COMS W4115 - Fall 2011
Prof. Stephen Edwards
PLT Language Proposal

Eric Chao | [ehc2129]
Susan Fung | [sgf2110]
Jim Huang | [jzh2103]
Jack Shi | [xs2139]

DESCARTES

Introduction

Descartes, or "some cards," is a game design programming language. The language is limited to any operating system capable of running Objective CAML, since all of the programs will be passed through an OCAML compiler. The focus of Descartes will be to implement card games strongly associated with the standard 52-card deck, although it can be used equally well to write any other non-real-time card game of various complexities.

Motivation

Using a general-purpose language such as Java or C would require many lines of code dedicated to designing the decks, players, different cards, etc. Descartes will allow programmers of all levels to focus solely on creating the game design, the rules behind the games, and encourage beginner coders to quickly sample games for their own learning or entertainment uses.

Below are some card game types that can be created using Descartes followed by a more in-depth explanation of the different card game types.

Card Game Types	Examples
Trick-Taking	Bridge, Whist, Euchre, Spades, Hearts, Twenty-Eight, Tarot Card
Matching	Rummy, Go Fish, Old Maid
Shedding	Phase 10, Rummikub
Accumulating	War, Egyptian War
Fishing	Scopa, Cassino
Comparing	Poker, Blackjack, Baccarat
Solitaire (Patience)	Solitaire
Drinking	Presidents, Daihinmin
Multi-Genre	Pinochle. Belote, Poke, Flaps, Skitgubbe, Tichu
Trading Card Games	Magic: The Gathering, Pokemon, Yu-Gi-Oh!, Marvel Comics
Casino/Gambling	Poker, Blackjack
Fictional	Pyramid, Exploding Snap

Card Game Types

There is a wide variety of card game types. Descartes is meant to cover this vast spectrum of card games. Some card game example types are trick-taking, matching, shedding, accumulating, fishing, comparing, patience, drinking, collectible card, casino, and many more. Although these are the general categories, some games are a combination of multiple genres.

1. Trick-taking Games

   A trick-taking game is a turn-based game where multiple rounds take place until all players run out of cards. During each round, each player will play a card from their hand and based on the values of the card, the winner will take the played cards.

2. Matching Games

   The objective of matching games is to acquire as many matching cards before your opponents.

3. Shedding Games

   Shedding games' objective is to discard all your cards in your hand before your opponents.

4. Accumulating Games

   Accumulating games require you to acquire all the cards in the deck in order to win the game.

5. Fishing Games

   In fishing games, cards from the hand are played against cards laid out on the table. Table cards are captured if they match.

6. Comparing Games

   Comparing card games involve each player's hand values being compared with those of other players. The player with the highest-value hand is the winner.

7. Patience Games

   Patience games are also known as solitaire games which are only played by one player. Most games start with a specific layout and to win the game, the player must construct another layout or clear the table of cards.

8. Drinking Games

   As the name implies, these games involve drinking or forcing other players to drink. There are many typical card games used as drinking games without their original objective. For example, Poker can be played as a drinking game in that the losers may have to consume a beer instead of losing money.

9. Multi-Genre Games

   These games are a combination of two or more types of games. The most common combinations are matching and shedding.

10. Trading Card Games

    Collectible card games consist of decks of proprietary cards that differ among players. The contents of these decks are a subset of a very large pool of available cards that have different attributes. A player accumulates his or her deck by purchasing or trading desirable cards. Each player uses his or her own deck to play against the other players of the same proprietary card types.

11. Casino Games

    Casino card games revolve around wagers of money. The obvious objective is to win as much money as possible. These games are designed for using some sort of strategy to reach this goal.

12. Fictional

    These card games revolve around science fiction. The game is often used to depict a story that involves background activities in a room.

Data Structures

The syntax and structures will be most similar to Java, but also with influences from O'Caml and Python.

Primitive Types
boolean
int
double
String
List

Card

1.	Initialize	- Creates a card easily
2.	ToString	- Printable String value
3.	ID	- Unique identifier for card
4.	Attributes	- Parameters of the card (Value, Suit, etc.)
5.	Move	- Moves the card from a CardStack to another
6.	Visibility	- Players who can access the card

CardStack

1.	Initialize	- Creates a CardStack easily
2.	ToString	- Printable String value
3.	Attributes	- Parameters of the CardStack (Bet Size, Points, etc.)
4.	Shuffle	- Shuffles the cards
5.	Reverse	- Reverses the order of the cards
6.	Remove	- Removes a certain card
7.	Contain	- Returns boolean denoting whether it contains a particular card
8.	Visibility	- Default visibility for cards added or moved to this card stack

Player

1.	Attributes	- Parameters of the player (Bet Size, Points, etc.)

Field (Fields may help in dividing up the board.)

1.	Attributes	- Parameters of the field

Game (Default game type that allows fast implementation of games.)

1.	Players	- List of players
2.	CardStacks	- List of CardStacks
3.	Attributes	- Other parameters of the game

Operators and Syntax

==	Equals
<	Less Than
>	More Than
<=	Less Than or Equal to
>=	More Than or Equal to
!=	Not Equals
<>	Within (int within range)
&&	And
||	Or
>>	Move single or multiple cards in a CardStack
+	Concatenate CardStack, Strings, etc.
@	Location of CardStack
/**/	Comment

Sample Code

This sample code shows what the setup for a game of Texas Hold'Em might look like:

```
Texas{
        CardStack deck, p1Hand, p2Hand;
        Player[] players;

        Setup(int numPlayers){
                /* If number of players is not 2, print "Wrong number of players." and quit.*/
                if(numPlayers!=2){
                        print "Wrong number of players.";
                        quit();
                }

                /* Create a deck composing of 2 default 52 card decks. */
                /* Using var will allow the type inference to decide the structure. */
                var Deck1 = DefaultCardStack; /* DefaultCardStack is part of standard lib.*/
                var Deck2 = DefaultCardStack;
                deck = Deck1 + Deck2;
                deck.visibility = [];

                /* Initialize players and hands with default visibility. */
                var P1,P2 = Player();
                players = [P1;P2];
                p1Hand.visibility = [P1];
                P1.CardStacks = p1Hand;
                p2Hand.visibility = [P2];
                P2.CardStacks = p2Hand;

                /* Deal card to each player. */
                Deal(2, [p1Hand,p2Hand]);
        }

        /* Deal function. Moves numCards from top of deck to designated card stack.*/
        Deal(int numCards, CardStack[] cardStacks){
                /* For each card stack, move numCards from deck to that stack. */
                for(cardStack:: cardStacks){
                        deck[0.. numCards-1]>> hand@0;
                }
        }
}
```