CASE STUDY

twenty five

Projectile Motion

case study

OVERVIEW

CS25.1 *Application Overview and Model Development*

Projectile motion is an important topic in physics. In this application, we allow a user to visualize and analyze the motion of a projectile in three projectile situations: when an object is shot from cannon; when it is dropped from a plane; and when it is thrown off a cliff. This tool can be used by instructors while teaching projectile motion or by students in order to better understand this subject.

CS25.1.1 Model Definition and Assumptions

We use standard physics equations to calculate the projectile motion. Given the initial condition of the object, we display its motion and plot its velocity and acceleration as a function of time until it hits the ground. The values that define the object's initial condition are the initial height, initial velocity, and initial angle. The user provides these initial values.

For the projectile calculations, we will also use values for time, a time increment for animation, x-position (or distance) and y-position (or height), and x-velocity, and y-velocity. Additionally, we will employ a constant for the gravitational force, which we set at 9.81. We assume that, initially, the x-position and y-position are zero.

With the following equation, we calculate the total time that the projectile motion requires:

$$t = (v_0*\sin(\theta_0) + \sqrt{((v_0*\sin(\theta_0))^2 + 2*g*y_0)}\ ) / g$$

[Total Time = (Initial Velocity * Sin(Initial Angle) + √((Initial Velocity * (Sin(Initial Angle)))^2 + 2 * Gravitational Force * Initial Height)) / Gravitational Force]

We calculate the total distance traveled as follows:

$$d = v_0*\cos(\theta_0)*t$$

[Total Distance = Initial Velocity * (Cos(Initial Angle)) * Total Time]

We calculate the initial x-velocity and y-velocity with the following two equations:

$$v_{x0} = v_0*\cos(\theta_0)$$

[Initial X-Velocity = Initial Velocity * (Cos(Initial Angle))]

$$v_{y0} = v_0*\sin(\theta_0)$$

[Initial Y- Velocity = Initial Velocity * (Sin(Initial Angle))]

We calculate the highest point of the projectile motion and the time at which this maximum y-position is reached as follows:

$$t_{ymax} = v_{y0} / g$$

[Time at Max Y-Position = Initial Y-Velocity / Gravitational Force]

$$y_{max} = y_0 + v_0*\sin(\theta_0)*t_{ymax} - (g/2) * (t_{ymax})^2$$

[Max Y-Position = Initial Height + Initial Velocity * (Sin(Initial Angle)) * Time at Max Y-Position – (Gravitational Force / 2) * (Time at Max Y-Position)^2]

As we animate the projectile motion, we determine its x-position, y-position, velocity, and angle for each time increment. We do so using the following equations:

$$d = v_0*\cos(\theta_0)*t$$

[X-Position = Initial Velocity * (Cos(Initial Angle)) * Time]

$$y = y_0 + v_0*\sin(\theta_0)*t - g/2$$

[Y-Position = Initial Height + Initial Velocity * (Sin(Initial Angle)) * Time – Gravitational Force / 2]

$$v = \sqrt{(v_x^2 + v_y^2)}$$

[Velocity = $\sqrt{(\text{X-Velocity}^2 + \text{Y-Velocity}^2)}$]

$$\theta = \arctan(v_y / v_x)$$

[Angle = Arctan(Y-Velocity / X-Velocity)]

We assume that the projectile motion stops at the ground, at a y-position equal to zero. For more details on projectile motion and the equations described above, please see Fundamentals of Physics by Halliday, Resnick, and Walker.

CS25.1.2 Input

The input for this application is the initial height, initial velocity, and initial angle for whichever projectile scenario the user chooses.

- Projectile scenario (cannon, plane, or cliff)
- Initial height
- Initial velocity
- Initial angle

CS25.1.3 Output

The output for this application is the path of the projectile motion.

- Chart of projectile motion
- X-position, y-position, velocity, and angle for each time increment
- Total time for projectile motion
- Highest y-position reached

CS25.2 *Worksheets*

We use two worksheets in this application: the welcome sheet and the projectile sheet. The welcome sheet contains the title and the description of the application, as well as an image. (See Figure CS25.1.) The “Start” button on the welcome sheet brings the user to the projectile sheet.

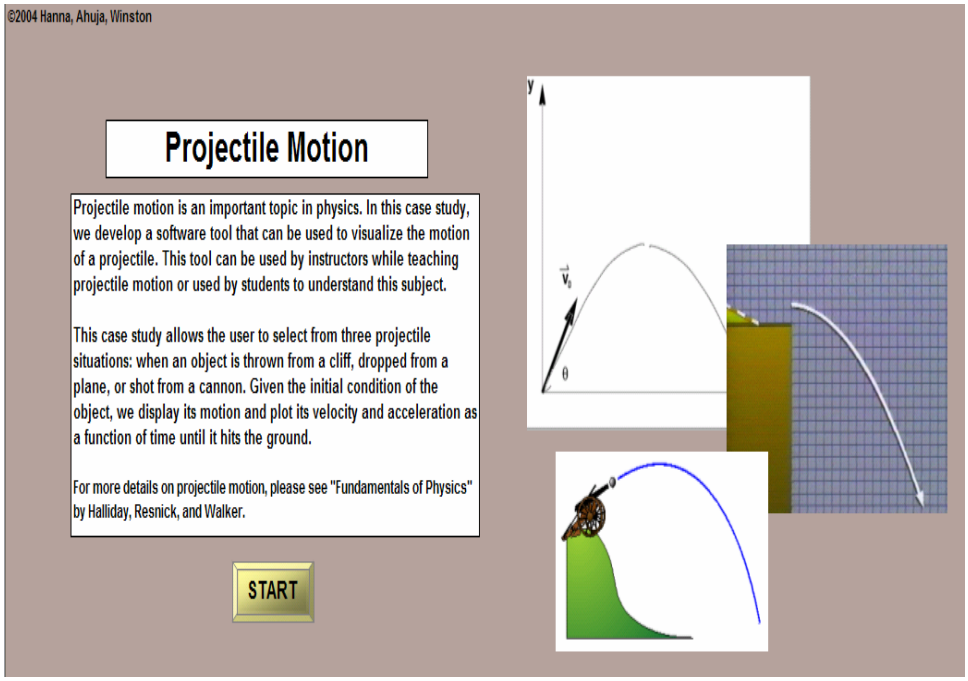


Figure CS25.1 The welcome sheet.

The projectile sheet includes a chart to animate the projectile motion and a table to display the values of the x-position, y-position, velocity, and angle for each time increment. (See Figure CS25.2.) The picture next to the chart corresponds to the projectile scenario, and the “Solve” button begins the projectile animation. The “Re-solve” button allows the user to change the projectile scenario and/or input values, and, finally, the “End” button brings the user back to the welcome sheet.

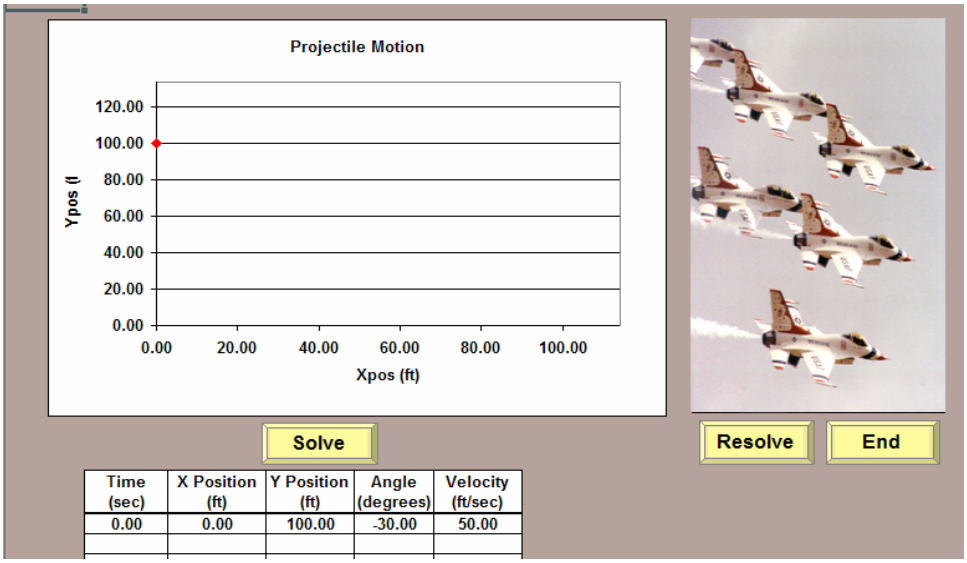


Figure CS25.2 The projectile sheet.

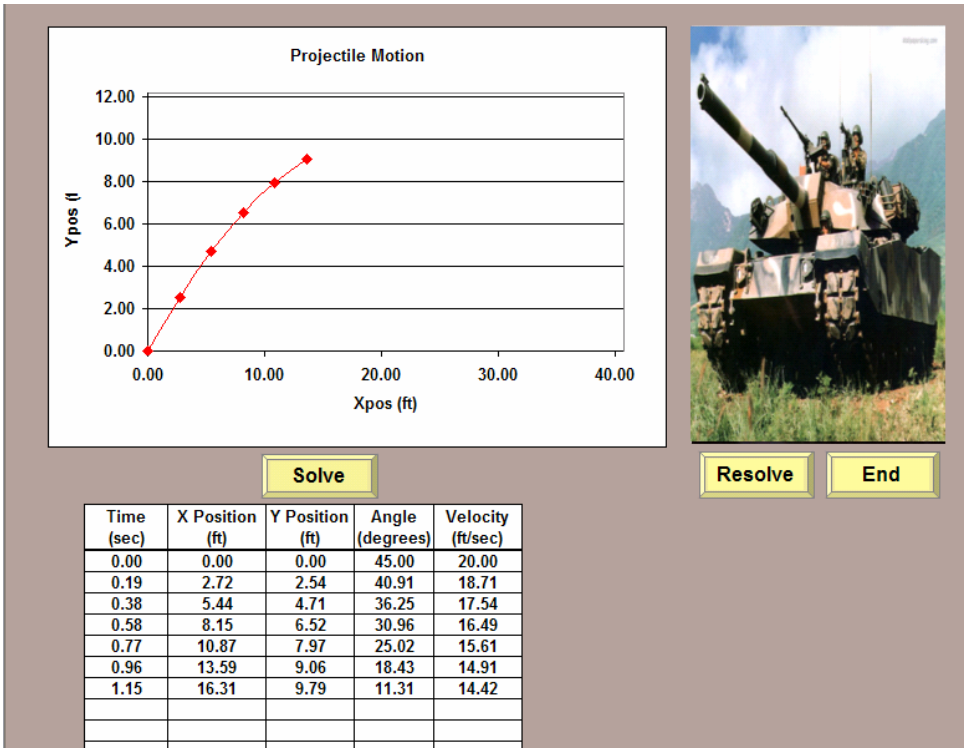
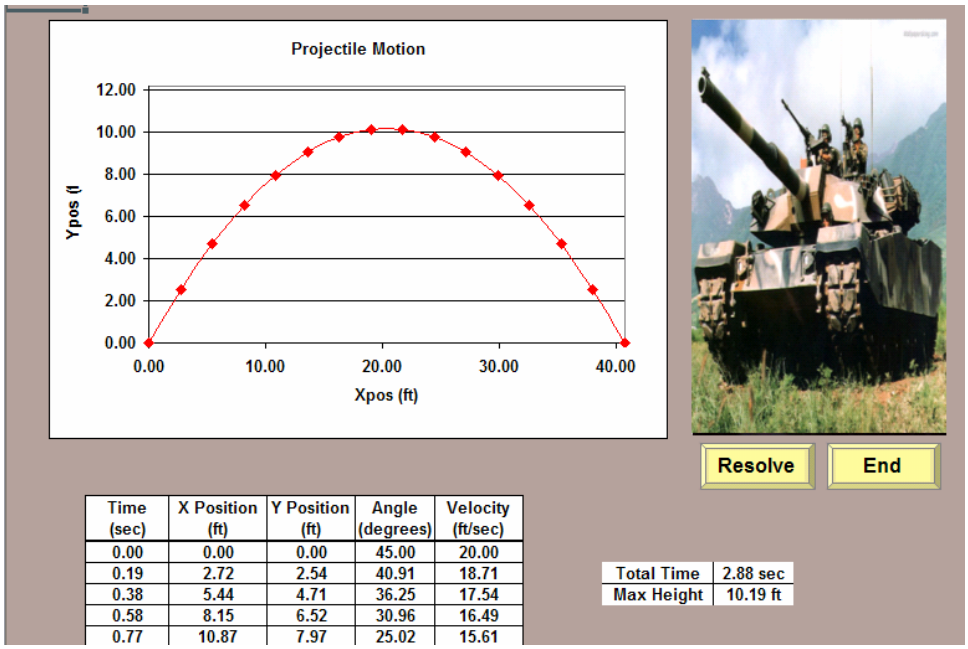


Figure CS25.3 Animating the projectile motion.

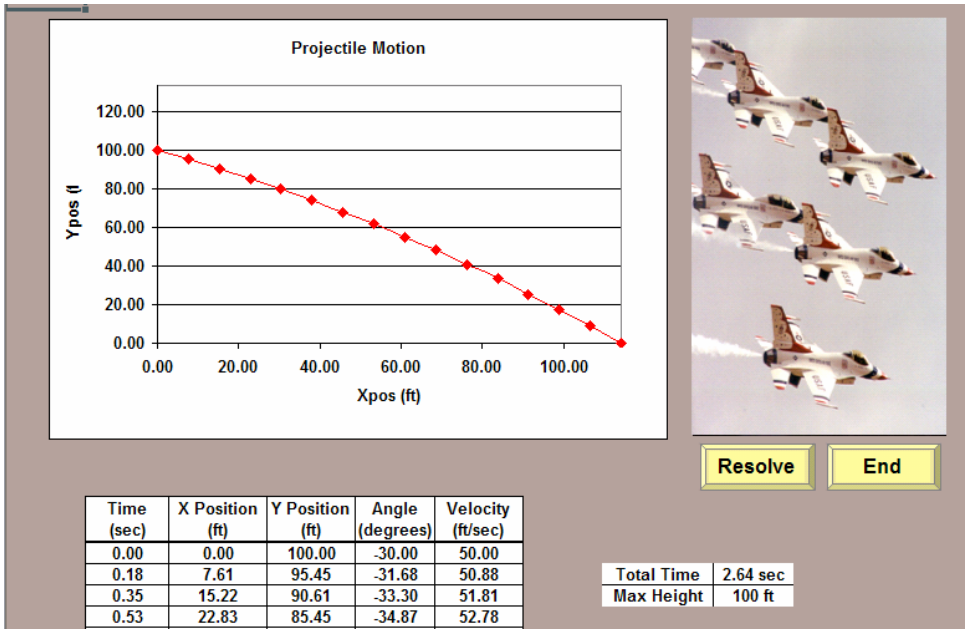
Figures CS2.4, CS2.5, and CS2.6 exhibit the final projectile motions for the cannon, plane, and cliff scenarios, respectively. (These particular projectile motions were calculated using default input values.)



Time (sec)	X Position (ft)	Y Position (ft)	Angle (degrees)	Velocity (ft/sec)
0.00	0.00	0.00	45.00	20.00
0.19	2.72	2.54	40.91	18.71
0.38	5.44	4.71	36.25	17.54
0.58	8.15	6.52	30.96	16.49
0.77	10.87	7.97	25.02	15.61

Total Time	2.88 sec
Max Height	10.19 ft

Figure CS25.4 The projectile motion of an object being shot from cannon.



Time (sec)	X Position (ft)	Y Position (ft)	Angle (degrees)	Velocity (ft/sec)
0.00	0.00	100.00	-30.00	50.00
0.18	7.61	95.45	-31.68	50.88
0.35	15.22	90.61	-33.30	51.81
0.53	22.83	85.45	-34.87	52.78

Total Time	2.64 sec
Max Height	100 ft

Figure CS25.5 The projectile motion of an object being dropped from a plane.

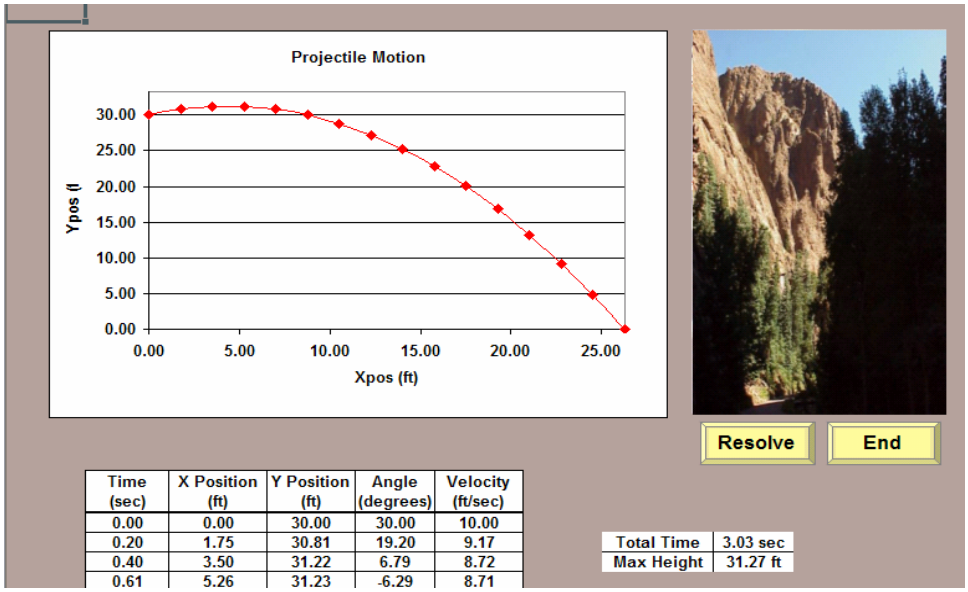


Figure CS25.6 The projectile motion of an object being thrown off a cliff.

Summary		
	Welcome sheet	Contains an application description and a "Start" button.
	Projectile sheet	Displays the projectile's motion and values.

CS25.3 *User Interface*

For this application's user interface, we use navigational and functional buttons as well as two user forms. On the welcome sheet, the "Start" button brings the user to the projectile sheet. The user is then shown a user form for selecting which projectile scenario he or she wants to analyze. (See Figure CS25.7.) The user can select one of three option buttons: "Cannon," "Plane," and "Cliff."

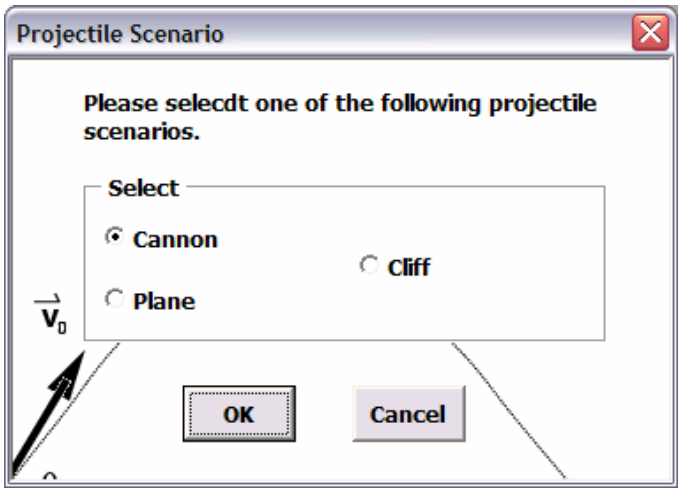


Figure CS25.7 The "Projectile Scenario" form.

Once the user has selected a projectile scenario, the input form then appears. (See Figure CS25.8.) This form has three text boxes to receive the values for the initial height, initial velocity, and initial angle. We use labels to clarify the units assumed for each value, and we also provide the user with default values. These default values differ depending on the specific projectile scenario. Figures CS2.8, CS2.9, and CS2.10 reveal the default values for the cannon, plane, and cliff projectile scenarios, respectively.

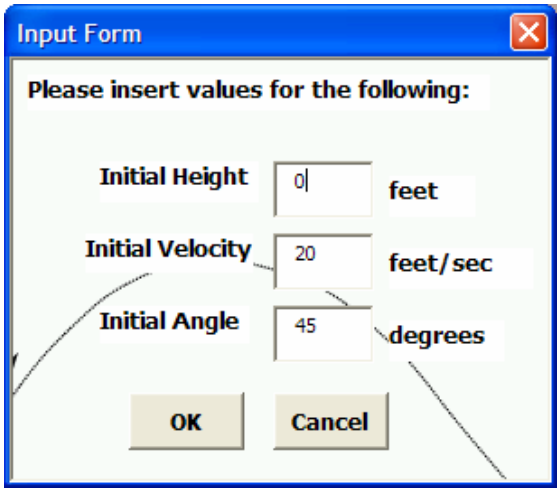


Figure CS25.8 The input form with the default values for the cannon projectile.

The projectile sheet additionally has three buttons: "Solve," a functional button that begins the projectile animation; "Re-solve," also a functional button that allows the user to change the projectile scenario and/or input values and re-perform the projectile calculations and animation; and "End," a navigational button that brings the user back to the welcome sheet.

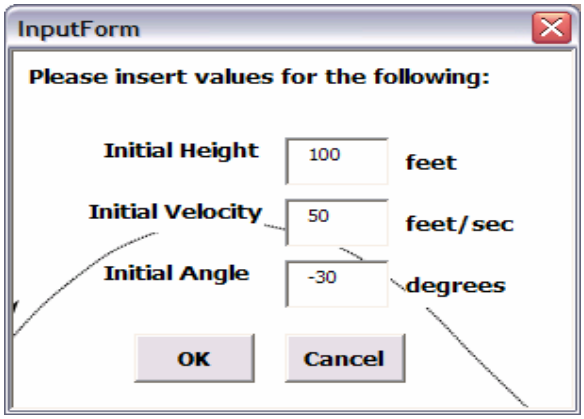


Figure CS25.9 The input form with the default values for the plane projectile.

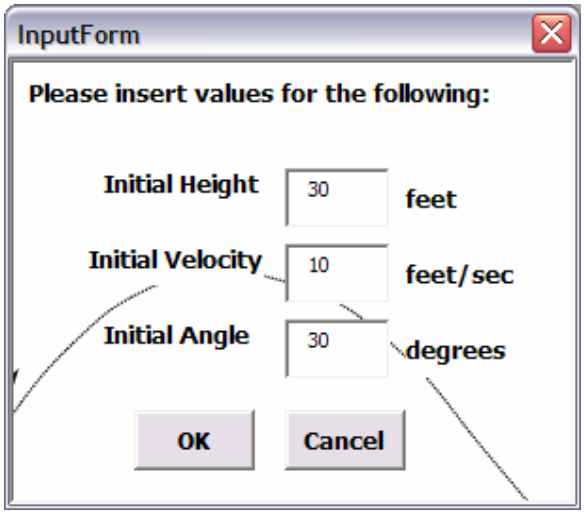


Figure CS25.10 The input form with the default values for the cliff projectile.

Navigational Buttons	The "Start" button on the welcome sheet; "Solve," "Re-solve," and "End" buttons on the projectile sheet.
Projectile Scenario user form	The user selects from three projectile scenarios.
Input user form	The user specifies values for the initial height, initial velocity, and initial angle.

CS25.4 *Procedures*

We will now outline the procedures for this application, beginning with the initial sub procedures and the variable definitions. (See Figure CS25.11.) The "Main" procedure initializes the variables and calls the "ClearPrevious procedure," which clears the previous data and initializes the variable values. The "Main" procedure then takes the user to the projectile sheet and displays the projectile scenario and the input forms. At the end of the "Main" procedure, the "InitializeData" procedure is called.

```
Option Explicit
Const PI = 3.14159265358979, GRAV = 9.81

Public i As Integer, InitHt As Integer, InitVel As Double, InitAngle As Double, StartCell As Range, _
Sim As Boolean, Time As Double, XPos As Double, YPos As Double, Angle As Double, Vel As Double, _
TTotal As Double, TIncrem As Double, Increm As Integer, XVel As Double, YVel As Double, _
InitYVel As Double, Cancl As Boolean, ws As Worksheet, XFinal As Double, YStop As Boolean, _
YMaxTime As Double, YMaxPos As Double

Sub MAIN()  'called from the START button
    Worksheets("Projectile").Activate
    Set StartCell = Range("B26")
    Call ClearPrevious
    ActiveSheet.Shapes("SolveButton").Visible = True

    Worksheets("Projectile").Visible = True
    Worksheets("Welcome").Visible = False
    frmImage.Show
    frmInput.Show

    Call InitializeData
End Sub

Sub ClearPrevious()     'clears previous data
    Application.ScreenUpdating = False
    Worksheets("Projectile").Activate

    Range(StartCell, StartCell.Offset(1, 4).End(xlDown)).ClearContents

    With Range(Range("Results").Offset(1, 0), Range("Results").Offset(2, 1))
        .ClearContents
        .Interior.ColorIndex = xlNone
        .Borders(xlInsideHorizontal).LineStyle = xlNone
        .Borders(xlInsideVertical).LineStyle = xlNone
    End With

    YStop = False
    Application.ScreenUpdating = True
End Sub
```

Figure CS25.11 Variable declarations, the “Main” procedure, and the “ClearPrevious” procedure.

The procedures for the “Projectile Scenario” form determine which scenario the user has selected. (See Figure CS25.12.) The input form’s default values are then set depending on this selection. The corresponding image for the scenario selected also appears next to the chart on the projectile sheet.

The procedures for the input form record the values for the initial height, the initial velocity, and the initial angle. (See Figure CS25.13.) The application also performs some error checking.

The “InitializeData” procedure uses the values entered on the input form to compute the total time of the projectile motion, a time increment for animation, the x- and y-positions, and the x- and y-velocities. (See Figure CS25.14.) These calculations are performed with the equations explained above in Section CS2.1.1. We set the initial x- and y- positions to zero. Next, we display all the initial values in the table on the projectile sheet. The user can now choose to press the “Solve,” “Re-solve,” or “End” button.

```
Sub cmdCancel_Click()
    Worksheets("Welcome").Visible = True
    Worksheets("Projectile").Visible = False
    Unload Me
    End
End Sub

Sub cmdOK_Click()
    Worksheets("Projectile").Activate
    'check which option button has been selected
    'then update picture on chart and give input default values
    If optCannon Then
        ActiveSheet.Shapes("Cannon").ZOrder msoBringToFront
        frmInput.txtInitialHeight.Value = "0"
        frmInput.txtInitialVelocity.Value = "20"
        frmInput.txtInitialAngle.Value = "45"
    ElseIf optPlane Then
        ActiveSheet.Shapes("Plane").ZOrder msoBringToFront
        frmInput.txtInitialHeight.Value = "100"
        frmInput.txtInitialVelocity.Value = "50"
        frmInput.txtInitialAngle.Value = "-30"
    ElseIf optCliff Then
        ActiveSheet.Shapes("Cliff").ZOrder msoBringToFront
        frmInput.txtInitialHeight.Value = "30"
        frmInput.txtInitialVelocity.Value = "10"
        frmInput.txtInitialAngle.Value = "30"
    Else
        MsgBox "Please make a selection."
        Exit Sub
    End If

    Range("A1").Select
    Unload Me
End Sub
```

Figure CS25.12 The procedures for the “Projectile Scenario” form.

```
Sub cmdCancel_Click()
    Worksheets("Welcome").Visible = True
    Worksheets("Projectile").Visible = False
    Unload Me
    End
End Sub

Sub cmdOK_Click()
    'ensure that values have been entered for each input
    If txtInitialHeight.Value = "" Or txtInitialVelocity.Value = "" Or txtInitialAngle.Value = "" Ther
        MsgBox "Please enter a value in the empty fields"
    End If

    'record text box values to variables
    InitHt = txtInitialHeight
    InitVel = txtInitialVelocity
    InitAngle = txtInitialAngle

    Unload Me
End Sub
```

Figure CS25.13 The procedures for the input form.

```
Sub InitializeData()    'called to initialize value before projectile motion is calculated
    'display initial values
    With StartCell
        .Offset(0, 0).Value = 0            'initial Time
        .Offset(0, 1).Value = 0            'initial XPos
        .Offset(0, 2).Value = InitHt       'initial YPos
        .Offset(0, 3).Value = InitAngle    'initial Angle
        .Offset(0, 4).Value = InitVel      'initial Velocity
    End With

    'initialize more values
    Time = 0
    TTotal = 0
    TIncrem = 0
    Increm = 15
    InitYVel = 0
    YVel = 0
    XVel = 0
    XPos = 0
    YPos = 0
    InitAngle = InitAngle * PI / 180   'convert angle from degrees to radians for calculations

    'calc total time
    TTotal = (InitVel * Sin(InitAngle) + Sqr((InitVel * (Sin(InitAngle))) ^ 2 + 2 * GRAV * InitHt)) / GRAV

    'calc distance values and adjust chart x-scale
    XFinal = InitVel * (Cos(InitAngle)) * TTotal
    ActiveSheet.ChartObjects("Chart 8").Activate
    ActiveChart.Axes(xlCategory).MaximumScale = XFinal
    ActiveChart.Axes(xlCategory).MinimumScale = 0

    'calc time increment and velocities
    TIncrem = TTotal / Increm
    InitYVel = InitVel * (Sin(InitAngle))
    XVel = InitVel * (Cos(InitAngle))

    'calc height and adjust chart y-scale
    YMaxTime = InitYVel / GRAV
    YMaxPos = InitHt + InitVel * (Sin(InitAngle)) * YMaxTime - GRAV / 2 * (YMaxTime ^ 2)
    ActiveSheet.ChartObjects("Chart 8").Activate
    ActiveChart.Axes(xlValue).MaximumScale = YMaxPos + 2
    ActiveChart.Axes(xlValue).MinimumScale = 0

    'ensure that simulation will stop at height of 0
    If YMaxTime < 0 Then
      YMaxTime = 0
      YMaxPos = InitHt + InitVel * (Sin(InitAngle)) * YMaxTime - GRAV / 2 * (YMaxTime ^ 2)
    End If
    Range("A1").Select
End Sub
```

Figure CS25.14 The “InitializeData” procedure.

When the user presses the “Solve” button, the “Projectile” procedure begins. (See Figure CS25.15.) This procedure performs the projectile motion animation by calculating the x- and y-positions, the velocity, and the angle for each time increment. We use a “Do, Until” loop to perform these calculations until the y-position value is zero or negative. We assume that the projectile motion stops at the ground: at a y-position equal to zero.

```
Sub Projectile()        'main sub which calculates projectile motion
   i = 1
   Do Until YStop
        Time = Time + TIncrem
        'update values
        XPos = InitVel * (Cos(InitAngle)) * Time
        YPos = InitHt + InitVel * (Sin(InitAngle)) * Time - GRAV / 2
        Vel = Sqr(XVel ^ 2 + YVel ^ 2)
        Angle = Atn(YVel / XVel)
        Angle = Angle * 180 / PI        'convert to degrees

        'check if height is negative
        If YPos < 0 And Time > 0 Then
         YStop = True
         YPos = 0
         Time = TTotal
         XPos = InitVel * (Cos(InitAngle)) * Time
         YVel = InitYVel - GRAV * (Time)
         Vel = Sqr(XVel ^ 2 + YVel ^ 2)
         Angle = Atn(YVel / XVel)
         Angle = Angle * 180 / PI        'convert to degrees
        End If

        'display values
        With StartCell
             .Offset(i, 0).Value = Time
             .Offset(i, 1).Value = XPos
             .Offset(i, 2).Value = YPos
             .Offset(i, 3).Value = Angle
             .Offset(i, 4).Value = Vel
        End With
        i = i + 1
        Application.Wait (Now() + TimeValue("0:00:01"))
    Loop

    With Range("Results")
        .Offset(1, 0).Value = "Total Time"
        .Offset(1, 1).Value = Round(Time, 2) & " sec"
        .Offset(2, 0).Value = "Max Height"
        .Offset(2, 1).Value = Round(YMaxPos, 2) & " ft"
    End With
    With Range(Range("Results").Offset(1, 0), Range("Results").Offset(2, 1))
        .Interior.ColorIndex = 2
        .Borders(xlInsideHorizontal).Weight = xlThin
        .Borders(xlInsideVertical).Weight = xlThin
    End With

    ActiveSheet.Shapes("SolveButton").Visible = False
End Sub
```

Figure CS25.15 The “Projectile” procedure.

The navigational procedures for the “Re-solve” and “End” buttons appear in Figure CS25.16.

```
Sub Resolve()
    Call MAIN
End Sub

Sub Cancel()
    Worksheets("Welcome").Visible = True
    Worksheets("Projectile").Visible = False
End Sub
```

Figure CS25.16 The navigational procedures.

Main	Initializes the application and takes the user to the input sheet.
ClearPrevious	Initializes the variables and clears the previous values.
Procedures for Projectile Scenario Form	Update the chart image and the default values for the input form depending on the user's projectile scenario.
Procedures for Input Form	Receive values for the initial height, the initial velocity, and the initial angle.
Initialize Data	Initializes values used in the projectile motion calculations.
Projectile	Performs projectile motion calculations and animation.
Navigational Procedures	Apply to the "Re-solve" and "End" buttons.

CS25.5 *Re-solve Options*

The user can re-solve this application with the projectile sheet's "Re-solve" button, which is assigned to the "Main" sub procedure. In other words, once this button is clicked, the program restarts. All previous values are then cleared and the projectile scenario and input forms are shown again to the user. This lets the user compare projectile scenarios or one scenario's projectile motions using various input values.

"Re-solve" Button	Calls the "Main" procedure to restart the application.

CS25.6 *Summary*

- The projectile motion application allows the user to analyze three projectile situations: when an object is shot from cannon; when one is dropped from a plane; and when one is thrown from a cliff.
- This application requires two Worksheets: the welcome sheet and the projectile sheet.
- This application uses buttons and two user forms as the user interface.
- Several procedures for this application initialize and perform the projectile motion calculations and animation.

- The user can re-solve the application by pressing the “Re-solve” button on the projectile sheet.

CS25.7 *Extensions*

- Calculate the total distance traveled by the projectile motion. Display the result to the user with the total time and maximum y-position values at the end of the animation.
- Add “object weight” as an input value. How do the projectile motion calculations change?
- Add another projectile scenario. What modifications do you need to make to the Worksheets, user interface, and procedures?
- Enhance the re-solve options by allowing the user to perform a comparison between all the scenarios using similar input values. In other words, if an object of the same weight, velocity, and angle is dropped in each scenario, determine in which scenario the object:
  - reaches the maximum height (assuming all initial heights are equal).
  - travels the farthest.
  - reaches the ground in the shortest amount of time.
- What are some other re-solve options you could add to this application?