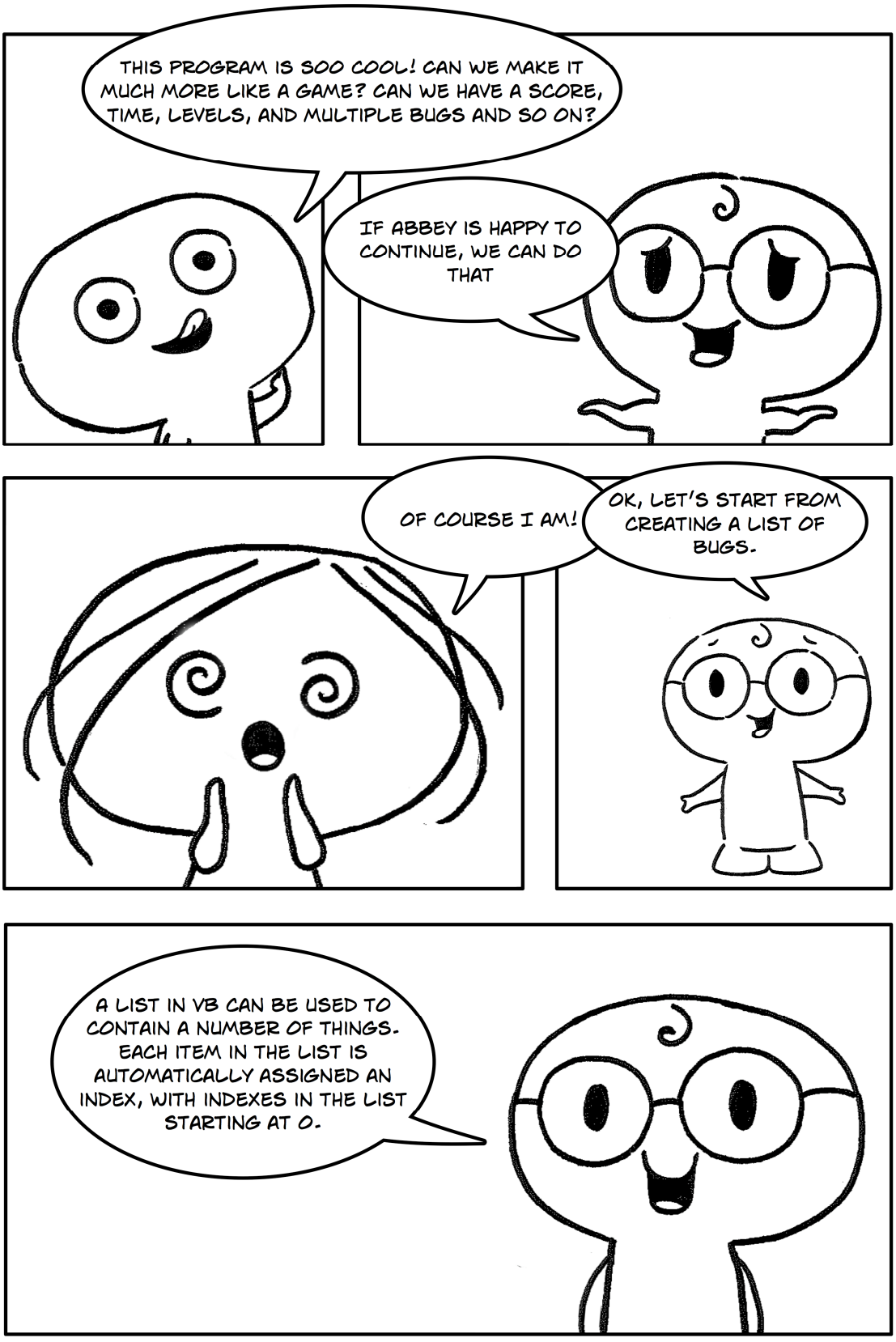
Chapter 8

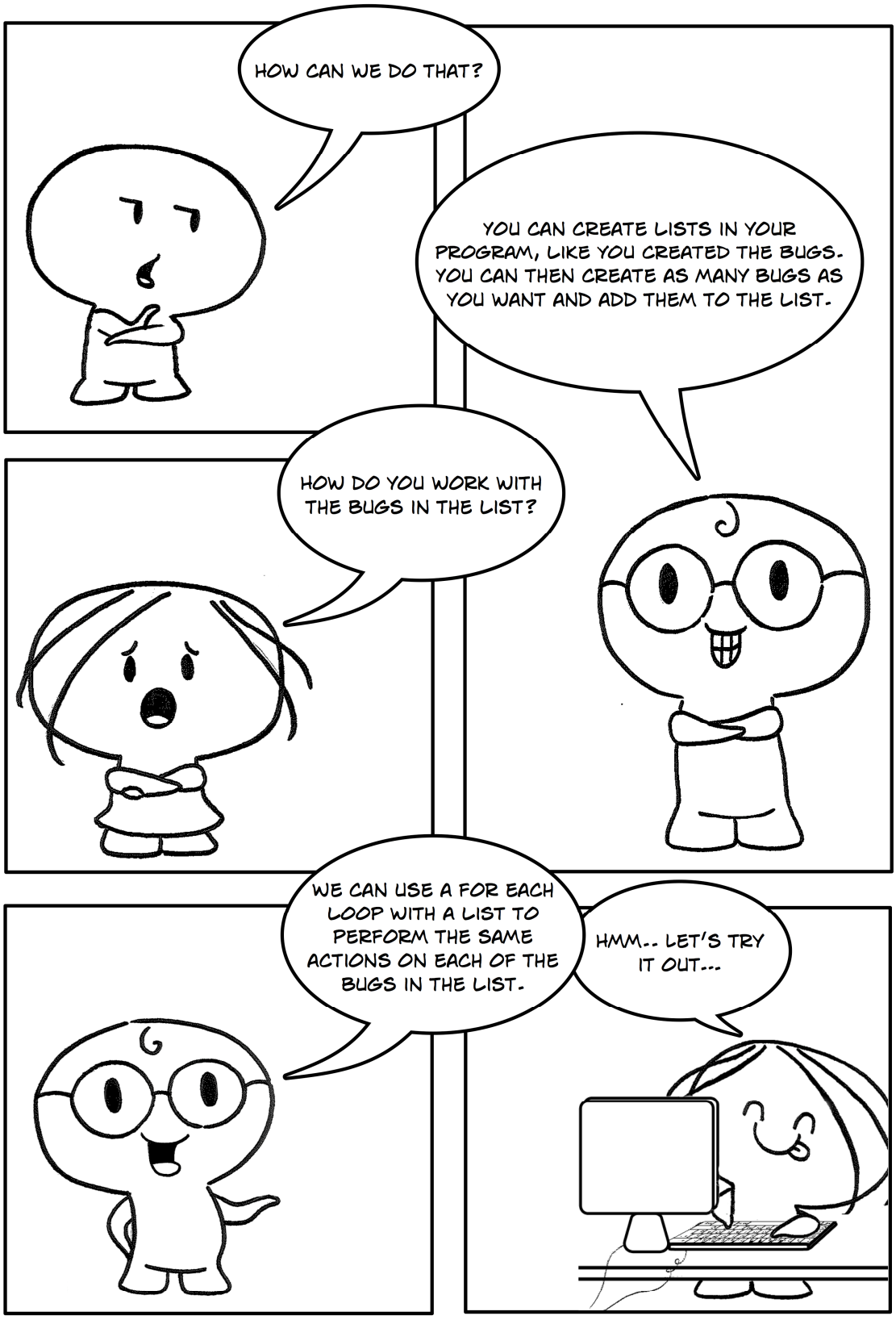
"Level and Score"

Summary:

This chapter will help you create different levels for your project, show how to make your program graphically look like a game. To complete this chapter you are required to use the solution from the previous one. You will also need new fonts, which will be given to you.

This chapter is the last one in a relation to "Bug Squash" game. Next series of chapters will be based on knowledge collected from Chapter 1 to 8 and will develop a new game – "Buggy Loves Food".





Part 1

A list is a collection of things. You can perform a number of actions with things in the list.

In our case, we need to declare a field which will create a list of Bugs – list of objects. To do so, type the following right after the `Module GameLogic` line: `Private listBugs As List(Of Bug)`. Now we need to assign a value to the list. Because a list is basically a class we need to create a new list object. To do so, type `listBugs = New List(Of Bug)` after `Randomize()` sub call inside the `Main()` method.

To perform any action on the list such as draw and update each bug in the list, we can use for each loop. The logic is shown in Figure 1 below:

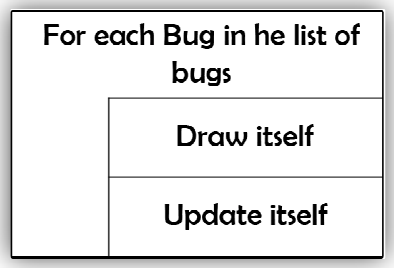


Figure 1

To tell each bug in the list to draw and update itself, use the following code inside the Game Loop after Graphics.ClearScreen(Color.White):

```
For Each Bug As Bug In listBugs
            Bug.Draw()
            Bug.Update()
Next
```

Because we are using multiple sprites now, we need to free each of them. To do so, we need to add the following code in the Bug class:

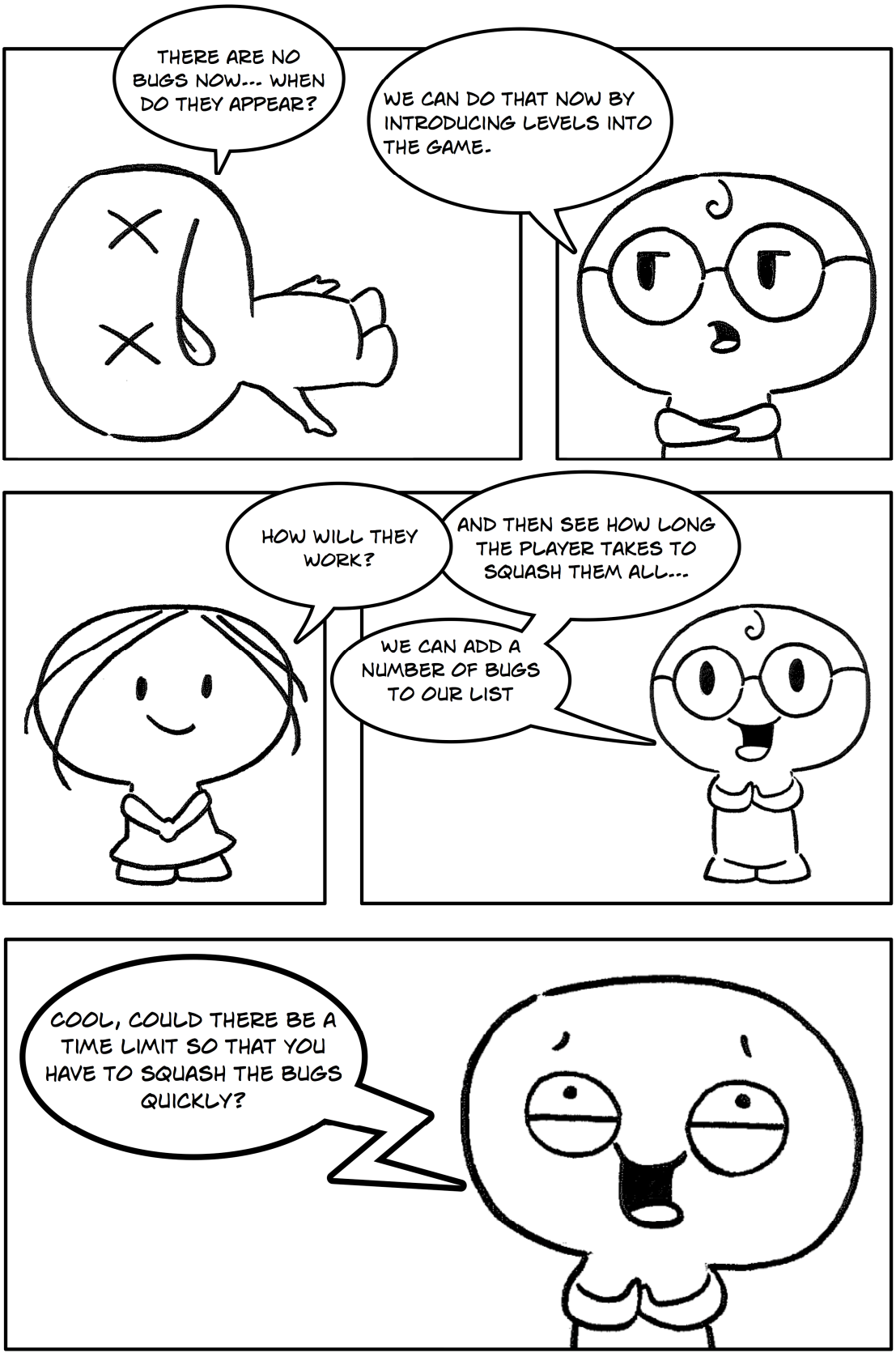
```
Public Sub CleanUp()

    DeadSprite.Dispose()
    AliveSprite.Dispose()
End Sub
```

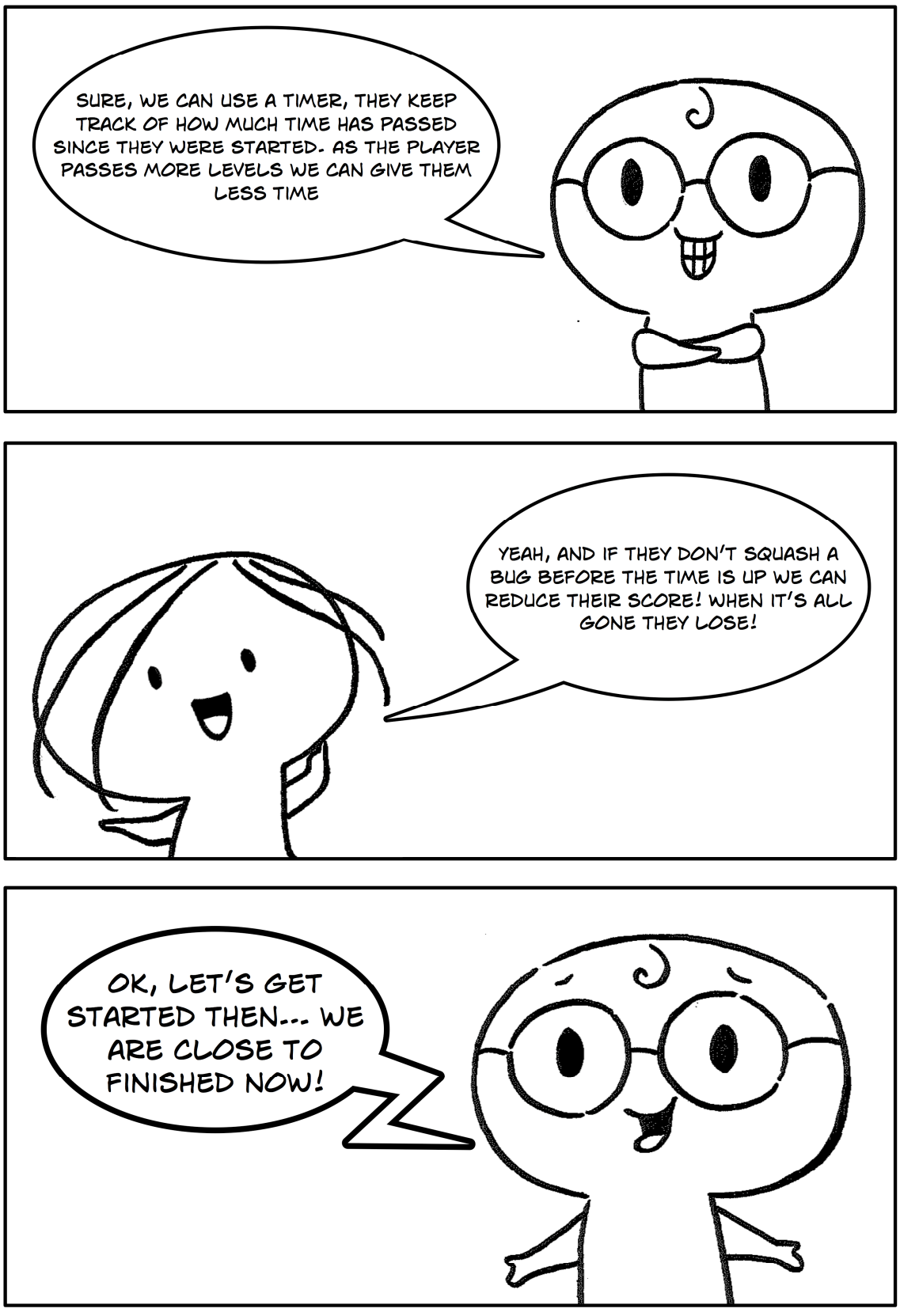
We will call this function from GameLogic.vb whenever we need to free our sprites.

Exercise 1: *Creating a list*


Make the following changes in your program and write your solutions to the worksheet:

1. Create a list of bugs in your program. Use the information above in order to achieve this.
2. Write a code that will allow each bug in your program to draw and update itself.
3. Create a function that will free a sprite inside the Bug class.





Part 2

Our list of bugs is empty for now. We need o fill it up, we also need to introduce level variable, and timer, and score, and the point at which the level will end. To do so, type in the following code right after the level field in the GameLogic.vb:

```
    Private level As Integer
    Private gameTimer As Timer
    Public score As Integer
    Private endLevelAt As Integer
```

Where level is a level number; gameTimer is an engine that will count time as a game goes; score is a number which represents a score n your program; and endLevelAt is an ending point for each level.

Now we need to have a procedure called LevelSetUp() in GameLogic.vb. The code for this procedure is shown below:

```
Public Sub LevelSetUp()
        Core.StopTimer(gameTimer)

        For Each Bug As Bug In listBugs
            Bug.CleanUp()
        Next

        listBugs.Clear()

        For i As Integer = 1 To level * 3
            listBugs.Add(New Bug)
        Next

        endLevelAt = 10000 - 500 * (level - 1)
        If endLevelAt < 500 Then endLevelAt = 500

        Core.StartTimer(gameTimer)
 End Sub
```

This procedure is basically starts timer for each level, frees each sprite created at previous levels, clear the list of bug so next level will not depend on previous one, fills up a list with a number of bugs (the number is tripled each level); this procedure is also tell each level at which point current level stops and when the next one starts.

In order to jump to another level we need to have a function EndOfLevel() which will check whether all of bugs in the listBugs are dead bugs and return true of false. The code for this function is shown below:

```
Public Function EndOfLevel() As Boolean
    For Each myBug As Bug In listBugs
        If myBug.IsAlive Then
            Return False
        End If
    Next
    Return True
End Function
```

In the main method we need to set up the starting point for the game. To do so, we should create a timer ad a time in our game inside the Main() method, right after `listBugs = New List(Of Bug)`. We need to tell our program to start from level 1 with score = 0. The code for this is shown below:

```
        gameTimer = Core.CreateTimer()
        Dim time As Integer

        level = 1
        LevelSetUp()
        score = 0
```

Inside the game loop we need to set up time, so it will go backwards: `time = (endLevelAt - Core.GetTimerTicks(gameTimer)) / 100`. That should be dore right after `SwinGame.Graphics.ClearScreen(Color.White)`. Als insde the Game Loop we need to tell when our game has to change the level to the next one, and when time goes off, we need to take 1 point from the score – so there will be a possibility to lose the game. To do so, type the following after the for each loop inside the Game Loop:

```
      If EndOfLevel() Then
           level = level + 1
           LevelSetUp()
      End If

      If time < 0 Then
           score = score - 1
           Core.StopTimer(gameTimer)
           Core.StartTimer(gameTimer)
      End If
```

Exercise 1: *Setting up the level and score*


Make the following changes in your program and write your solutions to the worksheet:

1. Make changes in your progra as it explained above.

Our game so far has no level separation, and no graphic support. To make our game look better, we need new fonts. You are provided with three new fonts - `cat_scratch.ttf, bear.ttf, comic.ttf`. Copy them into the Resources->fonts folder as you did in chapter 2 and type the following inside the `LoadFonts()` sub in GameResources.vb:

```
    NewFont("cat_scratch", "cat_scratch.ttf", 40)
    NewFont("bear", "bear.ttf", 120)
    NewFont("bear_huge", "bear.ttf", 170)
    NewFont("comic", "comic.ttf", 16)
```

Now we can beautifully draw the current score and current time on the screen uring the game. Put the following code right after `time = (endLevelAt - Core.GetTimerTicks(gameTimer)) / 100` inside the Game Loop:

```
Text.DrawText("Bugs killed: " & score, Color.Green,
GameFont("comic"), 2, 2)


Text.DrawText("Time: " & time, Color.Green,
GameFont("comic"), 730, 2)
```

This will show your current scpre at the left-top corner and time at the right-top corner.

Exercise 2: *Printing current score and time on the screen*

Make the following changes in your program and write your solutions to the worksheet:

1. Load new fonts into your program.
2. Draw current score and time n the screen.

Our program will look more like a game if you will use level separators and "lose" screen. To have a start level screen we need to create a new sub which will just print current level and score on the screen. To do so, in GameLogic.vb create a new procedure called `DrawLevelStart()`. The code for this is below:

```
Public Sub DrawLevelStart()
        For i As Integer = 1 To 22
            Graphics.ClearScreen(Color.White)
            Text.DrawText("Level " & level, Color.Green,
GameFont("bear"), 280, 200)
            Text.DrawText("Score: " & score, Color.Green,
GameFont("cat_scratch"), 320, 300)
            Core.RefreshScreen(30)
            Core.ProcessEvents()
        Next
 End Sub
```


This procedure is basically draws the text at the beginning of each level which is level number and current score. We need to call `DrawLevelStart()` inside the `LevelSetUp()` procedure. Just type `DrawLevelStart()` right after the `Core.StopTimer(gameTimer)` in `LevelSetUp()` procedure.

We also can have a procedure that will draw the messege "You looose!", the level at which you have lost, and it will ask you to press enter to go to the first level. The procedure could be called `DrawLevelEnd()` and the code for tis is shown below:

```
Public Sub DrawLevelEnd()
        Do
            Graphics.ClearScreen(Color.White)
            Text.DrawText("YOU ", Color.Green,
GameFont("bear"), 80, 150)
            Text.DrawText("LOOOOSE!", Color.Red,
GameFont("bear_huge"), 275, 125)
            Text.DrawText("Level " & level, Color.Green,
GameFont("cat_scratch"), 300, 340)
            Text.DrawText("Press ENTER to start",
Color.Green, GameFont("cat_scratch"), 180, 480)
            Core.RefreshScreen(30)
            Core.ProcessEvents()
        Loop Until Input.WasKeyTyped(Keys.VK_RETURN) Or
SwinGame.Core.WindowCloseRequested() = True
        score = 0
        level = 1
        LevelSetUp()
End Sub
```

This procedure must be called inside the if statement that checks whether time went off which is located in the Game Loop. The complete version of this statement is shown below:

```
    If time < 0 Then
          score = score - 1
          Core.StopTimer(gameTimer)
          Core.StartTimer(gameTimer)

          If score < 0 Then
                DrawLevelEnd()
          End If
    End If
```

Exercise 3: *Printing a start and end point of the game*


Make the following changes in your program and write your solutions to the worksheet:

1. Build into your program DrawLevelStart() and DrawLevelEnd() procedures.