

Written by Billy D. Spelchan for www.BlazingGames.com

Copyright © 2003-2005 Blazing Games Inc. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the file called fdl.txt

Chapter 14

Card Games Summary

Contents

This is a simple summary of what was learned in this part of the book and some suggestions on how you can applied what was learned here towards your own projects.

- Chapter 10: Card Games Overview - Summarized
- Chapter 11: Building Card Classes - Summarized
- Chapter 12: Video Poker - Summarized
- Chapter 13: Cribbage Square - Summarized
- Projects - Where to go from here

Chapter 10: Card Games Overview

Why Card Games

Computer versions of card games are fairly popular. The question is what are the advantages and reasons why someone would play a computer version of a card game when decks of cards are easy to come by?

- Fair Referee - computer card games make sure all the players follow the rules and prevent solitaire players from rule benders.
- Solo Play - some card games require other people to play against. If you can't find a person to play your favourite card game with, the computer is always there.
- Scoring - It is always nice to have someone else handle the scoring, especially if they don't make mistakes.

Part Plans

In this section of the book we first created a common card class (learning how to create classes in Action Script 1.0) and then created a video poker game and a cribbage square game.

Common Classes

First, we created a card movie. While fairly simple to build, creating a deck of cards is a time consuming process. Thankfully, we only needed to do it once.

Next we created some Flash classes. As this book was written before Macromedia's Flash MX 2004 was released. MX 2004 introduced Action Script 2.0, which greatly improved the way classes were created in Flash. Still, knowing how to create classes in Action Script 1.0 is probably an important skill, even if it used only to understand older Flash movies.

Video Poker Design

Video Poker is like the casino machines. You may bet one or five dollars. Once you have bet, the cards will be drawn. You can then choose which cards you want to draw by clicking on the card. The word "draw" will appear above the card to indicate that it is going to be drawn. If you change your mind, just click on the card again. Once you are done click on the draw cards button to see the results.

Cribbage Square design

There are at least three ways that a cribbage squares game can be created. We are going to go with a four by four grid of locations where they can place cards with an additional slot for the start card. The player keeps drawing cards until all the slots have been filled. And then the results - using standard cribbage scoring for each row, column and diagonal - is calculated.

Chapter 11: Building Card Classes

How to create a Deck of Cards

Before you can create a card game you need a deck of cards. This means that you are going to need to create 52 cards. One way would be to create the cards in a paint program and import the card images into a card movie. The problem with that technique is that you lose the advantages of vectors. For that reason, we created a deck of cards entirely in Flash.

Card Components

When you think about it, a set of cards can be broken into 34 components. Better yet, most of these are small components that only take a few curves to represent. These parts are the suits, the red numbers and letters, the black numbers and letters, the Jack, the Queen, and the King.

Creating the Card movie

The card movie is a combination of a card back image and the 52 card images. Each card has a numeric ID, with 0 being no card. Cards are sorted by their group and then by their order within the suit. As every card is a single frame, we can mathematically calculate the proper frame to go to in order to show the card.

We needed some code to initialize the card movie, set the card's ID, get the card's ID, show the card's face, and to hide the card's face (show the back of the card).

Testing the Card Movie

It is always wise to test any code that you create. When you are creating a movie or class that is going to be used by more than one movie, then testing becomes even more important. With that in mind, we wrote a movie to test the Card movie that we just created. The test simply loops through every card in the deck. While displaying the state of a card it shows both the front and back of the card. This way you can simply watch all the cards reveal themselves.

Flash Classes

This book was written before Flash MX 2004 was released. For that reason, all the games created in this book were created before Action Script 2.0 was released. Action Script 2.0 greatly increases the programmability of Flash by adding proper classes, interfaces, and even typed variables. However, as Flash MX and Flash 5 programs use Action Script 1.0, it is a good idea to learn how to create classes the hard way.

We had cards, but we needed a way of shuffling and dealing cards. To solve that problem we created a separate deck class. To create this class, we had to learn how classes are created. The new operator is used to create an instance of the class. It works by trying to find a constructor function for the indicated class name. To create a class in Action Script 1.0, you simply had to create a new constructor function.

Adding functions to a class is a bit tricky to explain. All classes are given a prototype object that holds functions and other variables that are available to any instance of that class. To add a function to a class, you need to add the function to the prototype object. This can be done two ways. First, you can create the function directly in the constructor by assigning the function name to the code for a function. The other way is to assign a function to the prototype, by using `classname.prototype.functionname = function(...) { ... };`

Creating the Deck class

While we could have created the deck class right inside the Flash program, this makes the code less portable. Instead we created the class file outside of the Flash editor. To use this file, we only need to use Flash's `#include` command to include the file. Creating the class this way makes it much more portable and easier to use in other projects.

The class creates an array of 52 elements. Elements are assigned the card ID values of 1 through 52 (see above). The first concern was to shuffle the deck. To do this I used my swap method of shuffling a deck. Other functions in the class are for drawing a card, and cloning the deck.

Testing the Deck

The deck class test is very similar to the card test we created earlier. This class creates two copies of the deck class. It shuffles one deck and then clones the deck with the other class. We then reveal all the cards and end the demo once all 52 cards have been shown.

Chapter 12: Video Poker

Game Layout

As the game is a simulation of a video game found at many casinos it only makes sense to have a background that reflects this fact. This is actually done in two layers. The "Screenback" layer holds the screen and the "Console" layer holds the console (outer part of the machine). In order to actually play the game, we are going to need a few more layers. The layers used in the movie are "Code," "Info," "Bet," "Console," "FRCard," "RCard," "CCard," "LCard," "FLCard," and "Screenback."

Betting

Video poker, and poker in general for that matter, is a casino game. This means that betting is an important aspect of the game. In a casino, the video poker games I have seen are based around a unit of currency. That unit, be it nickles, quarters, or dollars is the amount of a bet. Most machines allow for more than one unit be bet at a time. I have decided that my virtual machine will allow for 1 to 5 units to be used.

In the first frame we import our deck class. The second frame of the "Code" layer is where we placed our initialization. The initialization is designed to run only once. We also have a function, `updateText`, that sets the four dynamic text variables to the values contained within game variables.

The third frame of the code layer we labelled "BetWait." This is the location the movie moves to whenever the player has to make another bet. On the fifth frame (though technically, the specific frame doesn't matter), we created a keyframe in the code layer and label it "ClearCards."

Revealing the Hand

The revealing of the initial drawing of cards is done in three phases. The first phase the cards are removed from the game play field. The second phase is dealing the cards. Finally we animate the cards going from their top position to their proper final locations.

Selecting the Cards

We needed a way of letting the player know that the card is selected so we created a draw movie. This single frame movie simply contains the word “DRAW” in red bold letters. Why a movie? Quite simple, we need to be able to hide the movie, and for that we need to use a movie clip.

The draw movies was placed over the five cards. An `initGame` function makes all the draw movies invisible. Handling the mouse clicking is done in two parts. First, we have the mouse support code, which we placed at the top of frame 2 in the "Code" layer. The second step is activating the draw button and make sure the mouse button handling will work by adding code to the "select" frame of the "Code" layer.

Drawing the Cards

Drawing cards has to be done on a per card basis so that we can allow Flash to do the animation. The code to handle the drawing of the cards is fairly similar for all the cards so we can create a simple function to handle the drawing called `drawCard`. The draw animation consists of the card being removed followed by the new card being moved into place.

Right now all the cards are being drawn, and the value of the drawn card is the same value that the card already was. To fix these problems we are going to need to add some action script as well as a few more labels. The code simply sees if the card is being drawn. If not it skips to the next draw, otherwise is calls the `drawCard` function for that card.

Calculating the Win

The results of the hand is now calculated. While this work is all done in a single function, the work is kind of complicated. The first thing that is done is the cards are sorted by their face values. Next, we prepare all the variables that we are going to need to find the results. With the variables set, we can now determine if the results of the hand is a straight, a flush, or both as well as checking how many groups of cards there are. Next we analyse the results to determine what the hand was.

Handling the Results

If the result of the hand wasn't a winning hand the play head goes to the bet loop immediately otherwise the movie continues playing to give us time to show the results. To finish up the game we add a bit of sound to the game.

Chapter 13: Cribbage Square

Layout

The game layout consists of four layers. The background layer consists of a bitmap fill, using a wood texture that was created in fireworks. The cards layer consists of the card movies that we are using in the game for the cards in their final positions. The total layer holds all the totals of the columns, rows and diagonals. The code layer contains the code.

Card Slots

The purpose of a card slot is simply to show the player where cards can be placed and place the current card at the indicated location when the player clicks on the slot. A button is the ideal thing to use for a card slot. The slot buttons go on the button layer over top of the cards.

Initializing the Game

The Initial goals for initialization are threefold. First, we need a deck to play with. Second we need an easy way of accessing the cards and the slot buttons on the layout. Finally we need to activate all the slot buttons.

Dealing the Game

We could have put the code to handle the startup right in the start frame. Instead, we wrote a function (placed in frame 2) that handles this activity. This way all the code is in one location which helps makes the code easier to understand.

The startGame() function's first task is to reset the card count. The next thing the function does is the task of resetting the slots and making sure that the deck card is not showing some value. It then adjusts the score by calling an updateScore() function and finally it calls the nextCard() function which deals the player a card.

The nextCard function is going to first increase the card count. Then it makes sure that there are still slots that need to be filled. If there are no slots left it will start playing the movie at the gameOver frame.

We replace the placeCard function with code that actually does something. The first thing the function does is assigns the card underneath the selected button with the value that the deck card has showing. Next we make sure that the selected card is showing its face. We then make the button covering the card invisible. We call the nextCard function so the next card is dealt. Finally, we update the scores by calling the updateScore function.

Scoring Hands

What is the point of playing a cribbage square's computer game if at the end of the game you need to take out a piece of paper and add up your score? This section we will create the low-level scoring function that takes five raw card values and turns them into a score. Next section, we will create some glue functions that tie the scoring to the game.

The first thing we needed were some convenience functions for getting the face value and the suit. We then wrote a monster of a function that calculates the score of the hand. While it is possible to break up this function into a bunch of smaller functions, the procedure is very linear in nature so I don't see any point in doing that other than making a printout look nicer.

The scoreHand function takes five values. The first four cards are the cards in the player's hand (a horizontal, vertical, or diagonal line on the square) while the fifth value holds the start card. The first step in calculating the score is to build an array of the face value, as almost all of cribbage scoring is based on the face values of the cards. At the same time we are building this array, we can see if there is a flush (4 or 5 cards). If you recall from our discussion of the cribbage scoring system in chapter 10, then you will know that if a hand contains a jack of the same suit as the start card, the hand gets a point. This point is known as the nob.

The next task is a fairly complex task. We need to find the number of runs and the number of pairs. It is fairly efficient to do both of these things at the same time. In fact, counting pairs is partially how we dealt with multiple runs. This task is done in three steps. Sorting, checking, and scoring.

Finally, we finish of scoring by seeing how many combinations add up to 15. This is done by using bits to represent cards. There are 5 cards, meaning five bits. If you can't remember how binary works, look back at chapter 7. Five bits give you 32 combination. Loop through the combinations and add up the face values (making sure that face cards only count as 10). Any combination that adds up to 15 earns two points.

Scoring the Game

The scoring routine that we wrote is designed for five cards, so in order to score any line in the square all five cards that make up that line must be available. To make sure the user isn't confused about the lack of a score, we should display some type of message. I have chosen to use the letters NA, which stands for Not Available, as the value to assign any unfinished hand.

I have written a function, called `checkScore`, that takes four index values and returns the score text to show. Only four index numbers are needed as the start card index is always going to be used. If the hand is not valid for scoring we return the "NA" message, otherwise we make a call to the `scoreHand` function with the id's of the five cards that make up the hand.

Finally, we put together the final version of the `updateScore` function. This function just calls the `checkScore` function for all the rows, columns, and diagonals assigning the result to the appropriate dynamic text block.

Replay

When people finish playing a round of cribbage squares, there are three things that most people will do. Grab up the dealt cards and play through that particular shuffle again, shuffle the deck and deal a new hand, or quit playing. Creating a popup menu that gives the player the above choices while also showing the final game total and best total is not that difficult of a task.

The `newGame` function already exists, so all that was needed to make the make this movie usable to the main program is to write a `replayGame` and `quitGame` function. The `replayGame` function simply clones the official deck without shuffling the deck first. The `quitGame` function simply returns to the title screen.

We needed the animation that reveals this menu. As we already had a `gameOver` label, we only need to worry about adding a layer for the menu animation. The ending menu, however, doesn't have any scoring information. Creating a function called `finalTotals` that calculates the final total was simple enough. In order for there to be a current best, we need to reset the best score every time a new shuffle occurs.

Title

The final thing that we had to do is the title sequence. For this sequence, I simply wrote up the title text. I then did a motion tween which consisted of the text scaled down and located off screen to the text in the final position in the final size. Adding a few rotations to the motion finishes of this title sequence.

Projects

The code and movie clips we created in this part of the book should give the average reader a very good foundation for building other card games. For those of you who don't want to jump into a large project, here are some side projects that I have done myself that are fairly simple and are largely built with the code created in this part.

Tens or Better Video Poker

One variation of Video Poker that I have seen is one that has tens or better for the even payout instead of the more common jacks or better. This is a very simple variation of video poker. All that needs to be done is some slight modification to the scoring and an adjustment of the payout table.

Deuces Wild Video Poker

This is video poker but with the deuces acting like a wild card. This means that a two can be any card the player desires. While on the surface this sounds like an easy change, a lot of work will be involved. The payout tables are drastically different when wild cards are present. If that wasn't enough, the scoring system needs a lot of changes in order to deal with the concept of wild cards.

Joker Video Poker

This variation adds a couple of jokers to the deck. Jokers act like wild cards in the same way the deuces do in the above game. If you have already created the above game, this variation is not that difficult as all that is needed is the addition of jokers to the deck.

Totally Wild Video Poker

If the above game two games have been created, this game is a breeze. Like the above, this game features wild cards. This time, however, the wild cards consist of both deuces and jokers! Obviously, with so many wild cards the payout tables need big adjustments.

Cribbage Square 4x4

This variation of cribbage squares omits the start card and just has a 4 by 4 grid of hands. Quite simple to do as all that is needed is some slight adjustment to the code to remove the start card.

Cribbage Square 5x5

This variation of cribbage squares replaces the start card with a start card for every row and column. There is a lot of work in getting the new layout to work. Also, instead of one common start card, there are now nine start cards that have to be dealt with.

Poker Square

Having the five by five grid for the above cribbage game should give readers of this book an obvious idea for combining the work done in chapter 12 with the work done in chapter 13. Poker Square is a square game that uses poker rules for scoring instead of cribbage rules. As you seen with the video poker games above, there are lots of options for how to score this game.