University of Puerto Rico - Mayagüez Campus
Department of Electrical and Computer Engineering

# Card ++
# Final Report

Eliemer Vélez
Jonathan Irizarry
Ariel Torres

# Introduction

A card game is any game that uses cards as the primary device with which the game is played. Card games have always been a big part of human history. As such, we want to give people a tool to be able to continue this tradition via computer programs.

As part of our Structure and Properties of Programming Languages course we have been tasked with the creation of a new programming language which, in our case, would result in being able to create card games in a short amount of time, with little experience in programming required.

## Language Details

We want Card++ to be as high level as possible, readability is a high priority as we want anyone to be able to pick up our language and start creating. A key feature of our language is that of definition blocks. These blocks contain clear workspaces in which to work different aspects of the game. For example, at the start of your code, you declare a Deck{...} block in which you list all available cards in the entirety of your game. Poker for example, contains fifty-two casino suite cards, thus each item in the Deck{...} block would contain a number (from Ace to King) and a suite (hearts, clubs, spades, diamonds). These blocks contain different aspects of your game code, and are segregated for better readability.

Another key aspect of our language is that every action revolves around three things: player resources (money, energy, etc.), turn control (skip turns, reverse turn order), or fields. The latter is a critical part of our vision. A field is essentially a container for cards. These fields may have special properties to them, such as the Deck field being invisible to every player, that is to say, no one can view the values of the cards within. Other common fields include a player’s hand, the playing field, and card graveyards. Common actions that manipulate fields include: drawing cards (move 1 From Deck to Player) and playing cards (move 1 From Player to Game). Actions that manipulate player resources may include betting, or actions be conditioned to a value, say a mana bar. Thus, from this framework of actions we can implement anything we’d like the player to do.

**Simple Keywords**

- Card: A tuple representing an instance of a game card
- Draw: Adds an instance of a game card to a Field
- Move: Adds a particular instance of a game card to a Field
- Flip: Changes a card's visibility to certain Players
- Shuffle: Mixes all cards' locations in a given Field
- Rules: Boolean statements describing a particular scenario or condition in the game

**Tools Used**

PLY - PLY (Python Lex-Yacc) is an implementation of lex and yacc parsing tools for Python, developed by David Beazley. This tool is used to establish and interpret our language's grammar.

# Language Tutorial

**Installation**

1. Download Card++ from our GitHub repository at https://github.com/Eliemer/Card-plus-plus .
2. Make sure that you have Python 3 (3.6) installed in your machine with the PLY (3.10) library.
3. Run the *Card++.py* file on your IDE or terminal.
4. Start using Card++.

**Using Card++**

Card++ allows you to create your card game using the fifty-two casino suit cards. A card declaration in Card++ is simple. The following is the syntax used to declare a card:

Card *cardname* = ("*value*","*suit*");

For example, if we want to declare the three of diamonds, we would type the following in Card++:

Card Diamonds3 = ("3", "Diamonds");

A valid card declaration receives a valid value and suit as parameters, though we can declare the name however we want.

After you declare the desired number of cards, a field can be created to represent the different types of fields in a card game. The following is the syntax used to declare a field:

Field *fieldname* [*Card 0, Card 1, Card 2, ... , Card N-1*];

We could declare a Field composed of only the three of diamonds:

Field Deck [Diamonds3];

It is also valid to declare an empty field, for example an empty hand would be declared as follows:

Field Hand [];

After the hands and fields are declared, it is valid to operate on them using Card++'s functions.

Some functions are:

Shuffle(Deck); #Shuffles the deck

flip (Diamonds3); #Flips the card

Draw(Deck); #Draws a card from the deck

# Language Development

The language was developed using JetBrains PyCharm IDE. We used PLY in order to generate our language's grammar. A test program was developed that would demonstrate the execution of Card++ commands in an orderly manner that users of the language should use. Below is the test program used during development.

```
Card ClubsAce = ("Ace", "Clubs");
Card Clubs2 = ("2", "Clubs");
Card Clubs3 = ("3", "Clubs");
Card Clubs4 = ("4", "Clubs");
Card Clubs5 = ("5", "Clubs");
Card Clubs6 = ("6", "Clubs");
Card Clubs7 = ("7", "Clubs");
Card Clubs8 = ("8", "Clubs");
Card Clubs9 = ("9", "Clubs");
Card Clubs10 = ("10", "Clubs");
Card ClubsJack = ("Jack", "Clubs");
Card ClubsQueen = ("Queen", "Clubs");
Card ClubsKing = ("King", "Clubs");

Card SpadesAce = ("Ace", "Spades");
Card Spades2 = ("2", "Spades");
Card Spades3 = ("3", "Spades");
Card Spades4 = ("4", "Spades");
Card Spades5 = ("5", "Spades");
Card Spades6 = ("6", "Spades");
Card Spades7 = ("7", "Spades");
Card Spades8 = ("8", "Spades");
Card Spades9 = ("9", "Spades");
Card Spades10 = ("10", "Spades");
Card SpadesJack = ("Jack", "Spades");
Card SpadesQueen = ("Queen", "Spades");
Card SpadesKing = ("King", "Spades");

Card DiamondsAce = ("Ace", "Diamonds");
Card Diamonds2 = ("2", "Diamonds");
Card Diamonds3 = ("3", "Diamonds");
Card Diamonds4 = ("4", "Diamonds");
Card Diamonds5 = ("5", "Diamonds");
Card Diamonds6 = ("6", "Diamonds");
Card Diamonds7 = ("7", "Diamonds");
Card Diamonds8 = ("8", "Diamonds");
Card Diamonds9 = ("9", "Diamonds");
Card Diamonds10 = ("10", "Diamonds");
Card DiamondsJack = ("Jack", "Diamonds");
Card DiamondsQueen = ("Queen", "Diamonds");
Card DiamondsKing = ("King", "Diamonds");
```

```
Card HeartsAce = ("Ace", "Hearts");
Card Hearts2 = ("2", "Hearts");
Card Hearts3 = ("3", "Hearts");
Card Hearts4 = ("4", "Hearts");
Card Hearts5 = ("5", "Hearts");
Card Hearts6 = ("6", "Hearts");
Card Hearts7 = ("7", "Hearts");
Card Hearts8 = ("8", "Hearts");
Card Hearts9 = ("9", "Hearts");
Card Hearts10 = ("10", "Hearts");
Card HeartsJack = ("Jack", "Hearts");
Card HeartsQueen = ("Queen", "Hearts");
Card HeartsKing = ("King", "Hearts");

Field Deck
[
    ClubsAce,
    Clubs2,
    Clubs3,
    Clubs4,
    Clubs5,
    Clubs6,
    Clubs7,
    Clubs8,
    Clubs9,
    Clubs10,
    ClubsJack,
    ClubsQueen,
    ClubsKing,

    SpadesAce,
    Spades2,
    Spades3,
    Spades4,
    Spades5,
    Spades6,
    Spades7,
    Spades8,
    Spades9,
    Spades10,
    SpadesJack,
    SpadesQueen,
    SpadesKing,

    DiamondsAce,
    Diamonds2,
    Diamonds3,
    Diamonds4,
    Diamonds5,
    Diamonds6,
    Diamonds7,
    Diamonds8,
    Diamonds9,
    Diamonds10,
    DiamondsJack,
    DiamondsQueen,
    DiamondsKing,
```

```
    HeartsAce,
    Hearts2,
    Hearts3,
    Hearts4,
    Hearts5,
    Hearts6,
    Hearts7,
    Hearts8,
    Hearts9,
    Hearts10,
    HeartsJack,
    HeartsQueen,
    HeartsKing
];

Shuffle(Deck);

Field Hand [];

Field Table [];

Draw(Table);
Draw(Table);
Draw(Hand);
Draw(Hand);
Draw(Hand, 3);

Shuffle(Hand);

flip(Diamonds3);

Move(3, Table, Hand);

Hand[1];
Table[3];

id = Hand[1];
flip(Hand[1]);
```

Card++ Test Program

The test results are as follows:

```
Field Deck initialized to:

1 of Clubs
2 of Clubs
3 of Clubs
4 of Clubs
5 of Clubs
6 of Clubs
7 of Clubs
```

8 of Clubs
9 of Clubs
10 of Clubs
11 of Clubs
12 of Clubs
13 of Clubs
1 of Spades
2 of Spades
3 of Spades
4 of Spades
5 of Spades
6 of Spades
7 of Spades
8 of Spades
9 of Spades
10 of Spades
11 of Spades
12 of Spades
13 of Spades
1 of Diamonds
2 of Diamonds
3 of Diamonds
4 of Diamonds
5 of Diamonds
6 of Diamonds
7 of Diamonds
8 of Diamonds
9 of Diamonds
10 of Diamonds
11 of Diamonds
12 of Diamonds
13 of Diamonds
1 of Hearts
2 of Hearts
3 of Hearts
4 of Hearts
5 of Hearts
6 of Hearts
7 of Hearts
8 of Hearts
9 of Hearts
10 of Hearts
11 of Hearts
12 of Hearts
13 of Hearts

Deck shuffled to:

6 of Hearts
4 of Hearts
10 of Clubs
9 of Hearts
13 of Hearts
13 of Spades
11 of Clubs
2 of Hearts

7 of Hearts
2 of Spades
9 of Clubs
3 of Spades
12 of Clubs
10 of Diamonds
5 of Clubs
5 of Hearts
11 of Spades
6 of Diamonds
13 of Clubs
2 of Clubs
1 of Hearts
10 of Hearts
4 of Clubs
8 of Clubs
3 of Clubs
11 of Hearts
7 of Spades
1 of Spades
6 of Spades
3 of Hearts
9 of Spades
1 of Diamonds
7 of Diamonds
2 of Diamonds
1 of Clubs
5 of Diamonds
8 of Diamonds
9 of Diamonds
11 of Diamonds
13 of Diamonds
4 of Spades
6 of Clubs
12 of Diamonds
8 of Hearts
3 of Diamonds
12 of Hearts
5 of Spades
12 of Spades
8 of Spades
10 of Spades
4 of Diamonds
7 of Clubs

Field Hand initialized to:

EMPTY

Field Table initialized to:

EMPTY

Table updated to:

7 of Clubs

Table updated to:

7 of Clubs
4 of Diamonds

Hand updated to:

10 of Spades

Hand updated to:

10 of Spades
8 of Spades

Hand updated to:

10 of Spades
8 of Spades
12 of Spades
5 of Spades
12 of Hearts

Hand shuffled to:

5 of Spades
10 of Spades
8 of Spades
12 of Hearts
12 of Spades

Flipping: 3 of Diamonds
Current visibility: ***

Hand updated to:

5 of Spades
10 of Spades

Table updated to:

7 of Clubs
4 of Diamonds
12 of Spades
12 of Hearts
8 of Spades

```
Index 1 of Hand :

<GameElements.Card.Card object at 0x0000028A4F3C7320>

Index 3 of Table :

<GameElements.Card.Card object at 0x0000028A4F3C7358>

Index 1 of Hand :

<GameElements.Card.Card object at 0x0000028A4F3C7320>

Index 1 of Hand :


Flipping: 10 of Spades
Current visibility: ***


Process finished with exit code 0
```

Test program results

# Conclusion

The development of Card++ has given us the opportunity to truly see the complexity behind a programming language. The completed language implementation gives users the opportunity to develop their own card games in a simple, high-level environment. Card++'s simple keywords allow the readability of the code to be superior to any implementation using another programming language. The goal of this project is to allow card game creators to innovate by programming their own card games in a simple way. Card++ achieves this goal by providing the users with a simple interface that does not require much knowledge outside of basic programming.