



CONSTRUCTION PLAN FOR **COIN FLIP**

Coin Flip is an introductory activity that will prepare you for the more detailed projects that you will develop as you work with the *Programming Games with Microsoft Visual Basic 6.0* book.

In the Coin Flip project, you will create a computer program that simulates the flipping of a coin. The player sees a picture of the head face or the tail face of a quarter. The program keeps track of the number of times the coin lands on each side. You can experiment with the Coin Flip activity by running the `coinflip.exe` file, which is found on this Web site.

The Coin Flip project will give you experience in:

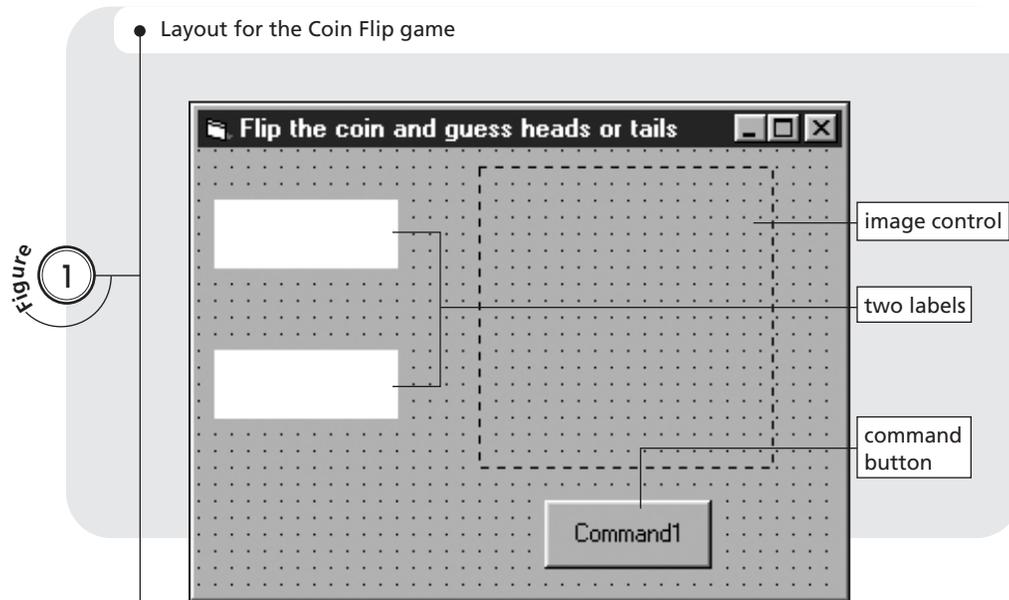
- Constructing a Visual Basic project, including placing controls on a form and writing code for event procedures
- Simulating random events, such as the flip of a coin, using Visual Basic built-in functions
- Manipulating images

This exercise will use three pictures stored in the following files: **question.bmp**, **head.bmp**, and **tail.bmp**. In the folder where you store your Visual Basic projects, create a folder named Coin Flip. Copy these image files from this text's Web site to your Coin Flip folder.

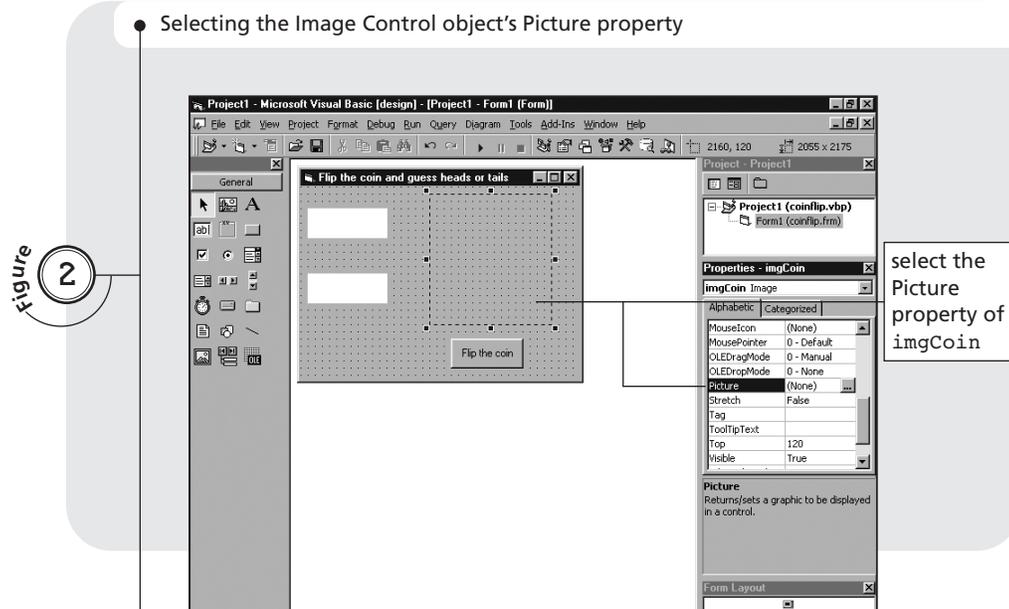
DESIGNING THE FORM

The first step is to design the user interface by placing controls on the form.

1. Start a new project. Change the Caption property of the form to **Flip the coin and guess heads or tails**.
2. Add to the form two Label objects , an Image Control object , and a command button . Follow the layout shown in Figure 1.



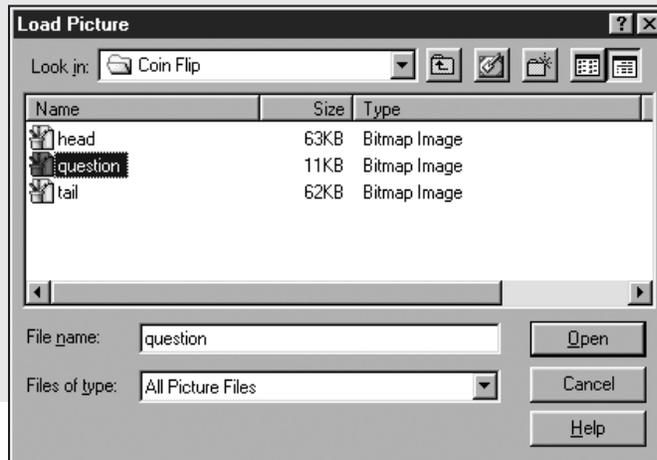
3. Change the name of the upper label to **lblHeadCt** and the name of the lower label to **lblTailCt**. Change the BackColor property of both labels to white (or another color of your choice). Change the name of the Image Control object to **imgCoin** and the name of the command button to **cmdFlip**. Change the command button's Caption property to **Flip the coin**.
4. Select the Image Control object **imgCoin** to access its Properties window, then select its Picture property, as shown in Figure 2.



5. Click the **three dots** on the right side of the Picture property. This invokes a dialog box that lets you assign a picture to the **imgCoin** object. Navigate to your Coin Flip folder and select the file **question.bmp**, then click **Open**. See Figure 3.

• Loading a picture into an Image Control object

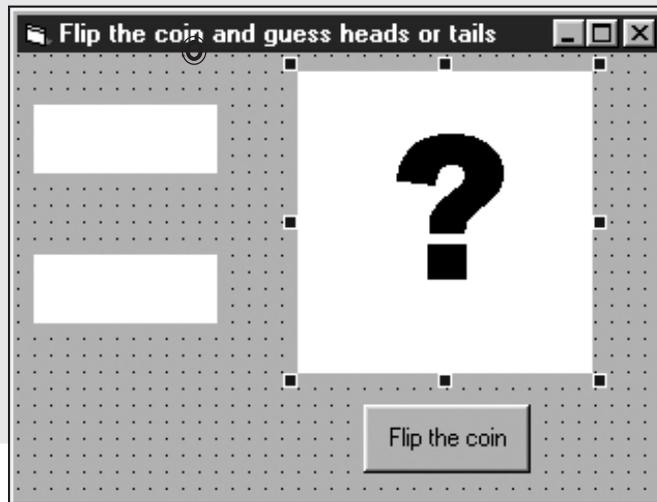
Figure 3



Once you have assigned the picture in **question.bmp** to the Image Control object, your form layout should look like Figure 4.

• The Image Control object contains the question mark picture

Figure 4



PROGRAMMING THE EVENTS

Now that you have designed the interface, you must tackle the next part of building a Visual Basic project: writing code. Specifically, this means:

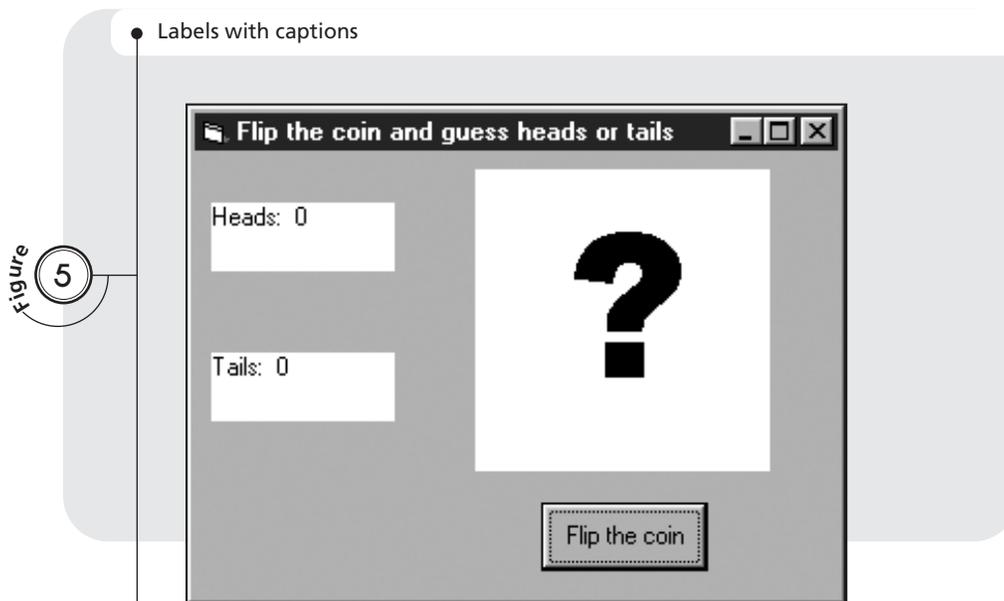
- Determining what information is required by the application and defining variables to hold that information
- Writing the event procedures for the relevant events. For this project, there are two events: starting the project, which is indicated by the `Form_Load` event, and clicking the command button to flip the coin, which triggers the `cmdFlip_Click` event.

The objective of this project is to keep track of the number of heads and the number of tails that occur as you flip the coin (by pressing the button) many times. The label `lblHeadCt` will display the number of times the coin lands heads up, and `lblTailCt` will display the number of times the coin lands tails up. You will need two variables to store this information: one to count the number of heads-up results, and one to count the number of tails-up results. Because this information needs to persist throughout play (execution of the game), you will need to declare these two as global variables. The global variables are declared as follows:

1. Access the (**General**) section of the project and insert the following code:

```
Option Explicit 'to force declarations of variables
Dim headCount As Integer
Dim tailCount As Integer
```

2. Access the `Form_Load` event procedure. In this event, you will assign an initial value to the caption of both labels, as shown in Figure 5. You need to determine what code sets the `Caption` property of each label so that the labels appear as they do in the figure.



The simple solution is to include the following line of code, called an **assignment statement**, in the `Form_Load` event procedure:

```
lblHeadCt.Caption = "Heads: 0"
```

The **"Heads: 0"** part of the statement is called a **literal**, which means that it is a fixed value. But how does your code represent a flip—or several flips—that comes up heads? Your label will need to display the appropriate number. A better solution is to use string concatenation. Remember: strings can be expanded or built up using the **"&"** operator.

You can still keep the first part of the line of code and the first part of the literal since they do not change. You must display the value of the variable `headCount`, which contains the number of flips that resulted in heads. But remember, `headCount` is an integer, so you need to use the string conversion function `Str`, like this:

```
lblHeadCt.Caption = "Heads: " & Str(headCount)
```

Add your own line of code to set the caption for `lblTailCt`.

3. Access the Click event for the command button `cmdFlip`. When the user clicks this button, code will execute the following steps:
 - a. Generate a random number to represent either heads or tails
 - b. Place the correct picture (heads or tails) in the Image Control object `imgCoin`
 - c. Increment the correct counting variable (`headCount` or `tailCount`)
 - d. Update the label caption with the new count

To accomplish this with code, first declare a variable to store the random number:

```
Dim choice As Integer
```

Next, use the random number generator to generate either a 0 (to represent heads) or a 1 (for tails):

```
Randomize  
choice = Int(Rnd * 2)
```

As you will see in several of the games in the book, `Randomize` initializes the random number generator, and the function `Rnd` returns a random fractional number between 0 and 1. Multiplying the return value of the `Rnd` function by 2 results in a fractional number between 0 and just under 2. Using the `Int` function truncates the fractional portion, leaving either a 0 or a 1.

Once the number representing heads or tails has been generated, the Click event must update the graphical display and increment the correct count variable. Loading the correct picture is done with the function `LoadPicture`, which can load a graphic into an Image Control object from a file. To indicate to Visual Basic the location of the pictures, the code concatenates `App.Path`, which indicates the folder of the current application, with a string holding a slash and the name of the image file.

These steps need to be handled in an If statement so that the correct code executes regardless of which side of the coin faces up. The code is shown in Figure 6.

Figure 6 Code that increments the heads and tails counters and displays the correct picture

```

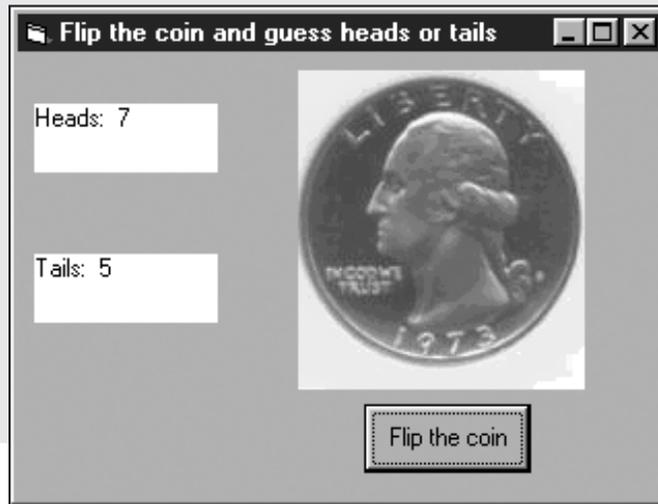
If choice = 0 Then
    headCount = headCount + 1
    ImgCoin.Picture = LoadPicture(App.Path & "/head.bmp")
Else
    tailCount = tailCount + 1
    ImgCoin.Picture = LoadPicture(App.Path & "/tail.bmp")
End If
  
```

4. The last step is to update the count labels with the latest count. You can use the same code you used in the `Form_Load` event (*Hint*: See Step 2).
5. Run the program and flip the coin several times. Do you notice any pattern—Does heads come up more often than tails, or vice versa? Figure 7 shows the results of 12 coin flips (your results may differ).

Figure

7

• Twelve coin flips show seven heads and five tails



Step 3 suggested one method for determining a value for choice, but there is an alternate way of handling this action. Since `Rnd` returns a fractional value between 0 and 1, the return value of `Rnd` can be split into values from 0 to .5 and from .5 to 1.

Change the code to reflect this approach. (*Hint:* You must also change the `If` statement that tests the value of `choice`.)