Using Sound and Keyboard Control

What Will I Learn?

Objectives

- Write programming statements to include sound in a program
- Write programming statements to include keyboard movements in a program

Why Learn It?

Purpose

You have learned how to make objects perform simple movements, however, you haven't created a true game until the person who plays it has the ability to manipulate the game's objects with their own keyboard. Programming keyboard movements allows a game to be intereactive and interesting.

Keyboard Controls

Games are controlled by a human or computer player, either with a remote control or keyboard controls.

To make a scenario behave like a true game, program statements that include keyboard controls so the player can control one or more objects in the game.

The isKeyDown Method Signature

The `isKeyDown` method:

- Checks if a key on the keyboard has been pressed
- Is located in the Greenfoot class
- Is a Static method (associated with a class)
- Returns true or false
- Expects a String argument in the parameter list
- Can be used as a condition in an if-statement

Examine the `isKeyDown` method signature:

```
Public static boolean isKeyDown(String key)
```

String Parameter in isKeyDown Method

The String parameter expects the name of the key to press on the keyboard.

- A String is a piece of text (e.g., word or sentence) written in double quotes.
- Examples:
  - “This is a String”
  - “A”
  - “name”

Find a key's name by looking at your keyboard. Sometimes the key name isn't evident (e.g., right cursor key is called “right”).

Example isKeyDown Method

This code in the act method uses the left and right keys on the keyboard to control the Duke object's direction as he moves.

```
/**
 * Act - do whatever the Duke wants to do. This method is called whenever
 * the 'Act' or 'Run' button gets pressed in the environment.
 */
public void act()
{
    move(1);
    if (Greenfoot.isKeyDown("left"))
    {
        turn(-3);
    }
    if (Greenfoot.isKeyDown("right"))
    {
        turn(3);
    }
```

Include Sound

Sounds are a great way to:

- Give feedback to the player when they win, lose, or achieve minor victories throughout the game.
- Include background noise in a game.

The `playSound` method is used to include sound in a game. This method:

- Is located in the Greenfoot class.
- Expects the name of a sound file (as String) as an argument in the parameter list.
- Does not return data.

Sound Example

Code to include sound is added to the lookForCode method, so that whenever the Duke object eats code, he makes a sound.

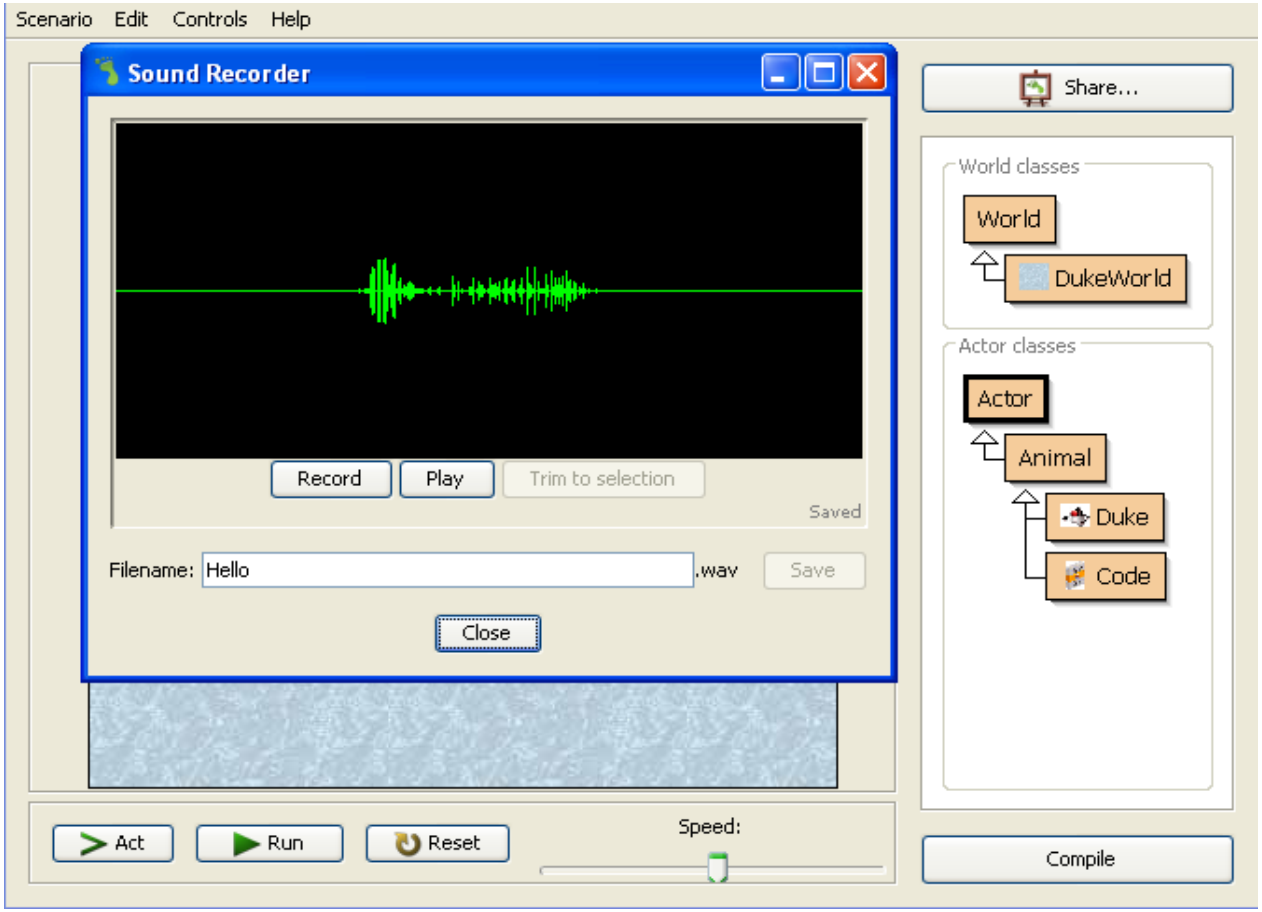
```
/**
 * Look for Code. If Duke finds code, he eats it. If Duke doesn't find code, he does nothing.
 */
public void lookForCode()
{
    if (canSee(Code.class))
    {
        eat(Code.class);
        Greenfoot.playSound("Chomp.wav");
    }
}
```

Create Original Sounds in Greenfoot

Follow these steps to create a sound recording:

1. In the Controls menu in the environment, select Show Sound Recorder.
2. Press Record, then talk into your computer's microphone to record the sound.
3. Press Stop Recording when finished.
4. Press Play to play back the sound.
5. Re-record if necessary.
6. Enter a file name, then click Save to save the file to your scenario. The file is now ready to use in your code.

Greenfoot Sound Recorder



Summary

In this lesson, you learned how to:

- Write programming statements to include sound in a program
- Write programming statements to include keyboard movements in a program

Practice

The exercises for this lesson cover the following topics:

- Enhancing programs with sound and keyboard movement
- Concept summary review