C++ (CS 2024) FINAL PROJECT WRITE-UP

Ope Oladipo, Joel Hoover, Neil Farnham, Junhan Zhao, Kyle Easop
ooo25
jah552
naf48
jz694
kfe6
Github: https://github.com/joelhoover/BlackJack

App Overview:

We built a basic BlackJack app that includes card and dealer graphics, win/lose logic and allows for multiple plays.

When the game is launched the “dealer” deals you two cards and himself one card. Each of the current hand scores are displayed above the images of the cards in your hand.

Once the initial setup has been taken care of the user gets the option to either select “hit” or “hold”. If the user selects hit then they are dealt another card and their score is updated. If the player’s score goes over 21 then the game ends and a “You Lose” image is displayed.

If the player hits the “Hold” button then the game continues to deal more cards to the dealer until the dealer’s score is higher than 16.

Once the dealer’s hand is 16 or higher the game checks for a winner. Whichever player has the higher score, but still less than 22, wins the game and an image is displayed showing the outcome. Then the next hand is started

Our game also detects if you get a blackjack and has a black duck as the image that pops up when that happens. .

Development Process:

C++:
Developing the game starts with writing behind the scenes C++ classes that hold: the deck, hands and cards. The deck class contains methods to deal a card, shuffle, and refill the deck. The card class has a face, a suit and a toString() function that returns the name of the card (ex: “TwoGiraffe”). The hand class contains a vector of card objects, an integer of the card total, and a boolean clarifying whether or not it has blackjack. It has a methods to add a card to the back of the vector, to check the value of the hand, and to clear out the hand. We found the Card & Hand classes we used in our HW to be beneficial when developing our own Card, Hand and Deck classes.

Adding C++ Code to Objective-C:

For this part of our project, we used Objective C++. This is source code that makes use of C++ classes and Objective C classes in one file. To do this, we change the file extension names from '.m'(Native Objective C extension) to '.mm'(Obj-C++ extensions) as well as '.h' to '.hh'. This was one of the most important things we did, as it was responsible for bridging our C++ classes and our IOS front end together. Declaring new class variables in Obj-C++ is just as easy as calling it as we do in C++ (e.g. "Card topCard = deck.dealCard(); "). Obj-C++ works nicely in terms of integrating and allowing C++ code to run in the same script. The downside though is it is Objective - C++ (and not swift or a more cohesive language to understand). We didn't explore the option of using swift as we all wanted to learn a bit about Objective C.

Animations:

The animations are relatively basic. The key interfaces images that are animated are the DuckWing, the back of the card, the front of the card and the WinLoseImage. When the dealer deals a card the duck wing merely rotates 90 degrees about the top of the duck's "shoulder" joint. At the same time that the wing is rotating the back of the card is translated across the screen to the middle. Then the wing rotates back to the starting point and the same "CardBack" image is enlarged to fill most of the screen. When the "CardBack" image is large enough it is returned to the top of the deck and in its place the "CardFront" image is programmatically instantiated with the correct card image. After the image is displayed it is delayed for a second and then shrinks to the appropriate position in the Player's Hand display box (these locations are hard-coded). The same is repeated for every subsequent card, but the destination of the card in the Player Hand display box is dependent on the size of the PlayerHand vector.

At the end of the game the program checks if the dealer of the player won the game and then loads the appropriate win, lose or draw image to a UIImageView then displays that in the front of the view. The cards in the hand then move off the screen and everything else is reset for the next game.

We were able to get various effects by using the build-in Objective-C animation blocks. These blocks allowed us to chain together animations by having one animation launch only when another finished and they also allowed us to add delays to certain animations occurring so that they looked more elegant.

Reproduction

To be able to reproduce our app, i've listed a few key steps below

1. Write your deck/hand/card classes in C++ and make sure they perform all the functions you intend them to.
2. Start an XCode project and set it to objective C (single page template)
3. Change the extension of the ViewController.m (and its accompanying header) from '.m' to '.mm' and '.h' to '.hh' respectively. This also applies to any other Objective-C files you plan to use C++ code in.
4. Call/define your C++ classes in you Obj-C++ just like you would in any C++ main function. Don't forget to define logic
5. Design your UI to integrate/apply your blackjack logic. This includes defining buttons that calls parts of your C++ code and labels(or images) that displays the result.