Sapphira2D Engine Manual

Wolfylink

October 24, 2015

Contents

Chapter 1

Introduction

Sapphira2D, or S2D, is a java game engine which helps to create easily and quickly a game with two dimensional graphics.

S2D is a java engine, and is obviously usable only if you are planning to code your game in java. S2D creates a computer-launchable application ; other versions creating java applets or smartphone-launchable applications shall be developped later.

To install S2D, you have to use the empty project given with this manual. The explanation of the file tree is the next step of this manual.

Chapter 2

Main directories

2.1 Graphics

This directory is where all your images have to be. You can use as much subdirectories as you want inside.

2.2 Sounds

This directory is where all your songs and sound effects have to be. You can use as much subdirectories as you want inside.

2.3 Data

This directory contains all the data of the game. S2D handles that directory, so you don't need to touch it. However it can be useful to know how it works.

S2D introduces the abstract idea of data : it is a portion of information about the game. It can be all the information of a certain level of the game, the options choosen by the player, or his save data...

All these kinds of data are saved in this directory, and managed by the DataEngine of S2D, which can create, edit, or read all the information that the game needs.

See the DataEngine (3.7) for more information about it.

2.4 System

This directory is where you can find all the java code of the game.

2.4.1 game

You have to put all your code in this directory — or this package. Of course, you can arrange this directory how you want to. It can be useful to split the

code in several packages, for example the scenes(3.4), the data (3.7), etc...

2.4.2 sapphira2D

This directory contains all the S2D engine, plus several generic contents that can be useful in many games. The explanation of this complex directory follows.

Chapter 3

How to use S2D ?

All this chapter is about how to use the S2D with java. We place ourselves in the System directory.

3.1 GameLauncher

The GameLauncher is the class which contains the main() method which you have to call to launch the game. It is located in the System directory.

3.2 Game

The Game class is the main class of the game ; it is located in the package sapphira2D. This class contains all the S2D managers as public attributes, which can (and should) be used. This class also handles the good working of the game, calling the update methods at each frame.

3.3 library

This package contains a lot of generic elements that many games could need. It can be graphic elements, scenes, etc... See the library chapter (4) for more details about these elements.

3.4 sceneEngine

3.4.1 SceneManager

3.4.2 Scene

3.5 graphicEngine

3.5.1 GraphicManager

The Game class contains a GraphicManager attribute. GraphicManager is only an interface, because it can be useful to have different managers, to have the same result in different platforms (applets, smartphones...). For the computer, it is the GraphicManagerComputer, using Swing.

3.5.2 GraphicElement

A GraphicElement is an object that can be drawn in the screen. Several basic elements already exist in the library : GraphicImage, GraphicImage-Portioned, GraphicText, GraphicShape.

It is possible for you to create others elements, using these basic elements as bricks of your bigger element. To do that, you have to make a class extend GraphicElement and you have to redefine the paint(), erase(), and update() methods. Actually, redefining these methods means calling the bricks' same methods, so that the GraphicManager knows what to do...

Note that there exists some pre-coded elements in the library : Window (and its sons), ...

3.6 eventEngine

3.6.1 EventManager

The Game class contains an EventManager attribute. EventManager is only an interface, because it can be useful to have different managers, for the different platforms (applets, smartphones...). This interface has not a lot of methods, because for each platform, the kind of events can be different... For the computer, it is the EventManagerComputer, using Swing.

The EventManagerComputer handles all kinds of events : mouse, keyboard, and also the game window's state. For these 3 kinds of events, there is a specific manager. It is possible (and necessary) that you should install different managers during the game, because, for example, the left click of the mouse has not the same utility in each case.

3.6.2 KeyBoardManager

This manager is, of course, especially for the computer version of S2D. Using the observator pattern, it defines 3 methods which are automatically called when the event occurs, if the manager is installed.

The method whenClicked is called when the key is just pressed, and the whenReleased when the key is just released. The method whichPressed allows to know, for each key and at each frame, the number of frames in a row in which the key is pressed. A value of -1 means that the key is not pressed currently.

To create a new KeyBoardManager, you just have to create a class extending this one, and to define what is done in each method. Then, you can install it in the EventManager.

3.6.3 MouseManager

This manager is, of course, especially for the computer version of S2D. Using the observator pattern, it defines several methods which are automatically called when the event occurs, if the manager is installed.

The method whenClicked is called when the button is just pressed, and the whenReleased when the button is just released. The method whichPressed allows to know, for each button and at each frame, the number of frames in a row in which the button is pressed. A value of -1 means that the button is not pressed currently. The methods whenExited and whenHovered are called when the mouse exits or hovers the game window ; and the whenScrolled method is called when the player is scrolling.

To create a new MouseManager, you just have to create a class extending this one, and to define what is done in each method. Then, you can install it in the EventManager.

3.6.4 WindowManager

This manager is, of course, especially for the computer version of S2D. Using the observator pattern, it defines several methods which are automatically called when the event occurs, if the manager is installed.

The method whenActivated and whenDeactivated are called when the window just becomes the priority window in the computer, or just loses its priority. The methods whenOpened and whenClosed are called when the window just opened or is just closed ; while whenClosing is called just before the closing of the window. Finally, whenIconified and whenDeiconified are called when the window is just iconified or when it's just deiconified.

To create a new WindowManager, you just have to create a class extending this one, and to define what is done in each method. Then, you can install it in the EventManager.

3.7 dataEngine

Chapter 4

Library

Chapter 5

Examples