers and can be extended.
- Define features of the league such as draft and trade functions.
- Connect the league to a database of the user's choice.
- Trigger the build for the front end GUI and deploy the program to users.
- Allows programmers different degrees of customization.

2.1 Data Types

- Array
- Boolean
- Number
- String

2.2 Keywords

- League
- Rule
- Player
- User
- Team

2.3 Control Flow

- if/else/elseif
- for...in
- while

2.4 Commenting

- // inline comments
- /* ... */ block comments

2.5 Predefined League Libraries

- Sports (football, baseball, soccer, basketball, rugby, hockey, etc.)
- Stock markets/financial data
- College courses/professors
- Politics (elections/polls)
- Movies (box office gross/critical reception)
- Music (iTunes ratings)

3 Sample Code

Listing 1: FLOOD sample code

```
createLeague(leagueObject1)
createLeague(leagueObject2)

leagueObject1.setName(Rugby)
leagueObject2.setName(Soccer)
```

```
6
7  leagueObject1.Team.setMaxPlayers(16) //number of players per
       team, not the number of users
8
9  leagueObject1.loadRules(Rugby) //rugby rules or any other pre-
       defined rules
10 leagueObject1.Rules.setNumberOfUsers = 10
11 leagueObject1.Rules.setEvaluateFreq(daily, 12:00)
12 leagueObject1.Rules.setNumberOfSubstitutions(10)
13 leagueObject1.Rules.loadDraftPattern(Snake) //or create a new
       draft as follows
14
15 leagueObject1.Rules.createDraftPattern()
16 loop Player(i).turn to end, return in alternate hops //any
       algorithm of the programmer's choice
17 leagueObject1.Rules.setInitialMoney = 2000
18 leagueObject1.Rules.createActions
19   Action goal(10)
20   Action redCard(-10)
21
22 Rules.evaluateWinner(RemainingResources + totalPoints * 200)
       //programmer determines what formula is be used
23 openDatabase(file path or tcp/ip to any network connection to
       the server)
24
25 build()
```

4 Challenges & Open Questions

- How extensible should our language be?
- How to design the syntax?

In designing our language, we face certain technical challenges which we must first overcome in order to move forward. The first decision regards the level of control the programmer will have over the existing library of functionality. For instance, will a programmer be allowed to override the existing draft or will he or she be confined to changing attributes of the pre-existing function?

Another design decision concerns the simplicity of the language and the skill a programmer will need in order to use it to build a fantasy league. Keeping the language simple allows a wide spectrum of programmers of varying skill level and experience to utilize the language. However, will doing so make the language too simple to allow custom libraries and more advanced functionality?

And with regard to syntax, most of the team is comfortable with Java and its associated format and structure. However, will certain other syntactical standards be more suitable for our programming language, e.g. Python and its use of whitespace to discern scope?

These and countless other technical questions and hurdles will undoubtedly arise as our team begins fleshing out the details of our programming language—challenges we are all looking forward facing head on.