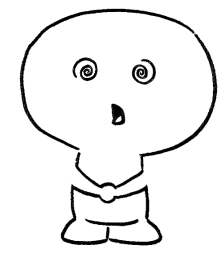
## VBugs

## Chapter 4

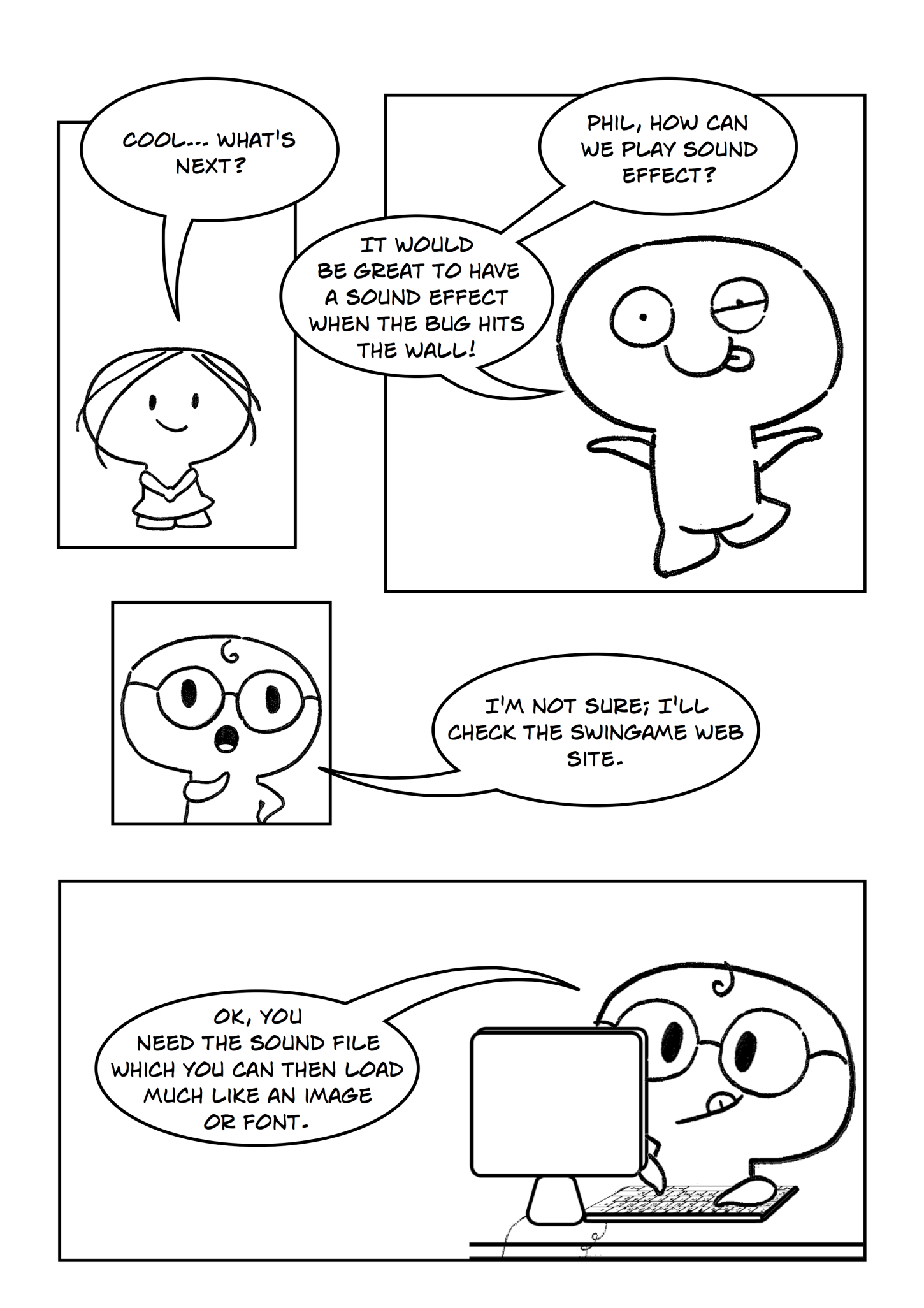
###### Sound and Keyboard

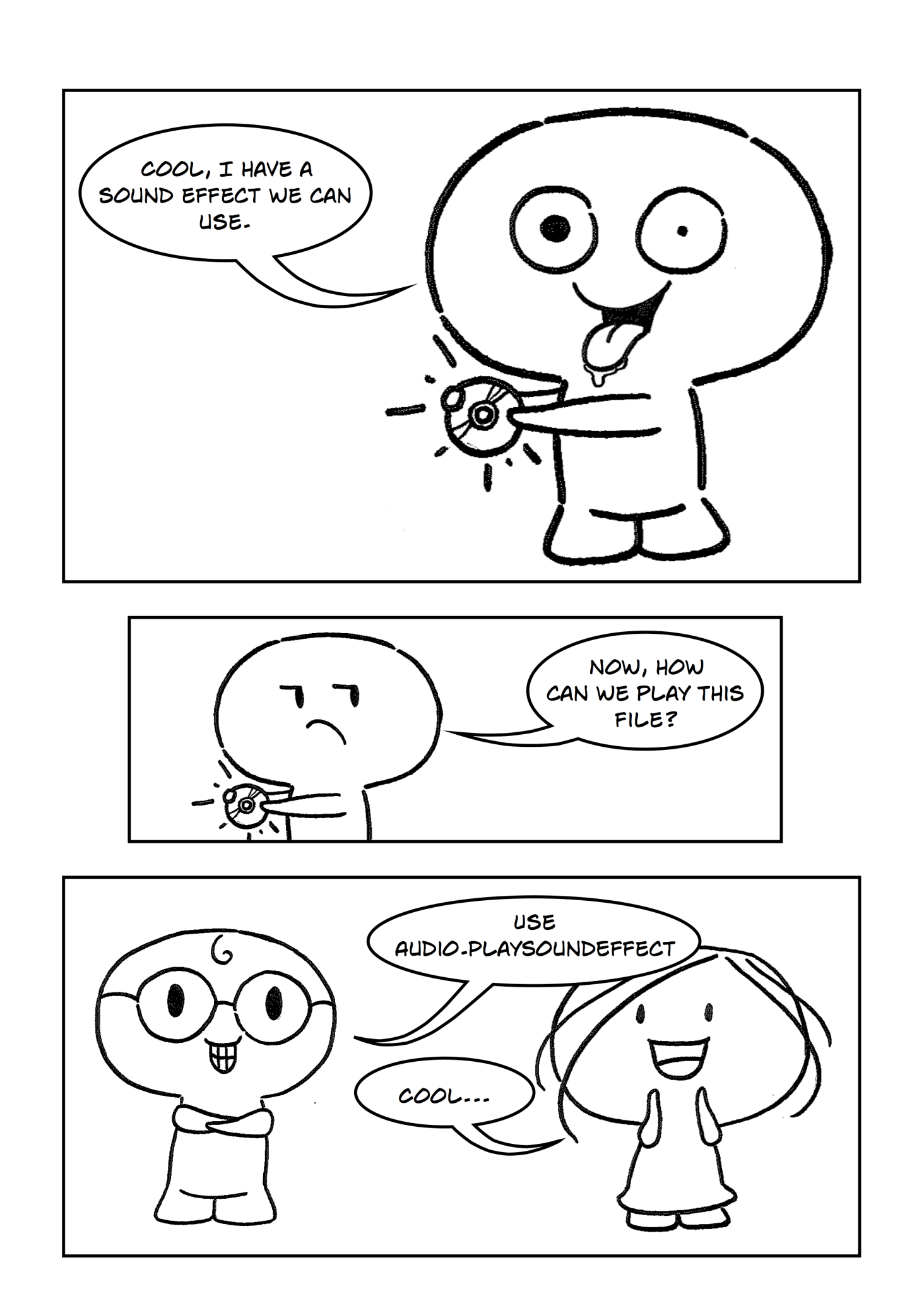




# Summary:

In this chapter you will modify the solution from the previous chapter. You will add sound effects and music and handle a keyboard input. All resources such as music or sound effects will be provided to you.

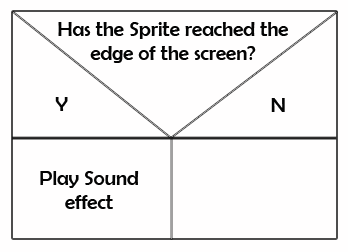
****

****

## Loading Sounds

We want a sound effect to play when the bug hits the wall. The first step is to load sound effect to your program. In order to do so, follow the same steps as the previous tutorials for loading images and fonts. The audio file provided needs to be placed within the Resources > Sounds folder.

To play sound when the bug hits the wall you need to follow the logic in the NS diagram below: (Figure 1)



Figure

Open the project from the previous exercise (Bugs.sln) and continue to work with this project.

Exercise 1: *Loading the sound effect*

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

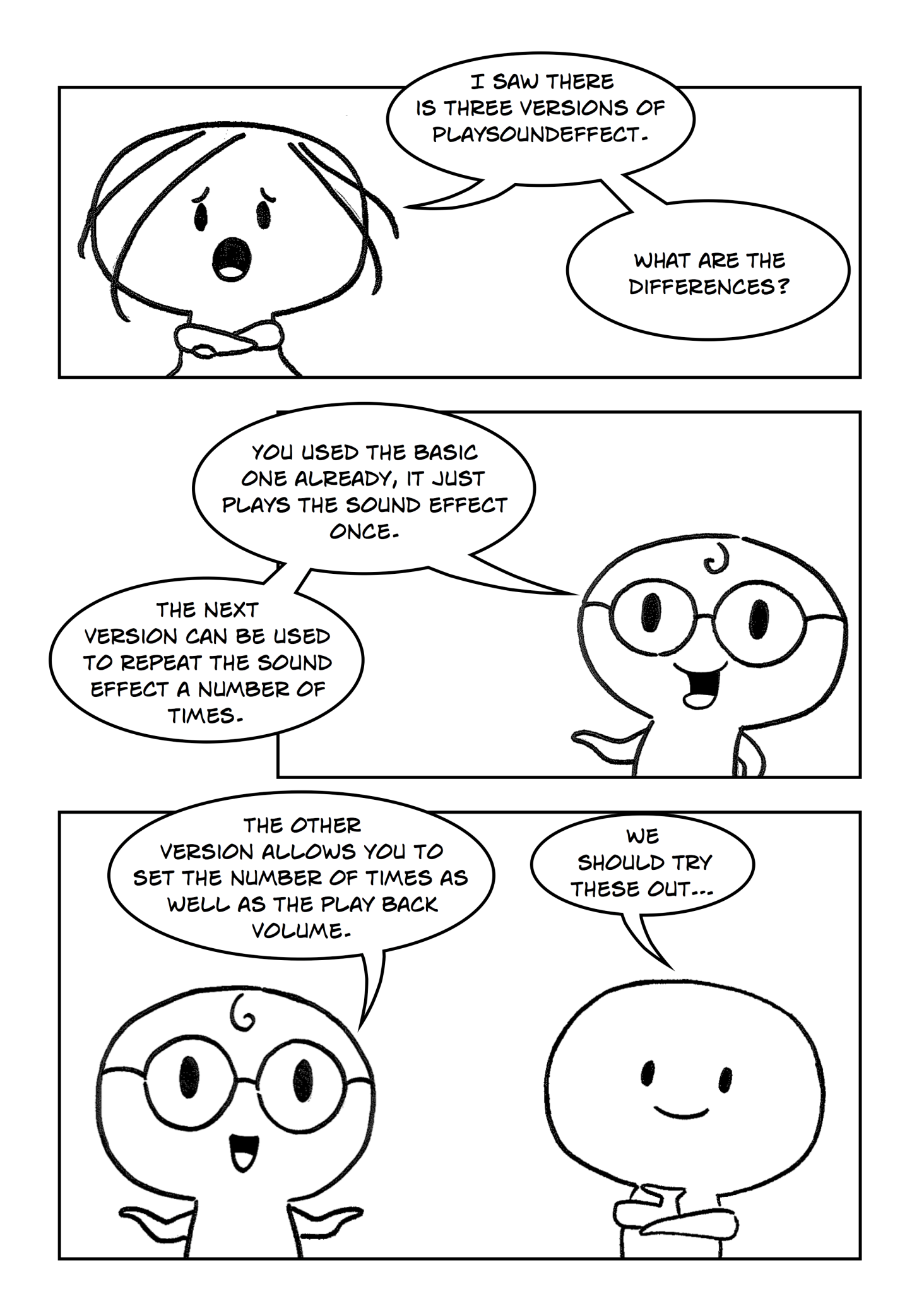
Load “hit20.wav” file into your program. Use NewSound() function inGameResources.vb.

Exercise 2: *Playing sound effect*

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

Play sound effect each time bug hits the wall. Use Audio.PlaySoundEffect(GameSound("soundName"))

\*You will need to add this line in four places for the four different walls.

****

## Playing Sounds

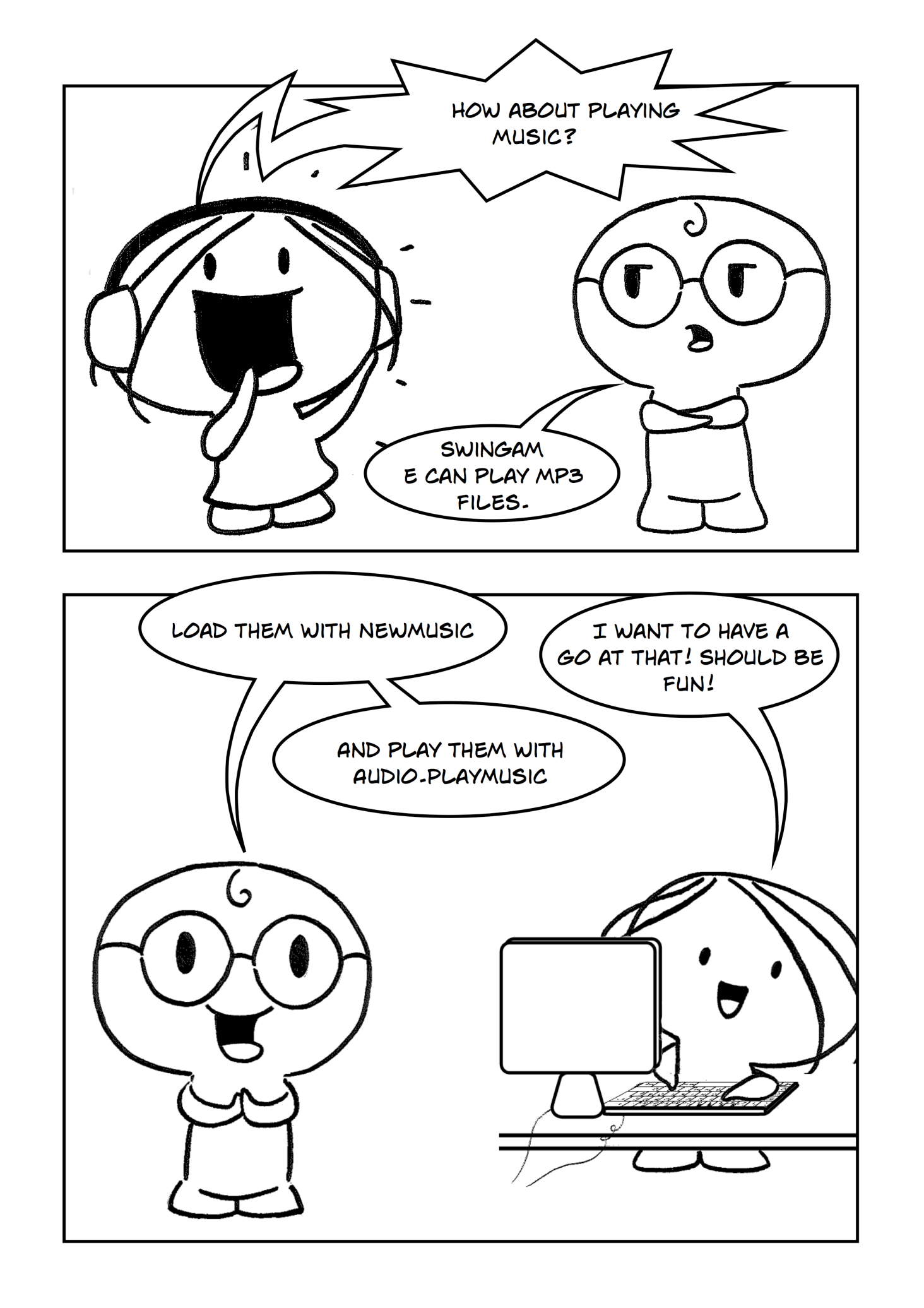
A sound effect can be played in three possible ways; so far we have used the most basic version which simply plays the sound effect once. The second version enables us to play the sound effect for a number of times and by using the third version we can tell the program to play the sound effect for a number of times as well as set the play back volume.

Exercise 3: *Using different versions of PlaySoundEffect*

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

1. Use Audio.PlaySoundEffect(GameSound(“soundname”), NoOfLoops). Use this function with “hit20.wav” sound effect, which you previously loaded, before the start of the Game Loop.
2. Use Audio.PlaySoundEffect(GameSound(“sooundname”), NoOfLoops, Volume). Use ths function with “hit20.wav” before the start of the Game Loop.

**Note:** NoOfLoops is number of times that the sound will be played. Volume is numeric representation of volume in your program (preferably use between 0 and 10).

****

## Playing Music

In addition to playing sound effects, we can also play music. We can load a music track into our program and play it once, or play in a certain amount of times, or loop it infinitely. All of these varieties are enclosed in one function – PlayMusic().

Exercise 4: *Playing music*

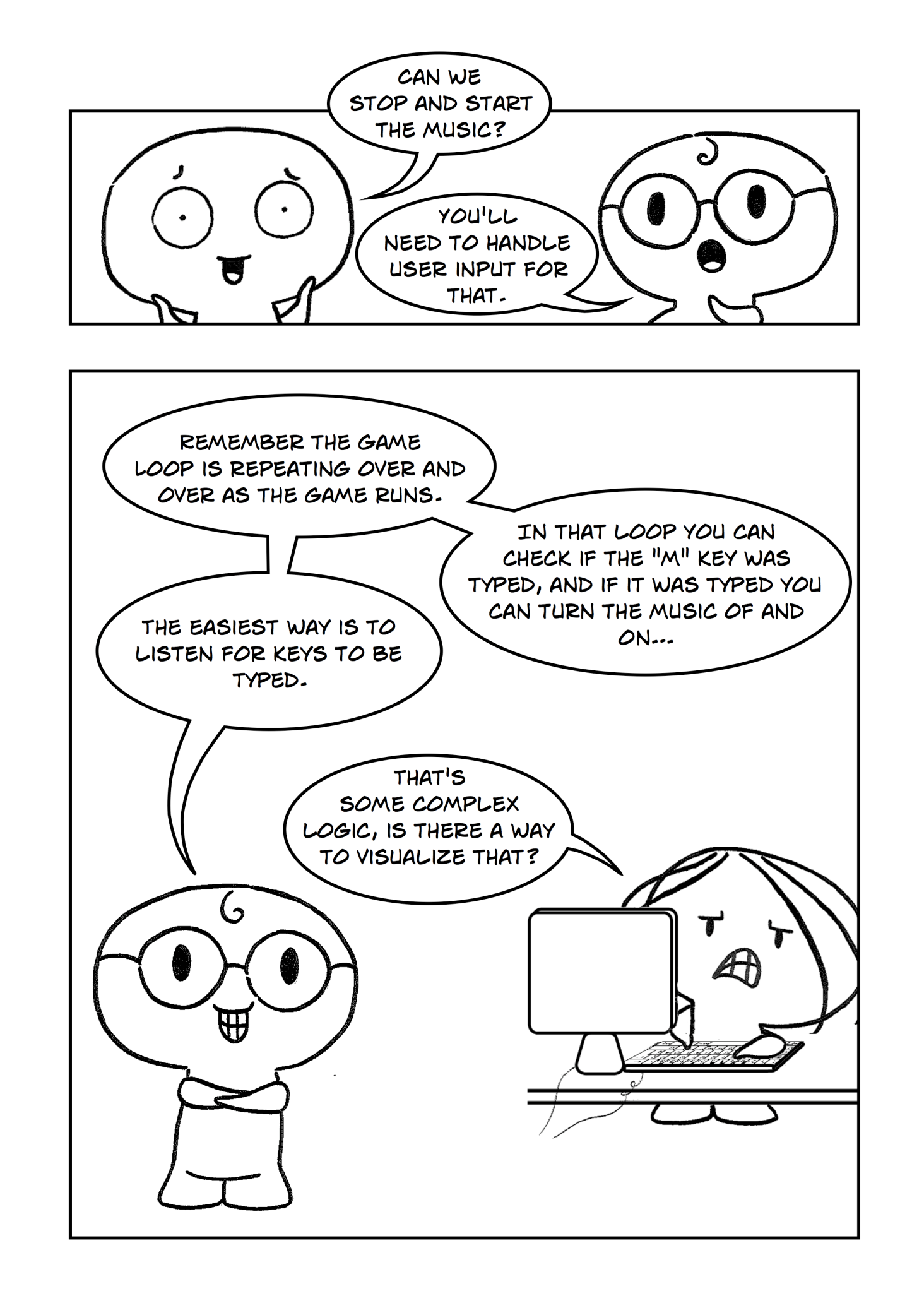
cha 2 - worksheet.pngMake the following changes in your program and write your solutions to the worksheet:

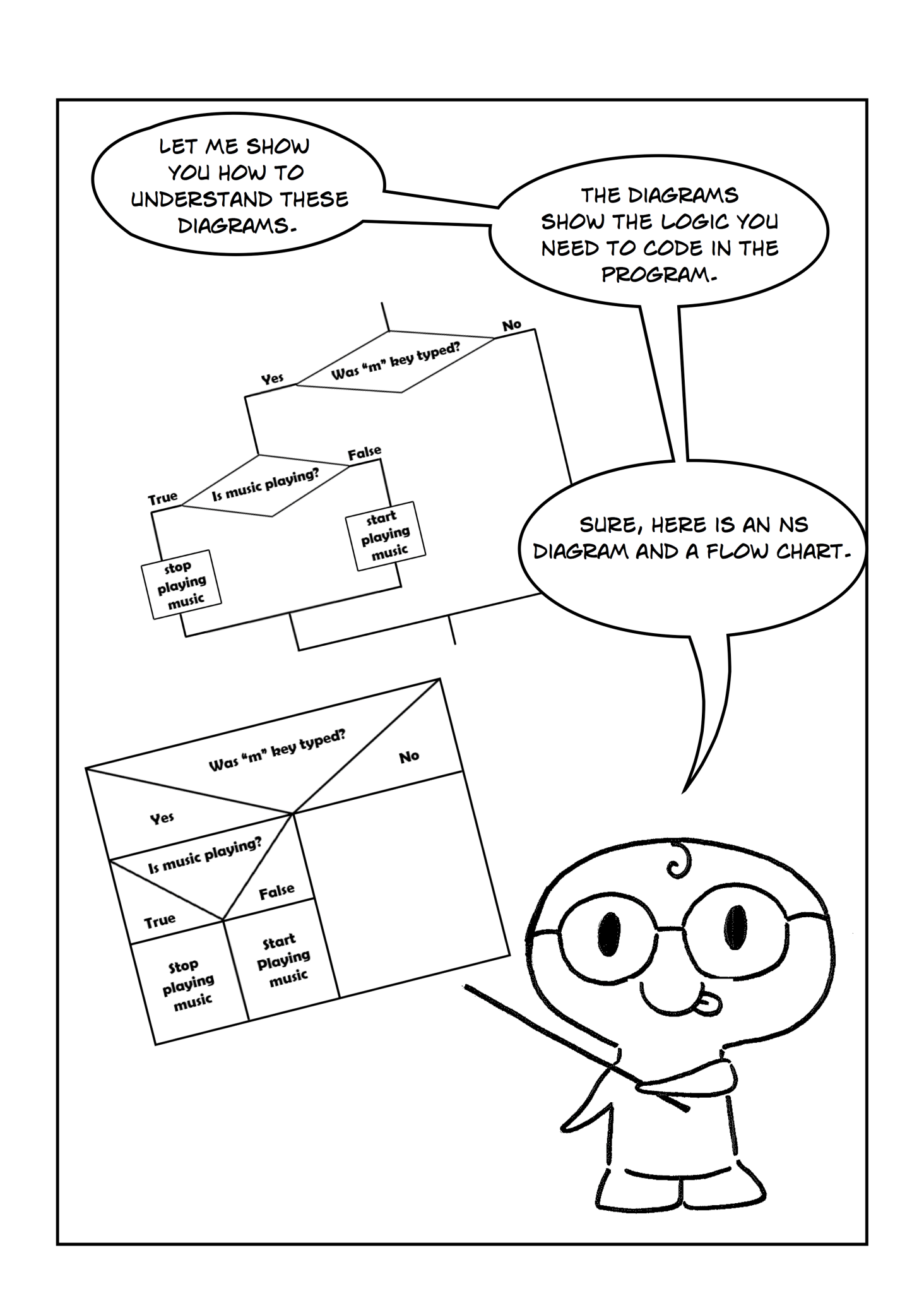
1. We want to play a music track which is infinitely repeated. To do so, load “lion.mp3” into your program (Resources – Sounds then add it to GameResources.vb). Instead of using NewSound() in GameResources.vb use NewMusic in the LoadMusic sub.
2. Then use Audo.PlayMusic(GameMusic(“trackname”), -1) in GameLogic.vb where -1 is indicator of looping to infinity.
3. We finally need to tell the program to stop playing the music when the program is closed, add Music.Stop()after Audio.CloseAudio() at the end of the program.

**NOTE:** To load music track use NewMusic(“trackname”, “track”) inside the LoadMusic Sub,

***Did you know?***

**** You can find your own music and sound effects online and implement them using the same methods as described above, you can use a program like iTunes to import them onto your computer and convert them to mp3 or wav files.

****

****

## Stopping and Starting Music

You can stop music and start playing it any time you want. In order to do so you have to handle user input in your program. You can do this by using If..Else statement. The logic for this is shown in Figure 3 and 4.

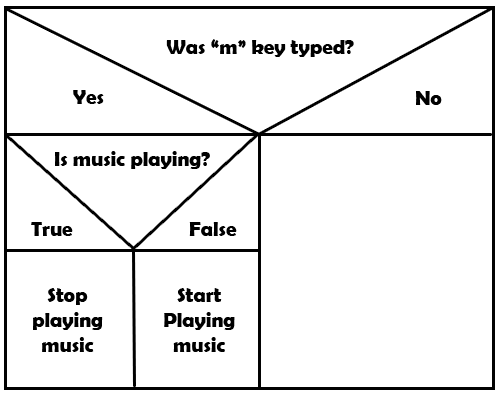
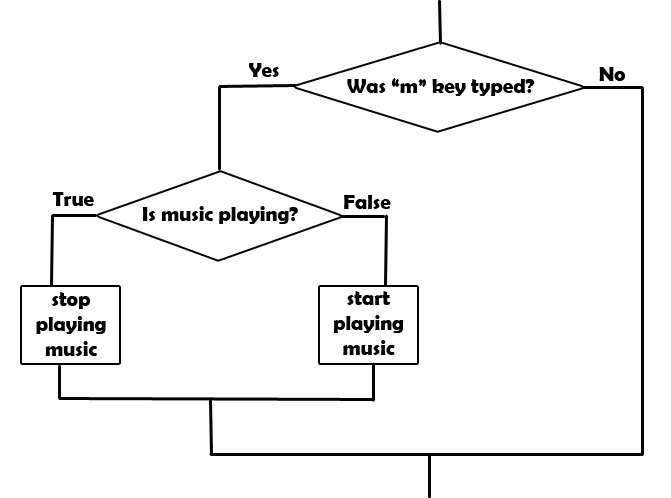


Figure 3

Figure 2

***Did you know?*** Figure 3 is a flowchart. A flowchart is common type of [chart](http://en.wikipedia.org/wiki/Chart) that represents an [algorithm](http://en.wikipedia.org/wiki/Algorithm) or [process](http://en.wikipedia.org/wiki/Process_(general)), showing the steps as boxes of various kinds and their order by connecting these with arrows. Flowcharts are used in designing or documenting a process or program in various fields of work.

In order to check whether “m” key was typed you should use: Input.WasKeyTyped(SwinGame.Keys.VK\_M). This is a built-in function in the SwinGame that checks whether the key was typed and returns true if it was and false if wasn’t.

To check whether music is playing we use Audio.IsMusicPlaying(). This function returns true if the music is playing and false if it is not. You can stop playing music by calling Audio.StopMusic() function and start to play it again by calling Audio.PlayMusic(GameMusic("trackname"), -1).

Exercise 5: *Handling user input.*

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

Allow to start and stop music in your program when typing the “m” key. Use the logic represented in Figure 1 and 2 and functions introduced above.

**NOTE:** The code for this exercise has to be placed inside the game loop.

****

## Changing the Volume

To change the volume you should follow the logic shown in Figure 4.

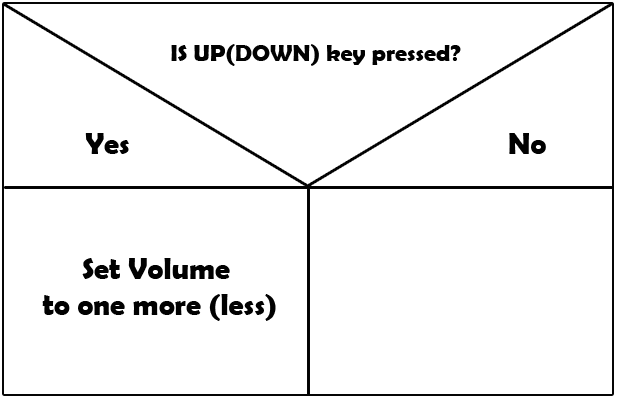


Figure 4

You can check whether the key was pressed by using Input.IsKeyPressed(SwinGame.Keys.VK\_Up). This is a built-in function in SwinGame which returns True or False. To tell to your program to change the volume when arrows were pressed use Audio.SetMusicVolume(Audio.MusicVolume + 0.01F).

Exercise 6: *Changing the volume*

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

Write the code that allows you to change the volume up when UP arrow key was pressed and down when DOWN arrow key was pressed.

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.