NSBasic – App Studio

10 - SweetShop

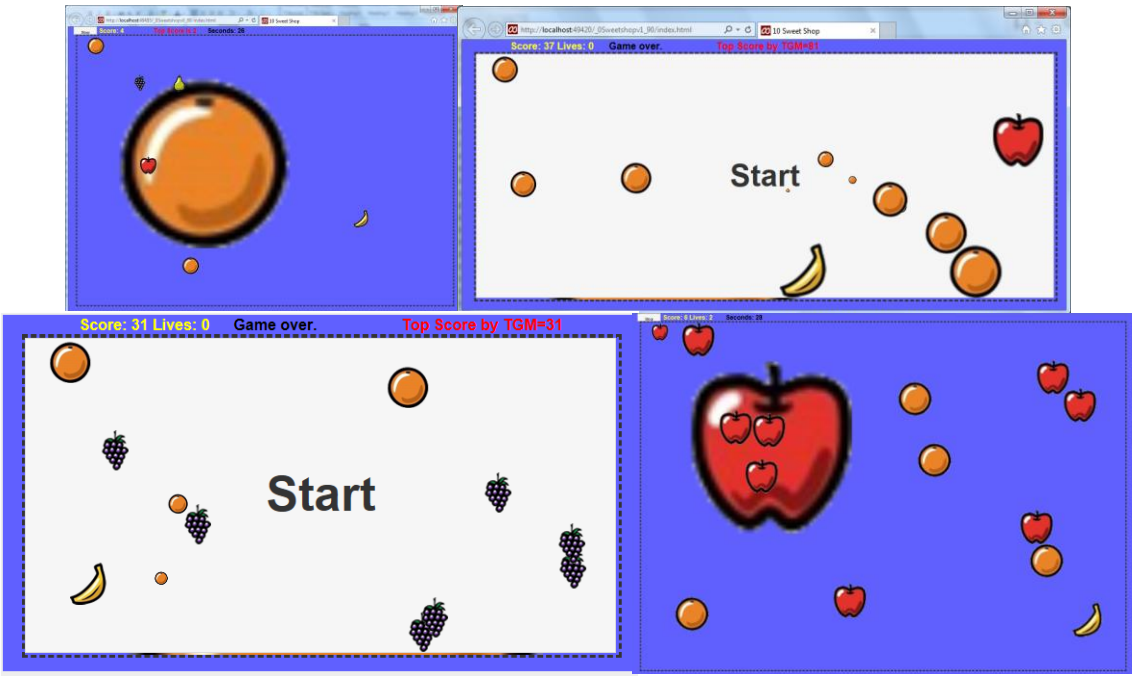


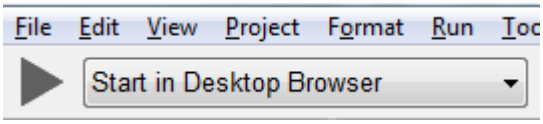
Table of Contents

NSBasic –Key Steps…

1. Always use your P Drive!!!!

2. To Test your code.

Click the play button shown …>



3. To 'Deploy' your working code…

Following Testing you MUST deploy the code. To do this select the Run Menu and Deploy
This creates a folder within your P drive folder with all the html code to run in a website apart from 2 extra lines of code below…

4. Additional code for Internet Explorer

Additional html Code to make page run outside of NSBasic. Insert via Dreamweaver or Notepad…

These 2 lines should be placed directly after the <Head> line…

```
<META content="IE=11.0000" http-equiv="X-UA-Compatible">
<META http-equiv="Content-Type" content="text/html; charset=utf-8">
```

5. Add to your website.

Move the folder onto a folder within your VLE page or website. Then provide a link to it, by hyperlinking to the index file in the folder created by your NSBasic deploy. The html file index.html can also be embedded into the VLE.

6. Using this booklet…

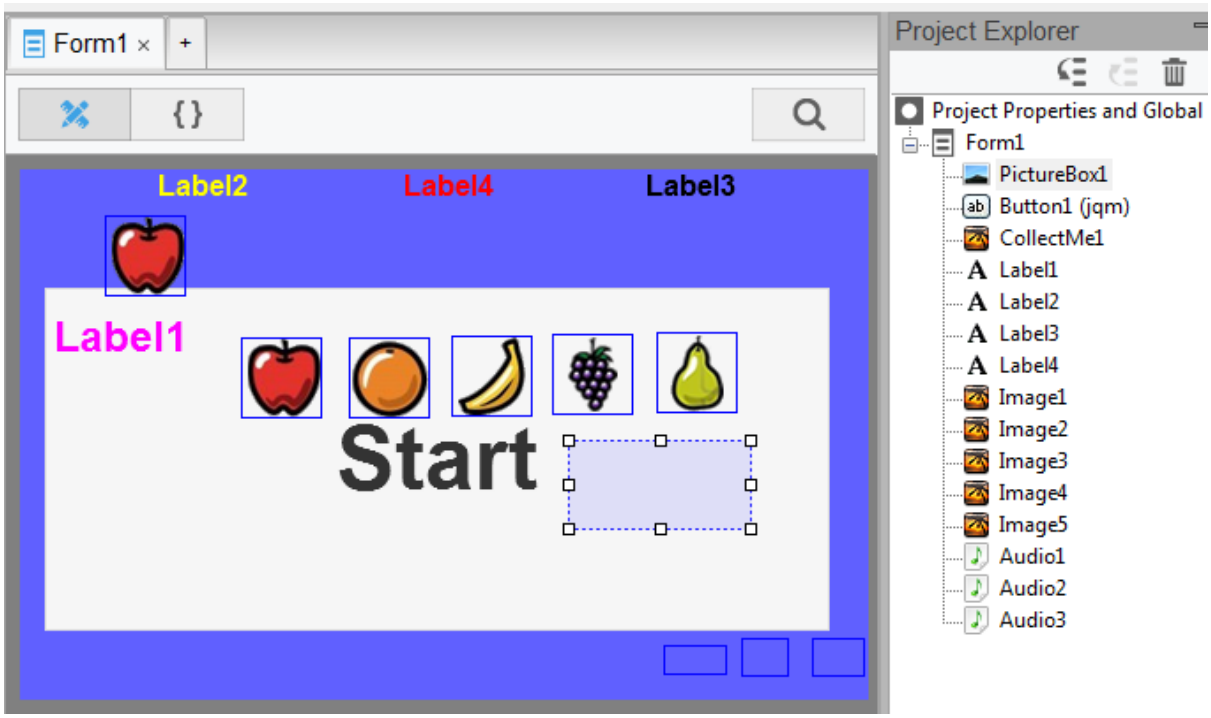
- You are advised to read a page / section / version before attempting it.
- Aim to understand what the code is trying to do and how it goes about doing it.
- After reading a part through carryout the changes carefully and crucial with 100% accuracy.
- Make sure you keep numerous 'version named' copies of the program as if it does not work you can go back and try again!
- If it works then do experiment, but make sure you save a copy before you experiment.

10 SweetShop

Use a folder 'Sweetshop' on the P drive for all your work (code and files) – This maybe inside a NSBasic folder!

Form Layout..

You will need a range of Objects on your Form. The Form size is 480 (width) x 300 (height)



There are 4 labels, 6 Images, 3 Audio, 1 button and 1 PictureBox objects to create. Arrange as shown see the next few lines for additional information.

You will, initially, need 5 'sweets'. They are png files of size 44 x 44. They should have a transparency. CollectMe1 has the same image as Image1. Store these in the Sweetshop folder.

You will need 3 mp3 files attached to Audio1 – Audio3. Store these in the Sweetshop folder.

Version 0.10

Copy the code below (it is on 3 pages) into NSBasic…

```
Rem version 0.10
Dim NumberOfImages = 5
Dim AImage(NumberOfImages),Movehorizontal(NumberOfImages), Movevertical(NumberOfImages)
Dim timerRef, i, TopScore, Score, MoveHoriz, MoveVert, Border, pictbox1, GameEnd,
Collected, Collect, Lives
pictbox1 = PictureBox1.getContext("2d")
SetUp()

Rem *************** Code to create the Movement *****************
Function nextAction()
  UpdateNow()
  For L=1 To NumberOfImages
    MoveHoriz = Movehorizontal(L) : MoveVert = Movevertical(L)
    MoveImage(AImage(L))
    Movevertical(L) = MoveVert : Movehorizontal(L) = MoveHoriz
  Next

  i = i + 1
  Label3.textContent = "Seconds: " + CStr(CInt(i / 10))
```

```
  If GameEnd = 1 Then Stop()
End Function

Sub MoveImage(BYREF TheImage)
  TheImage.Left =  TheImage.Left + MoveHoriz

  If TheImage.Left + MoveHoriz < Form1.Left + Border Then
    MoveHoriz =  Abs(MoveHoriz)
    MoveVert = MoveVert + 1
    TheImage.Top =  TheImage.Top + 50
End If

  If TheImage.Left + MoveHoriz + Border > Form1.Width - TheImage.Width Then
    MoveHoriz = -1 * Abs(MoveHoriz)
    TheImage.Top =  TheImage.Top + 50
    MoveVert = MoveVert + 1
  End If

  TheImage.Top =  TheImage.Top + MoveVert
  If TheImage.Top + MoveVert < Form1.Top + Border Then MoveVert = Abs(MoveVert)
  If TheImage.Top + MoveVert + Border > Form1.Height - TheImage.Height Then
    Audio3.play()
    Lives = Lives - 1
    MoveVert = -1 * Abs(MoveVert)
End If

  If Lives < 1 Then GameEnd = 1
End Sub

Rem ******************* Main Code *************************

Function Button1_onclick()
  MsgBox("Version 0.10")
End Function

Sub Button1Display()
  $(Button1).css("font-size", "50px");
  Button1.resize(Border + 3 ,Border + 3, Form1.Width - (3+(Border * 2)), Form1.Height -
(3+(Border * 2)))
  Button1.value = "Start"
End Sub

Function CollectMe1_onclick()

End Function

Sub DisplayScore()

End Sub

Function Image1_onclick()
  ImageHit(Image1)
End Function

Function Image2_onclick()
  ImageHit(Image2)
End Function

Function Image3_onclick()
  ImageHit(Image3)
End Function

Function Image4_onclick()
  ImageHit(Image4)
End Function

Function Image5_onclick()
  ImageHit(Image5)
```

```
End Function

Function ImageHit(ImageClicked)

End Function

Function PictureBox1_onclick()

End Function

Sub SetUp()
  TopScore = 0
  Score = 0
  Border = 20

  AImage(1)=Image1
  AImage(2)=Image2
  AImage(3)=Image3
  AImage(4)=Image4
  AImage(5)=Image5
  For L = 1 To NumberOfImages
    AImage(L).hidden = True
  Next
  Label1.hidden = True
  Label2.textContent = "Press Start for the First game to commence.."
  Label3.hidden = True
  Label4.hidden = True
  CollectMe1.hidden = True
  PictureBox1.hidden = True
End Sub

Sub StartUp()

End Sub

Sub Stop()

End Sub

Function UpdateImage(ImageCode)

End Function

Sub UpdateNow()

End Sub
```

Now make sure the Objects have their Onclick set to the correct Function!

The program does not do much, apart from a message box appearing when Start is clicked!

Version 0.20

Now we can start to develop the code…

We will get button1 to toggle between the Start and Stop mode.

Modify This Function.
 	
It will make a decision depending on what Button1 says. If it says Start then Startup is called otherwise Stop() is called.
 	
Now we need to add code to StartUp and Stop!
	`Function Button1_onclick()`
	`  MsgBox("Version 0.10")`
	`  If Button1.value = "Start" Then`
	`  StartUp()`
	`  Else`
	`  Stop()`
	`  End If`
	
	`End Function`

StartUp changes the font size of button1, then it chooses any position and size for the button, before changing the text to read Stop.
 	
Stop on the otherhand calls a subroutine Button1Display. If you look at this subroutine you will see similar code to StartUp. The resize is more complex as it resizes it to a larger area that can change!!
	`Sub StartUp()`
	`  $(Button1).css("font-size", "10px");`
	`  Button1.resize(14,0,59,20)`
	`  Button1.value = "Stop"`
	`End Sub`
	
	`Sub Stop()`
	`  Button1Display()`
	`End Sub`

Test the code to see Start and Stop toggle as well as the button change size.

Now remove the Messagebox code from Button1_onclick!

Version 0.25

The Yellow Message to start the first game needs to be replaced by the Score.

To do this we need to enter code into Display Score that changes the text in Label 2. It also checks and fixes the Score so that it cannot go below 0.
 	
Add this code.
	`Sub DisplayScore()`
	`  If Score < 0 Then Score = 0`
	`  Label2.textContent = "Score: " + CStr(Score)`
	`End Sub`

In addition the call to this routine needs to be present in StartUp()
 	
The extra line has been added towards the bottom of the code, just above End Sub at this stage.
	`Sub StartUp()`
	`  $(Button1).css("font-size", "10px");`
	`  Button1.resize(14,0,59,20)`
	`  Button1.value = "Stop"`
	
	`  DisplayScore()`
	`End Sub`

Version 0.30

We now plan to start to make things move… This is handled in startup…

Label 3 is the time elapsed display, by setting hidden to False it will be displayed. The same is true for the item the game player is trying to collect.

The For L…. Next section this loops round all the images (presently 5, but that number can and will be changed later on). Every Image is displayed by the Hidden property becoming False. Then the initial movement is set. And the AImage Top / Left set the location.

Key variables are reset. GameEnd will be changed to 1 when we want the Game to end. Score I reset to 0. I is used to count the time and Lives will be lost when the fruit falls to the ground!

You must add the timeRef line. This sets up the Motion part of the code to work. A line later will stop the motion once we reach the Game End.

```
Sub StartUp()
  Label3.hidden = False
  CollectMe1.hidden = False

  For L = 1 To NumberOfImages
    AImage(L).hidden = False
    Movehorizontal(L) = -1
    Movevertical(L) = 0
    AImage(L).Left = Border + (L*100)
    AImage(L).Top = 100 + Border
  Next

  GameEnd = 0
  Score = 0
  i = 0
  Lives = 3

  $(Button1).css("font-size", "10px");
  Button1.resize(14,0,59,20)
  Button1.value = "Stop"
  timerRef = SetInterval(nextAction, 100)
  DisplayScore()

End Sub
```

Versions 0.31-0.39

After each version go back to 0.30 and try the next set of variations. Try to understand what each part does.

Experiment with the code given (only make 1 change and test each of these)…
0.31 What happens if you change the -1 to 1 or 0 or 2?
0.33 What happens if you change the 0 to -1 or 1 or -2 or 2?
0.35 What happens if you change (L*100) to (L*50) or (L*200) or (L*300)?
0.37 What happens if you change the 100 of 100 + Border? Try 90, 150, 50, 110, 200, 400
0.38 Try changing the 100 in timeRef – what does this do?

0.39 Suppose we want all objects to start with the same image. Add the highlighted line (shown on the right) just before the Next statement and below .Top….

```
    AImage(L).Top = 100 + Border
    AImage(L).src = "Move1.png"
  Next
```

Restore the code to V0.30 before moving on! Hopefully you have seen what controls the initial position, location and speed!

Version 0.40

Now we are going to create an Action on the Object being hit. The Action will be to increase the score by 1.

To do this make sure all Images have their on_Click property set and the code simply has the line ImageHit(ImageX) where X changes according to the Image Number. An example is shown below for Image 4…

```
Function Image4_onclick()
  ImageHit(Image4)
End Function
```

Now all we need to do is add the code to Function ImageHit() – find and adjust the code to…

```
Function ImageHit(ImageClicked)
  Score = Score + 1
  DisplayScore()
End Function
```

Test – Does the Score now change upon hitting any of the 5 Images? Try changing +1 to +2 or *2 – What happens to the score? Stop the game- can you still score after the game ends?

Version 0.41

Reset the code to just one point per hit. But now update the Function to…

```
Function ImageHit(ImageClicked)
  If Button1.value <> "Start" Then
    Score = Score + 1
  Else
    MsgBox ("Game over - you cannot score points.")
  End If
  DisplayScore()
End Function
```

By adding a decision, based on the text in the Start / Stop Button you can stop the score from being updated at the wrong point!

Now we can go further by if the object is hit, we can move it to a new position. This code is very technical and a bit tricky. So treat it as a “black box” for now and just let it work… this should be placed after DisplayScore() and before End Function…. Be very careful to check that you have typed it in 100% accurately!

```
ImageClicked.Left = Border + Int(Rnd()* (Form1.Width - (Label1.Width + Border)))
ImageClicked.Top = Border + Int(Rnd()* (Form1.height - (Label1.height + (Border*3))))
```

Version 0.42

We are now going to make a different noise depending on whether you have hit the object shown. That is the one that appears in the object CollectMe1. So we will need a decision (IF) that compares the image clicked with the CollectMe1. Add this code directly after Score = Score + 1..

```
  If CollectMe1.src = ImageClicked.src Then
    Audio1.play()
  Else
    Audio2.play()
  End If
```

Version 0.43

Now we can score points by moving the Score = Score + 1 line into either the Then or Else side of the decision. Move the Scoring line so that you only score a point if it is the same image as the one shown!

Version 0.45

Instead of adding to the score at this stage we are going to build a collection of items clicked on. Remove the Score = Score + 1 line.

Change the code so that it now reads… Lines like MsgBox have been deleted!

```
Function ImageHit(ImageClicked)
  If Button1.value <> "Start" Then
    If CollectMe1.src = ImageClicked.src Then
      CollectedUpdate(Collected + 1)
      Audio1.play()
    Else
      CollectedUpdate(0)
      Audio2.play()
    End If

    DisplayScore()
    ImageClicked.Left = Border + Int(Rnd()* (Form1.Width - (Label1.Width + Border)))
    ImageClicked.Top = Border + Int(Rnd()* (Form1.height - (Label1.height + (Border*3))))
  End If
End Function
```

Test and you will notice that the sound does not play when you click an object!

That is because we have not added the subroutine CollectedUpdate…

Add this code after the End Function of UpdateImage and before Sub UpdateNow()… Close to the end of your code.

```
Function UpdateImage(ImageCode)

End Function

Function CollectedUpdate(NewCollect)
  If NewCollect = 0 Then
    pictbox1.clearRect(100, 0, 44 * (Collected + 1), 44)
  Else
    pictbox1.addImage("Move" + CStr(Collect) +".png",100 +(Collected * 44),0,44,44)
  End If
  Collected = NewCollect
End Function

Sub UpdateNow()
```

This code is a partial “black box”. Test and see what happens.

We are back to where we were with sound playing. However the collection does not yet appear. We need to set all the details in StartUp…

On the lines above End Sub of Sub StartUp() add this code…

`CollectedUpdate(0)`
 	
`PictureBox1.Left = Border`
`PictureBox1.Top = Border`
 	
`PictureBox1.Height = Form1.Height - (Border * 2)`
`PictureBox1.Width = Form1.Width - (Border * 2)`
`PictureBox1.hidden = False`
 	
`Collect = 1`
`End Sub`
	This just resets the score to 0 (zero).
	These lines set the top & left sides to the top left of the screen but within Border (20 – as in SetUp). The next two lines set the bottom right corner. It is not to be hidden.
	Collect sets the image to Image1.
	
	End Sub was already present.

Test and you will see a Collection of Image1’s appear!

Version 0.50

We need a way to harvest the collection and score the points. To do this the player must click on the shape being shown, that is the Image being displayed in CollectMe1. Check that CollectME1 onclick property is set to the Function below and add the code… (The Function & End lines are already present!)

`Function CollectMe1_onclick()`
`  If Collected = 0 Then`
`  GameEnd = 1`
`  Audio3.play()`
`  Else`
`  Audio1.play()`
`  Score = Score + Collected`
`  CollectedUpdate(0)`
`  DisplayScore()`
`  End If`
 	
`End Function`
	If Nothing has been collected then the game ends!
	
	Else…
	Sound 1 plays,
	Score is adjusted
	and the collection is updated back to none.
	
	The score is displayed.

Version 0.55

Now we need to change the Image that is being Collected! To do this we add a line of code to the empty Sub UpdateNow() to read…

```
Sub UpdateNow()
  If Rnd() < 0.02 Then Collect = UpdateImage(1)
End Sub
```

The above code means that there is a 1 in 5 that is 1/5 = 0.2 as a decimal chance that it will change.

And we need the Function below (presently empty so find and enter its code…

```
Function UpdateImage(ImageCode)
  Dim UpImage = new Object
  Dim location
  location = "CollectMe" + Right("0"+CStr(ImageCode),1)
  UpImage = Eval(location)

  ImageCode = 1 + Int(Rnd()*NumberOfImages)
  UpdateImage = ImageCode
  UpImage.src = Eval("Image" +CStr(ImageCode)+".src")
  pictbox1.clearRect(100, 100, 440, 440)
  pictbox1.addImage("Move" +CStr(ImageCode)+".png",100,100,440,440)
End Function
```

This is a partial black box, but it selects the CollectMe Object (first 4 lines) Then selects a random number upto the NumberOfImages and places that in ImageCode – this is also the value returned by the Function. Then the line starting UpImage.src changes the Image. The last 2 lines clear an area of Picturebox1 before adding a bigger copy of the image to be added to PictureBox1 which is in the background of the window displayed!

Version 0.60

If you miss one of the targets then you probably will be clicking on PictureBox1. This can be used to reduce the score or just play a negative beep type sound.

We should test to see if the Game is in play (IF), and if true we will play Audio3…	`Function PictureBox1_onclick()`
	`  If Button1.value <> "Start" Then`
	`  Audio3.play()`
	`  End If`
	
	`End Function`

Version 0.61

Add two further lines within the IF statement added in Version 0.60 to reduce the score by 1 and redisplay the score by calling an appropriate function. Hint: Look at Version 0.41 for ideas.

Version 0.70

We have introduced many changes to the code which have increased what the program can do as well as the way that it does this. This will lead to old code that can be improved further with minor “tweaks”. This is an area that many programmers do not like going back to – probably from the view it works so why change it further! However we will as this is good practise. Another key skill is in checking code – you probably have experienced this whilst entering this program and others before

it! This is a chance to practise….

One line has changed – spot it and change it. To help some lines are longer than displayed – it is not those lines!

Old Version

```
Sub StartUp()
  Label3.hidden = False
  CollectMe1.hidden = False

  For L = 1 To NumberOfImages
    AImage(L).hidden = False
    Movehorizontal(L) = -1
    Movevertical(L) = 0
    AImage(L).Left = Border + (L*100)
    AImage(L).Top = 100 + Border
  Next


  GameEnd = 0
  Score = 0
  i = 0
  Lives = 3

  $(Button1).css("font-size", "10px");
  Button1.resize(14,0,59,20)
  Button1.value = "Stop"
  timerRef = SetInterval(nextAction, 100)
  DisplayScore()

  CollectedUpdate(0)

  PictureBox1.Left = Border
  PictureBox1.Top = Border

  PictureBox1.Height = Form1.Height - (Bor
  PictureBox1.Width = Form1.Width - (Borde
  PictureBox1.hidden = False

  Collect = 1
End Sub
```

Latest Version 0.70

```
Sub StartUp()
  Label3.hidden = False
  CollectMe1.hidden = False

  For L = 1 To NumberOfImages
    AImage(L).hidden = False
    Movehorizontal(L) = -1
    Movevertical(L) = 0
    AImage(L).Left = Border + (L*100)
    AImage(L).Top = 100 + Border
  Next


  GameEnd = 0
  Score = 0
  i = 0
  Lives = 3

  $(Button1).css("font-size", "10px");
  Button1.resize(14,0,59,20)
  Button1.value = "Stop"
  timerRef = SetInterval(nextAction, 100)
  DisplayScore()

  CollectedUpdate(0)

  PictureBox1.Left = Border
  PictureBox1.Top = Border

  PictureBox1.Height = Form1.Height - (Bor
  PictureBox1.Width = Form1.Width - (Borde
  PictureBox1.hidden = False

  Collect = UpdateImage(1)
End Sub
```

Did you notice how blank lines helped you with reading your code?

Version 0.80

The game is looking quite good, but there are problems when them game ends or Stops. At this point Sub Stop() is called and all it presently does is to change the Button1Display word!

What does it need to do?

The list below is perhaps what we need to do…

1. Stop the objects moving.
2. Change Label3 to say that the Game is Over.
3. If the Score is greater than the top Score then update the Top Score.

4. Change the Start/Stop button (present in the code!)

How do we do each…

ClearInterval(timerRef)	This will stop the movements!
Label3.textContent = "Game over."	Changes the Label to say what we want!

If Score > TopScore Then
TopScore = Score
Label4.textContent = "Top Score is " + TopScore
Label4.hidden = False
End If
	The decision..
	Set the Highest score
	Display the new High Score
	Make sure it is not hidden
	End the decision

Now modify Sub Stop() – don't forget that the sequence matters and that the last bit is to do the part that the code is already doing!

Test your changes. If happy call this version 1.0!

Ideas for development… Post Version 1.00

Animation upon a hit / reduces size?

To do this we need to adjust HitImage…

```
Function ImageHit(ImageClicked)
  If Button1.value <> "Start" Then
    If CollectMe1.src = ImageClicked.src Then
      CollectedUpdate(Collected + 1)
      Audio1.play()
    Else
      CollectedUpdate(0)
      Audio2.play()
    End If

    DisplayScore()
    'ImageClicked.Left = Border + Int(Rnd()* (Form1.Width - (Label1.Width + Border)))
    'ImageClicked.Top = Border + Int(Rnd()* (Form1.height - (Label1.height + (Border*3))))
    ImageClicked.Width = ImageClicked.Width - 2
  End If
End Function
```

And… Function nextAction()

```
Function nextAction()
  UpdateNow()
  For L=1 To NumberOfImages
    MoveHoriz = Movehorizontal(L) : MoveVert = Movevertical(L)
    MoveImage(AImage(L))
    Movevertical(L) = MoveVert : Movehorizontal(L) = MoveHoriz
    If AImage(L).Width < 5 Then
      AImage(L).Width = 44
      AImage(L).Left = Border + Int(Rnd()* (Form1.Width - (Label1.Width + Border)))
      AImage(L).Top = Border + Int(Rnd()* (Form1.height - (Label1.height + (Border*3))))
    End If
    If AImage(L).Width < 44 Then AImage(L).Width = AImage(L).Width - 2
  Next
```

You can also change the height. The -2 can be -1 for a slower change or -3 for a faster change. The widths / heights will need to be reset in StartUp otherwise the game restarts with smaller images.

Items re-appear too close to the bottom of the screen…

The line that ends…

```
.Top = Border + Int(Rnd()* (Form1.height - (Label1.height + (Border*3))))
```

Controls how far down the screen an image may appear. Yet the calculation is far from ideal..

Try…

```
.Top = Border + Int(Rnd()* (PictureBox1.height - 40))
```

This should be 'better' as the top will at least be at the Border, plus a random number upto 40 less than the height of PictureBox1 which has been set to the background of the 'field of play'. However the left position used a similar formula. This too could / should be adjusted.

```
.Left = Border + Int(Rnd()* (Form1.Width - (Label1.Width + Border)))
```

The name of the Top Scorer is entered and displayed on screen.

The Routine Stop() needs to be updated to collect in the Player's name and display it. Try this code…

```
Sub Stop()
  Dim TopScoreName
  ClearInterval(timerRef)
  Label3.textContent = "Game over."
  If Score > TopScore Then
    TopScore = Score
    TopScoreName=InputBox("New Top Score - Enter Your Name")
    Label4.textContent =  "Top Score by " + TopScoreName +"=" + TopScore
    Label4.hidden = False
  End If

  Button1Display()
End Sub
```

You may need to resize or reposition the Label items on the screen / form.

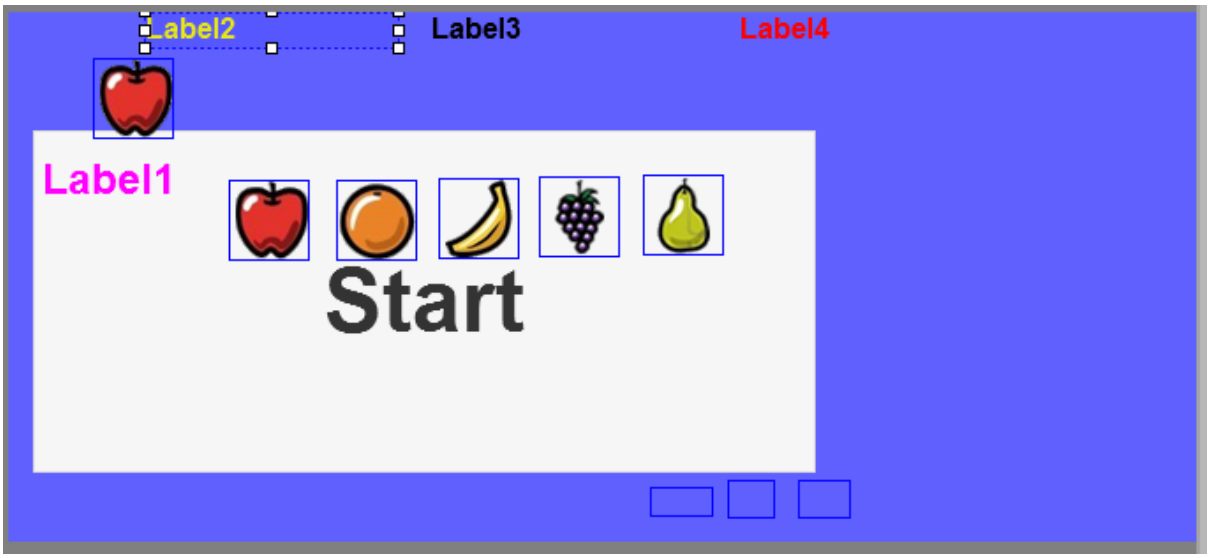
Display the number of Lives Left

Three changes are needed to achieve this. (v1-20)

1. Add the extra details to the Label that displays the score.

```
Sub DisplayScore()
  If Score < 0 Then Score = 0
  Label2.textContent = "Score: " + CStr(Score) + " Lives: " + CStr(Lives)
End Sub
```

2. Realign / position the labels (Make the Form wider too).



3. Add DisplayScore() code to where Lives are lost… (Find "Lives -")

```
If TheImage.Top + MoveVert + Border > Form1.Height - TheImage.Height Then
  Audio3.play()
  Lives = Lives - 1
  DisplayScore()
  MoveVert = -1 * Abs(MoveVert)
```

Bug: Sometimes all Lives are lost

This can occur if an item remains below or on the bottom of the game area from one screen update to the next. To attempt to solve this we must force the item to be moving up (Which means MoveVert must be -1 **or lower**) and be above the bottom line (So while it is below we must keep moving it up). (v1-21)

```
If TheImage.Top + MoveVert + Border > Form1.Height - TheImage.Height Then
  Audio3.play()
  Lives = Lives - 1
  DisplayScore()
  MoveVert = -1 * Abs(MoveVert)
  If MoveVert > -1 Then MoveVert = -1
  Do While TheImage.Top + MoveVert + Border >= Form1.Height - TheImage.Height
    TheImage.Top =  TheImage.Top + MoveVert
  Loop
End If
```

Award Extra Lives

Lives = Lives + 1 is the code to increase lives. The question is when to reward the play with an extra life. Then we need to enable the computer to make that decision.

Let's say it is when you "bank" or "cash in" 5 identical "sweets". We will need to keep a separate count which is reset to zero when the CollectMe1 image is updated. We need a new area to store this counter in and a name for it. Amend

```
Dim timerRef, i, TopScore, Score, MoveHoriz, MoveVert, Border, pictbox1, GameEnd, Collected, Collect, Lives, ExtraLife
```

So that it has a comma and ExtraLife at the end. ExtraLife will be our variable Counter. This needs to be set to zero every time the image is amended. This happens in UpdateNow(). So we will need to modify this to…

```
Sub UpdateNow()
  If Rnd() < 0.02 Then Collect = UpdateImage(1) : ExtraLife = 0
End Sub
```

Now we need to increase by 1, every time and object is added to the store… This happens inside ImageHit() Which can now be updated to…

```
Function ImageHit(ImageClicked)
  If Button1.value <> "Start" Then
    If CollectMe1.src = ImageClicked.src Then
      CollectedUpdate(Collected + 1)
      ExtraLife = ExtraLife + 1
      Audio1.play()
    Else
      CollectedUpdate(0)
      Audio2.play()
    End If

    DisplayScore()
    ImageClicked.Width = ImageClicked.Width - 2
  End If
End Function
```

Now we can add the decision… (v1-25)

```
Function CollectMe1_onclick()
  If Collected = 0 Then
    GameEnd = 1
    Audio3.play()
  Else
    Audio1.play()
    Score = Score + Collected
    If ExtraLife >= 5 Then Lives = Lives + 1
    CollectedUpdate(0)
    DisplayScore()
  End If

End Function
```

Extend the Number of Images

Not that many changes to make per extra Image.

Create the extra Image in NSBasic. (ie Image6 < They do need to go up in sequence). Add the ‘new’ graphic to be shown, that is change the source (src) property (It must be a new image – not for NSBasic, but for the game as the same image will cause a problem with the collection sign). The

filename **must be** Move6.png.

Select the OnClick so that it points to Image6 onclick…

onclick	Image6_onclick()

Change the code to read…

```
Function Image6_onclick()
  ImageHit(Image2)
End Function
```

Now add the line AImage(6)=Image6 into SetUp() where shown…

```
  AImage(4)=Image4
  AImage(5)=Image5
  AImage(6)=Image6
  For L = 1 To NumberOfImages
    AImage(L).hidden = True
```

Finally change the variable NumberOfImages to 6 at the start…

```
Rem version 1.30
Dim NumberOfImages = 6
Dim AImage(NumberOfImages),Movehor
```

Repeat for 7, 8 & 9. Will 10 work?? (v1-30)

Bug: Game ends quickly on narrow screen.

If you make a narrow window so that any of the target objects are outside the game ends rapidily…

For example…

As shown on the right…

The fruit to the right of the screen stay of the screen and their down speed build rapidly. The game ends before it has start!

The problem comes from lines inside Sub MoveImage(BYREF TheImage). Use the ‘ symbol to rem out the key lines… (They will turn green…)

```
  TheImage.Top =  TheImage.Top + 50
End If

If TheImage.Left + MoveHoriz + Border > Form1.Width - TheImage.Width Then
  MoveHoriz = -1 * Abs(MoveHoriz)
  'TheImage.Top =  TheImage.Top + 50
  'MoveVert = MoveVert + 1
End If

TheImage.Top =  TheImage.Top + MoveVert
If TheImage.Top + MoveVert < Form1.Top + Border Then MoveVert = Abs(MoveVert)
```

Test again and it is not a problem. So why is the code there? It is to stop objects going off screen to the right. It will bounce many back, but if they are outside then it is a problem. The solution is to test for the object being outside and moving to the right. That is MoveHoriz is greater than zero..

Try this revised code…. Then rem statements are removed and a 2nd clause to the IF statement is added…

```
  TheImage.Top =  TheImage.Top + 50
End If

If TheImage.Left + MoveHoriz + Border > Form1.Width - TheImage.Width And MoveHoriz > 0 Then
  MoveHoriz = -1 * Abs(MoveHoriz)
  TheImage.Top =  TheImage.Top + 50
  MoveVert = MoveVert + 1
End If

TheImage.Top =  TheImage.Top + MoveVert
If TheImage.Top + MoveVert < Form1.Top + Border Then MoveVert = Abs(MoveVert)
```

Test again and see the objects move back onto the screen.

More than 1 of the same Sweet

The game could start with all objects containing the image Move1.png….
When you click on the image it reappears as the next item. That is Move1 changes to Move2… Move 2 to Move3 and so on. The maximum one, presently Move6 or Move10 either remains as is or reverts back to Move1!

This has the potential to lead to a whole variety of different rules and scoring systems!

We must try to set all the images to Move1.png within StartUp(). This is a case of adding 1 line…

Spot the new line below and add…

```
Sub StartUp()
  Label3.hidden = False
  CollectMe1.hidden = False

  For L = 1 To NumberOfImages
    AImage(L).hidden = False
    Movehorizontal(L) = -1
    Movevertical(L) = 0
    AImage(L).Left = Border + (L*100)
    AImage(L).Top = 100 + Border
    AImage(L).Width = 44
    AImage(L).Height = AImage(L).Width
    AImage(L).src = "Move1.png"
  Next
```

Now we need some more complex code to change the Image once clicked.

In the current version the shape disappears and then re-appears in a new place. It is at this point that we need to change it to Move2.png…

Again find the added line in the revised code below…

```
Function nextAction()
  UpdateNow()
  For L=1 To NumberOfImages
    MoveHoriz = Movehorizontal(L) : MoveVert = Movevertical(L)
    MoveImage(AImage(L))
    Movevertical(L) = MoveVert : Movehorizontal(L) = MoveHoriz
    If AImage(L).Width < 1 Then
      AImage(L).Width = 44
      AImage(L).Height = AImage(L).Width
      AImage(L).Left = Border + Int(Rnd()* (Form1.Width - (Label1.Width + Border)))
      AImage(L).Top = Border + Int(Rnd()* (PictureBox1.height - 40))
      AImage(L).src = "Move2.png"
    End If
    If AImage(L).Width < 44 Then
      AImage(L).Width = AImage(L).Width - 1
      AImage(L).Height = AImage(L).Width
    End If
  Next
```

Test your code (v1.35). This is fine to change the image from Move1 to Move2. Yet much more work is needed to change to the next number in the sequence…

As it is more complex we will create a new Function that carries this out for us. The Function will be called ChangeImage. It will need to know the present Image name, that is what is presently stored in the .src property (src stands for Source as in the Source Image file).

Now change the recently added line…

```
AImage(L).src = "Move2.png"
```

to

```
AImage(L).src = ChangeImage(AImage(L).src)
```

The code will not work until we define the Function ChangeImage… Add this code

```
Function ChangeImage(TheOldFile)
  ChangeImage = "Move2.png"
End Function
```

Make sure you place it after any existing End Sub or End Function. If it is situated within one it will not work! One place for this is after the End Sub of Sub Button1Display() and before the Function CollectMe1_onclick() – why – no programming logic, just that the Functions and Subs have generally been ordered alphabetically!

This is v1.36 and works the same as v1.35! However we can now develop the function to move the image on in the planned sequence…. We need to calculate the 'number' of the image we are presently on. TheOldFile is just a string of letters and other characters and needs to be cut down to just the number. To do this need to know the value after "Move" and before ".png". This takes 2 lines ① cuts off everything to the 4 place after "Move" (hence the 4 +) then ② just takes everything before the "." (hence the – 1). In ③ the new FileNumber becomes 1 more, but we need to Cint (Convert to Integer) first. Finally line ④ checks to see if the number is too large and if so resets it back to 1. Modify the code as shown below…

```
Function ChangeImage(TheOldFile)
  Dim FileNumber
①  TheOldFile = Mid(TheOldFile,4+InStr(TheOldFile,"Move"),3)
②  TheOldFile = Left(TheOldFile, InStr(TheOldFile,".") - 1)
③  FileNumber = 1 + CInt(TheOldFile)
④  If FileNumber > NumberOfImages Then FileNumber = 1
  ChangeImage = "Move" + CStr(FileNumber) +".png"
End Function
```

This is v1.40. You can try V1.41 where the IF statement changes to…

```
If FileNumber > NumberOfImages Then FileNumber = NumberOfImages
```

What difference does this change make?

Different sweets score different points.

Now that the Sweets change on the click we can award different points.

We will award 1 point for Move1.png, 2 for Move2.png and so on.

Once again we need to extract the number from the source (src) of the object clicked on. The code to do this already exists from the last amendment. We have two choices – either to copy and paste the code or (and from a programming point of view) a better way is to create a Function to do it for us. This function can be used in multiple locations – once it works!

We will name this function FileImageNumber we need to pass the source to it which we will name FileSource….

```
Function FileImageNumber(FileSource)
  FileSource = Mid(FileSource,4+InStr(FileSource,"Move"),3)
  FileSource = Left(FileSource, InStr(FileSource,".") - 1)
  FileImageNumber = CInt(FileSource)
End Function
```

Hint: Place this function alphabetically - so between DisplayScore and Image1_onclick.

Now to test we can modify ChangeImage to…

```
Function ChangeImage(TheOldFile)
  Dim FileNumber
  FileNumber = 1 + FileImageNumber(TheOldFile)
  If FileNumber > NumberOfImages Then FileNumber = NumberOfImages
  ChangeImage = "Move" + CStr(FileNumber) +".png"
End Function
```

This is V1.42 and behaves the same as V1.41. At least we know that the new Function works and we can use this to help with changes to the score.

The Score is increased by the number of items Collected. Collected is incremented by 1.

Modify…

```
Function CollectedUpdate(NewCollect)
  If NewCollect = 0 Then
    pictbox1.clearRect(100, 0, 44 * (Collected + 1), 44)
    Collected = NewCollect
  Else
    pictbox1.addImage("Move" + CStr(Collect) +".png",100 +(Collected * 44),0,44,44)
    Collected = Collected + 1
  End If
```

This is V1.50 and test that it works in the same way. That is one point for those items collected.

If you have not noticed before there is a problem with the image that is displayed – this bug came into the code as we only partially finished changing the program to use all the same images! Again the new Function solves the problem Check and adjust the two lines ✓ (ticked) below… (V1.51).

```
Function UpdateImage(ImageCode)
  Dim UpImage = new Object
  Dim location
  location = "CollectMe" + Right("0"+CStr(ImageCode),1)
  UpImage = Eval(location)

  ImageCode = 1 + Int(Rnd()*NumberOfImages)
  UpdateImage = ImageCode
  UpImage.src = Eval("Image" +CStr(ImageCode)+".src")
  pictbox1.clearRect(100, 100, 440, 440)
✓ pictbox1.addImage("Move" +CStr(FileImageNumber(UpImage.src))+".png",100,100,440,440)
End Function

Function CollectedUpdate(NewCollect)
  If NewCollect = 0 Then
    pictbox1.clearRect(100, 0, 44 * (Collected + 1), 44)
    Collected = NewCollect
  Else
✓   pictbox1.addImage("Move" + CStr(FileImageNumber(CollectMe1.src)) +".png",100 +(Collected * 44),0,44,44)
    Collected = Collected + 1
  End If
```

Now we can adjust the scoring in CollectedUpdate from Collected = Collected + 1 to… (v1.55).

```
Collected = Collected + FileImageNumber(CollectMe1.src)
```

This has now caused a 2nd problem. As it is fine when Move1.png's are collected, but the spacing spreads out when Move2.png or Move3.png are collected. This is because Collected had 2 roles. To count the points of those items collected (which it does successfully), but also to workout the position of this item on the screen. It is this later role that we have caused the problem with! We need to do both and that they now have different roles the only option is to have two separate counters or in programming speak variables. As the points are working fine, we just need to create the new variable for CollectedLocation and manage when this changes and is used. This will be a new Global variable and will need to be added to the Dim line at the top of the program…

```
Dim timerRef, i, TopScore, Score, MoveHoriz, MoveVert, Border, pictbox1, GameEnd, Collected,
CollectedLocation, Collect, Lives, ExtraLife
```

This line is a single line rather than as shown.

Now there are 2 new lines and one line changed… in Function CollectedUpdate

```
Function CollectedUpdate(NewCollect)
  If NewCollect = 0 Then
    pictbox1.clearRect(100, 0, 44 * (Collected + 1), 44)
    Collected = NewCollect
    CollectedLocation = 0
  Else
    pictbox1.addImage("Move" + CStr(FileImageNumber(CollectMe1.src)) +".png",100 +(CollectedLocation * 44),0,44,44)
    Collected = Collected + FileImageNumber(CollectMe1.src)
    CollectedLocation = CollectedLocation + 1
  End If

End Function
```

This now gives you a working v1.56.

Suppose we want to have our own scoring system rather than progressing through the numbers and points in sequence…

We will need to score the points associated with each image and then add that number on.

Image Filename	Image Index	Points
Move1.png	1	1
Move2.png	2	2
Move3.png	3	4
Move4.png	4	8
Move5.png	5	12
Move6.png	6	16

To implement this table as our scoring system it is perhaps best to use an array which we will call TargetPoints! The size of the table (number of Rows) is controlled by the NumberofImages! Amend the Dim line…

```
Dim AImage(NumberOfImages),Movehorizontal(NumberOfImages), Movevertical(NumberOfImages), TargetPoints(NumberOfImages)
```

Now we need to populate the array with the points. Add these 6 extra lines to the bottom of Sub SetUp()	PictureBox1.hidden = True TargetPoints(1)=1 TargetPoints(2)=2 TargetPoints(3)=4 TargetPoints(4)=8 TargetPoints(5)=12 TargetPoints(6)=16 End Sub

Then we must change the way Collected is increased. Spot the one line that has changed and amend..

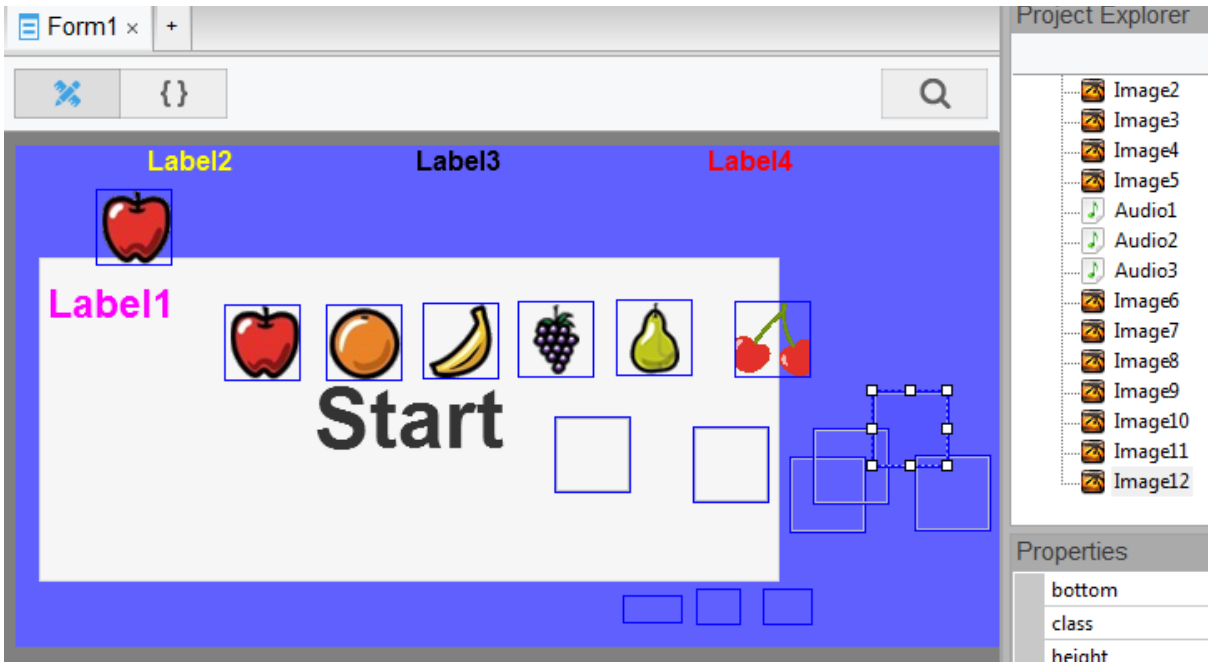
```
Function CollectedUpdate(NewCollect)
  If NewCollect = 0 Then
    pictbox1.clearRect(100, 0, 44 * (Collected + 1), 44)
    Collected = NewCollect
    CollectedLocation = 0
  Else
    pictbox1.addImage("Move" + CStr(FileImageNumber(CollectMe1.src)) +".png",100 +(CollectedLocation * 44),0,44,44)
    Collected = Collected + TargetPoints(FileImageNumber(CollectMe1.src))
    CollectedLocation = CollectedLocation + 1
  End If

End Function
```

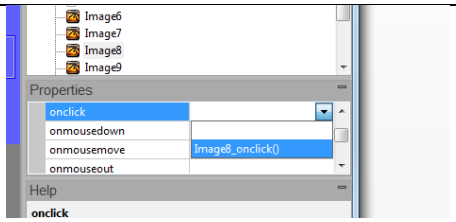
This gives a working v1.60!

More Sweets but less flavours!

Due to the changes made it is now even easier to have more images on screen and without the need to create a new pictorial file (Move7.png) to represent the “flavour”. To do this we need to divide the number of Image Objects from the pictorial file. Suppose we want to have 6 flavours as we have from Move1 to Move6, but with 12 Image objects / targets moving around the screen.

Firstly create the extra Images by dragging and dropping the Image object onto the Form…



For Each Object set the OnClick…	

Unlike earlier you do not need to set the src!
 	
You do need to enter the onclick code…
	``
	Function Image7_onclick()
	  ImageHit(Image7)
	End Function
	
	Function Image8_onclick()
	  ImageHit(Image8)
	End Function
	
	Function Image9_onclick()
	  ImageHit(Image9)
	End Function
	
	Function Image10_onclick()
	  ImageHit(Image10)
	End Function
	
	Function Image11_onclick()
	  ImageHit(Image11)
	End Function
	
	Function Image12_onclick()
	  ImageHit(Image12)
	End Function
	``

In Sub SetUp change from below ``` border = 20 AImage(1)=Image1 AImage(2)=Image2 AImage(3)=Image3 AImage(4)=Image4 AImage(5)=Image5 AImage(6)=Image6 For I = 1 To NumberOfImages ```	To

```
Border = 20

AImage(1)=Image1
AImage(2)=Image2
AImage(3)=Image3
AImage(4)=Image4
AImage(5)=Image5
AImage(6)=Image6
AImage(7)=Image7
AImage(8)=Image8
AImage(9)=Image9
AImage(10)=Image10
AImage(11)=Image11
AImage(12)=Image12
For I = 1 To NumberOfImages
```

Then Change the Dim NumberOfImages to 12 and add a line about Targets….

```
Rem version 1.60
Dim NumberOfImages = 6
```

```
Rem version 1.70
Dim NumberOfImages = 12
Dim NumberOfTargets = 6
```

Then change the IF line in this Function…

```
Function ChangeImage(TheOldFile)
  Dim FileNumber
  FileNumber = 1 + FileImageNumber(TheOldFile)
  If FileNumber > NumberOfTargets Then FileNumber = NumberOfTargets
  ChangeImage = "Move" + CStr(FileNumber) +".png"
End Function
```

To give a working V1.70.

The items randomly change speed / direction upon some event.

Try this amendment… (v1.80)

```
Sub UpdateNow()
  Dim Chosen
  Chosen = 1 + Int(NumberOfImages * Rnd())
  If Rnd() < 0.02 Then Collect = UpdateImage(1) : ExtraLife = 0
  If Rnd() < 0.9 Then Movehorizontal(Chosen) = Movehorizontal(Chosen) + Int(5 * Rnd()) - 2

End Sub
```

The line above End Sub has a 0.9 a 5 and a 2. These all play a vital role in the change of left / right movement. Every 100 / 1000 of a second this subroutine is carried out. Chosen selects one of the NumberOfImages (presently 12). The 0.9 means that there is 9 chances in 10 of the horizontal movement being changed. The “Int(5 * Rnd()) – 2” changes this from -2 to +2. The 5 and 2 numbers are related otherswise objects will move more quickly to either the right or left. The higher number

should be double the lower plus 1 to stop this. 0.9 is probably too high for the game but shows the effect. Experiment with values from 0.01 to 0.80.

In addition to Left / Right… Up / down speed can be varied – Try V1.81

```
Sub UpdateNow()
  Dim Chosen
  Chosen = 1 + Int(NumberOfImages * Rnd())
  If Rnd() < 0.02 Then Collect = UpdateImage(1) : ExtraLife = 0
  If Rnd() < 0.9 Then Movehorizontal(Chosen) = Movehorizontal(Chosen) + Int(4 * Rnd()) - 2
  If Rnd() < 0.9 And Score > 10 Then Movevertical(Chosen) = Movevertical(Chosen) + Int(4 * Rnd()) - 1
End Sub
```

By testing the score in the IF statement the effect only happens once the user has progressed to a score of more than 10. This time the 4 and 1 numbers mean that +2 to -1 are produced so the objects will move down more quickly and the number lives will be under threat, thus making the game harder at this point.

A variety of this If statement could be used to create harder levels as the score progresses.

If you want different objects to change each 1/10 of a second then reselect a Chosen value before the IF statement..

Below is a more subtle and Complex variation (v1-88) though even more is possible..

```
Sub UpdateNow()
  Dim Chosen
  If Rnd() < 0.02 Then Collect = UpdateImage(1) : ExtraLife = 0

  Chosen = 1 + Int(NumberOfImages * Rnd())
  If Rnd() < 0.1 And Score > 10 Then Movehorizontal(Chosen) = Movehorizontal(Chosen) + Int(3 * Rnd()) - 1
  'If Rnd() < 0.1 And Score > 10 Then Movevertical(Chosen) = Movevertical(Chosen)
  Chosen = 1 + Int(NumberOfImages * Rnd())
  If Rnd() < 0.3 And Score > 50 Then Movehorizontal(Chosen) = Movehorizontal(Chosen) + Int(5 * Rnd()) - 2
  If Rnd() < 0.3 And Score > 70 Then Movevertical(Chosen) = Movevertical(Chosen) + Int(2 * Rnd()) - 0
  Chosen = 1 + Int(NumberOfImages * Rnd())
  If Rnd() < 0.4 And Score > 100 Then Movehorizontal(Chosen) = Movehorizontal(Chosen) + Int(7 * Rnd()) - 3
  If Rnd() < 0.4 And Score > 100 Then Movevertical(Chosen) = Movevertical(Chosen) + Int(4 * Rnd()) - 1

End Sub
```

Using Images other than of width 44.

This is possible, but will require many changes. The program was written with 44 “hard wired in to the code”. You can search and replace 44 with either a different size or a variable (ImageSize). Take care not to change 440 for example. 44 needs to be changed in 8 locations

You should use a variable this will need to be set at the top of the code with a Dim statement.

```
Dim NumberOfTargets = 6
Dim ImageSize = 44
Dim AImage(NumberOfImages),Mo
```

It will then be much easier to modify the width! Test with say 88 (V1-89) – Back to 44 for (v1-90)! Although the size changes the quality of the graphic will be distorted by any resizing carried out by NSBasic. Ideally your image sizes and the setting of ImageSize should be the same.

It is possible, but very complex, to have each image at a varying size. Likewise you could have two variables ImageWidthSize and ImageHeightSize to control non square shaped images.

Too much downward bounce on Sidewall

At the start when the object hits the left wall it moves down and then continues it journey. The first move down is too much and this can cause lots of lives to be lost later in the game. That is when a 'sweet' reappears quite low down on the screen.

This move down code has been present since the start. Look in Sub MoveImage() you should see this line appearing twice… *"TheImage.Top = TheImage.Top + 50".* Try changing the fifty to smaller values or zero. An alternative to zero would be to rem out the two lines. Test to see what works best. The Remmed out Version is 1.91.

Rate of Descent to quick in early stages..

Once the left wall is reached the 'sweets' descend at an angle of 45 degrees. This is perhaps too steep in the early stages of the game. There are two lines which presently control this. They are in Sub MoveImage(). … *"MoveVert = MoveVert + 1"* Try alternative values. You should find any whole number larger increases the rate of decent. 0 Means no decent and the objects just bounce from left to right. Fractional options either descend at a rate the same as 0 or 1. As pixels are measured in whole units you cannot use a fraction.

There is a solution, but it means much more coding changes as the Vertical movement and position needs to store the values between 0 and 1! In NSBasic the Images Top (and Left if finer horizontal movement is needed) property have to be whole numbers. To keep fractional parts this will need to be managed by additional storage and code. We need a TrueTop position for each image. The calculations are then held in here and only transferred to the Images Property at the end.

We can add TrueTop with a store for each image by changing this line

```
Dim AImage(NumberOfImages),Movehorizontal(NumberOfImages), Movevertical(NumberOfImages), Targe
```

To

```
Dim AImage(NumberOfImages), TrueTop(NumberOfImages), Movehorizontal(NumberOfImages), Movevertical(NumberOfImages), TargetPo
```

The location on the line does not matter.

Likewise a single variable TrueTp needs to be added to the Dim below…

From

```
Dim timerRef, i, TopScore, Score, MoveHoriz, MoveVert, Border, pictbox1, GameEr
```

To

```
Dim timerRef, i, TopScore, Score, TrueTp, MoveHoriz, MoveVert, Border, pictbox1, GameEr
```

Now wherever the Top Property of the Images is changed we need to do two things.

1) Update TrueTop first of all then 2) place TrueTop into .Top! As TrueTop will keep the fractional part the rounding will only occur at the end.

We shall start in Sub StartUp

Old Code…	New Code…

```
For L = 1 To NumberOfImages
  AImage(L).hidden = False
  Movehorizontal(L) = -1
  Movevertical(L) = 0
  AImage(L).Left = Border + (L*100)
  AImage(L).Top = 100 + Border
  AImage(L).Width = ImageSize
  AImage(L).Height = AImage(L).Width
  AImage(L).src = "Move1.png"
Next
```

```
For L = 1 To NumberOfImages
  AImage(L).hidden = False
  Movehorizontal(L) = -1
  Movevertical(L) = 0
  AImage(L).Left = Border + (L*100)
  TrueTop(L) = 100 + Border
  AImage(L).Top = TrueTop(L)
  AImage(L).Width = ImageSize
  AImage(L).Height = AImage(L).Width
  AImage(L).src = "Move1.png"
Next
```

Then in Function nextAction()…

Old Code…

```
For L=1 To NumberOfImages
  MoveHoriz = Movehorizontal(L) : MoveVert = Movevertical(L)
  MoveImage(AImage(L))
  Movevertical(L) = MoveVert : Movehorizontal(L) = MoveHoriz
```

New Code..

```
For L=1 To NumberOfImages
  TrueTp = TrueTop(L) : MoveHoriz = Movehorizontal(L) : MoveVert = Movevertical(L)
  MoveImage(AImage(L))
  TrueTop(L) = TrueTp : Movevertical(L) = MoveVert : Movehorizontal(L) = MoveHoriz
```

This transfers the value from the Lth Image being processed into the variable TrueTp, Moves the Image and then stores the value back in TrueTop(L) from TrueTp!

This means that in MoveImage the TrueTop is represented by TrueTp… So we make these changes..

From… `TheImage.Top = TheImage.Top + MoveVert`

To…

```
TrueTp = TrueTp + MoveVert
TheImage.Top = TrueTp
```

Now the Two lines which update MoveVert can have a fractional part. Test with 0.3 (as shown below) and with other values…

```
If TheImage.Left + MoveHoriz < Form1.Left + Border Then
  MoveHoriz = Abs(MoveHoriz)
  MoveVert = MoveVert + 0.3
   'TheImage.Top = TheImage.Top + 50
End If

If TheImage.Left + MoveHoriz + Border > Form1.Width - TheImage.Width And MoveHoriz > 0 Then
  MoveHoriz = -1 * Abs(MoveHoriz)
  'TheImage.Top = TheImage.Top + 50
  MoveVert = MoveVert + 0.3
End If
```

The green Remmed out code can be deleted. This gives V1.92!

Sweets Objects reappear to close to the bottom of the screen.

Randomly on re-appearing many objects end up very close to the bottom of the screen. This might be ideal for more advance players but frustrating for the new player as the game comes to a rapid end without any opportunity for the player to adapt.

The code where an item is repositioned is within Function nextAction()…

Presently..

```
AImage(L).Top = Border + Int(Rnd()* (PictureBox1.height - 40))
```

We need to modify it so that the range of numbers are less. One way would be to divide PictureBox1.height by 2 rather than taking a fixed value off. This will force the object into the top half of the screen…

```
AImage(L).Top = Border + Int(Rnd()* (PictureBox1.height / 2))
```

Test this.

Now we could have a more graduated system with a series of extra decisions.

The Decision could be based on Score (or Lives left)…. The one line becomes 3 or more lines!

```
AImage(L).Top = Border + Int(Rnd()* (PictureBox1.height - (2*ImageSize)))
If Score < 50 Then AImage(L).Top = Border + Int(Rnd()* (PictureBox1.height / 2))
If Score < 10 Then AImage(L).Top = Border + 100
```

Rather than taking away a fixed 40, the code now takes away twice the ImageSize so items will not appear too low down and terminate the game. The next two lines make decisions. Logically, it is important to modify the higher limit scores first. So All Scores below 50 generate the object in the top half of the screen. Then for Scores less than 10 the objects re-appear on the start line. <ore If's could be added. The If's too could be extended to modify the direction, for example if the score is less than 30 and the object vertical movement is greater than 0.5 then the Movement Vertical (MoveVertical) is halved…

```
If Score < 30 And Movevertical(L) > 0.5 Then Movevertical(L) = Movevertical(L) / 2
```

You can select whatever effects and changes for your game. (V1.93)

Stop item collection when image is small

Once you hit a "Sweet Object" it starts to get smaller. You can collect it multiple times. Suppose we want to stop collection of Multiple objects once the size has been reduce by half. This occurs inside the second If below…

```
Function ImageHit(ImageClicked)
  If Button1.value <> "Start" Then
    If CollectMe1.src = ImageClicked.src Then
      CollectedUpdate(Collected + 1)
      ExtraLife = ExtraLife + 1
      Audio1.play()
    Else
```

The code will not make any changes if the display of Button1 is set to Start, so we can simply add in the extra condition to this line…

```
Function ImageHit(ImageClicked)
  If Button1.value <> "Start" And ImageClicked.Width > ImageSize / 2 Then
    If CollectMe1.src = ImageClicked.src Then
```

So now the image must be at least half the size for a collection to take place! (V1.94)

How to Stop a Blank Top Score Name…

By adding a loop around the entry of the new TopScoreName the software can force the user to enter a name…

```
If Score > TopScore Then
  TopScore = Score
  TopScoreName = ""
  Do While TopScoreName = ""
  TopScoreName=InputBox("New Top Score - Enter Your Name")
  Loop
  Label4.textContent =  "Top Score by " + TopScoreName +"=" + TopScore
  Label4.hidden = False
End If
```

This gives V1.95

Different Start Heights and Movement directions..

This revision to StartUp will cause objects to have a 50/50 chance of starting to move left or right and a 50 /50 chance of being on a lower row (200)….

This gives V1.96

```
For L = 1 To NumberOfImages
  AImage(L).hidden = False
  Movehorizontal(L) = -1
  If Rnd() <.5 Then Movehorizontal(L) = 1
  Movevertical(L) = 0
  AImage(L).Left = Border + (L*50)
  TrueTop(L) = 100 + Border
  If Rnd()<.5 Then TrueTop(L) = 200 + Border
  AImage(L).Top = TrueTop(L)
  AImage(L).Width = ImageSize
  AImage(L).Height = AImage(L).Width
  AImage(L).src = "Move1.png"
Next
```

Creating a large Start Button at the Start of the Game

To do this we need to simply let the program think that a game is running and it has reached then end. The Game ends when Lives equals zero. The code which runs the movement can then be added after this!
At the top of your program add the 2 bottom lines shown directly below SetUp() giving V1.97…

```
SetUp()
Lives = 0
timerRef = SetInterval(nextAction, 100)
```