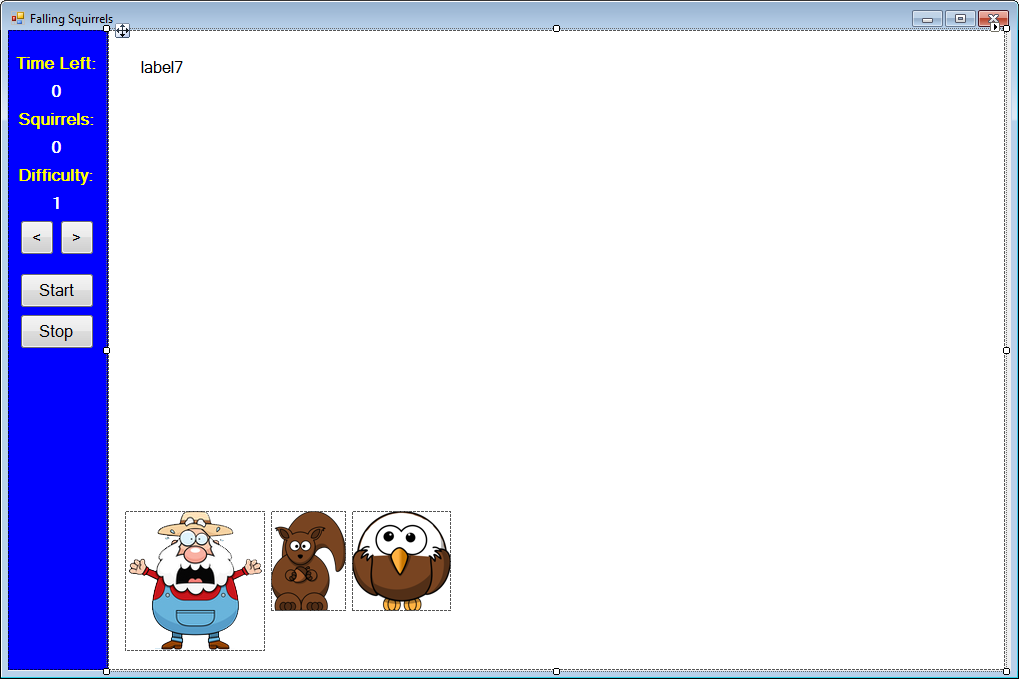
**Falling Squirrels**

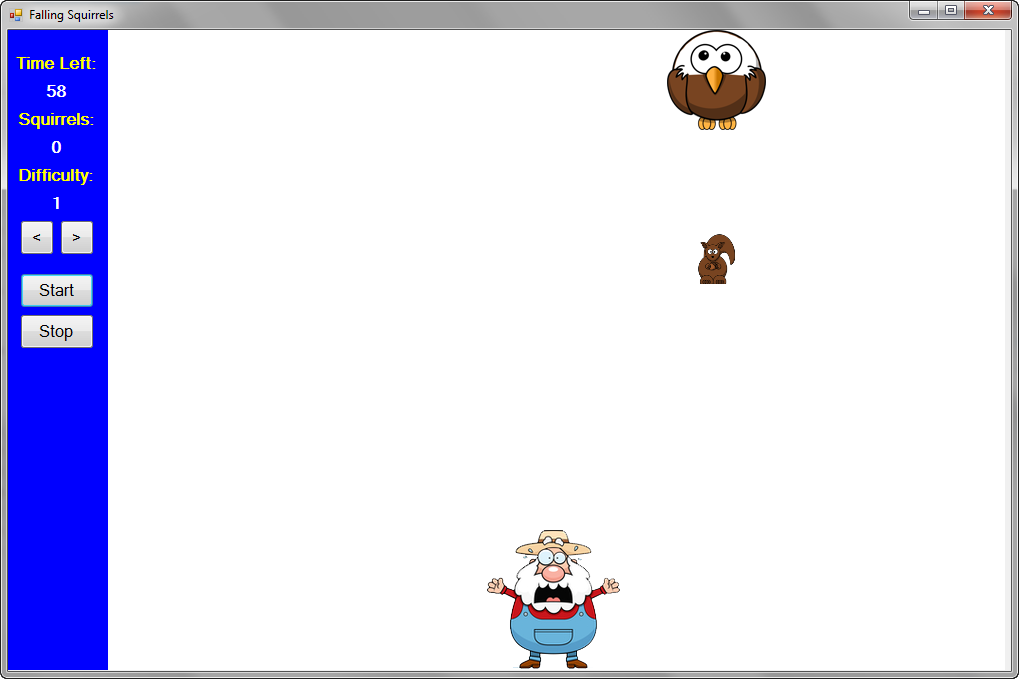
Flying eagles drop squirrels from the sky. You position the farmer under the falling squirrels to catch them.



Labels are used for instructions and to display game information (time remaining, squirrels caught, difficulty). Two buttons are used to change the difficulty, one to start the game and one to stop. Picture box controls hold images for Famer, the squirrel and the eagle.

Run the program (press <**F5**>). The game will appear in its ‘initial’ state. A game description and program instructions are given. At this point, you can click the **Start** button to start the game, click arrow buttons to change the game difficulty or click **Stop** to stop the program.

Change the difficulty by clicking the little arrows under **Difficulty**. Click the **Start** button to start playing. The program instructions disappear. An eagle appears at the top of the screen holding a squirrel. The Farmer is at the bottom of the screen. The squirrel will start falling. This is the game’s ‘playing’ state:



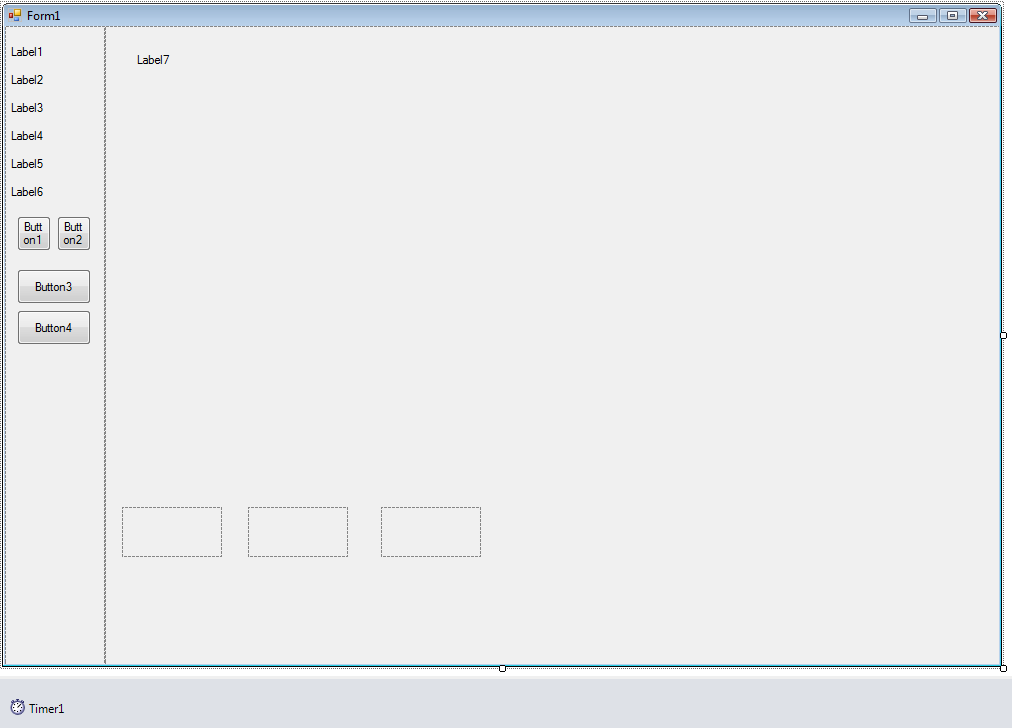
You move the farmer under the falling squirrel to catch it. You can move the farmer using the keypad arrow keys (**NumLock** must be on) or by holding down the left or right mouse button. When the squirrel reaches the farmer, you score a point (under **Squirrels**). If the squirrel passes without catching it, you score nothing. Either way, another eagle appears dropping another squirrel.

The game continues for 60 seconds. The remaining time is shown under **Time** **Left**. You are given your final score at this point.

Next, if you want, you can change the difficulty – higher difficulties give you faster dropping squirrel. You can also click **Start** to race again or click **Stop** to stop the program. Continue playing the game to understand its operation.

**Form Design**

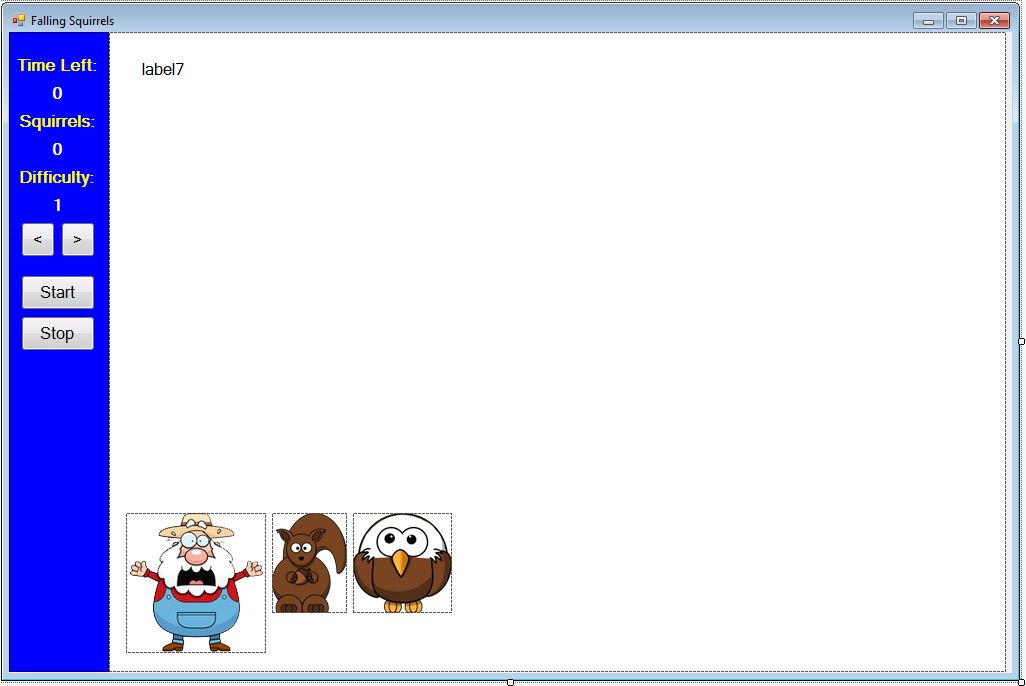
Put two panel controls on the form (one 100 pixels wide on the left, one 900 pixels wide on the right). Make both panels around 600 pixels high. Put six labels and four buttons on the left panel. Put a label control and three picture box controls on the right panel. Add a timer control. The form should look like this:



The labels on the left are used to display game information. Two buttons are used to change the difficulty, one to start the game, and one to stop the program. The label control in the second panel is used to display game instructions. The picture boxes hold the images of the farmer, a squirrel and the eagle. The timer controls game play.

Set the control properties using the properties window:

When done, the form should look like this:



We now begin writing code, we will write the code in several steps. As a first step, we will write the code that starts the program and establishes its ‘initial’ state. Then, we look at how to go to ‘playing’ state (positioning all the graphics elements) following clicking of the **Start** button. We then write code to move the squirrel down the screen, move the farmer (using mouse clicks or keyboard presses), and check for caught squirrels. During the code development process, recognize you may modify a particular procedure several times before arriving at the finished product.

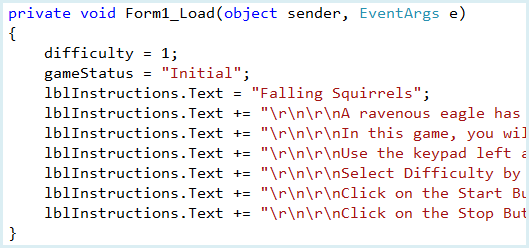
**Code Design – Initial State (Instructions)**

Any time we start a program, there are certain initializations that must take place. When the program first begins, we want to display the game instructions. We also initialize the difficulty (**Difficulty**) and the game status (**GameStatus**). We set up the game screen and display the instructions first. These initializations are in the form **Load** procedure.

We declare two variables (**difficulty** and **gameStatus**). Do this in the general declarations area:



Add this code to the **Form Load** event procedure:

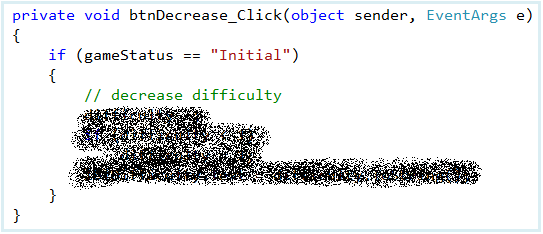


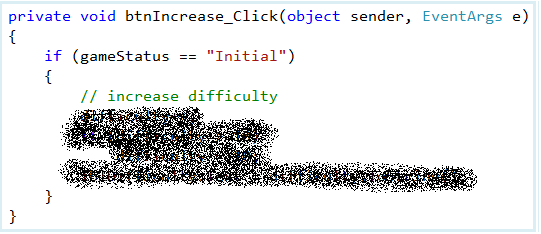
As mentioned, we set **difficulty** to **1** and the **gameStatus** to “**Initial**”. Next, using a series of statements, we write out the game instructions in **lblInstructions**.

**Save** and **Run** the program to see:

**Code Design – Select Difficulty**

The **difficulty** variable is used to determine how fast the squirrel falls (the higher the difficulty, the faster it falls). To change the difficulty, a user clicks on either of the two arrow buttons. The code to do this goes in the **btnDecrease** and **btnIncrease** **Click** event procedures:





The code is straightforward. The difficulty can only be changed when the game is in “**Initial**” state. If the **<** button is clicked, we decrease the difficulty by one (minimum of 1). If the **>** button is clicked, we increase the difficulty (maximum of 10). The new value for the difficulty is then displayed in **lblDifficulty**.

**Save** and **Run** the program. Make sure the arrow buttons work – make sure the difficulty is always between 1 and 10.

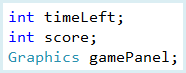
**Code Design – Position the Farmer**

Once the difficulty is set, the next step is to click the **Start** button to allow the game to begin. Hence, we need code to move from the “**Initial**” state to what we’ll call “**Playing**” state. When the **Start** button is clicked in “**Initial**” state, we need to take the following steps:

* Change **gameStatus** to “**Playing**”
* Clear the screen.
* Initialize score and time left.
* Place the Farmer at bottom of screen
* Randomly place the eagle and squirrel at the top of the screen
* Start dropping squirells

There is a lot to do here, so we will take it in steps. First, we’ll initialize variables and position the farmer at the bottom of the screen.

Two variables are used in game control. **timeLeft** is an integer value telling us how much time is left in the game (in milliseconds). We choose to let a game last 60 seconds. **score** tells us how many squirrels have been caught by the farmer. **gamePanel** is the graphics region we play the game in. Add these general variable declarations:



The image stored in **picFarmer** is used for the farmer:

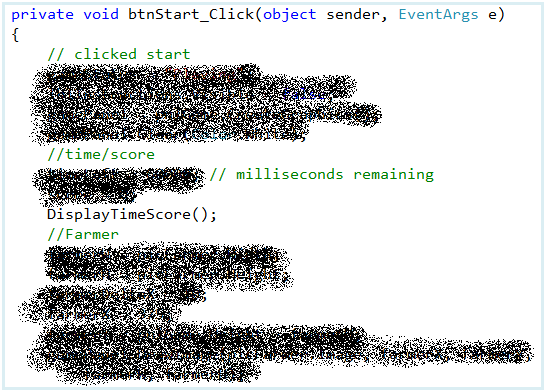


**picFarmer**

In the code, we determine both the width (**farmerW**) and height of this graphic (**farmerH**). The Farmer is positioned at the bottom of the screen, near the middle. The variables **farmerX** and **farmerY** tell us where the farmer is. **farmerDeltaX** is used to tell us how much to move the farmer with each button click or mouse press. All of these variables will be helpful when checking to see if the farmer catches or misses falling squirrels. Add these new variable declarations:



Add this code to the **btnStart** **Click** event procedure:

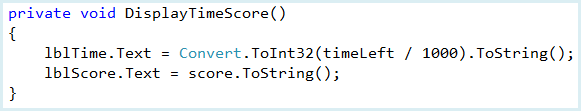


You should be able to identify each step in this new code. We clear the area and set variables. We establish the variables for Farmer and position him at the bottom.

Notice this line of code:

DisplayTimeScore();

This line calls a procedure that updates the displayed value for **timeLeft** and **score**. Add this procedure at the end of your program:



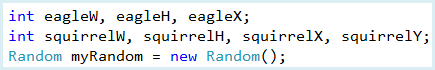
Here, we set the appropriate values in **lblTime** and **lblScore**. Note how we convert milliseconds to seconds.

**Save** and **Run** the program. Notice the displayed values for **Time Left** and **Squirrels**. Notice the farmer down there waiting for squirrels to fall. Let’s get an eagle and squirrels on the screen.

**Code Design – Position Squirrel and the Eagle**

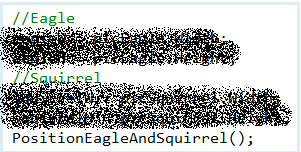
The last initialization step is to randomly place the eagle and the squirrel at the top of the screen. The images stored in **picEagle** and **picSquirrel** are used:

We determine the width and height of the eagle (**eagleW** and **eagleH**) and the width and height for the squirrel (**squirrelW** and **squirrelH**). These help in determining if the farmer catches a squirrel. Add these general declarations:

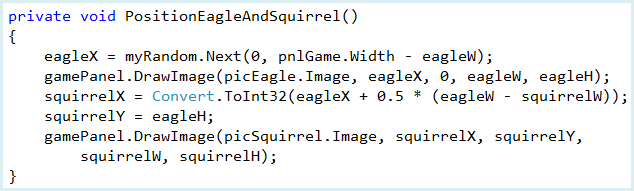


**eagleX** is the x position of the eagle (the y position is zero – the top of the screen). The squirrel is a (**squirrelX**, **squirrelY**). **myRandom** is needed for random number generation.

Add the this code to the **btnStart** **Click** event procedure under the current code:

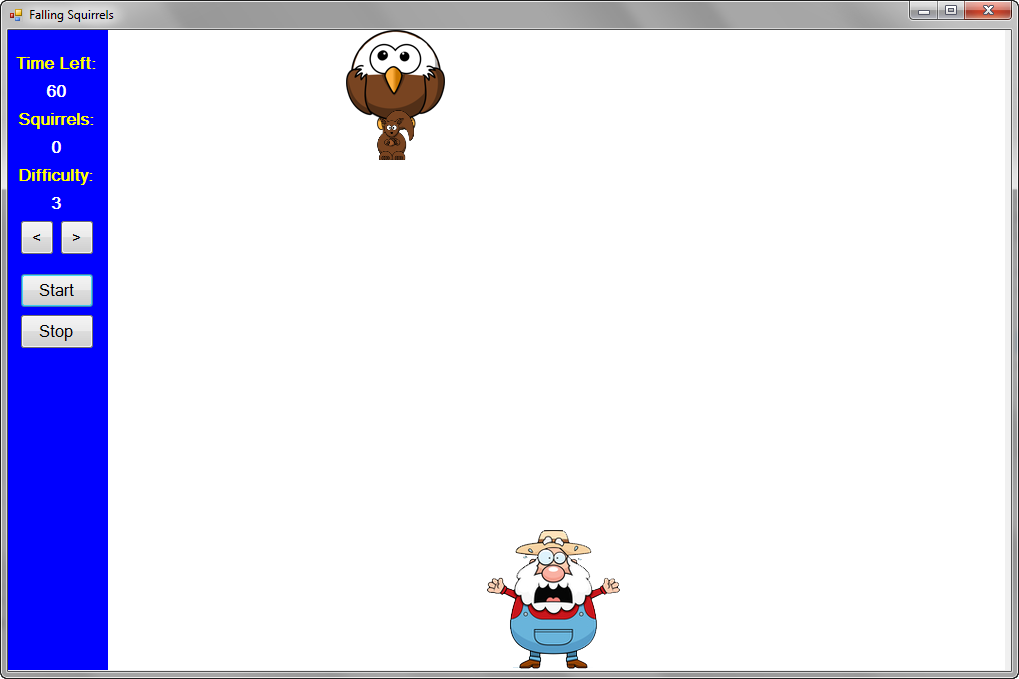


The above code uses a separate procedure (**PositionEagleAndSquirrel**) to randomly place the eagle and squirrel on the screen. Add this procedure to your program:



Notice how the random horizontal position is selected. The coordinates **squirrelX** and **squirrelY** let us know where a falling squirrel is at all times.

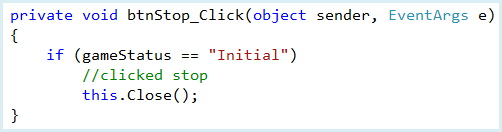
**Save** and **Run** the program. Set the **Difficulty** (I used 3). Click **Start**. The complete initialized game is now seen:



We still need to code to get the squirrel to drop and to move the farmer along the bottom of the screen. We write code to do these steps soon. But first, let’s take care of one last thing you can do while in “**Initial**” state – stop the program.

**Code Design – Stop the Program**

One last thing you can do while the game is in “**Initial**” state is to click on the **Stop** button. This will stop the program. The code here is simple. Add these lines to the **btnStop** **Click** event procedure:



As desired, if the **Stop** button is clicked, the program ends.

**Save** and **Run** the program. Make sure the **Stop** button works. This completes the code for moving from “**Initial**” game status to “**Playing**” status. Now, let’s look at code to move Farmer along the bottom of the screen.

**Code Design – Dropping Squirrel**

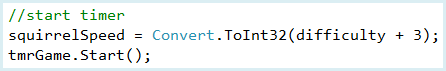
Once the game begins, the squirrel will drop. You move the farmer along the bottom of the screen to try and catch the dropping squirrel. Here we do the code that drops the squirrel. A variable **squirrelSpeed** will determine how fast the squirrel drops down the screen. Add this general declaration:



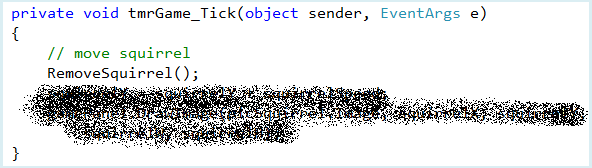
We assign **SquirrelSpeed** based on the value of **Difficulty**:  
  
squirrelSpeed = difficulty + 3;

Note, the higher the difficulty, the faster the speed. You can change this if you like.

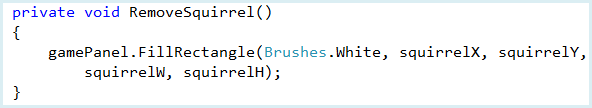
The timer control (**tmrGame**) is used to control squirrel motion. Add these lines to the **btnStart Click** event procedure at the bottom to turn on the timer and set the value for **squirrelSpeed**:



The squirrel position is updated in the **tmrGame Tick** event procedure. Use this code:



With each **Tick** event, the squirrel vertical position changes. This uses a procedure **RemoveSquirrel** that erases the squirrel from its current location prior to moving it to its new one. Add this procedure:



**Save** and **Run** the program. Click **Start**. The squirrel is falling:

But it just keeps on falling! We need to add two more pieces of code – one to move the farmer under the squirrel and one to determine whether the farmer has caught the squirrel.

**Code Design – Moving Farmer**

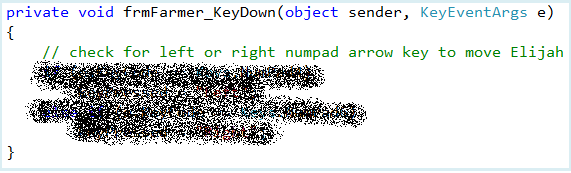
The object of the game is to move the farmer under falling squirrel so he can catch and save the squirrels. The farmer can be moved using two different methods. First, he can be moved by using the left and right arrow buttons on the numeric keypad (this will not work with the cursor control keys). With each key press, we change **farmerX** by **farmerDeltaX** (defined a while ago) in the appropriate direction. We use a new variable **keyPressed** to keep track of which key is being held. Add the general declaration (initializing **keyPressed** to “**None”**):



Change the Form’s KeyPreview property to true:

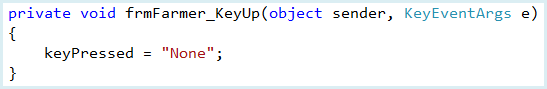


We detect key presses in the form’s **KeyDown** event procedure. The code is:



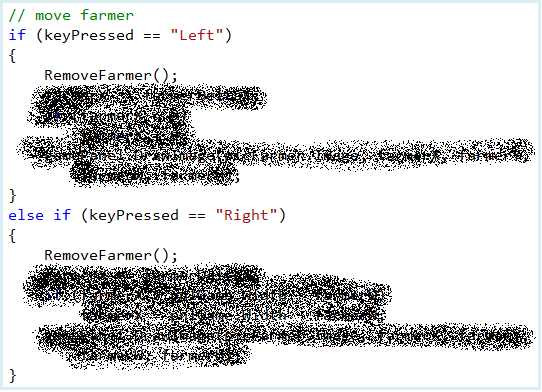
This tells us which key is being pressed (**keyPressed**). It is used in the **tmrGame** **Tick** event procedure to update the position of Farmer.

We also need code to tell us when a pressed key is released. This goes in the form’s **KeyUp** procedure:



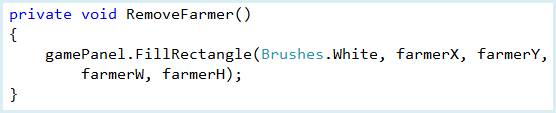
When a key is released, **keyPressed** is set back to “**None**”.

Add this code to the **tmrGame** **Tick** procedure:



If the left key is pressed, Farmer’s position is decremented by **farmerDeltaX** – if the right key is pressed, the position is incremented by **farmerDeltaX**.

Like the squirrel moving code, this code uses another procedure **RemoveFarmer** to erase Farmer from his current location. Add this procedure:

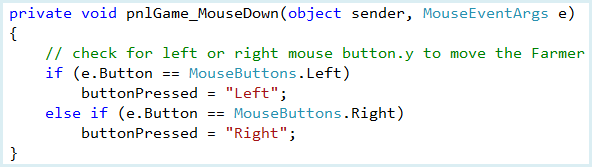


**Save** and **Run** the program. Click **Start**. Try moving Farmer using the left and right arrow keys on the numeric keypad (make sure **NumLock** is on):

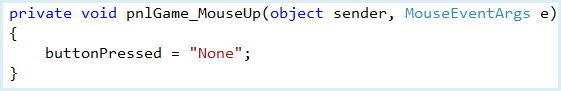
When we started this section, we said there were two ways to move Farmer. Let’s write code for the second method – using the left and mouse button keys. It’s a fairly easy change. Define a variable to keep track of the pressed button (**buttonPressed**):



Now, add this code to the **pnlGame** **MouseDown** event procedure:

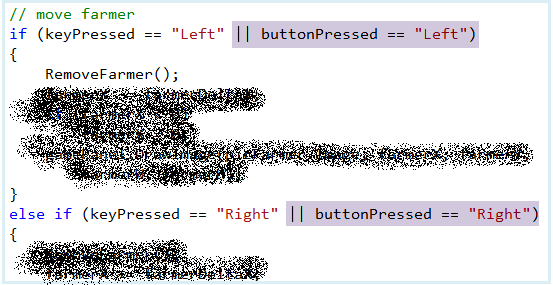


With this, when the mouse is over the game panel, the left and right mouse buttons replicate the action of clicking on the left and right arrow keys on the keypad. Add this code to the **pnlGame** **MouseUp** event procedure:



This tells us when a mouse button is released.

Finally, add the shaded changes to the **tmrGame Tick** event procedure:



In this code, if the left mouse button is being held down, the Farmer moves to the left. Similarly, if the right mouse button is being held down, the Farmer moves to the right. In the changes, we just use the code we used for the arrow keys.

**Save** and **Run** the program. Make sure you can move the Farmer with both the keypad arrow keys and by using the mouse buttons. Recall the mouse must be over the panel for the mouse buttons to work properly.

**Code Design – Catching Squirrels**

When the squirrel approaches the bottom of the screen, we need to see if Farmer catches it. The code to do this check is in the **tmrGame** **Tick** event procedure. We need to do two things:

* Check to see if the squirrel has been caught. If so, make the squirrel disappear, increment score and make the eagle and squirrel reappear at a random location at the top of the screen.
* Check to see if squirrel has reached the same vertical position of Farmer without being caught. If so, start a new eagle and squirrel.

How do we know if Farmer catches the squirrel? We want to see if the Farmer graphic collides with the squirrel graphic. The check cannot be made until the squirrel passes the top of Farmer. This happens when:

SquirrelY + SquirrelH > FarmerY

We will use a boolean variable (**passed**) to see if this is true. Once this condition is met, we can see that Farmer collides with squirrel (or catches it) when:

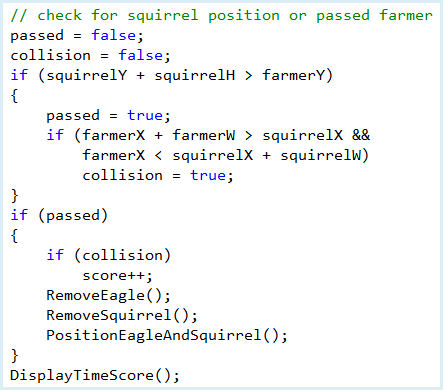
FarmerX + FarmerW > SquirrelX && FarmerX < SquirrelX + SquirrelW

This says that the right side of the Farmer must be “farther right” than the left side of the squirrel and the left side of the Farmer must be “farther left” than the right side of the squirrel. Another boolean variable (**collision**) is used to check this condition.

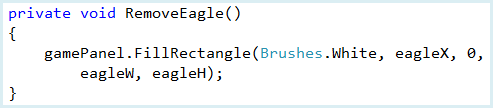
Declare the two new variables:



Add the shaded this code to the **tmrGame Tick** event procedure:



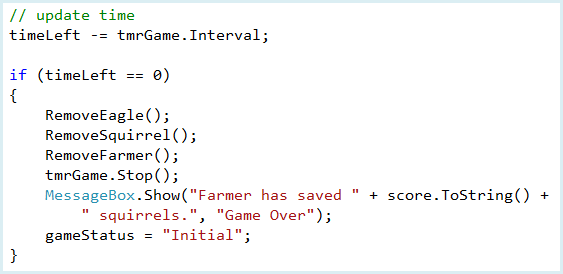
You should see how the listed steps are implemented. In particular, note how a new eagle is positioned once **passed** is **true** (and how the score is incremented if **collision** is also **true**. This new code requires **RemoveEagle** – a procedure to erase the eagle from its current location:



**Save** and **Run** the program.

**Code Design – Stop the Game**

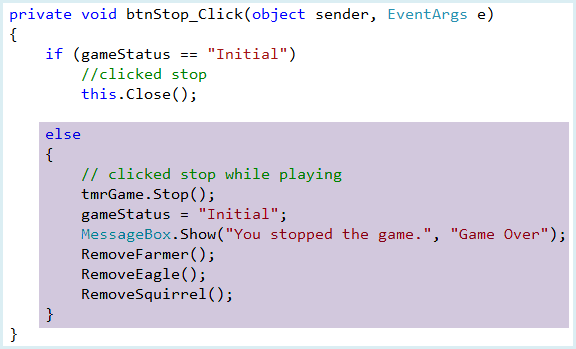
We’re almost done. With the current code, you can play the game as long as you like – notice the displayed **Time** **Left** is always 60. We need to implement the logic that ends the game after 60 seconds (and updates the time display). To do this, add this code to the **tmrGame Tick** event procedure:



In this new code, with each **Tick** event, the **timeLeft** is decremented by **Interval**. Once **timeLeft** is zero, the graphic elements (eagle, squirrel, Farmer) are removed from the screen, the timer is turned off and the results displayed.

**Save** and **Run** the program. Click **Start**. The time will count down.

Just one last change. You may want to stop a game before 60 seconds goes by. To do this, you can click the **Stop** button while the game is in “**Playing**” status. The code here is simple. Add the few shaded lines at the end of the **btnStop Click** event procedure:



Here, when stopped is pressed the game screen is cleared and a message displayed.

**Save** and **Run** the program. Click **Start**, then click **Stop**. You should see this message:



The game is now complete.

**Level 4+ Achievements**

* Have more than one type of object being dropped.
* Add a new feature to the game.