

H i g h R o l l e r s

C # G a m e

B y D e r e k S n e d d o n

Part 1: Game Design and Setup

“To roll or not to roll-that is the question,” -William Shakespeare if he was a gambler.

MAKE SURE YOU READ THIS, DON'T SKIP STRAIGHT TO THE CODE

(If you have any questions in this guide at all, feel free to come talk to me in class, email me at

19sneddderb@washk12.org or send me a message on Discord at Jexington#8042)

Are you excited for more C# fellow students?

No?

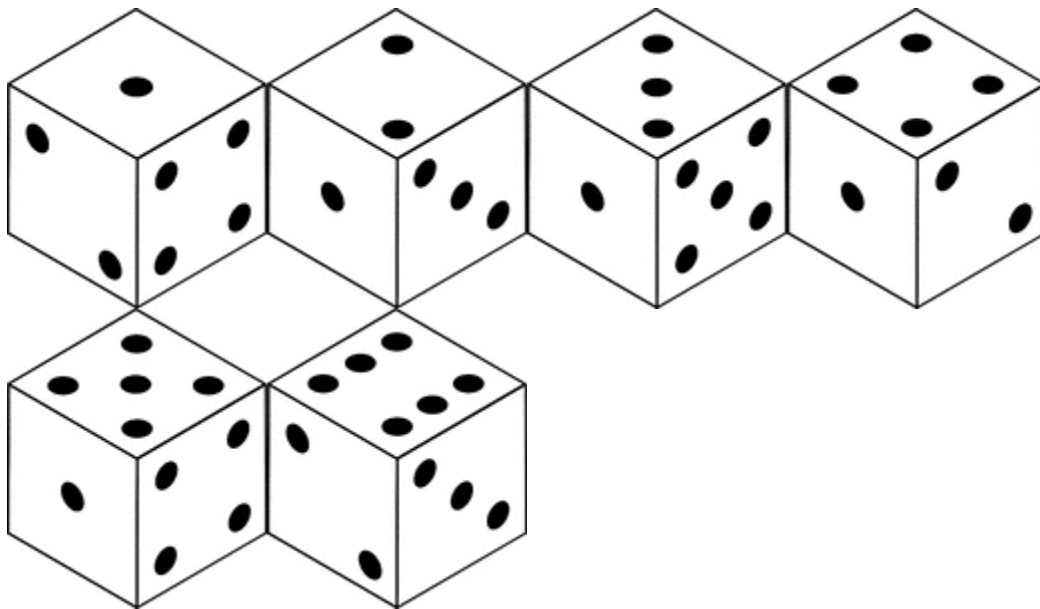
Well, you should be! It is personally one of my favorite languages, I have always had so much fun with it creating games, apps, calculators, you name it! In addition, knowing Visual Studio and C# can make you a lot of money! However, so can the game you are about to make, High Rollers!



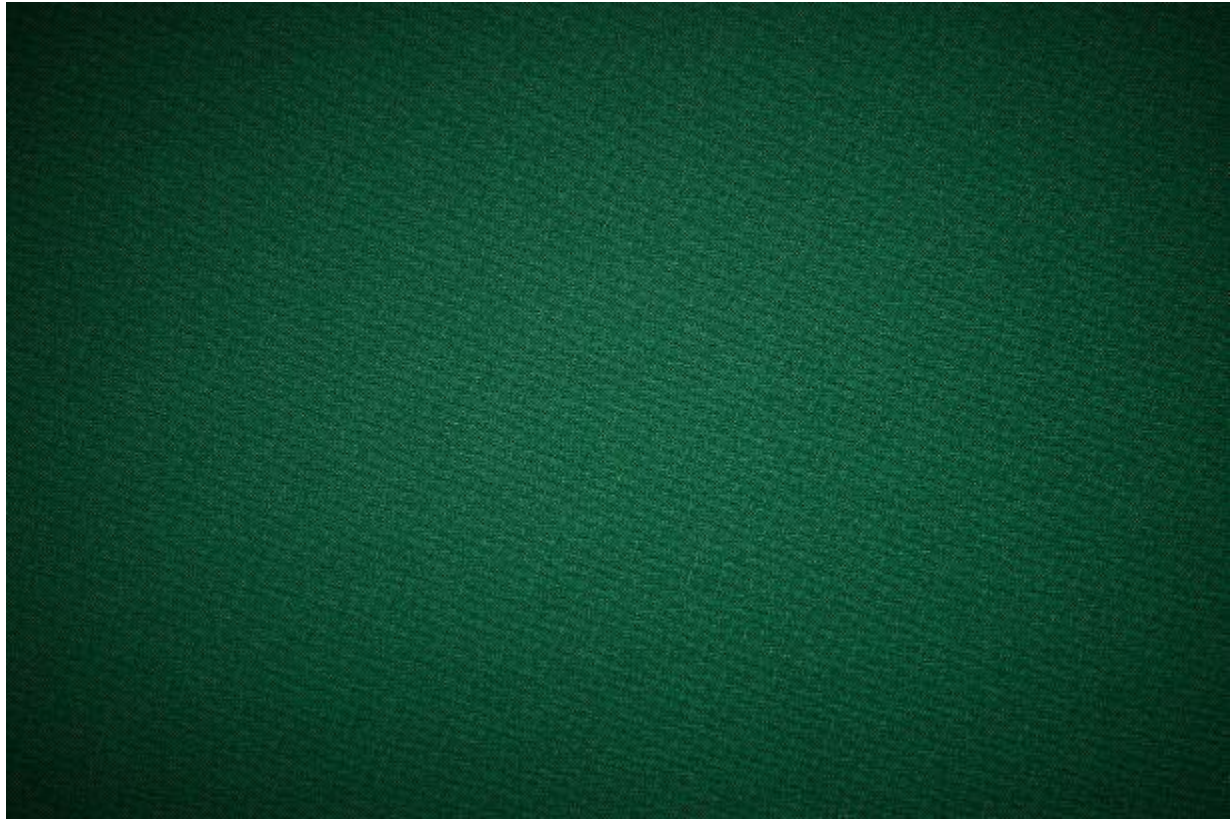
Disclaimer: Gambling is not allowed at school so, please do not use this game to actually bet cash.

High Rollers is a game where two players (or a player and an AI if I have enough time to code one for this curriculum) make bets and roll dice in attempt to make money! This game is all

about risk and reward, so place your bets wisely! (Wow, I sound cheesy) But, if a player runs out of money they lose. On the contrary, if the player makes at least \$20000 they win! An important thing to remember is that there are two victory conditions, one running out of money and one getting enough money to reach the goal.

[illegible]

After that, we need a game table image. Make sure you download each image and save it where you can find it. Basic logic stuffs. (Same rule applies with using your own images; refer to “really” list above)



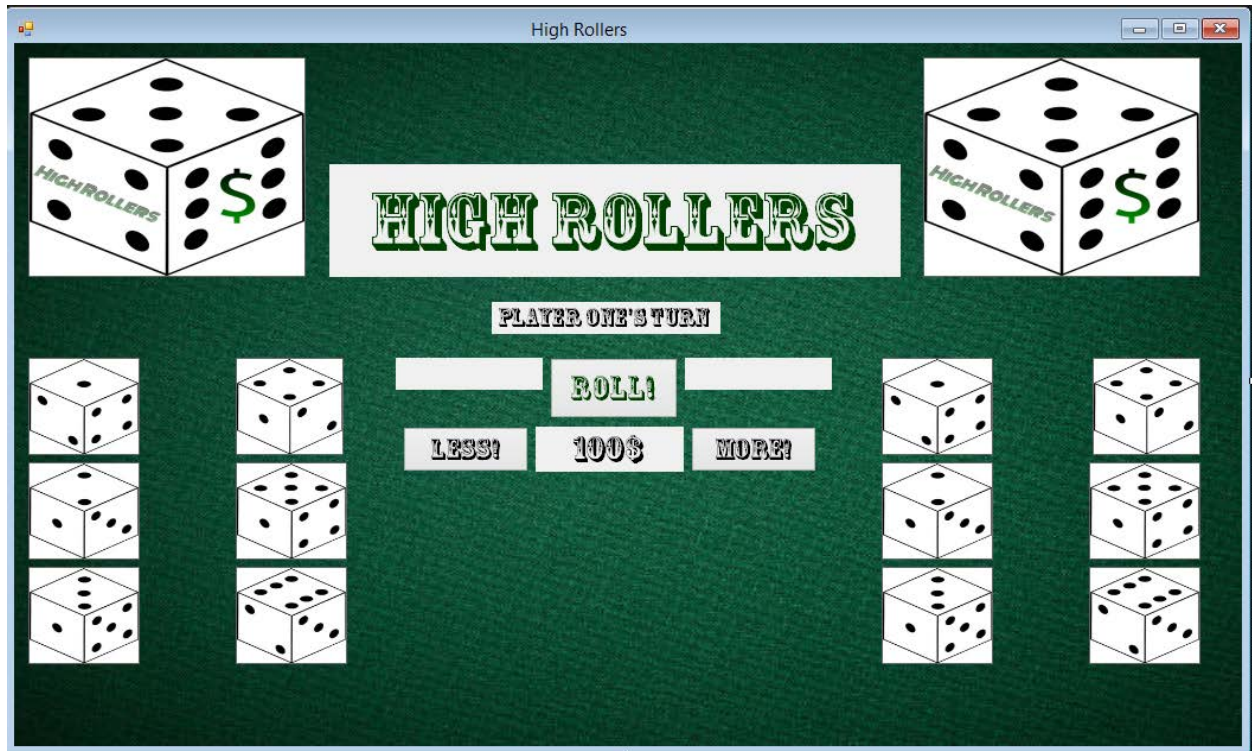
Nice table you got there. Looks good, eh?

Now, we need a game logo. I even made it myself! (It's true) You can thank me later.

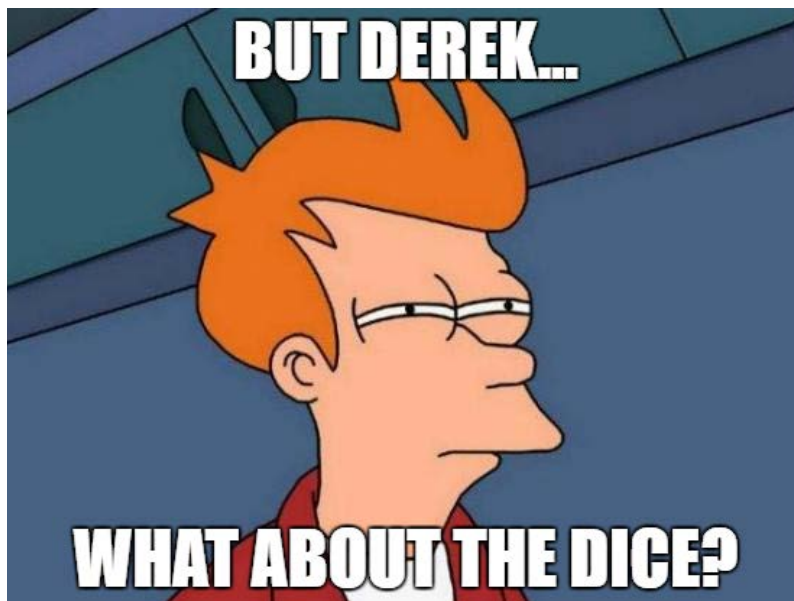
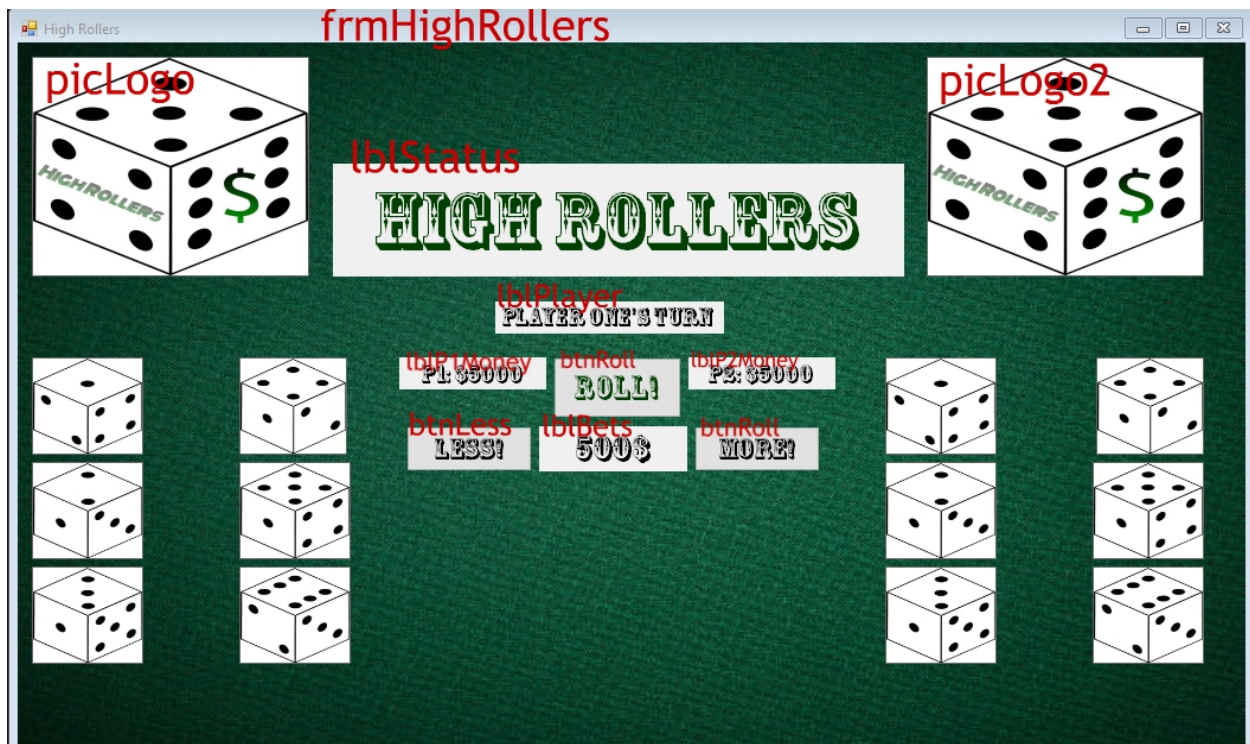


Man he is good at weird Photoshopped logos!

Now, downloading images that I have provided for is fun and all, but we need to work on putting it all together in visual studio. Here is my design. (Note that when the program is started that the dice are invisible).



I am not picky on how you design it, even if you don't want logos or a background. But, make a design that allows it to work. The following are the names of each object, make sure you get them down on the properties of the object or the code will not work.



That I will simply explain. I named my first die, the one on the far top left, picDice11, because it was a picture box, it is a dice, the first number (the first 1 in picDice) means the number on the top of the dice and the second 1 on picDice means the category it is in. The other

dice with a 1 is named picDice12 because it has a one on it and sits in the second category. I am sure it will make sense later. Here is all of the names I have used if you are still confused.

Names:

picDice11

picDice21

picDice31

picDice41

picDice51

picDice61

picDice12

picDice22

picDice32

picDice42

picDice52

picDice62

Let's get to properties. Before you do anything, make sure autosize is set to false on all objects, long story short; sometimes stuff does not work without them.

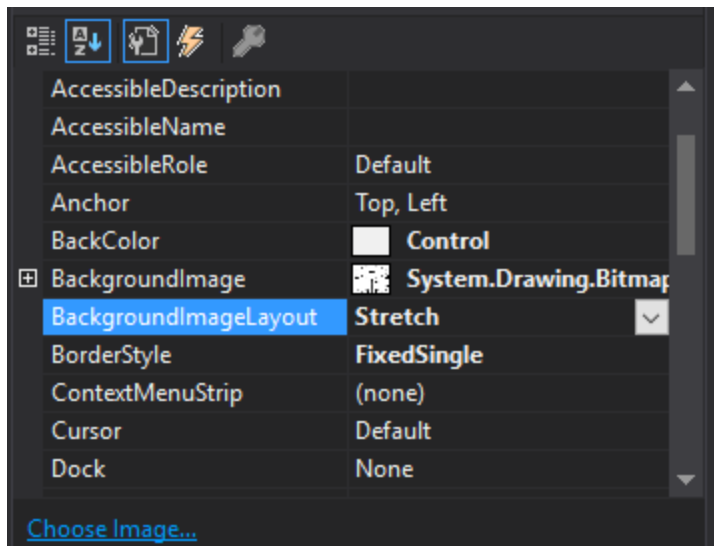
Disclaimer: Derek was lazy and did not want to do all the properties that made the design, he only wanted to list the properties that mattered in the actual program.



Because I am feeling generous, here is how to stretch your images so your background and the images in your image boxes do not look like absolute garbage.

First, click on the object with the image you want to stretch.

Then go to BackgroundImageLayout like so.



Then click on the dropdown and choose stretch.

WOW You did it! Hat diggity dog! Now your images look normal!

Can I get a thank you?

No.

Okay let's move on.

Next up, make sure all of the picDice objects have their visibility set to false, that way, when you roll the dice, we can make them visible.



Part 2: Coding

First off, we need to make sure we declare all of our variables. Open the code window and declare all of the following variables. Make sure to do it under the public partial class frmHighRollers (that is generated automatically)

```
public partial class frmHighRollers : Form
{
    public Random random = new Random();
    const int MONEY_GOAL = 20000;
    int roll1 = 0;
    int roll2 = 0;
    int player1Money = 5000;
    int player2Money = 5000;
    int betMoney = 500;
    int turnCounter = 1;
    int rollTotal = 0;
}
```

Now, allow me to explain each variable and what its purpose is in the program.

const int MONEY_GOAL: This the money goal, being a constant, I format it in all caps so I know that it is a constant. A constant is a variable that cannot be altered in a program no matter what, for my sake, I never want MONEY_GOAL to change, so I plan to keep it that way.

int roll1: I am setting it to 0 as a default value so it doesn't give me any errors or trigger any unwanted results. This variable will soon be randomized between 1 and 6, depending on what roll1 generates as a random number, it will display a dice corresponding to on the left side of the screen.

int roll2: Just like roll1 but applies to the other set of dice.

int player1Money: It is the amount of money the first player has and will increase and decrease based on how much money the player losses or gains. Starts at 5000.

int player2Money: Same thing as player1Money, but for the second player.

int betMoney: How much money is going to bet for a roll.

int turnCounter: Whose turn it is, always starts on the first player.

int rollTotal: roll1 and roll2 added together to determine how much money the player wins or losses.

Moving on.

Let's code our btnRoll button. Double click on it and it should bring you to the code window. Type the following code in.

```
private void btnRoll_Click(object sender, EventArgs e)
{
    Random rnd = new Random();
    roll1 = rnd.Next(1, 6);
    roll2 = rnd.Next(1, 6);
    rollTotal = roll1 + roll2;

    picDice11.Visible = false;
    picDice21.Visible = false;
    picDice31.Visible = false;
    picDice41.Visible = false;
    picDice51.Visible = false;
    picDice61.Visible = false;

    picDice12.Visible = false;
    picDice22.Visible = false;
    picDice32.Visible = false;
    picDice42.Visible = false;
    picDice52.Visible = false;
    picDice62.Visible = false;

    switch (roll1)
    {
        case 1:
            picDice11.Visible = true;
            break;
        case 2:
            picDice21.Visible = true;
            break;
        case 3:
            picDice31.Visible = true;
            break;
        case 4:
            picDice41.Visible = true;
            break;
        case 5:
            picDice51.Visible = true;
            break;
    }
}
```

(There is little bit of overlap, the next screenshot starts at case 5)

```

        case 5:
            picDice51.Visible = true;
            break;
        case 6:
            picDice61.Visible = true;
            break;
    }

    switch (roll2)
    {
        case 1:
            picDice12.Visible = true;
            break;
        case 2:
            picDice22.Visible = true;
            break;
        case 3:
            picDice32.Visible = true;
            break;
        case 4:
            picDice42.Visible = true;
            break;
        case 5:
            picDice52.Visible = true;
            break;
        case 6:
            picDice62.Visible = true;

```

Again, just like I had said, overlap between screen shots. But, once you type in `moneyManage();` it might give you an error, we are going to create that function later, so bear with me, if you want to, take it out and bring it back in later.

```

        case 6:
            picDice62.Visible = true;
            break;
    }

    moneyManage();
    if (turnCounter == 1)
    {
        turnCounter += 1;
        lblPlayer.Text = "Player Two's Turn";
    }
    else
    {
        turnCounter -= 1;
        lblPlayer.Text = "Player One's Turn";
    }
}

```

I know you are looking at my code and your like, what the fetch is the switch case stuff?! Well, how about we run from the top and go down; I will get to switch case stuff later.

The `Random rnd = new Random();` allows me to import a Random function from Visual Studio into my code, allowing me to generate random numbers. The next two lines roll the dice, generating a number between 1 and 6. The one after adds those two numbers together, giving us a number that will be used to calculate how much money a player makes or loses.

The stuff about `picDice11.Visible = false;` is so all the dice that may have been revealed previously from all other rolls are reset when the player rolls again.

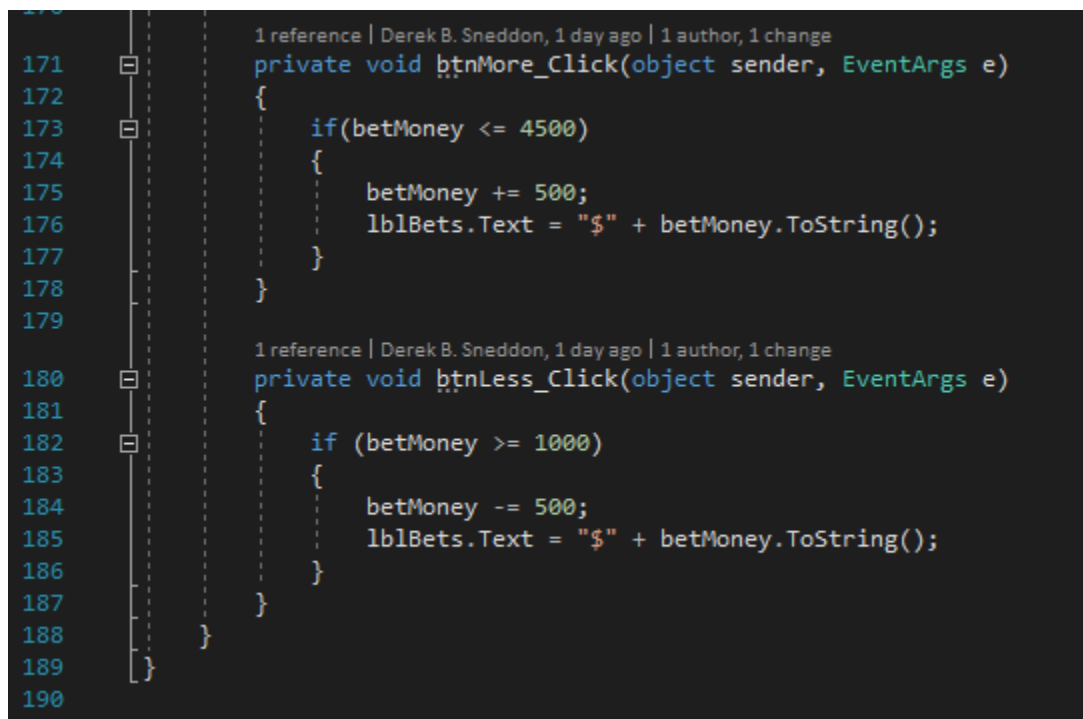
Now, you may ask, WHAT IS A SWITCH CASE?! This is a bit more of an advanced technique that is a bit less common than your standard if statement.



They are really good at making your code cleaner and more efficient, also they take less time to code. They function just like an if statement, evaluating a variable and executing a block

of code depending on what that evaluation returns. But, switch case statements are really good for evaluating simple integer algorithms. I love them, heck if I wasn't already going to prom with someone, I would probably take a switch case statement there. (That was humor, *canned laughter*) The moneyManage(); function we will create later. After that is what decides whose turn it is, then lets the player know through lblPlayer.Text = "Player One's Turn"; and lblPlayer.Text = "Player Two's Turn";

Now, before we make the moneyManage() function, we need to code our btnLess and btnMore buttons. This will allow the player to increase or decrease their bet amounts before they roll, adding strategy to the game, but we need limits at \$500 and \$5000. Type in the following:



```
171 | 1 reference | Derek B. Sneddon, 1 day ago | 1 author, 1 change
172 | private void btnMore_Click(object sender, EventArgs e)
173 | {
174 |     if(betMoney <= 4500)
175 |     {
176 |         betMoney += 500;
177 |         lblBets.Text = "$" + betMoney.ToString();
178 |     }
179 | }

180 | 1 reference | Derek B. Sneddon, 1 day ago | 1 author, 1 change
181 | private void btnLess_Click(object sender, EventArgs e)
182 | {
183 |     if (betMoney >= 1000)
184 |     {
185 |         betMoney -= 500;
186 |         lblBets.Text = "$" + betMoney.ToString();
187 |     }
188 | }
189 | }
190 | }
```

I think this is pretty self-explanatory.

Now we need to do the moneyManage function, I will put my screen shots in here now.

1 reference | Derek B. Sneddon, 2 days ago | 1 author, 1 change

```
public void moneyManage()
{
    if(turnCounter == 1)
    {
        if(rollTotal % 2 == 0 && rollTotal != 2)
        {
            player1Money = player1Money - betMoney;
            lblStatus.Text = "Bad Roll!";
        }
        else if(rollTotal % 2 != 0 && rollTotal != 7)
        {
            player1Money = player1Money + betMoney;
            lblStatus.Text = "Nice Roll!";
        }
        else if(rollTotal == 2)
        {
            player1Money = player1Money - (betMoney * 5);
            lblStatus.Text = "Ouch!!";
        }
        else if(rollTotal == 7)
        {
            player1Money = player1Money + (betMoney * 2);
            lblStatus.Text = "Wow!";
        }
        lblP1Money.Text = "P1: $" + player1Money;
    }

    if (turnCounter == 2)
    {
        if (rollTotal % 2 == 0 && rollTotal != 2)
        {
            player2Money = player2Money - betMoney;
            lblStatus.Text = "Bad Roll!";
        }
        else if (rollTotal % 2 != 0 && rollTotal != 7)
        {
            player2Money = player2Money + betMoney;
        }
    }
}
```

Remember, I have overlap between screenshots. So double-check everything.

```

    else if(rollTotal % 2 != 0 && rollTotal != 7)
    {
        player1Money = player1Money + betMoney;
        lblStatus.Text = "Nice Roll!";
    }
    else if(rollTotal == 2)
    {
        player1Money = player1Money - (betMoney * 5);
        lblStatus.Text = "Ouch!!";
    }
    else if(rollTotal == 7)
    {
        player1Money = player1Money + (betMoney * 2);
        lblStatus.Text = "Wow!";
    }
    lblP1Money.Text = "P1: $" + player1Money;
}

if (turnCounter == 2)
{
    if (rollTotal % 2 == 0 && rollTotal != 2)
    {
        player2Money = player2Money - betMoney;
        lblStatus.Text = "Bad Roll!";
    }
    else if (rollTotal % 2 != 0 && rollTotal != 7)
    {
        player2Money = player2Money + betMoney;
        lblStatus.Text = "Nice Roll!";
    }
    else if (rollTotal == 2)
    {
        player2Money = player2Money - (betMoney * 4);
        lblStatus.Text = "Ouch!!";
    }
    else if (rollTotal == 7)
    {
        player2Money = player2Money + (betMoney * 3);
        lblStatus.Text = "Wow!";
    }
    lblP2Money.Text = "P2: $" + player2Money;
}

if (player1Money >= MONEY_GOAL || player2Money <= 0)
{
    lblStatus.Text = "P1 Wins!";
    btnRoll.Enabled = false;
}
else if (player2Money >= MONEY_GOAL || player1Money <= 0)
{
    lblStatus.Text = "P2 Wins!";
    btnRoll.Enabled = false;
}

```

```

    if (player1Money >= MONEY_GOAL || player2Money <= 0)
    {
        lblStatus.Text = "P1 Wins!";
        btnRoll.Enabled = false;
    }
    else if (player2Money >= MONEY_GOAL || player1Money <= 0)
    {
        lblStatus.Text = "P2 Wins!";
        btnRoll.Enabled = false;
    }
}

```

Let me walk through what it does for you. First it takes the rollTotal and checks if it is even and not a two (lose bet money), odd and not a seven (gain bet money), a two (lose 5 * betMoney) and a seven (gain 2 * betMoney). Then it does the block of code associated with whatever rollTotal was. After that, it checks if either player has won or lost. If you left moneyManage() out of the btnRoll object, you might want to put it back in. I will even show where it is.

```

88         break;
89         case 6:
90             picDice62.Visible = true;
91             break;
92     }
93
94     moneyManage();
95     if (turnCounter == 1)
96     {
97         turnCounter += 1;
98         lblPlayer.Text = "Player Two's Turn";
99     }
00     else
01     {
02         turnCounter -= 1;
03         lblPlayer.Text = "Player One's Turn";
04     }
05 }

```

That should be all the code!

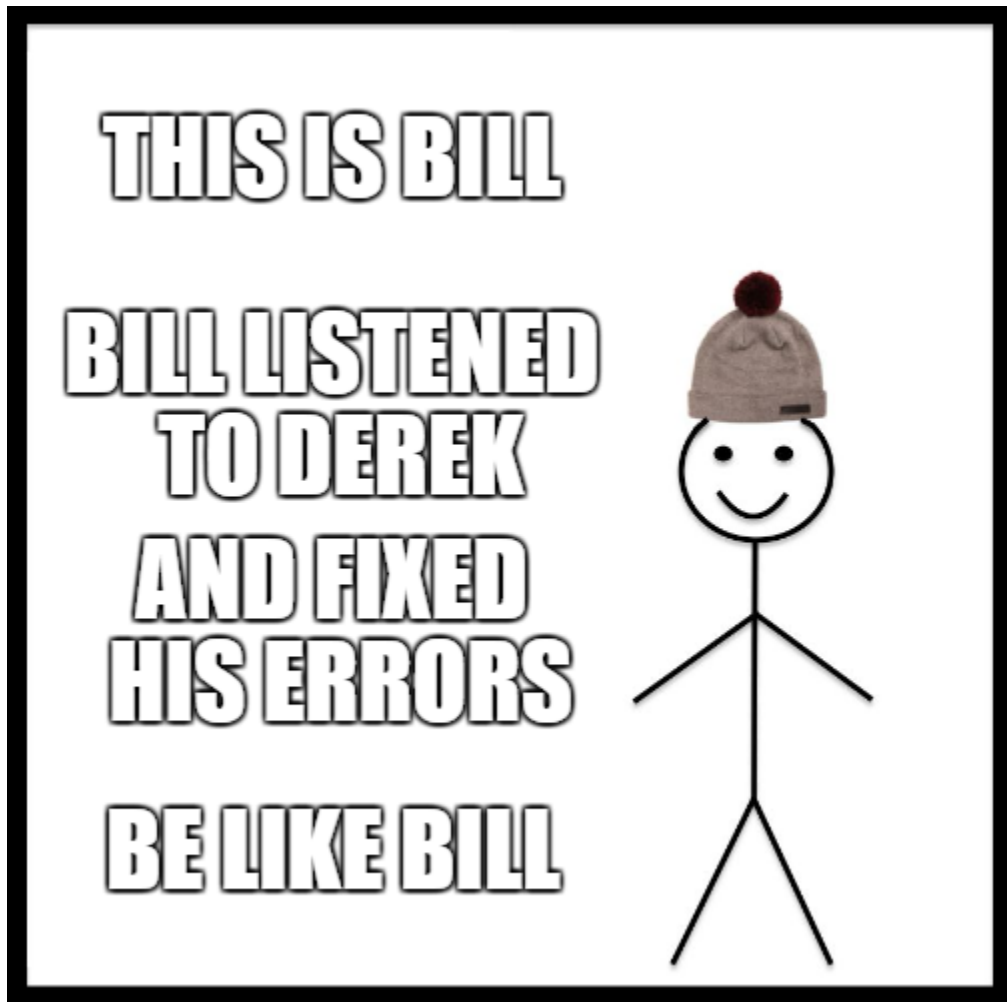


Part 3: Debugging

If you finish part 1 and part 2 and have a ton errors, you might look like this:



You can now go this part here and look like this:



My tips for debugging!

1. Check brackets, remember each one has a mate, and you choose their mate.

Programming is totalitarianistic, pick their marriages just like China does.

2. Retype lines of code, you might fix an error you didn't catch.
3. Check with your buddies.
4. If all else fails, check with me!

Now that you have reached the end of this part, you know what time it is:



All right, thanks for giving High Rollers a shot and keep programming!

(Also, please follow me on gitHub, my username is Jexington)