In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 'Getting Started with openFrameworks'
- A synopsis of the book’s content
- More information on openFrameworks Essentials

About the Authors

Denis Perevalov is a scientist at the Krasovsky Institute of Mathematics and Mechanics, Ekaterinburg, Russia. His main research interests are computer vision and interactive systems.

He has been teaching the information processing and algorithms of technical vision courses at the Ural Federal University since 2010. He is also the author of the book Mastering openFrameworks – Creative Coding Demystified, Packt Publishing, 2013.

Igor (Sodazot) Tatarnikov is a video artist, animator, and VJ from Moscow, Russia. He creates media content in various techniques, including animation, stop motion, and generative art.

Igor and Denis are cofounders of the visual laboratory Kuflex (2011), which made numerous interactive video installations and audio-visual performances for numerous exhibitions and commercial projects. Kuflex’s clients include the Garage Museum of Contemporary Art (Moscow), the Jewish Museum and Tolerance Center (Moscow), the Moscow International Festival «Circle of light», Microsoft, Nokia, Samsung, and Master Card. Most of these works were made using openFrameworks.
openFrameworks Essentials

openFrameworks Essentials is a guide that can be used to learn and use openFrameworks in order to develop creative and artistic real-time applications. The book contains an insight into how openFrameworks can be used to implement your creative ideas in powerful projects by working on the OS X, Windows, Linux, iOS, Android, and Raspberry Pi devices. Although openFrameworks is a very popular toolkit for creative coding, the number of books describing openFrameworks is not high.

This book is a fast-paced tutorial that begins with installing openFrameworks and then takes a step-by-step approach towards using openFrameworks to build a video synthesizer project. We will investigate and implement features such as 2D and 3D graphics, GUI, shaders, and reaction on sound, using the OSC networking protocol and Arduino.

The openFrameworks version considered is 0.8.4.

What This Book Covers

Chapter 1, Getting Started with openFrameworks, introduces openFrameworks by explaining its installation and running one of its examples. Additionally, it discusses the structure of the video synthesizer project, which will be developed in the course of the book.

Chapter 2, Creating Your First openFrameworks Project, guides you to create a new openFrameworks project and implement 2D graphics.

Chapter 3, Adding a GUI and Handling Keyboard Events, explains creating a graphical user interface consisting of controls such as sliders and checkboxes. Also, it covers implementing keyboard events, using system dialogs, and saving screenshots.

Chapter 4, Working with Raster Graphics – Images, Videos, and Shaders, explains drawing images and videos, capturing video from a camera, mixing videos using additive blending, and creating the kaleidoscope video effect using a fragment shader.

Chapter 5, Creating 3D Graphics, introduces the basics of 3D graphics with openFrameworks by drawing a sphere in 3D and then texturing and deforming it.

Chapter 6, Animating Parameters, discusses using various data sources, such as Perlin noise, sounds, and text files, to automatically control the parameters of the project.
Chapter 7, *Distributed and Physical Computing with Networking and Arduino*, explains how to control your project from other programs using the OSC networking protocol. It describes building such controlling programs using openFrameworks by itself as well as the Python and Max/MSP programming languages. Also, it covers receiving data from an Arduino device.

Chapter 8, *Deploying the Project on iOS, Android, and Raspberry Pi*, guides you to run the light version of the developed video synthesizer on mobile and Raspberry Pi devices.

Chapter 9, *Further Resources*, suggests the direction to further enhance the video synthesizer project, provides sources of additional information on openFrameworks, and discusses debugging and speeding up openFrameworks projects.

Appendix A, *Video Synthesizer Reference*, is a comprehensive documentation of the developed video synthesizer project. It contains a description of all GUI controls, control keys, and media files used by the project.

Appendix B, *openFrameworks Quick Reference*, is a useful reminder of the basic openFrameworks functions and classes used in the book.
Getting Started with openFrameworks

In this chapter, we will introduce openFrameworks by covering the following topics:

• What is openFrameworks?
• Installing openFrameworks
• Running an openFrameworks example

Finally, we will discuss the video synthesizer project, which we will develop throughout the book.

What is openFrameworks?

openFrameworks is a free, open source C++ framework that is intended to develop real-time projects, which use most modern computing devices' capabilities, such as video, audio, networking, and computer vision.

openFrameworks is developed by Zach Lieberman, Theodore Watson, and Arturo Castro, together with a big openFrameworks community. The current openFrameworks version is 0.8.4.

Its main features are:

• openFrameworks is a framework for the easy development of real-time applications. All you need to do is implement a number of events, such as starting project and drawing, and openFrameworks will maintain all the stuff behind this, such as creating a graphical window and listening for mouse and keyboard events.
Getting Started with openFrameworks

- Its conception and philosophy is oriented for use in creative and experimental projects. openFrameworks contain many ready-to-use functions and classes to work with 2D and 3D graphics, video, audio, and so on.
- It's a C++ framework, that is, you will code it using the C++ programming language. Your code will be compiled in native machine code and executed very efficiently. Many algorithms that work slowly when implemented in languages such as Python, Java, or Processing, could work significantly faster when implemented in openFrameworks.
- openFrameworks is highly extensible. Its extensions are called addons. They are open source and are built by members of the openFrameworks community. Several hundreds of existing addons give you an opportunity to extend your project with the capabilities of various software libraries, popular sensors, and gadgets.
- openFrameworks is cross-platform — the code you will write can be executed on many popular platforms, including OS X, Windows, Linux, iOS, Android, and Linux ARM devices such as Raspberry Pi.
- Finally, it has a very friendly and active community. If you get stuck during developing with openFrameworks, you can get help from the openFrameworks forum.

Begin your acquaintance with openFrameworks by visiting and exploring its site openframeworks.cc. It contains a lot of useful information about openFrameworks, including download links, tutorials, documentation, forums, and news.

Installing openFrameworks

To develop projects with openFrameworks, you need to install an Integrated Development Environment (IDE) with the C++ compiler and openFrameworks.

To install them, perform the following steps:

1. Go to openFrameworks' download page at openframeworks.cc/download.
2. Find your platform on the page (OS X, Linux, Windows, iOS, Android, or Linux ARM) and select the desired IDE (Xcode, Code::Blocks, Eclipse, or Visual Studio). For developing with iOS, an OS X computer is required. Development on a Linux ARM device is done using the C++ compiler directly, without an IDE.

All the IDEs mentioned are free. But, to run your project on an iOS device, you must buy an iOS Developer License from Apple for $99 per year.
3. Download the openFrameworks archive by clicking on the corresponding link and unzip it.

4. Click on the corresponding setup guide link and follow the instructions shown for installing and configuring the IDE and openFrameworks. Now you can develop and run your projects in openFrameworks.

Running openFrameworks projects on desktops (Mac OS X, Windows, or Linux) is simpler than on mobiles. So, if you are a novice in C++ or openFrameworks, we recommend that you start to develop your very first openFrameworks project for a desktop OS rather than a mobile one.

After a while, when you are a little more comfortable with C++ and openFrameworks, you can easily port your project to mobiles, if needed. You will learn how to deploy openFrameworks projects on mobiles and Raspberry Pi in Chapter 8, Deploying the Project on iOS, Android, and Raspberry Pi.

The openFrameworks folder structure

It’s time to look inside openFrameworks, so open the openFrameworks folder. It consists of a number of folders and files, as shown in the following screenshot (the screenshot is for OS X, but the folder structure is similar for all other platforms):

- addons
- apps
- CHANGELOG.md
- docs
- examples
- export
- INSTALL.md
- libs
- LICENSE.md
- other
- projectGenerator_osx
- README.md
- scripts
The most important folders for us now are **apps** and **examples**. The **apps** folder is the place where your own projects will be stored. The **examples** folder contains a collection of examples, demonstrating all the aspects of openFrameworks programming. The examples are categorized by topics, including 3D, graphics, sound, and video.

Now, let's run one of the examples.

**Running your first example**

We would like to introduce one of the funniest openFrameworks examples, which we like a lot. The example captures the images from a webcam and draws it on the screen as a fancy 3D surface, as shown in the following picture:

![3D surface generated by an openFrameworks example](image)

This is the **meshFromCamera** example located in the **examples/3d** folder.

This example requires a webcam. If your computer does not have a built-in webcam, and you have no external webcam to connect to, please work with another example, for instance, **examples/3d/3DPrimitivesExample**.
We cannot run the example immediately after installing openFrameworks. The reason is that openFrameworks examples are distributed as source files only, without executable files that you can run.

To obtain the executable file, we need to build the project, that is, compile its C++ source code to machine code and link it into an executable file. Let's do it and then run the example by performing the following steps:

1. Open the example folder examples/3d/meshFromCamera.
2. Find there the file named meshFromCamera.xcodeproj (Xcode project), meshFromCamera.sln (Visual Studio project), or meshFromCamera. workspace (Code::Blocks project).
3. Double-click on this file, and the meshFromCamera project will open in your IDE.
4. Build the project by pressing Command + B in Xcode, F7 in Visual Studio, or by clicking on the Build button in Code::Blocks.

**For Xcode users**

In the project, there exist several schemes that indicate the part of the project to build and a number of settings for it. Often at first run, the scheme is set to openFrameworks, as shown on the following screenshot:

![Scheme selector in Xcode](image)

If so, the compiler builds openFrameworks, but not the project. To resolve the issue, please click on the scheme name. A context menu will appear. Select the meshFromCamera Release scheme there. After this, perform building as described in step 4.

5. Run the project by clicking on the corresponding button in your IDE (normally, it's depicted as a small triangle, meaning the play symbol). After running, you will see your webcam's image drawn as a distorted 3D surface.
6. Press Esc to stop the execution.
This example involves several topics, such as capturing frames from the camera, creating 3D surfaces, and texturing them. You will discover these topics in Chapter 4, Working with Raster Graphics – Images, Videos, and Shaders, and Chapter 5, Creating 3D Graphics, and will be able to build similar and even more sophisticated projects easily.

The video synthesizer application

A video synthesizer is an electronic device or a computer program that generates and transforms a video signal in a complex way. Video synthesizers are used for creating live visuals for DJ sets and performances, or serve as a visual part of interactive installations and mobile apps.

Probably the most famous analog video synthesizer was made by John Whitney in the middle of the twentieth century. Visual effects, obtained with the synthesizer, are demonstrated in his film Catalog (1961). It's worth watching on YouTube.

Throughout the book, we will build our own video synthesizer application. We will start with a simple project in the next chapter and will extend it by adding new features with each new chapter. At the end of the book, we will get a fully-fledged video synthesizer, which reveals many openFrameworks capabilities.

Typically, a video synthesizer consists of several modules—video generators, video players, video effects, and video mixers—connected to each other. Our synthesizer will include all these basic modules and will have the following structure:

- A 2D image generator that produces pictures made from geometric shapes, (Chapter 2, Creating Your First openFrameworks Project)
- A graphical user interface (GUI) consisting of sliders and checkboxes (Chapter 3, Adding a GUI and Handling Keyboard Events)
- A player of image files, video files, and live videos from a camera (Chapter 4, Working with Raster Graphics – Images, Videos, and Shaders)
- A video mixer that mixes several pictures using additive blending and the kaleidoscope video effect (Chapter 4, Working with Raster Graphics – Images, Videos, and Shaders)
- A 3D surface generator that renders a textured and deformed sphere in 3D (Chapter 5, Creating 3D Graphics)
• Various sources for controlling the synthesizer’s parameters, such as LFO, sound analysis, text files (Chapter 6, Animating Parameters), networking, and an Arduino device (Chapter 7, Distributed and Physical Computing with Networking and Arduino)

This structure is shown in the following diagram:

Additionally, in Chapter 8, Deploying the Project on iOS, Android, and Raspberry Pi, we will see how to deploy the light version of the synthesizer on mobile devices: iOS, Android, and Raspberry Pi. In the last chapter, Chapter 9, Further Resources, we will consider ways to further enhance the synthesizer, get some advice on debugging and optimizing your future openFrameworks projects, and consider additional references on openFrameworks.

The synthesizer created will have quite a lot of GUI controls and control keys. If while working with the book you need a quick reference on some of them, please see Appendix A, Video Synthesizer Reference, where we have collected descriptions and references to all parts of the synthesizer.

For a short description of openFrameworks functions and classes discussed throughout the book, see Appendix B, openFrameworks Quick Reference.
Three reasons to create your own video synthesizer with openFrameworks

Why would you program a video synthesizer with openFrameworks? The reasons are the following:

• You will get a handy, standalone application that provides you with interactive video wherever you need it (on a stage at your live performance or in your mobile), running on any platforms, and controlled by other programs or devices, such as mobiles or Arduino.

• Our video synthesizer is exceptionally customizable. By changing just several lines of code, you will get a unique synthesizer. The more you change the code, the more you find that there are no limits to expressing your creative ideas.

• By splitting the video synthesizer project into parts and then rearranging and adding new parts, you will obtain something different from the video synthesizer. For example, you can construct an interactive installation that transforms users' faces into abstract images. It could even be a mobile 3D drawing application that generates fantastic 3D landscapes. It's worth noting that you can easily incorporate LeapMotion, Kinect, and many other gadgets into your openFrameworks project in order to realize the interactive experience that you want.

Summary

In this chapter, you learned what openFrameworks is, how to install it, and also how to build and run its examples.

In the next chapter, we will create our own openFrameworks project that draws simple 2D graphics.
Where to buy this book

You can buy openFrameworks Essentials from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.