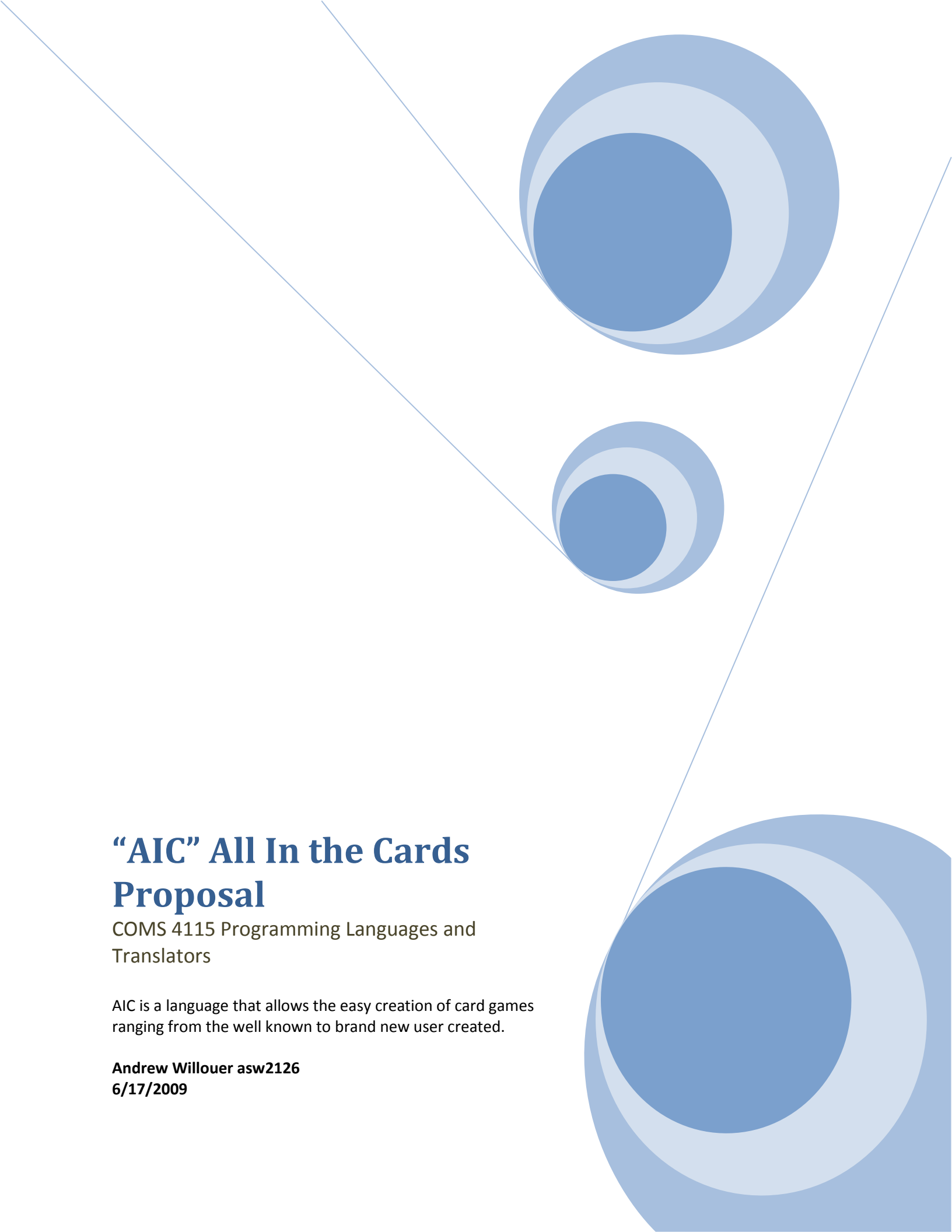

“AIC” All In the Cards Proposal

COMS 4115 Programming Languages and Translators

AIC is a language that allows the easy creation of card games ranging from the well known to brand new user created.

Andrew Willouer asw2126
6/17/2009

Introduction

There are thousands of card games that use a standard 52 card deck. AIC allows the quick creation of card games by only specifying certain parameters. AIC language describes the rules, players, dealer, number of decks, and card values and the AIC compiler takes this input and creates a console based card game. This allows the user to concentrate on the rules of the game and not worrying about the coding details.

Description

The AIC language only supports turn based card games. As the card games are played in a console window only one player can go at a time. The game designer has to design all the rules, how to win the game, legal moves, max number of players and everything else that goes into a game. AIC will come with defaults but the more rules specified the less "cheats" there will be in one's newly created game. By default the card values are initialized in order 2 through Ace with Ace being the highest value card. The game designer can change the card values though.

The AIC compiler than takes the "rules" and creates and executable that can be run in a console window. The card games are completely text based and cards will be displayed with simple text. For example 2H means two of hearts. D will be for diamonds, H for hearts, C for clubs and S for spades. There will be a number for cards 2-10, or a letter for Jack (J), Queen(Q), King(K), Ace(A) followed by a letter for the suite for the output. This will allow games with large hands to fit nicely on the screen. The games will then take text strings as input and valid input will depend on the game rules. They could be hit, fold, a specific card and so forth. A game will automatically stop when a winner is determined. There will no computer opponents except for a dealer. The AIC language will allow for simple rules to be programmed in for a dealer's moves.

Key Words

NDecks (To specify the number of decks, 1-10)

Players (To specify the number of players, 1-10)

Dealer (To specify whether to have a dealer as a separate player or not, true or false)

CName (The name of the card game)

Print (To print some text to the screen)

Shuffle (To randomly distribute the cards throughout the deck. Can be called at any time)

Deal (To give a player some cards)

Input (Waiting for a text string from the console)

Built in Types

Decks is a list that contains all the cards. Cards can be removed from the deck or put back into the deck depending on the rules of the game. Each player and the dealer will also have a hands which will contain all of their cards.

Cards is a list that contains a tuple, the card symbol and an integer value. So for example Cards.2H.value will return 2 for the 2 of hearts. Or the value can be changed

Cards.JH.setvalue=10

This will set the Jack of Hearts value to 10. Cards can be removed from a deck with simple subtraction and added to a hand.

Decks – Cards.JH

Hands.P1 + Cards.JH

The AIC language also supports:

+ - / * For addition, subtraction, division and multiplication

< > <= == != For less than, greater than, less than or equal, equal to, not equal to

If else while for and other standard conditionals and loop functions.

Example Code

Below is example code for Blackjack played with 2 decks and a dealer.

Decks = 2

Players = 2

Dealer = true

CName=Blackjack

Print(CName) //Print the name of the game to the console window

Shuffle //This will shuffle 2 decks of cards and randomly distribute the cards in the deck.

//Now need to set the value of the face cards.

Cards.J*.setvalue = 10 //Using matches all 4 suites.

Cards.Q*.setvalue = 10

Cards.K*.setvaule =10

Cards.A*.setvalue =11 //The value of 1 will be programmed into the rules

//Card game initialized at this point.

Deal(Dealer, 2) This will take the top 2 cards from the deck and add them to the dealer's hand.

Print(Hands.Dealer.Cards[1]) //Only prints the dealers second card and the first is hidden in blackjack

Print(Hands.Dealer.Cards.value[1]) //Only prints the dealers second card's value

//Now need to deal to all the players

For(J = 1 to Players.value)

{

Deal(J, 2)

Print(Hands.J.Cards) //Prints the cards the player has (For example 2H 9S)

Print(Hands.J.Cards.value) //Prints the player hand's value which would be 11 for the previous cards

}

For(J = 1 to Players.value)

{

Print("Player" J "Do you want another card? Yes or No?"

Input //wait for text from console

If(Input = "Yes")

{

Deal(J,1) //removes top card from deck and adds to players hand

Print(Hands.J.Cards) //Prints the cards the player has (For example 2H 9S)

Print(Hands.J.Cards.value) //Prints the player hand's value which would be 11 for the previous cards

If(Hands.J.Cards.value > 21)

```
        {
        Print("Player" J "You went over 21, you lose")
        }
}
//Player's turns over, time for the dealer to go
Print(Hands.Dealer.Cards) //Print both of the dealers cards
Print(Hands.Dealer.Cards.value]) //And their values
For(J = 1 to Players.value)
{
        //Deal the dealer cards until he goes bust or ties or beats the players hand and make sure the
        player hasn't already gone bust
        While(Hands.Dealer.Cards.value <= 21) && (Hands.Dealer.Cards.value <=
        Hands.J.Cards.value)&&(Hands.J.Cards.value<=21)
        {
            Deal(Dealer, 1)
            Print(Hands.Dealer.Cards) //Prints the cards the dealer has
           Print(Hands.J.Dealer.value)
        }
}
//At this point the dealer has compared his cards to all the players and either gone bust or has a higher
score or equal score then all of them.  Dealer wins if the scores are equal in this version.
        If(Hands.Dealer.Cards.value > 21)
        {
        Print("Dealer went bust everyone wins");
        }
        Else
        {
```

Print("Dealer wins")

}