A Fantasy League Language

FLOOD

Team 9:
- Stephanie Aligbe
- Dillen Roggensinger
- Tam Le
- Anuj Sampathkumaran
- Elliot Katz
● Introduction
● Language Basics
● Example
  ○ Input
  ○ Intermediate code
  ○ Application
● Architecture
● Challenges and Lessons Learned
Introduction

- Fantasy gaming has become part of the American lexicon.
- Over 30 million currently play some form of fantasy sports.
- Economic impact estimated to between $3-4 billion annually.
- Untapped application domain.
- Natural candidate for social networking integration.
Why FLOOD?

- Domain specific.
- Easy to learn to program in.
- Quickly create fantasy leagues.
- Comprehensive GUI as output with minimal input.
Architecture

Input sample.fld

Lexer
Jflex/
Parser
Byacci

Intermediate code
FloodProgram.java

OUTPUT

GUI
Stat File

BackEnd

Action.java
GUI.java
League.java
IOManager.java
Player.java
StatParser.java
Language Basics

● Simple structure

```
DefineLeague  /* League Setup Here */
```

```
DefineFunctions  /* Functions Here */
```

● Predefined functions for fast development
  ○ `draftFunction, draftPlayer, trade, dropPlayer`

● Types

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Flt</code></td>
<td><code>Str</code></td>
<td><code>User</code></td>
</tr>
<tr>
<td><code>Int</code></td>
<td><code>Bool</code></td>
<td><code>Player</code></td>
</tr>
</tbody>
</table>
# Language Basics

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
</table>
| **While**             | ```
While (numPlayers < 10) {
    footballDraft(); /* User defined draft function*/
};
``` |
| **Function (with:**   | /* */
`Bool isAllStar(Player p) {
    Flt points;
    Bool allStar = False;
    points = GetPlayerPoints(p);
    If (points > 100) {
        allStar = True;
    };
    Return allStar;
};` |
| **-If statement**     | `If (totalPoints > 1000) {
    Alert("Champion!", "You win!");
} Else {
    Alert("Game Over", "You Lose");
};` |
| **-assignment**       |                                                                         |
| **-comments**         |                                                                         |

---

**While**

- **Type:** While loop
- **Example:**
  ```
  While (numPlayers < 10) {
    footballDraft(); /* User defined draft function*/
  };
  ```

**Function (with:)

- **Type:** Function
- **Example:**
  ```
  /* */
  Bool isAllStar(Player p) {
    Flt points;
    Bool allStar = False;
    points = GetPlayerPoints(p);
    If (points > 100) {
      allStar = True;
    };
    Return allStar;
  }
  ```

**If Else**

- **Type:** Conditional statement
- **Example:**
  ```
  If (totalPoints > 1000) {
    Alert("Champion!", "You win!");
  } Else {
    Alert("Game Over", "You Lose");
  };
  ```
DefineLeague
Set LeagueName("PLS-Pseudo League Soccer");
Add User("Carlo Ancelotti");
Add User("Alex Ferguson");
Add Action("score goal", 2.0);
Add Action("block shot on goal", 1.0);
Add Player("Petr Cech","goal keeper");
Add Player("Branislav Ivanovic","defender");
Add Player("Ashley Cole","defender");
Add Player("Michael Essien","midfielder");
Add Player("Ramires","midfielder");
Add Player("Frank Lampard","midfielder");

DefineFunctions
/*Returns a Bool stating if the number of players is greater than the maximum allowed*/
Bool tooBig(Int players){
    Bool flag=False;
    If(players > 13){
        flag=True;
    }
    Return flag;
}

/*The draftPlayer function overwritten by the programmer*/
Bool draftPlayer(User u, Player p){
    Bool tooBig;
    Bool value=False;
    Int i;
    i=GetNumPlayers(u);
    i=i+1;
    tooBig=tooBig(i);
    If(!tooBig){
        AddPlayer(u,p);
        value=True;
    }
    Return value;
}

Example Input
DefineLeague:
This part of the program that builds the league by allowing programmer to add users, players and actions. Users are the teams that will participate in the league, players the athletes on the teams and actions the rules for point distributions.

DefineFunctions:
Any FLOOD program needs four functions to operate which are:
- draftPlayer which adds a given player to a given team. It must return a Bool stating if the draft is successful.
- draftFunction which returns which user's turn it is given a number representing the current draft pick. Users are indexed by the order they are added in DefineLeague which means that in this league, if the draftFunction returns 0, it's saying it's Carlo Ancelotti's turn.
- trade which is responsible for trading two given arrays of players between two given users. It must return a Bool stating if the trade is successful.
- dropPlayer which drops a given player from a given user. It must return a Bool stating if the drop is successful.

Any of these functions that are not defined explicitly will be added to the intermediate code with no rules in them.
public class FloodProgram {
    public static League myLeague;
    public static GUI run;

    public static void main(String[] args) {
        myLeague = new League("PLS-Pseudo League Soccer");
        myLeague addUser(new User("Carlo Ancelotti");
        myLeague addUser(new User("Alex Ferguson");
        myLeague addAction(new Action("score goal", 2.0));
        myLeague addAction(new Action("block shot on goal", 1.0));
        myLeague addPlayer(new Player("Petr Cech", "goal keeper");
        myLeague addPlayer(new Player("Branislav Ivanovic", "defender");
        myLeague addPlayer(new Player("Ashley Cole", "defender");
        myLeague addPlayer(new Player("Michael Essien", "midfielder");
        myLeague addPlayer(new Player("Ramires", "midfielder");
        myLeague addPlayer(new Player("Frank Lampard", "midfielder");
        run = new GUI(myLeague);
        run.drawBoard();
    }

    public static boolean tooBig(int players) {
        boolean flag = false;
        if (players > 13) {
            flag = true;
        }
        return flag;
    }

    public static boolean draftPlayer(User u, Player p) {
        boolean tooBig;
        boolean value = false;
        int i;
        i = u.getNumPlayers();
        i = i + 1;
        tooBig = tooBig(i);
        if (!tooBig) {
            u.addPlayer(p);
            value = true;
        }
        return value;
    }
}
Application

- Features of the Application

Start of the Application

Application in Progress
Application

Confirm a Draft

Confirm a Trade

Confirm a Drop
Application

Upload a Statistics File

View All Possible Actions
Testing methods

- **Unit Testing**
- **Integration Testing**
- **System Testing**

- Black Box testing on the front end and back end.
- Tested Frontend and backend together, after each major integration.
- Tested entire system by writing several Flood Programs.
Challenges

- Brainstorming and then deciding on an intellectually stimulating/novel subject matter for a programming language.
- Coordinating and managing the logistics involved in getting a group of strong willed individuals to work together towards a common goal. Easier said than done.
- Making tough technical decisions and be accepting of accountability when those choices don't always work out.
Lessons Learned

- Be realistic with project scope early on. Try doing too much too soon bogs down development from the get go.
- Be proactive in formulating/implementing language grammar. Do NOT defer because material not yet presented in class.
- Be open to compromise. Aggressive timelines dictate form follows function.
- Don't take yourself too seriously. In any collaborative undertaking, a good sense of humor goes a long way.
Questions?