Programming Assignment 6 | Coursera

Bloco de Notas: Programação
Data de Criaç… 28-05-2016 21:48
URL: https://www.coursera.org/learn/game-programming/peer/efDxQ/programming-assignment-6

Instructions

Title: XNA Blockjuck

Assignment Description

In this assignment, you'll be playing a single hand of a game called Blockjuck. This game is something like Blackjack, but it most definitely isn't Blackjack – it's much simpler!

Honor Code

Please remember that you have agreed to the Honor Code, and your submission should be entirely yours. Our definition of plagiarism follows t standard literature: passing off someone else's work as your own, whether from your peer or Wikipedia.

Starting the Assignment

To start your work, download

ProgrammingAssignment6Materials.zip

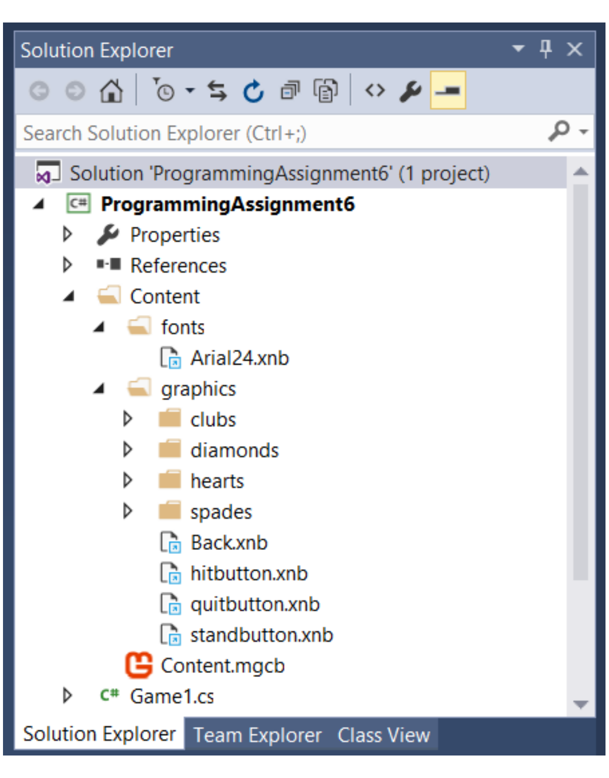
and extract the contents somewhere. The zip file contains a dll, source code, documentation, and content for the assignment.

The zip file contains documentation for the classes in the XnaCards namespace; you can use that documentation to figure out how to use thos classes.

Create a MonoGame Windows Project (or appropriate MonoGame project for your OS) called ProgrammingAssignment6. DON'T call the projec something else, it needs to be called ProgrammingAssignment6 for the next steps to work properly. Copy all the code from the Windows, Mac, code folder from the zip file into the appropriate place (you'll have to confirm replacing the template Game1.cs file the IDE generated for you) a all the files you just copied to the project (except Game1.cs, which is already in the project). These are all the classes you need as well as stub the methods you need. I even threw in some extra code snippets where I thought they might be helpful.

Copy the dll from the zip file into the same folder as your Game1.cs file. Add the dll as a reference for the project (detailed instructions for addi are included in Lab 10). Add a using statement for the XnaCards namespace to your Game1 class.

I've already used the Pipeline tool to build two separate content projects for you. Copy the XnaCardsContent and the BlockjuckContent folders the Windows, Mac, or Linux content folder from the zip file into the same folder as your Solution file. Add the content from BOTH content projec your project. Be sure to EXACTLY mirror the folder structure that appears in the content projects; when you're done adding your content, your should look like the image below (all the "suit" folders hold all the cards for each suit, but that doesn't fit in the image!).



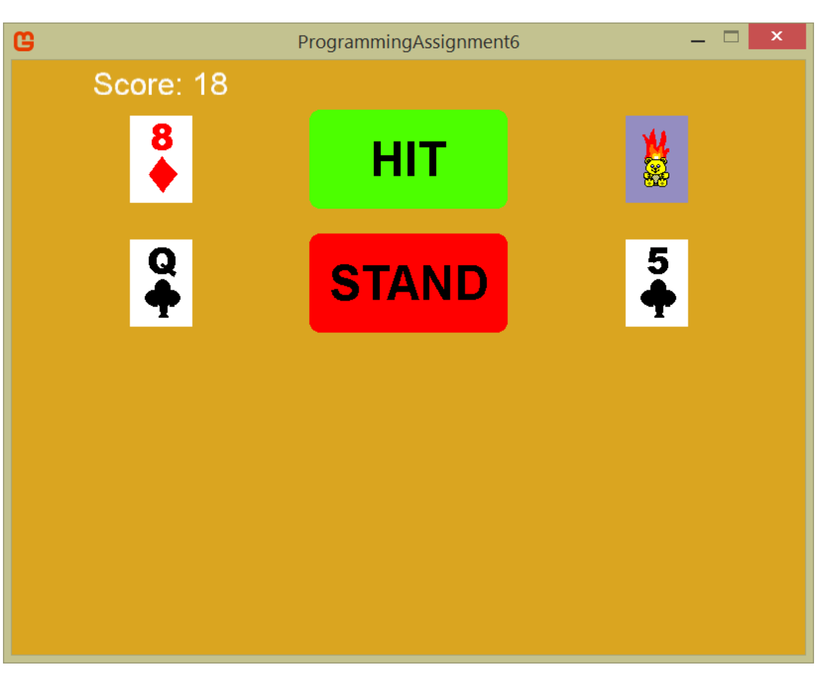
Game Behavior

Here's how your game needs to work. The game should start with two hands: one for the player and one for the dealer. You should deal two ca from a Deck into each hand (dealing like in Las Vegas, so DON'T just give the player 2 cards then give the dealer 2 cards!). The player doesn to see the deal happen. Both of the player's cards should be face up (so the player can see what they have) and only the second dealer card be face up (so the player doesn't know what the dealer has for their first card). You'll have to set the X and Y properties for the center of each get a reasonable display of all the cards, with the player cards on the left and the dealer cards on the right.

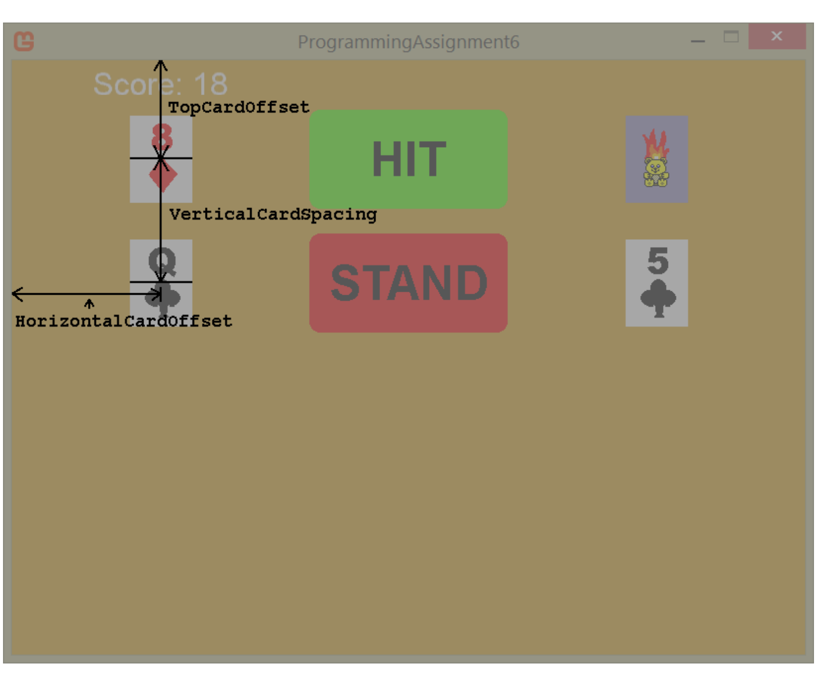
You'll calculate and display the score above the player's hand. You should just call the GetBlockjuckScore method that I've provided in the Ga class to calculate the score for the hand, but you should use the Message class I provided to actually display the score.

Finally, you'll also display two menu buttons to the right of the player cards -- one for the player to Hit (take another card) and one for the play Stand (don't take another card). You should definitely use the MenuButton class I've provided in the project for these menu buttons.

Here's a screen shot of my solution at the start of the game (you don't have to match the spacing exactly, but it should look approximately like t screen shot):



Here's a hint about how you can use some of the constants I've provided to place the cards properly (placing the menu buttons properly is very similar):



You should carefully read the Changing Game State section in the Additional Assignment Information below to see how the game transitions be the possible game states.

The game continues until either both the player and the dealer bust (have a hand with a score higher than MaxHandPoints), only the player bu the dealer busts, or both the player and the dealer decide to stand in a particular "turn". At that point, the dealer's hole card (the card that's fa down) is flipped over, the dealer's score is displayed, the Hit and Stand buttons are removed, and a Quit button is displayed. The player clicks button to quit the game.

Here's a screen shot of my solution at the end of the game:

Requirements

Your solution to this problem must:

- Meet the problem specification (e.g., do what it's supposed to)
- Comply with the Course Coding Standards

Review criteria

After submitting your work, you'll get the chance to grade the work of five of your peers. Your own work will also be assessed by your peers, fro you'll get your grade. Since you've worked hard on your submission and would like your peers to do a good job of assessing your submission, take your time and do a good job of assessing the submissions you've been assigned. Completing your peer review counts for 20% of your gra this assignment.

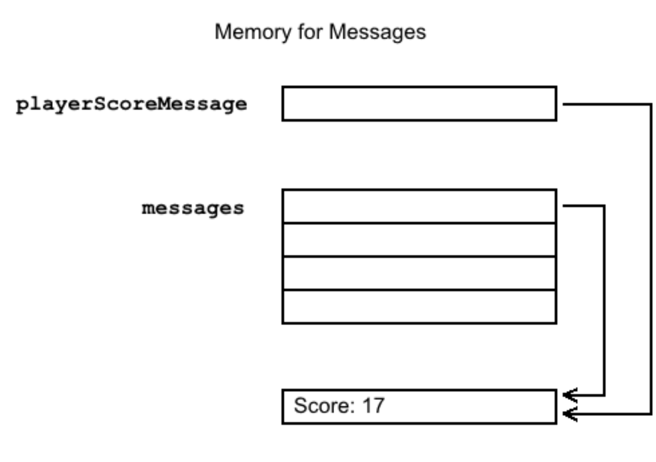
Additional Assignment Information

Changing Game State

Finite State Machines (FSMs) are a very helpful way to specify the behavior of software (and other systems). An FSM consists of a set of state we move between the states along transitions. That's exactly how you'll transition between the possible game states in this game. I've provided picture of the FSM following the description.

The game starts in the WaitingForPlayer state. If the player decides to hit (by clicking the Hit menu button), the game transitions to the PlayerH state. In the PlayerHitting state, you give the player another card, calculate and display their new score, then transition to the WaitingForDeale the player decides to stand (by clicking the Stand menu button) in the WaitingForPlayer state the game transitions to the WaitingForDealer sta

Changing the display for the new player score has caused students some confusion in the past. You do NOT have to create a new Message o each time the player score changes, you can simply change the Text property for the player score message. Why does that work? Look at the below.



In the code I provided to you in the Game1 LoadContent method, I created the player score message and also added it to the messages list. T means that the playerScoreMessage field and the messages[0] element both reference ("point to") the same Message object in memory. If I ch the Text property of the Message object by changing the playerScoreMessage field, the messages[0] element also "gets" that change because referring to the same object in memory.

In the WaitingForDealer state, if the dealer decides to hit (they have to hit on 16 or fewer points), the game transitions to the DealerHitting stat DealerHitting state you give the dealer another card then transition to the CheckingHandOver state. If the dealer decides to stand (they have t on 17 or more points) in the WaitingForDealer state the game transitions to the CheckingHandOver state.

In the CheckingHandOver state, the game checks to see if the hand is over. Hands are over when:

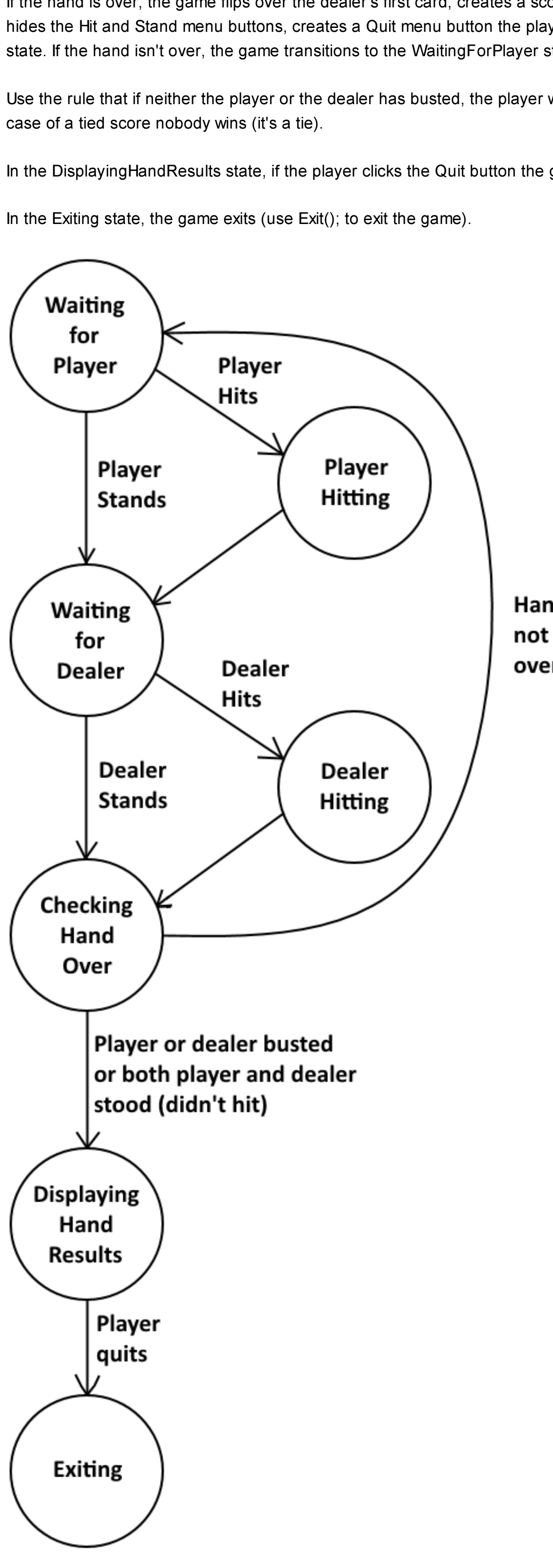
- the player or the dealer has busted (gone over MaxHandPoints in their hand), or
- both the player and the dealer decided to stand in a turn

If the hand is over, the game flips over the dealer's first card, creates a score message for the dealer's score, creates an appropriate winner m hides the Hit and Stand menu buttons, creates a Quit menu button the player can use to exit the game, then transitions to the DisplayingHandR state. If the hand isn't over, the game transitions to the WaitingForPlayer state.

Use the rule that if neither the player or the dealer has busted, the player wins if their score is highest, the dealer wins if their score is highest, case of a tied score nobody wins (it's a tie).

In the DisplayingHandResults state, if the player clicks the Quit button the game transitions to the Exiting state.

In the Exiting state, the game exits (use Exit(); to exit the game).



Helpful Hints

Add your functionality to the game a little bit at a time. The best way to develop a game is a small piece at a time.

I provided all the fields you'll need for your solution at the top of the Game1 class, including lots of constants you can use to properly place the in the game so they display properly.

Be sure to shuffle the deck before dealing the cards!

Managing the MenuButton and Message objects will be much easier if you maintain lists of them. By doing it this way, you can easily add and r menu buttons and messages as they become active or inactive.

Your Update method should use a switch statement or if/else if statements to do the appropriate game processing based on the current game

The menu buttons in the game should only be updated in certain states, specifically in the WaitingForPlayer and DisplayingHandResults states sure you do this properly.

My Implementation Steps

You can download my implementation steps for this assignment from

Programming Assignment 6 Implementation Steps.pdf

For the Strivers (or Lunatics) Only

The only source code you submit for this programming assignment is the Game1 class. In Week 9, we cover how to design and implement you classes. That means that, if you want to, you can design and implement all your own classes for the cards, deck, menu buttons, and so on. Bec peer reviewers will be looking at how you interact with those classes, you should leave the method names and parameters for the methods tha Game1 class interacts with the same as I have in the provided classes, but other than that you have full freedom to build those other classes h you want to. If you've been itching to do some "heavier lifting", here's your chance!