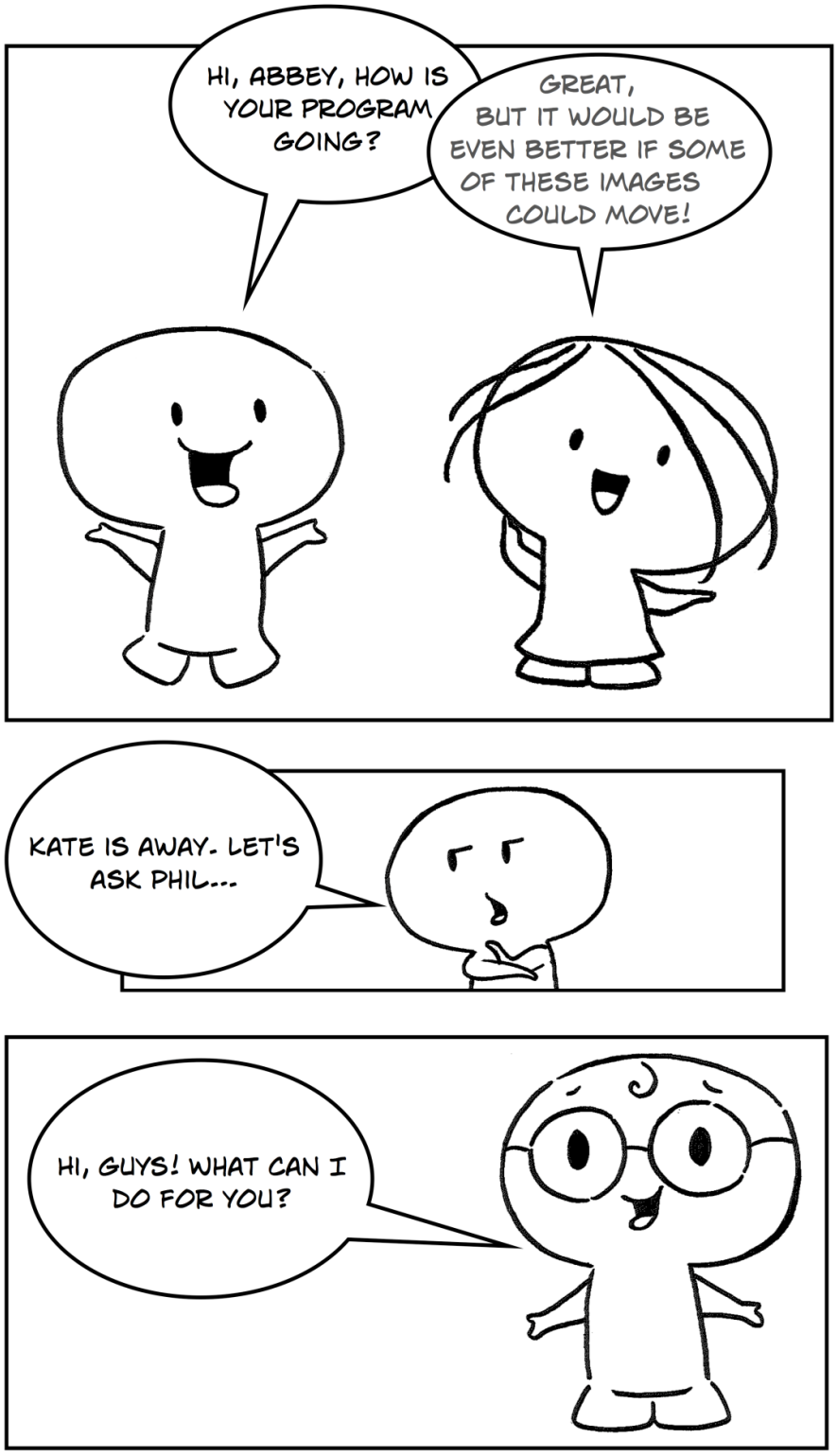
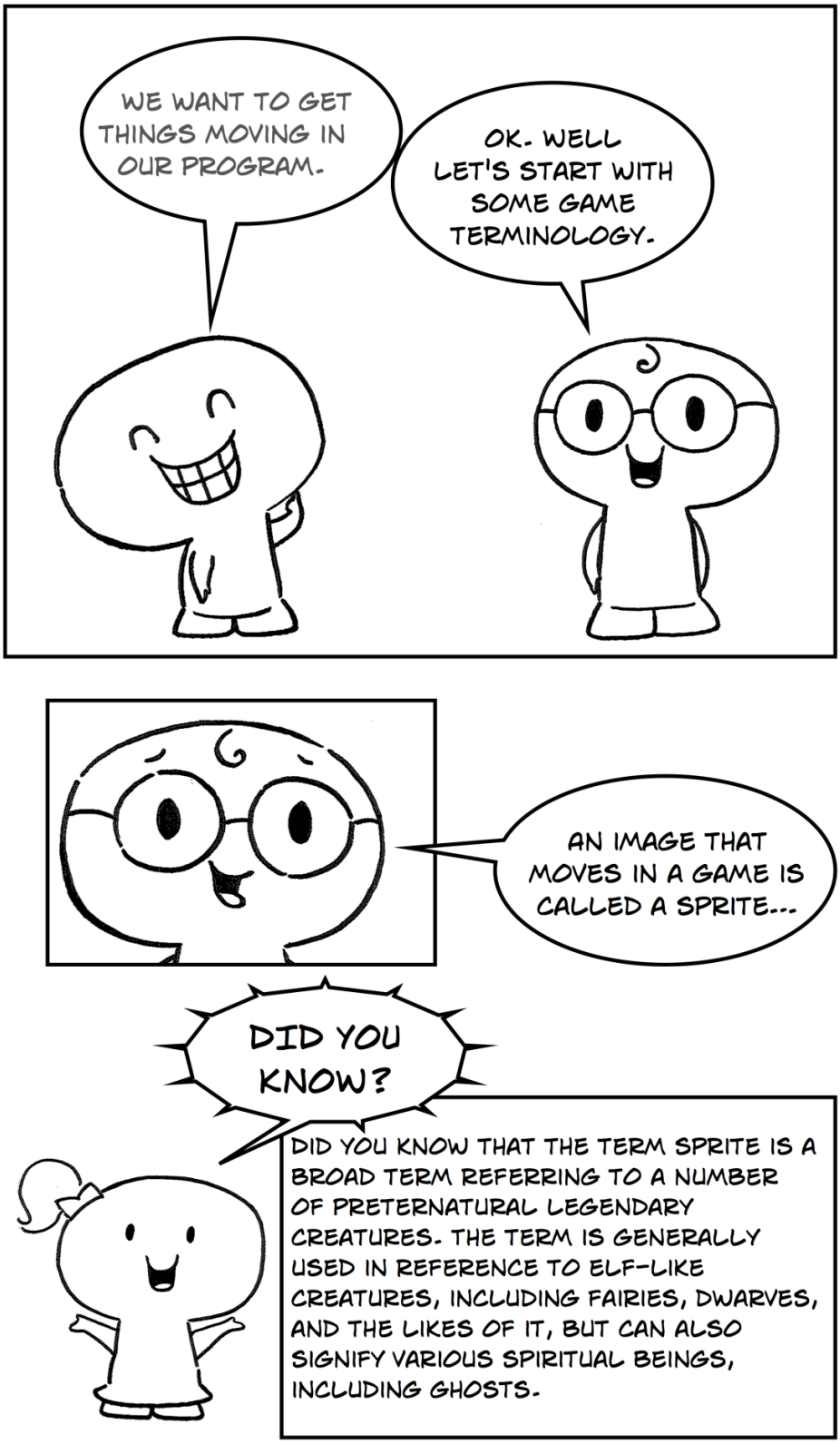
Chapter 3

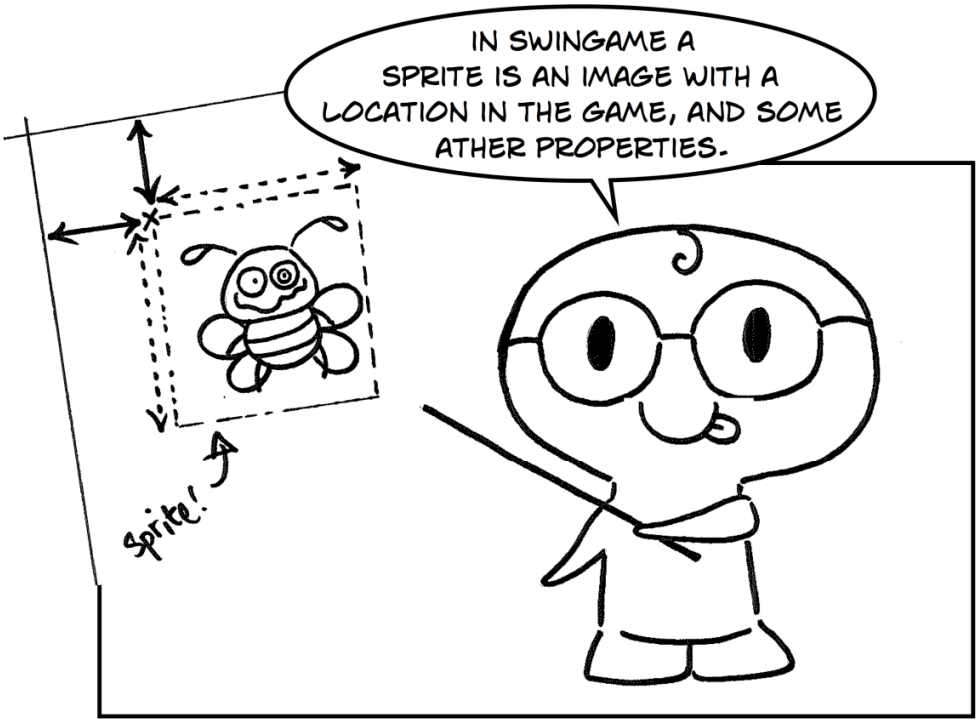
"Movement"

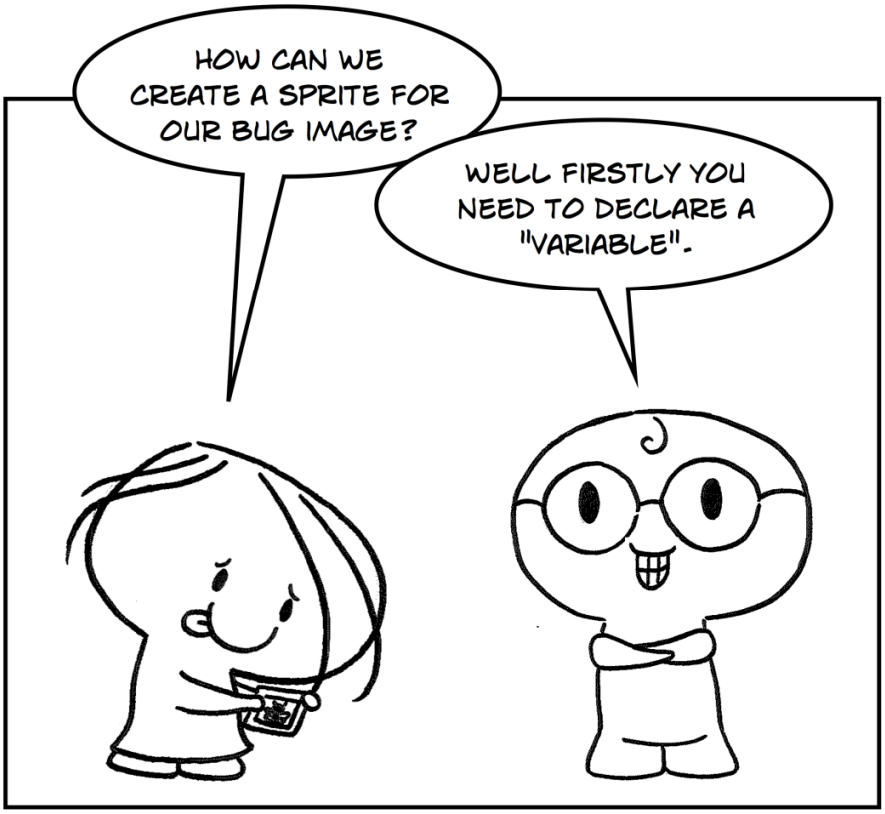
Summary:

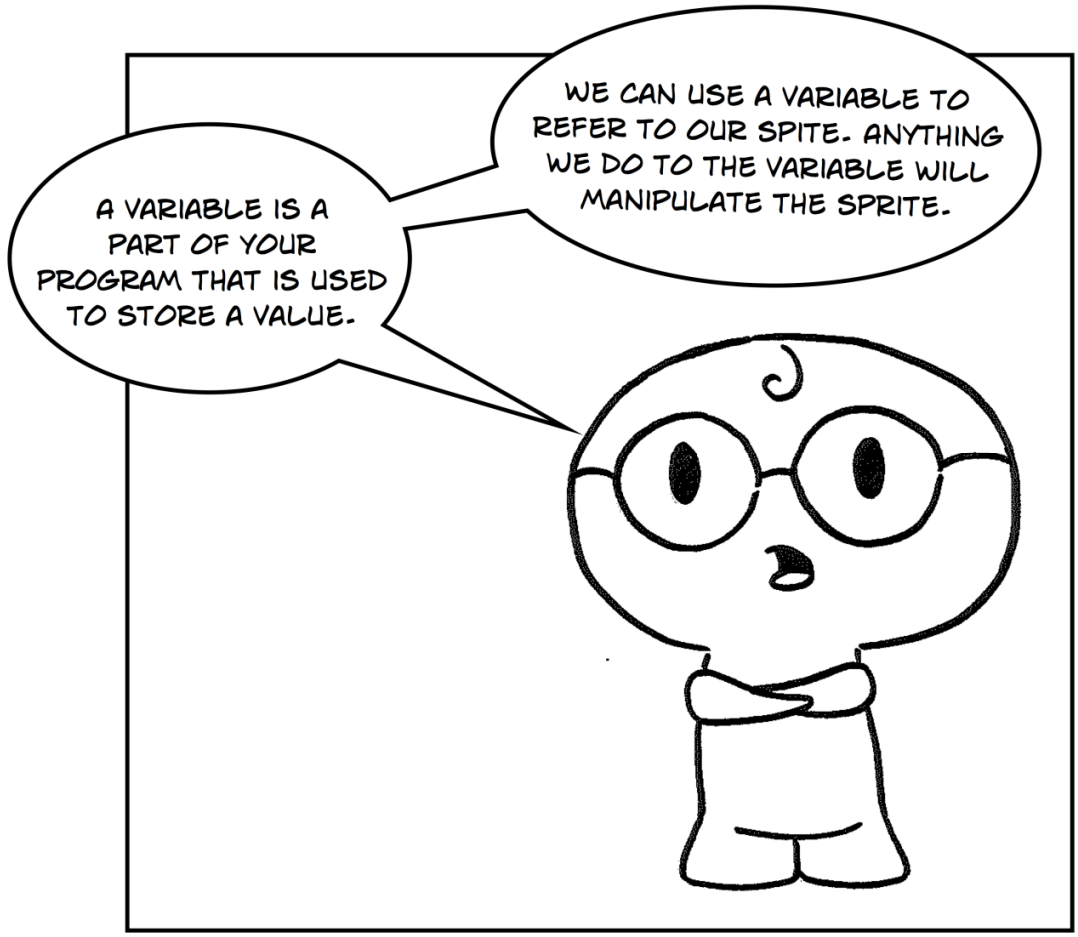
For this chapter you will need to create a new SwinGame project, and use your knowledge from the previous chapters. You will be introduced to new terms and some game features such as movement of the main character.

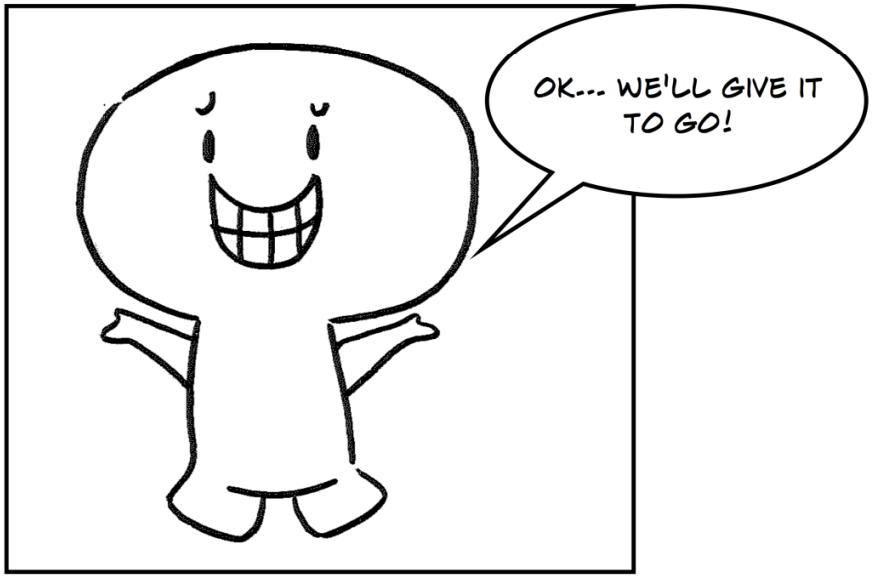












Part 1

A sprite is basically a small graphic that can be moved independently around the screen, producing animated effects. To create your spite you need to declare a variable which will refer to our sprite. This enables us to manipulate our sprite and make it to do what we want. Every time we create a sprite we need to free it at the end of our program.

Open Visual Studio and create a new SwinGame project called "Bugs". Delete everything between `SwinGame.Graphics.ClearScreen()` and `'Refreshes the Screen and Processes Input Events` as you did in Chapter 2. Load image called "sprite.png" into your program as you did in previous chapter. Name the new image as "sprite" (`NewImage("sprite", "sprite.png")`).

Exercise 1: *Creating a sprite*

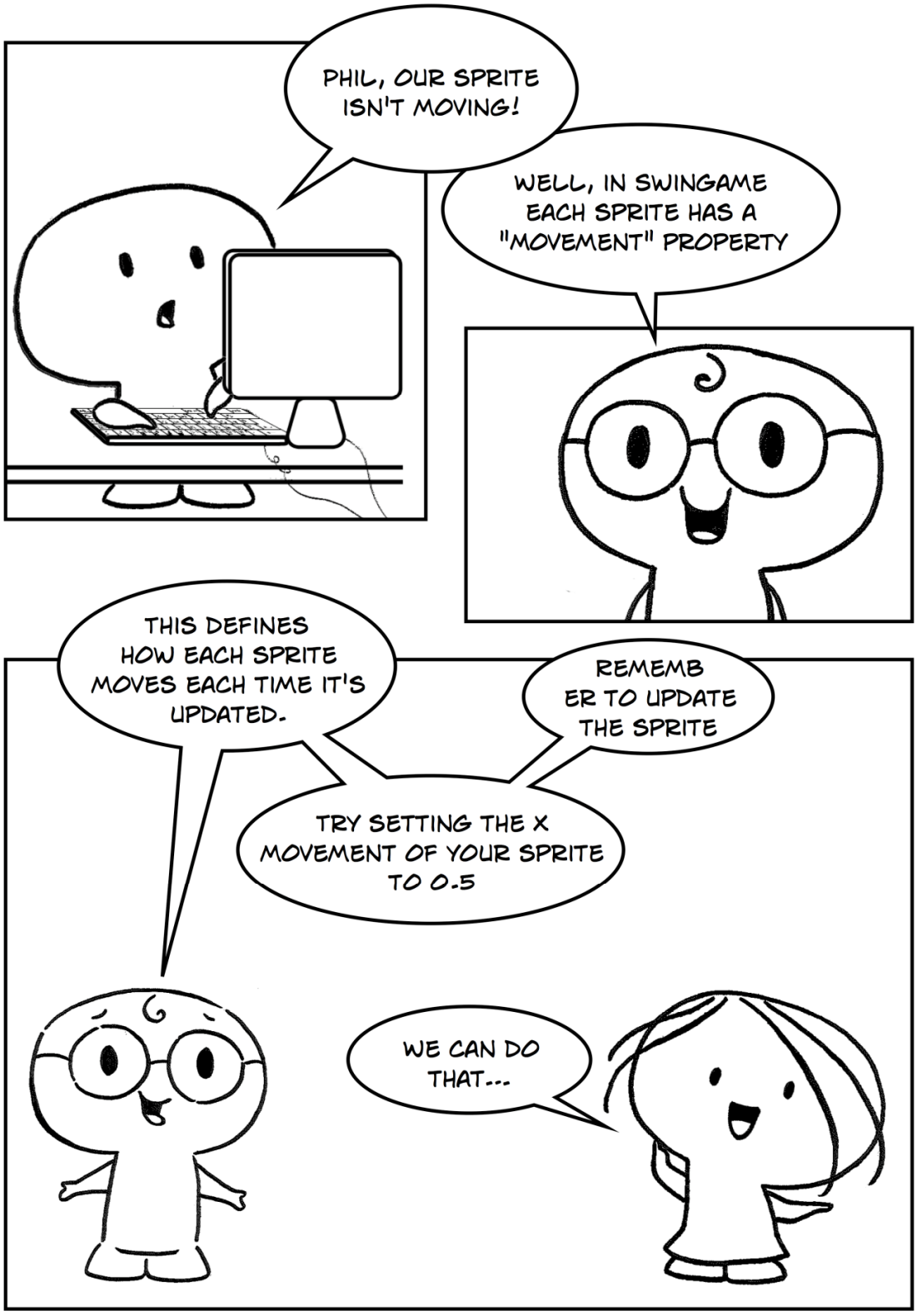
 Make the following changes in your program and write your solutions to the worksheet:

1. Declare the variable "bug" which is a Sprite. To do so, put the following code after the `LoadResources()` where `bug` is the name of your variable and `Sprite` is the type of the variable:

```
'Load Resources
LoadResources()

Dim bug As Sprite
```

2. Now we can create a sprite. To create the sprite use `variableName = Graphics.CreateSprite(GameImage("nameOfImage"))` after variable declaration.
3. To draw the sprite on the screen use `Graphics.DrawSprite(variableName)` inside the Game Loop.
4. Put `Graphics.FreeSprite(variableName)` at the end of your program.
5. Press the "StartDebugging" button at the top of the screen (looks like a green arrow  , F5 works too) to see what it does.



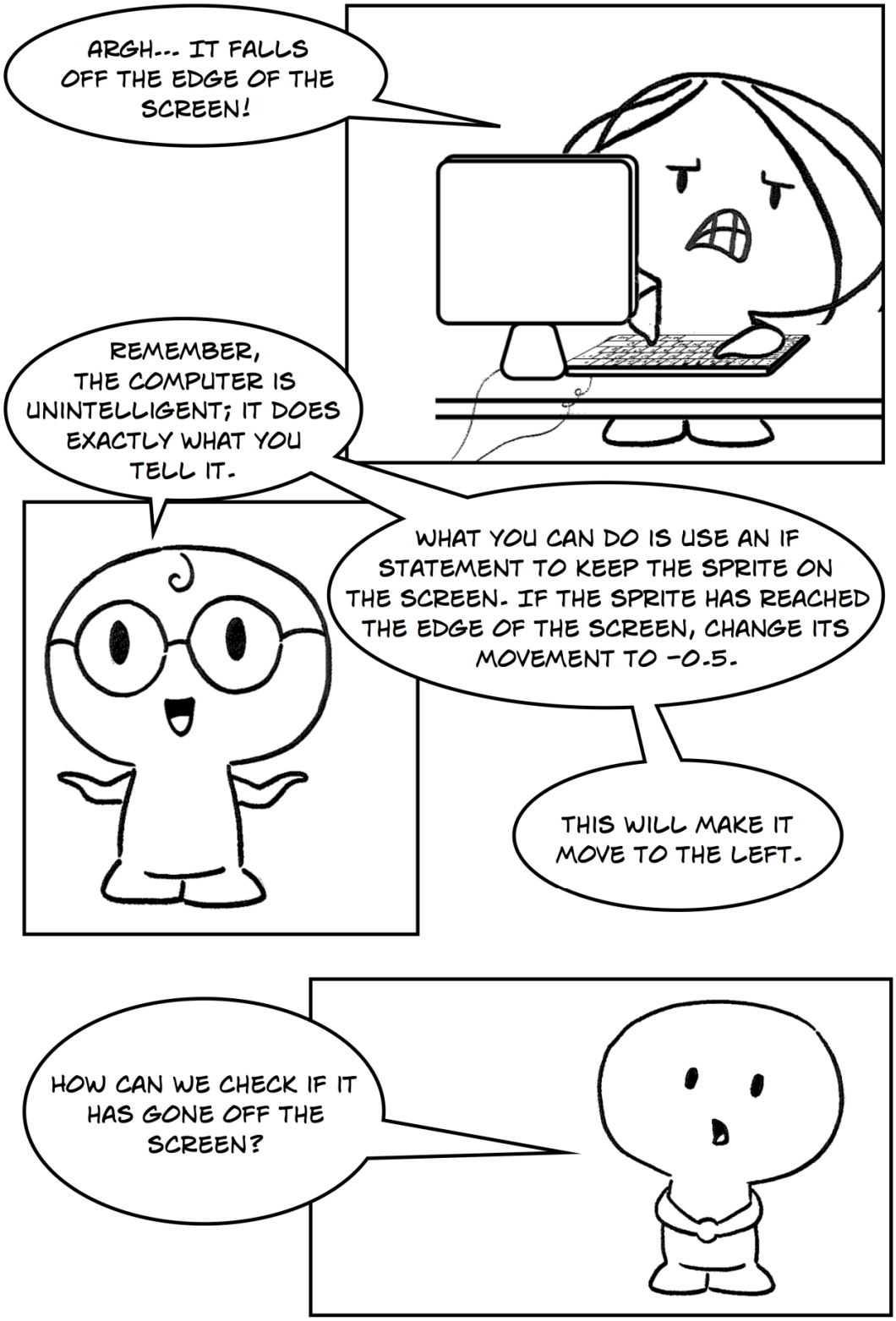
Part 2

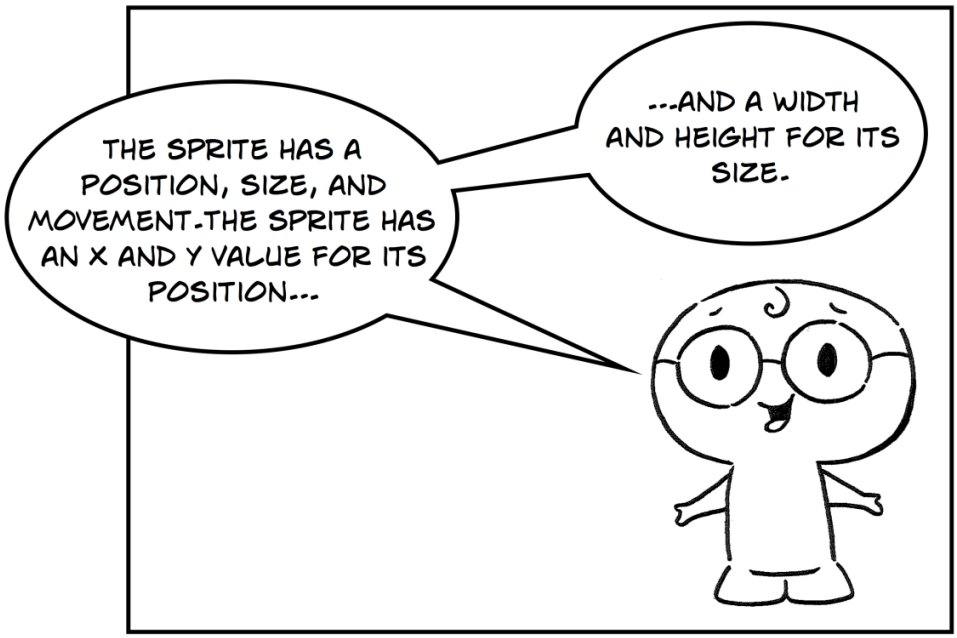
In SwinGame each sprite has its "Movement" property which defines how much the sprite moves each time it is updated. The movement is defined by X and Y values. X is the amount of pixels that the sprite moves horizontally, and Y is amount of pixels that the sprite moves vertically. In order to see the exact movement of our sprite on the screen, we need to update our sprite inside the Game Loop.

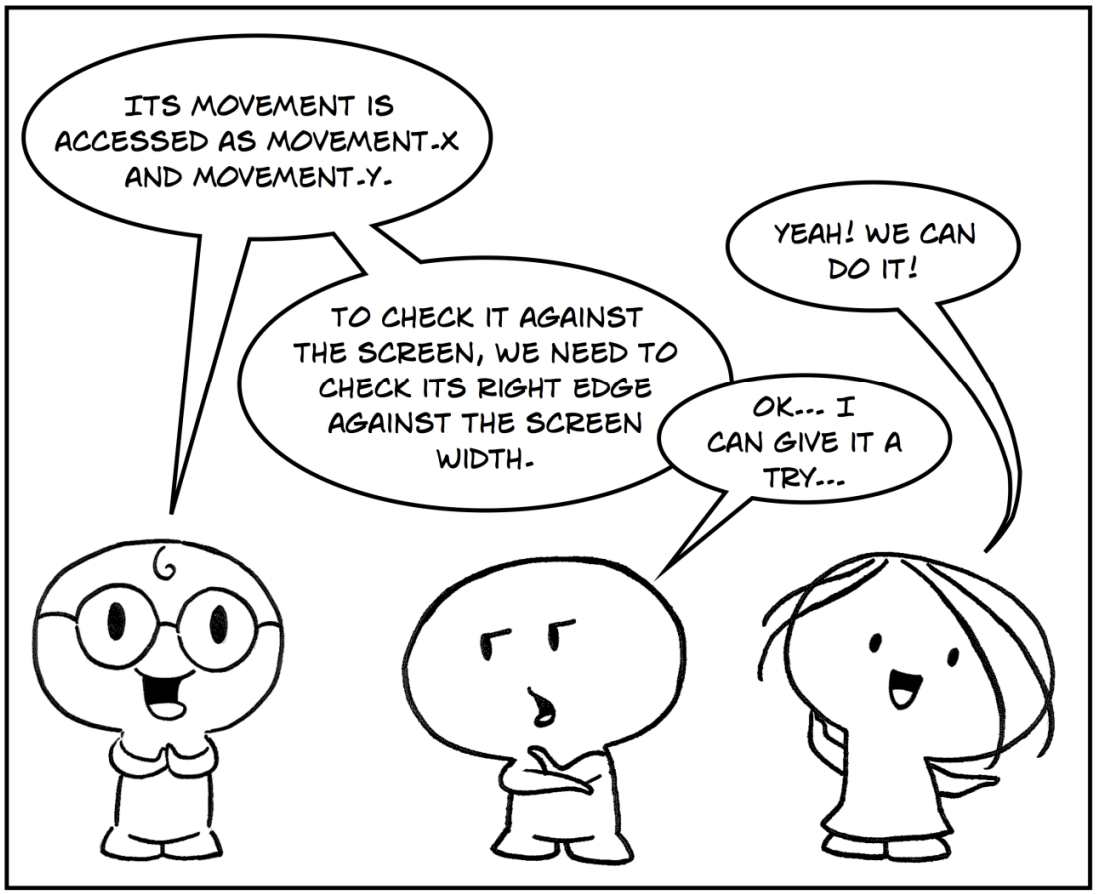
Exercise 1: *Making the sprite to move*


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

1. Assign Movement.X of your sprite to 0.5. To do so, use `variableName.Movement.X = 0.5`, and put this code before the start of the game loop.
2. In order to see how our sprite moves, it needs to be updated withing the loop. Use `Graphics.UpdateSprite(variableName)` after `Graphics.DrawSprite(bug)`.
3. Press the "StartDebugging" button at the top of the screen (looks like a green arrow  , F5 works too) to see what it does.







Part 3

Each sprite has a position, size and movement. The position is defined by X and Y values of the sprite. The size is defined by width and height of the sprite. The movement is determined by X and Y values which are the number of pixels the sprite is being moved. X and Y of movement can be accessed by Movement.X and Movement.Y.

In order to keep the sprite on the screen we need to use all parameters of the sprite and the `If` statement. `If` conditional expression is one of the most useful control structures which allows us to execute an expression if a condition is true. The syntax looks like this:

```
If condition Then
[statements]
End If
```

If the condition is true, the statements following the `Then` keyword will be executed. `If` statement could also contain the `Else` expression:

```
If condition Then
[statements]
Else If condition Then
[statements]
-
-
Else
[statements]
End If
```

That basically means that if the condition is true, the statements following the `Then` keyword will be executed, else, the condition following the `Else If` will be checked and if true, the second block of statements will be executed, else, the statements in the `Else` part will be executed.

In order to check whether our bug falls off the edge we need to use the first, simple `If` statement. To do so, we need to check it's right edge against the screen width. The right edge is the position of the sprite plus the width of the sprite, as shown in Figure 1:

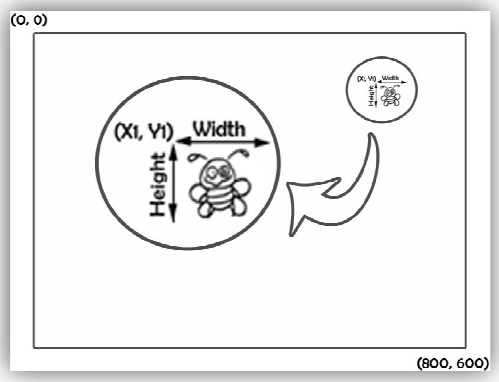


Figure 1

The logic for checking whether the sprite is within the screen is shown in the Figure 2:

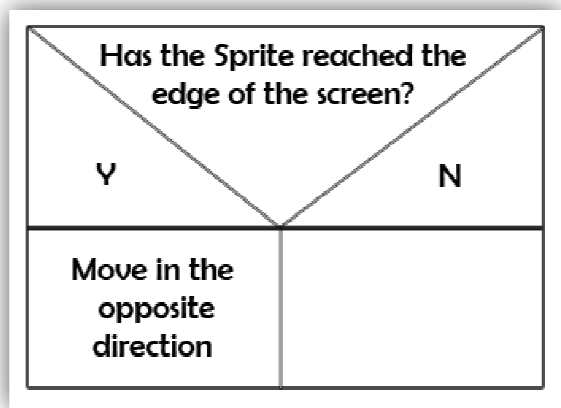


Figure 2

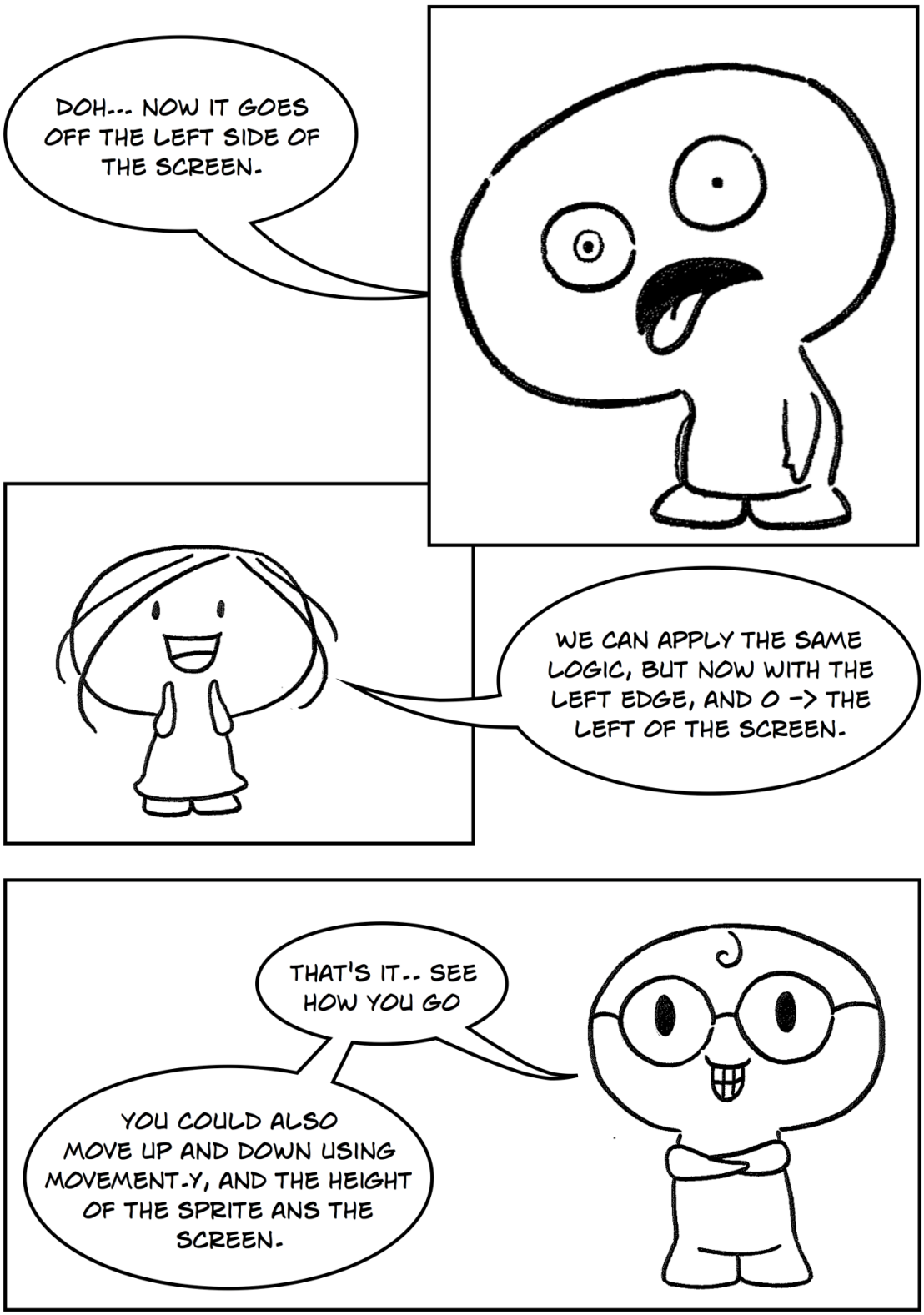
Exercise 1: *Stopping the Sprite from moving off the right edge of the screen.*

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

1. Add the following code to the Game Loop after the line containing `Graphics.UpdateSprite(bug):`

```
If variableName.X + variableName.Width >= Core.ScreenWidth Then
variableName.Movement.X = -0.5
End If
```

2. Press the "StartDebugging" button at the top of the screen (looks like a green arrow , F5 works too) to see what it does.



Part 4

In order to stop the Sprite from moving off the left edge of the screen we need to use the same logic as we used preciously. The difference here is that unlike the right edge which is determined by the position of the Sprite plus Sprite's width, the left side of the Sprite is defined only by its position.

Exercise 1: *Stopping the Sprite from moving off the left edge of the screen.*


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

1. Add the following code to the Game Loop after the coded from the previous exercise:

```
If variableName.X <= 0 Then
    variableName.Movement.X = 0.5
End If
```

2. Press the "StartDebugging" button at the top of the screen (looks like a green arrow  , F5 works too) to see what it does.

So far the Sprite is moving horizontally; by assigning a value to `Movement.Y` you can move the Sprite in different directions.

Exercise 2: *Changing the movement direction.*


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

1. Assign `Movement.Y` of the Sprite to 0.5, this can be done in the same way as shown in part 1 > exercise 1.
2. Write the code which will stop the Sprite from moving off the top edge of the screen; use the same logic as in part 3 & 4 exercises.
3. Write the code which will stop the Sprite from moving off the bottom edge of the screen; use the same logic as in part 3 & 4 exercises.

Extra Exercise:

If you want you can add a second Sprite to your program, follow the same steps to achieve this.