## VBugs

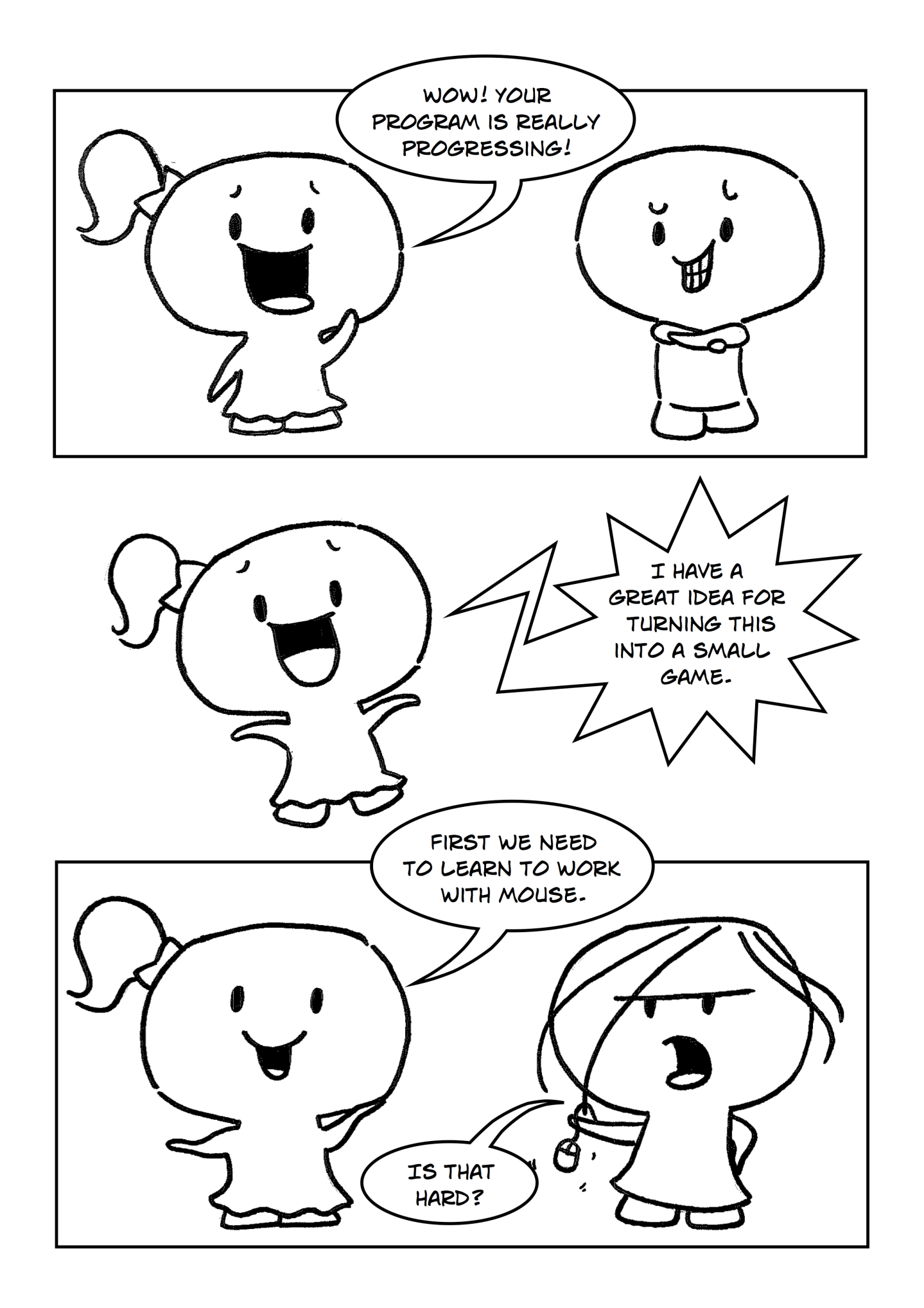
## Chapter 5

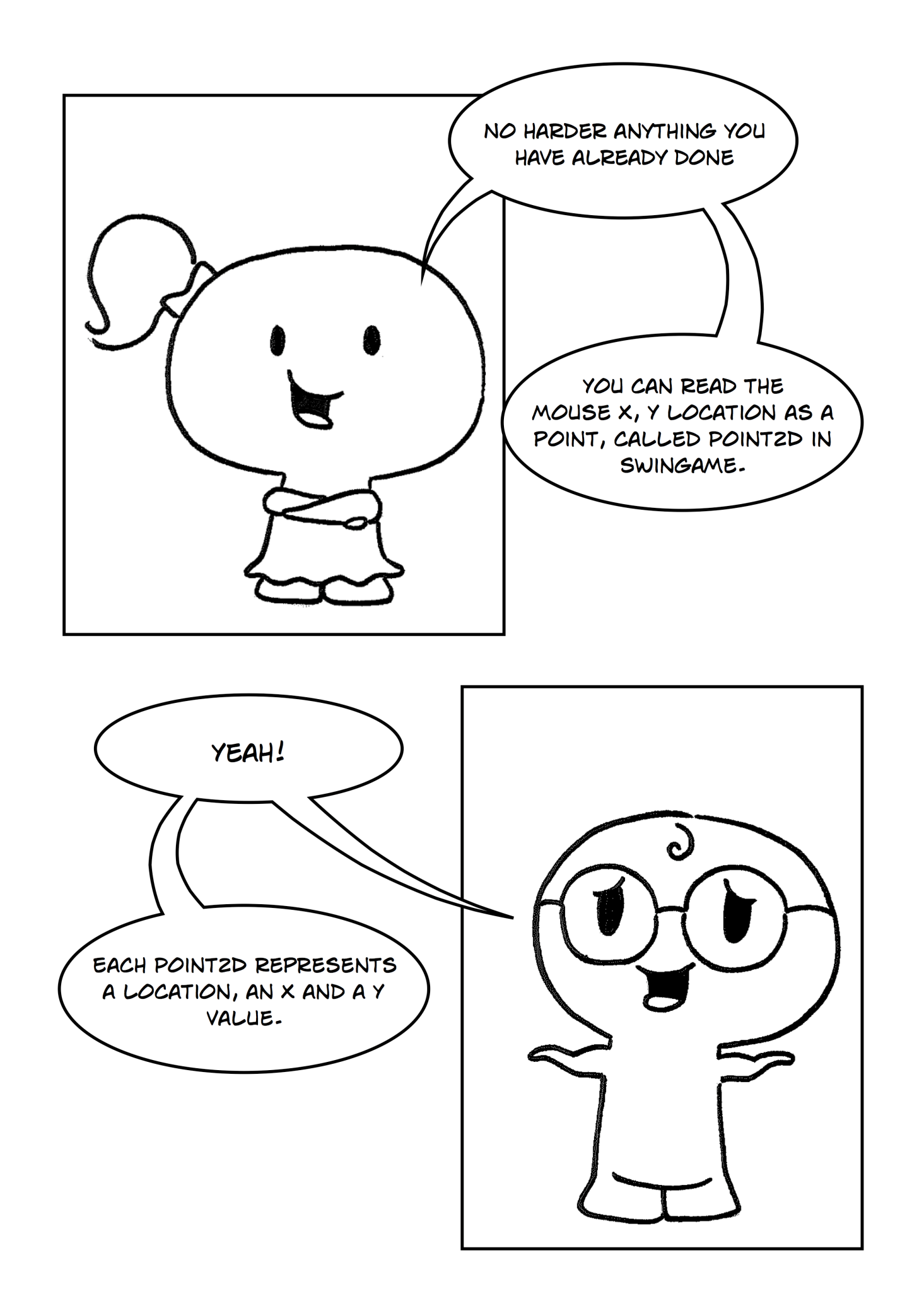
###### comic 1 - 8.pngMouse input and Animation

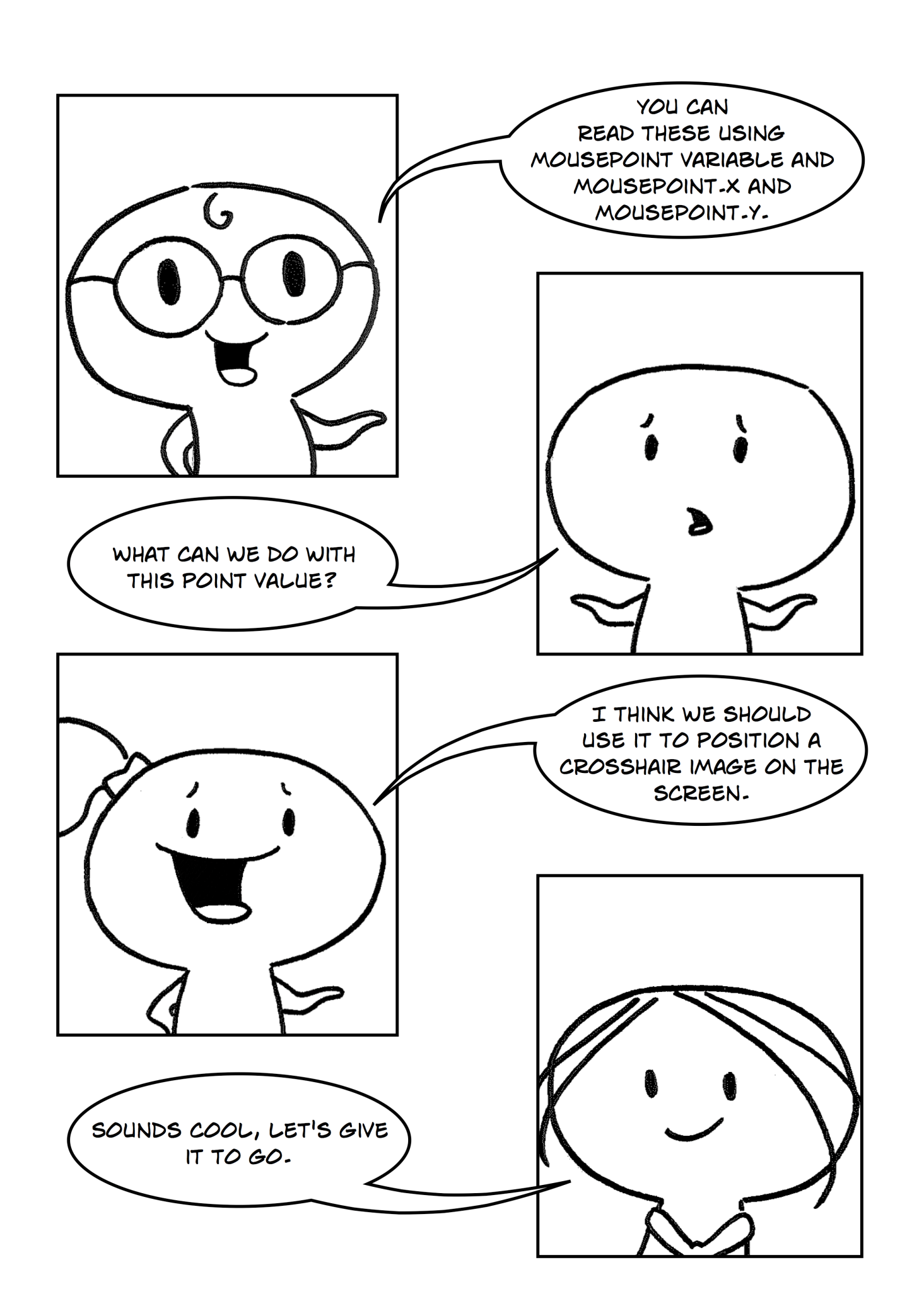


# Summary:

In this chapter you will be modifying the solution from the previous chapter. You will learn how to replace default mouse point with a custom picture, how to handle mouse input and use animated sprites. All resources will be provided.

****

****

****

## Replacing the Mouse Pointer

Point2D (2D = two-dimensional) is a variable that can be used to get the position of the mouse’s cursor on the SwinGame screen. So for example if we were to decare (Dim) a variable of type Point2D and name it mousePoint, we can then later on get its X position using mousePoint.X. This variable has to be declared outside the game loop.

If we know where the mouse is on the screen we can then hide it and replace it with whatever we want, in this example a “crosshair” image. The code for drawing the new image has to be inside the Game Loop as it constantly has to be re-drawn to the screen.

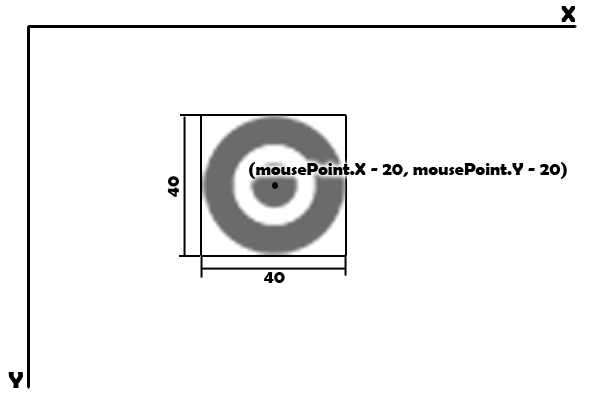
To hide the current mouse pointer use Input.ShowMouse(False) (this should be made outside the Game Loop to increase the speed of your program, if you put it in the loop it has to execute every time the program loops). The ShowMouse() function is a built-in function that is “True” (visible) by default.

To place the custom image (crosshair) instead of the original one, we need to know the position of the mouse on the screen. We can get mouse position at a time by assigning a variable mousePoint to Input.GetMousePosition() inside the Game Loop. Then we can draw the crosshair image at the position of the mouse with Graphics.DrawBitmapOnScreen(GameImage("ImageName"), mousePoint.X, mousePoint.Y) function.

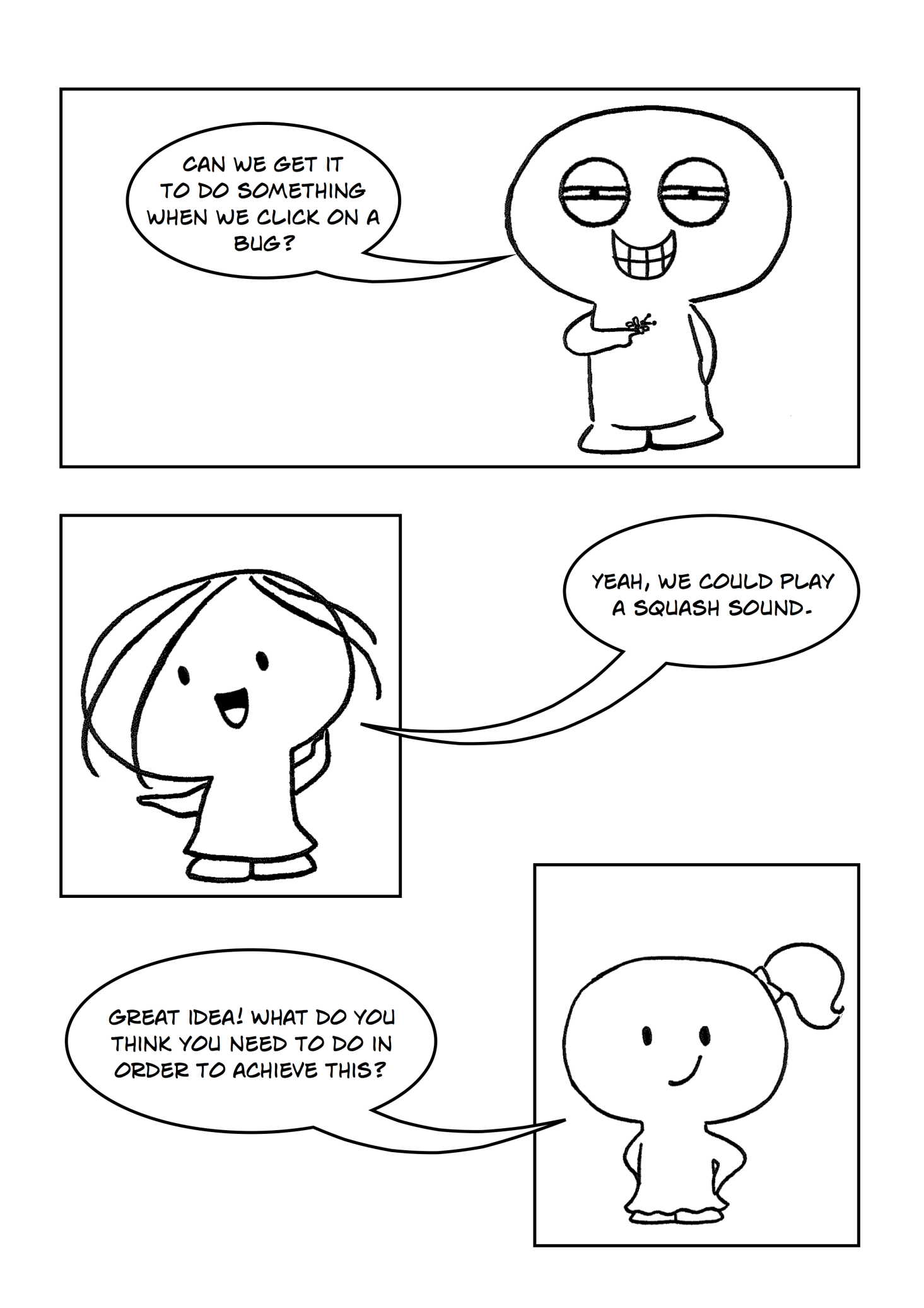
Exercise 1: *Replacing the original mouse point with the custom one*

cha 2 - worksheet.pngMake the following changes in your program and write your solutions to the worksheet:

1. Loadtarget.png into your program.
2. Inside the Game Loop (below ClearScreen) tell the program to draw target.png instead of the original mouse pointer using the methods described above.
3. The image has the size of 40x40 pixels. Remember the mouse position is going to be the position of the top left corner of the mouse on the screen. In order get the target in the right position (the middle of crosshair image) you need to consider the size of the image. If the image is 40x40 then the middle of the image will have a position of X = mousePoint.X – 20 and Y = mousePoint.Y – 20 as shown in Figure 1.

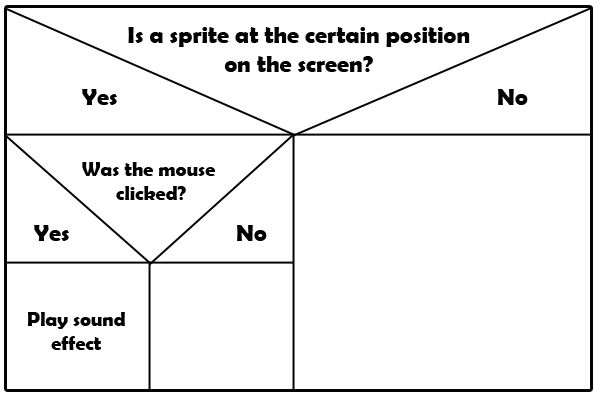


Figure

****

## Playing a Sound When the Bug is Clicked

In order to play sound when clicking on a bug, you need to follow the logic represented in Figure 2.



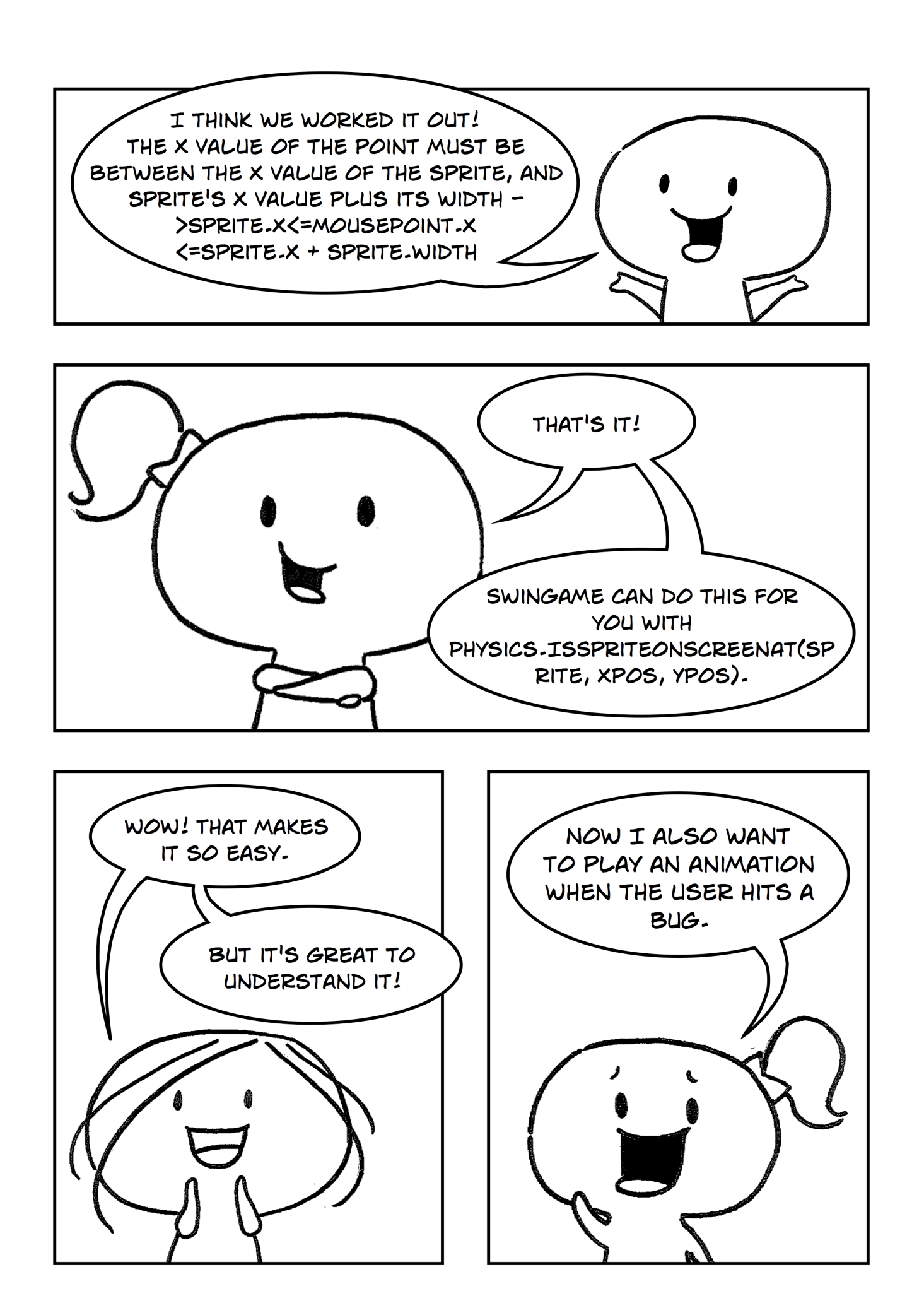
Figure

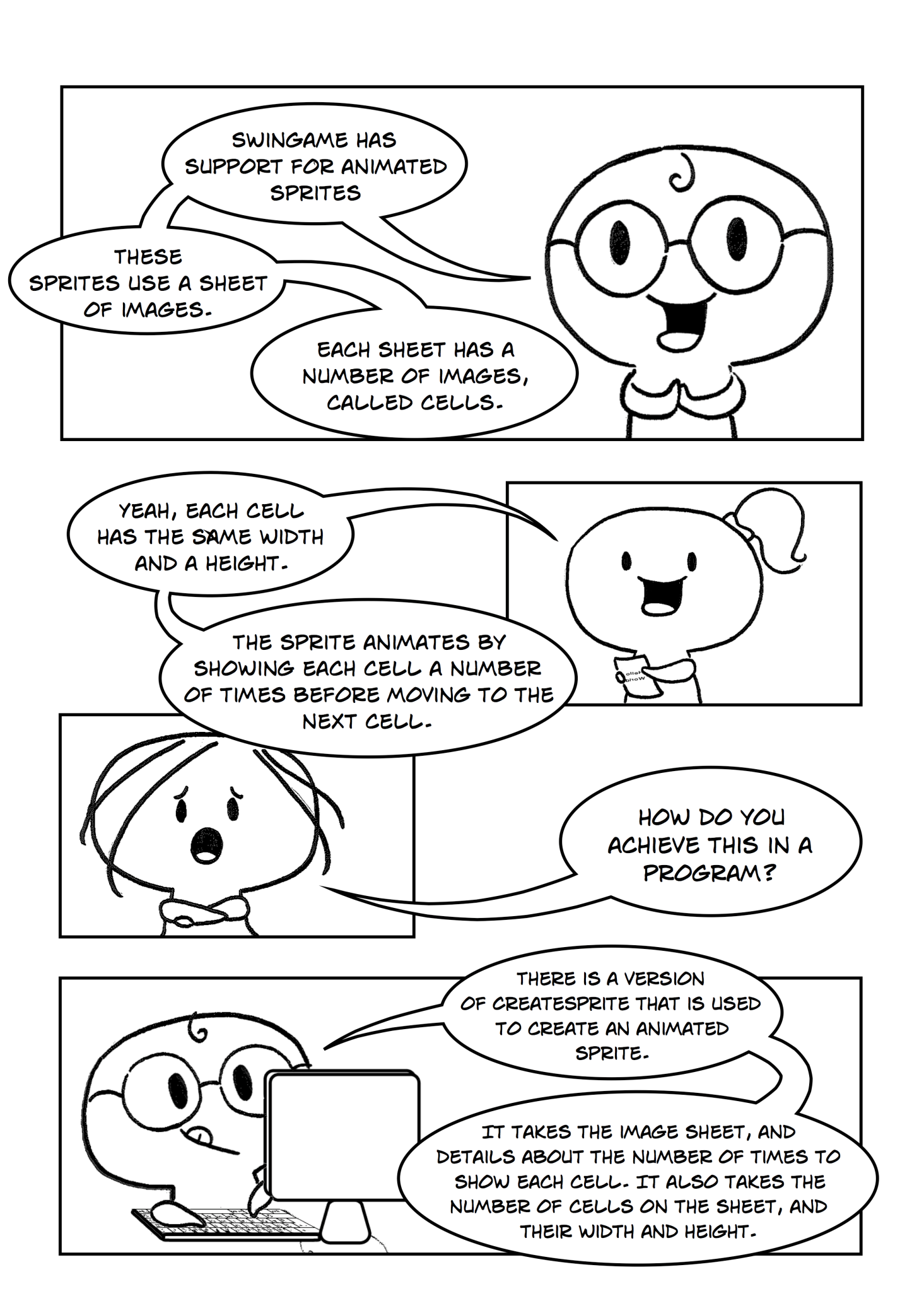
To check whether your sprite is at a certain position on the screen, you should use the Physics.IsSpriteOnScreenAt(NameOfSprite, mousePoint.X, mousePoint.Y mousePoint.X and mousePoint.Y are coordinates of the mouse on the screen. Therefore this code checks to see if the sprite is at the same position as the mouse. Then we have to check if the user has clicked, in order to do this you can use: Input.MouseWasClicked(MouseButton.LeftButton). Both of these are in-built functions which return True or False values. Because the mouse is moving we need to check its position all the time, therefore we need to write our code inside the Game Loop.

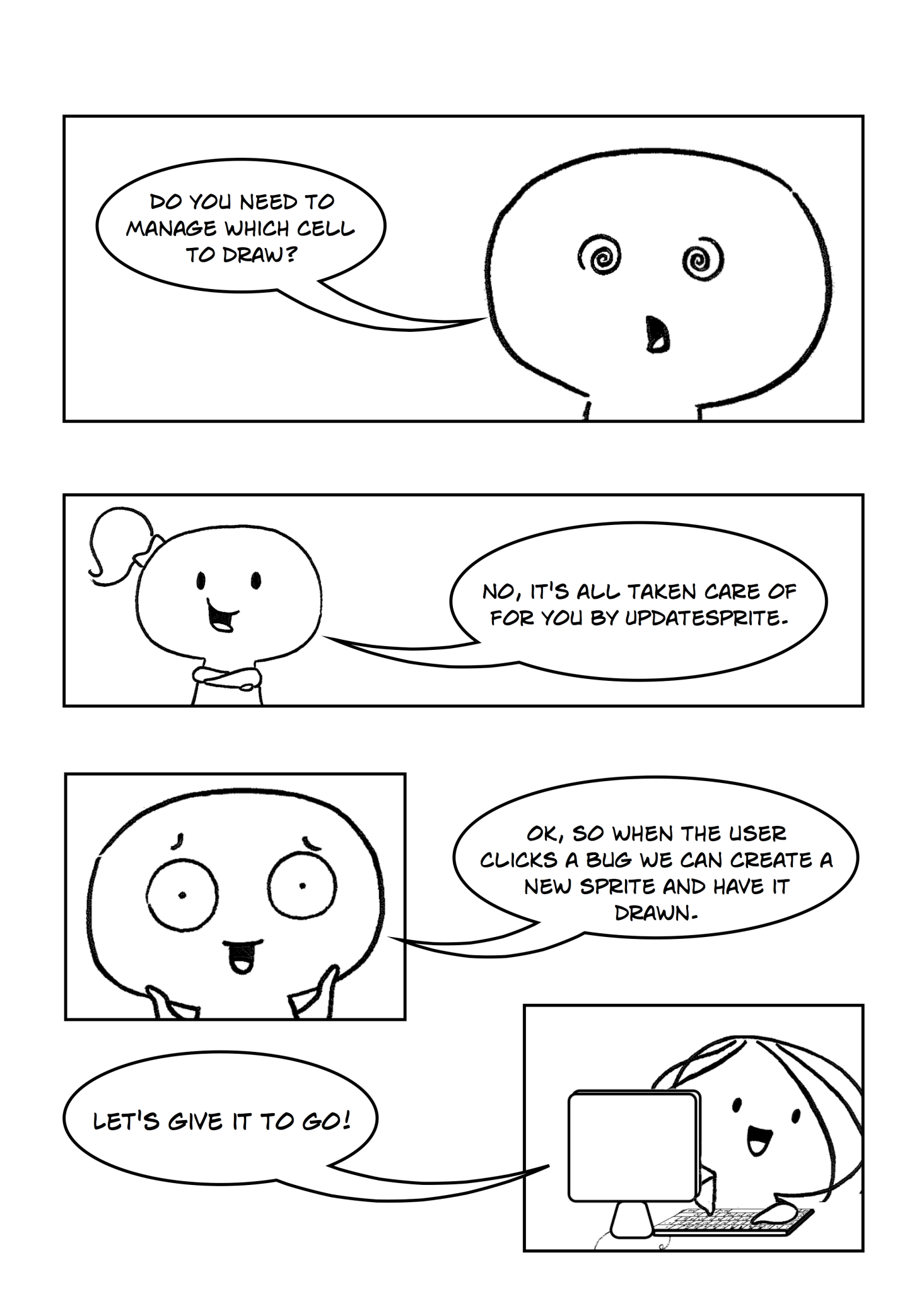
Exercise 2: *Playing sound when clicked on a bug*

cha 2 - worksheet.pngMake the following changes in your program and write your solutions to the worksheet:

Write the code to tell the program to play the “splat.wav” (file provided in the resources folder) when a user clicks on a bug sprite. Use knowledge from previous chapters and follow the logic above.

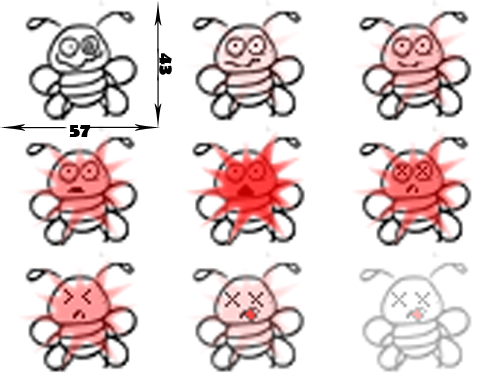
****

****

****

## Animation

SwinGame supports animated sprites. Basically animation in SwinGame is a sheet with a number of images which are divided into “cells”. Each cell has same width and height. Each cell is shown a number of times, before moving to the next one, the higher the number of times each cell is shown the slower the animation. You have to load a sheet of images into your program as a normal image. Figure 3 is the sheet of images that we will use as the “splat” animation in our program. In our case each cell has 57 pixels width and 43 pixels height.



Figure

To create an animation firstly we have to load the “deadBug.png” into our program. Then we need to declare the variable which will represent deadBug animated sprite. It should be done outside the Game Loop. You have to declare (Dim) the variable as a “Sprite” and then assign it to the following function:

Graphics.CreateSprite(GameImage("deadBug"), **40, 10, 57, 43**)

In this example:

**40** – Is the number of times each cell will be shown (speed of animation)

**10** – Is the total number of bug cells. The total number of cells is bigger than the number of cells in the picture; this allows you to hide the animation after it was played. If you will leave the exact 9 frames as total number of frames, the last image in the sheet will stay on the screen.

**57** – Is the width of each frame.

**43** – Is the height of each frame.

Exercise 3: *Declaring the animated sprite variable*

cha 2 - worksheet.pngMake the following changes in your program and write your solutions to the worksheet:

Declare a new animated sprite variable and name it “deadBug”. Use the CreateSprite()function as detailed previously and assign the deadBug image to the variable.

*cha 2 - worksheet.pngQuestion 1: In the following code what is the height of each frame in the animation: Graphics.CreateSprite(GameImage("monkey"),* ***50, 10, 60, 35****)*

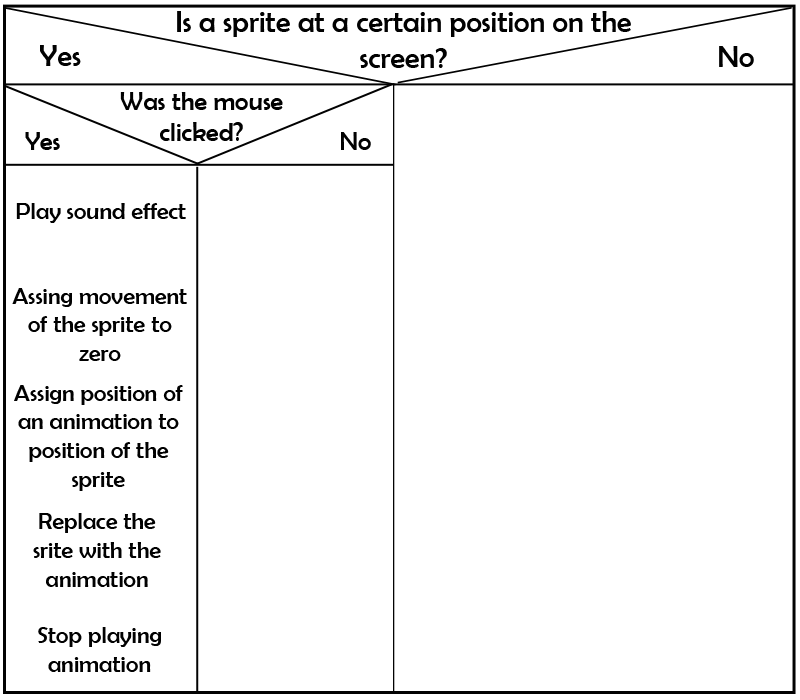
*cha 2 - worksheet.pngQuestion 2: In the following code how many cells are in the animation in the animation? Graphics.CreateSprite(GameImage("chimp"),* ***15, 5, 15, 10****)*

**NOTE:** Do not forget to load a sheet of images into your program before using it. If you have two or more bugs, do the same procedure for all of them.

Now we need to tell the program that when a bug is clicked the animation of dying bug should be played in its place. We can break this down into four steps:

1. When the bug is clicked stop the bug
2. Assign the position of bug to the position of the deadBug
3. Replace the bug sprite with the deadBug animation
4. Stop the animation after playing once

The logic for playing animation action is shown in Figure 4.



Figure

1. Stop the bug

To fix the position where the bug was killed find the if statement we created earlier which checks whether the bug was clicked and plays the splat sound if it was. Inside this statement we need to set the bug movement (x and Y) to 0 to stop it.

1. Assign the position of bug to the position of the deadBug

At this position where the bug was clicked we need to assign the position of the animation to the position of the bug:

deadBug.X = bug.X

deadBug.Y = bug.Y

1. Replace the bug sprite with the deadBug animation

At this point, if you run you program, you will see that the bug will just stop at the position it was clicked. Replace the bug sprite with the animation of deadBug by assigning bug variable to animation variable:

bug = deadBug

1. Play the animation once

If you run you program again and click on the bug, you will see that the animation keeps repeating. To make it so it only plays once we can use

deadBug.EndingAction = SpriteEndingAction

***Did you know?***

*When we assign variables in programs we make whatever is on the left side of the = the same as what is on the right side. For example consider this statement:*

*X = 10*

*Y = 20*

*X = Y*

*At the end of this statement both X and Y will be 20 because in the third line we made X equal whatever Y is. We made whatever was on the left side of the = the same as what was on the right. We made X the same as Y. As you can imagine understanding this is very important to getting our code to work like above where we made the position of the deadBug animation the same as the position bug sprite (e.g. deadBug.X = bug.X)if we had of got them the wrong way around the program would not work.*

*cha 2 - worksheet.pngQuestion 3: In the following code was run what would be the final value of String1?*

*String1 = “medium”*

*String2 = “high”*

*String3 = “low”*

String4 = String2

String2 = String3

String1 = String2

*cha 2 - worksheet.pngQuestion 4: In the following code was run what would be the final value of X?*

*X = 20*

*Y = 40*

*Z = 50*

*X = X × Z*

*Z = Z + X*

*X = X + Z - Y*

Exercise 4: *Playing animation when a bug was clicked.*

cha 2 - worksheet.pngMake the following changes in your program and write your solutions to the worksheet:

Write and implement code that stops the bug and plays the animation in its place when it is clicked. Use the previous information to complete this task.

saveicon.png Remember to save your project (File – Save All). Once you have finished then you can close Visual Studio or move on to the next chapter.