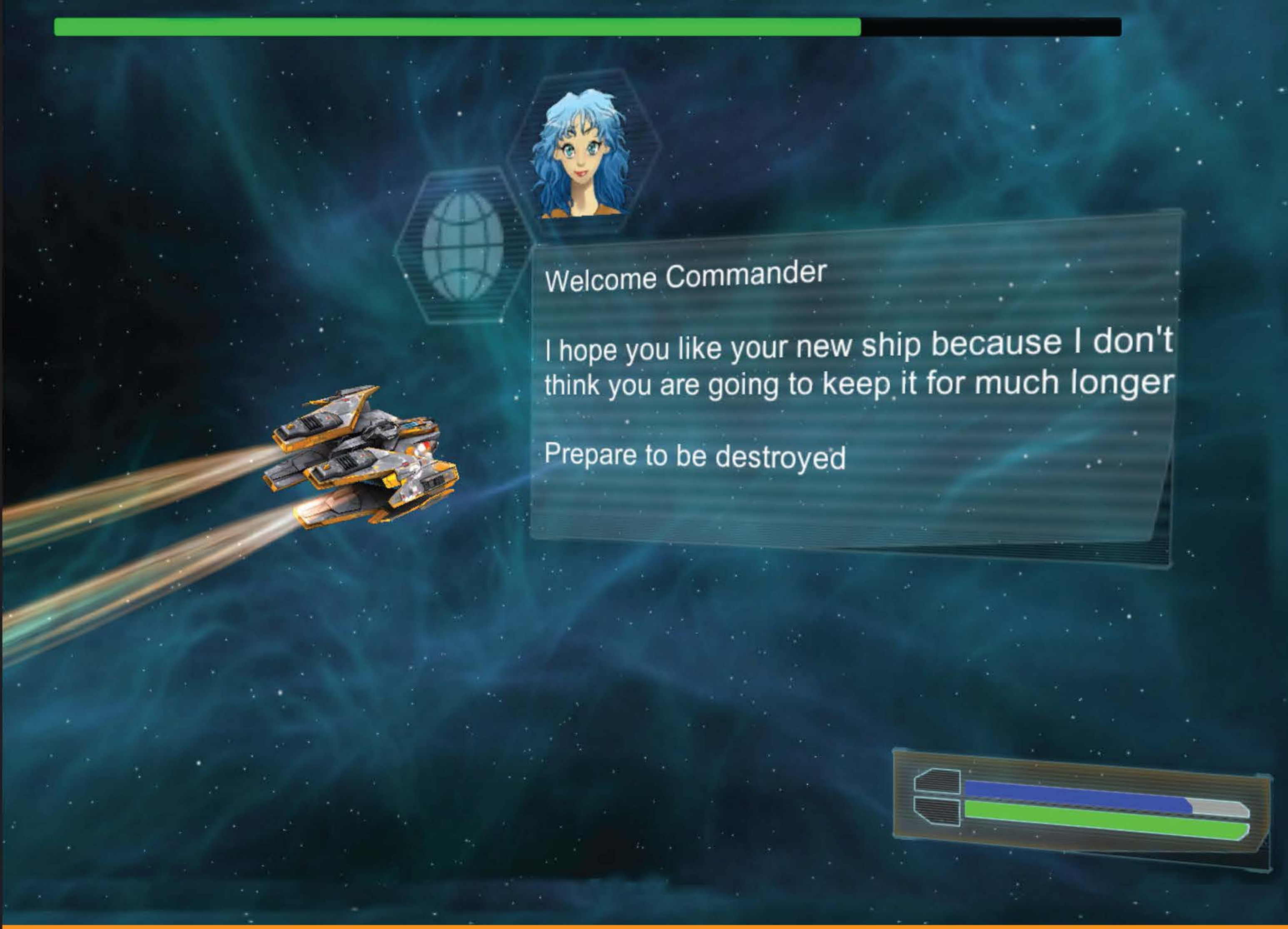
Welcome Commander
I hope you like your new ship because I don't think you are going to keep it for much longer
Prepare to be destroyed

Community Experience Distilled

Unity 3D UI Essentials

Leverage the power of the new and improved UI system for Unity to enhance your games and apps

Simon Jackson

[PACKT] PUBLISHING open source* community experience distilled

Unity 3D UI Essentials

Leverage the power of the new and improved UI system for Unity to enhance your games and apps

Simon Jackson

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Unity 3D UI Essentials

First published: January 2015

Production reference: 1270115

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN - 978-1-78355-361-7

www.packtpub.com

Credits

Author
Simon Jackson

Reviewers
Attilio Carotenuto
Adam Dawes
Javier García-Lajara Herrero
Dr. Sebastian T. Koenig
Simon Wheatley

Commissioning Editor
Akram Hussain

Acquisition Editor
James Jones

Content Development Editor
Sumeet Sawant

Technical Editor
Utkarsha S. Kadam

Copy Editors
Gladson Monteiro
Merilyn Pereira

Project Coordinator
Danuta Jones

Proofreaders
Simran Bhogal
Ameesha Green
Paul Hindle

Indexer
Priya Subramani

Graphics
Sheetal Aute

Production Coordinator
Nitesh Thakur

Cover Work
Nitesh Thakur

About the Author

Ever since my early years I have been a tinkerer, engineer, problem solver, and solution gatherer. In short, I love to break things apart, figure out how they work, and then put them back together, usually better than before.

I started way back when with my first computer, the Commodore Vic20. It was simple, used a tape deck, and forced you to write programs in basic or assembly; they were fun times. From there, I progressed through the ZX Spectrum +2 and the joyous days of modern graphics, but with the 30 minute load times from a trusty tape deck. Games were a passion of mine even then, which led to many requests for another gaming machine, but Santa brought me an Amstrad 1640, my first PC. From there, my tinkering and building exploded, and that machine ended up being a huge monstrosity with so many addons, tweaks, and fixes; I was Frankenstein, and this PC became my own personal monster crafted from so many parts. Good times.

This passion has led me down many paths, and I learned to help educate others on the tips and tricks I learned along the way; these skills have equipped me well for the future.

Today I would class myself as a game development generalist. I work with many different frameworks, each time digging down, ripping them apart, and then showing whoever would listen through my blog, videos, and speaking events how to build awesome frameworks and titles. This has been throughout many generations of C++, MDX, XNA (what a breath of fresh air that was), MonoGame, Unity 3D, The Sunburn Gaming Engine, HTML, and a bunch of other proprietary frameworks; I do them all. This gives a very balanced view of how to build and manage many different types of multiplatform titles.

I don't stop there, as I regularly contribute to the MonoGame project, adding new features and new samples before publishing it on NuGet. I also have several of my own open source projects and actively seek out any new and interesting ones to help with.

By day I am a lowly lead technical architect working in the healthcare software industry seeking to improve patients' health and care through better software (a challenge to be sure), but by night I truly soar! Building, tinkering, and educating whilst trying to push game titles of my own. One day they will pay the bills, but until then I still lead a double life.

More recently, I achieved the highly acclaimed reward of being a Microsoft MVP in the ID@Xbox program, for my evangelizing efforts in the game development space. This won't change much about me, but will give me additional tools to help game developers out there.

Lastly, you should check out my previous title, which has been one of Packt's best sellers since its release, especially if you want to learn more about Unity's 2D system. Check it out here: `http://bit.ly/MasteringUnity2DGameDevelopment`.

It truly has been a tormentous year but it looks to be getting better. Through it all, my wife Caroline has been my tiller, keeping me straight and true while I tend to my game development and writing efforts. Looking forward, I'll likely be crafting more of my own experiences with my kids pitching in; Caitlin, new to game development in school, being my sidekick, Jessica adding her colorful artistic talent to the mix, and the boys (Alexander and Nathan) no doubt trying to destroy my efforts through testing.

Additionally, a big thanks to my extended family (Mike and Marilyn) for helping out with the kids and keeping the writing area a kids-free zone for a few desperate hours. (It's amazing what a few hours respite can do.)

Also a big shout out to the PWSA (Prader-Willi Syndrome Association — `http://pwsa.co.uk`) for their help and support with Alexander, plus the Warrington Youth Club (`http://www.warringtonyouthclub.co.uk/`) for their exciting events to keep him entertained, especially in his more trying times. On that last thread, a very big thank you to Westland Drive respite (supported by Warrington council), who give us peace of mind and a night off from time to time; Alexander certainly loves his visits there.

Finally, thanks to the reviewers of this title (especially Simon W and Andrew D who joined me from my previous book); they kept me grounded and on target, although didn't help keeping the page count down. Thanks for your support guys!

About the Reviewers

Attilio Carotenuto is a senior game developer at Space Ape Games, based in London. There, he uses Unity to create awesome mobile and tablet strategy games such as *Samurai Siege*.

Attilio previously worked at King, developing *Farm Heroes Saga*, and EA Playfish, creating social games and tools based on the *The Sims* brand. Before that, he was a freelance game and web developer, using tools such as Unity, Cocos2D, Flash, and XNA.

He has previously worked with Packt Publishing on *Unity 3D Game Development* (a video tutorial) as a technical reviewer.

Recent projects, tutorials, and articles from Attilio can be found on his personal website, `www.attiliocarotenuto.com`.

Adam Dawes is a software developer and systems architect working at a cutting-edge online service development company.

He has long maintained a fondness for computer games. From the very first time Nightmare Park displayed its devious maze of pathways in green symbols back in 1980, he has been a player across a variety of genres and styles. Creating his own games has always been a hobby, and while he has no plans to become part of the professional games industry, Adam has a lot of fun developing his own titles nonetheless.

Over the last few years, Adam has also published three books of his own, the most recent of which, Windows 8 and Windows Phone 8 Game Development (published by Apress), explains everything you need to know to develop Windows 8 games in C# using the open source MonoGame platform.

More information is available from his website at `www.adamdawes.com`.

Javier García-Lajara Herrero was part of the booming video game industry in Spain, participating in the *Commandos* saga of Pyro Studios, where he developed as an artist, before continuing his career in different studies at Virtual Toys, Bitoon, and Unusual Studios.

He is now one of the professors at U-Tad University of Technology.

Always passionate about technical advances, he now researches and develops new proposals in games, virtual reality, and aerial photogrammetry of objects and environments with drones.

Dr. Sebastian T. Koenig received his PhD in human interface technology from the University of Canterbury, New Zealand, developing a framework for individualized virtual reality cognitive rehabilitation. He obtained his diploma in psychology from the University of Regensburg, Germany, in the areas of clinical neuropsychology and virtual reality rehabilitation.

Dr. Koenig is the founder and CEO of Katana Simulations, where he oversees the design, development, and evaluation of cognitive assessment and training simulations. His professional experience spans over ten years of clinical work in cognitive rehabilitation and over seven years of virtual reality research, development, and user testing. Dr. Koenig has extensive experience as a speaker at international conferences and as a reviewer of scientific publications in the areas of rehabilitation, cognitive psychology, neuropsychology, software engineering, game development, games user research, and virtual reality.

Dr. Koenig has developed numerous software applications for cognitive assessment and training. For his work on the Virtual Memory Task, he was awarded the prestigious Laval Virtual Award in 2011, for the Medicine and Health category. Other applications include the virtual reality executive function assessment in collaboration with the Kessler Foundation, NJ, USA, and the patent-pending Microsoft Kinect-based motor and cognitive training JewelMine/Mystic Isle at the USC Institute for Creative Technologies, CA, USA.

Dr. Koenig maintains the website `www.virtualgamelab.com` about his research and his software development projects. His website also contains a comprehensive list of tutorials for the game engine Unity.

Simon Wheatley first got into programming with the Sinclair ZX81 and then the Acorn BBC Micro. This hobby led onto a bachelor's degree in information technology, after which he embarked on an IT career working in the service, manufacturing, and higher education sectors.

Recently, he discovered Unity's new 2D tools and set about enthusiastically learning as much as possible about them while contributing plenty of errata to several recently published Unity books. Currently, he is developing an indie mobile game using Unity. When he isn't working, he can be found singing down at his local karaoke bar or out enjoying the fantastic British countryside!

Table of Contents

Preface

A new era has dawned, and Unity Technologies have taken a big, bold step. Not only have they delivered on some big promises for an all new and improved UI system for Unity projects, but they have also made the source for the new UI completely open source, giving everyday developers access to the inner workings of the new UI.

These are bold steps indeed. Many felt that the new UI wouldn't live up to the dream that was sold, as it had been years since they announced it was coming. Delays and rewrites made it look like it was never going to happen, leaving developers with either having to live with the existing legacy GUI or pay for some of the more advanced GUI systems on the asset store (such as NGUI).

Now, after a long and highly deliberated beta program, the new UI system is finally upon us. In some areas, it meets our expectations; in some, it falls a bit short (however, this is only the beginning). In other areas however, it has gone far beyond.

Throughout this title, we will peel back the layers of all this new technology to understand what each component does, how it fits together, and how to use it to build a fantastic new UI in our projects. Each chapter builds upon the last, to arm you (the reader) with all the knowledge required to assemble your UI within your projects. You will not just build on screen menus and options, but to embed UI elements within your 3D game world.

Not only have Unity released the new UI system, they have also given every developer access to the source that builds the UI, allowing you to better understand how things are built and enable you to extend the existing controls or even build your own. If you are feeling adventurous, you can even submit fixes or new features back to Unity for them to include within Unity itself.

Finally, we can now build what we want, how we want and best of all, it's completely *free* and available with the Free license for Unity. All hail and rejoice!

Now what are you waiting for? Pack up your towel, brew a freshly hot cup of tea, crack open this guide, and start exploring the all new universe of UI.

What this book covers

Chapter 1, *Looking Back, Looking Forward*, is a retrospective look at what Unity3D had to offer prior to 4.6 and an overview of what 4.6 and beyond brings to the table, including a high-level overview of all the new UI features.

Chapter 2, *Building Layouts*, covers the core elements of the new Unity UI system, the Canvas and Rect Transforms. These elements are the foundations of the new Unity UI system.

Chapter 3, *Control, Control, You Must Learn Control*, Unity UI introduces a heap-load of new UI controls to suit just about any UI need, from buttons and checkboxes to entire scrollable areas and layout masks. Here, we will delve deep into how to make the most of all the controls available.

Chapter 4, *Anchors Away,* provides a detailed walk-through of how to make the most of the new Unity UI anchor system and build responsive layouts/designs.

Chapter 5, *Screen Space, World Space, and the Camera*, Here we finally delve into one of the most highly anticipated parts of the new UI system: the ability to easily build perspective UI layouts and add UI elements as 3D objects within a scene.

Chapter 6, *Working with the UI Source*, looks at all the coding behind the UI framework and explores the new Event System and UnityEvent frameworks. The chapter finishes with a walk-through, the open source project for the UI system, allowing you to see just about every line of code Unity has written for the new UI.

Appendix, *The 3D Scene Sample*, talks about a flashy 3D demo scene, which was discussed in *Chapter 5*, *Screen Space, World Space, and the Camera*, to show off the UI. Because this wasn't the focus of the book, it was added as an optional appendix that you could follow if you wish. The instructions are also available online and as a downloadable package to enable developers of all levels to make use of it.

What you need for this book

- Unity3D V4.6+
- Visual Studio 2012 (Express, Pro, or higher); optional but recommended

Who this book is for

This book is for anyone with a solid understanding of Unity's core functionality and a decent grasp of C# scripting in Unity (although not required for just the core editor portions of the new Unity UI system). With this book, you'll be well placed to take advantage of the new UI feature set.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "For standards stake, you should add scripts into a folder called `Scripts` and scenes into a folder called `Scenes`."

A block of code is set as follows:

```
void OnGUI() {
  GUI.Label(new Rect(25, 15, 100, 30), "Label");
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
public Texture2D myTexture;
void Start() {
  myTexture = new Texture2D(125, 15);
}
void OnGUI() {
  GUI.DrawTexture(new Rect(325, 15, 100, 15), myTexture,
    ScaleMode.ScaleToFit,true,0.5f);
}
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "With the new **Unity UI** system, you can define several layout groups."

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at `http://www.packtpub.com` for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Additionally, the author has provided a support forum for the book. This forum provides direct support from the author on your queries and any forthcoming announcements regarding the title. You can find this forum at `http://bit.ly/Unity3DUIEssentialsForums`.

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from: `https://www.packtpub.com/sites/default/files/downloads/3617OS.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

Additionally you can post questions directly to the author about the content of the title on the book's support forum at `http://bit.ly/Unity3DUIEssentialsForums`.

1
Looking Back, Looking Forward

The new Unity UI has long been sought by developers; it has been announced and re-announced over several years, and now it is finally here. The new UI system is truly awesome and (more importantly for a lot of developers on a shoestring budget) it's free.

As we start to look forward to the new UI system, it is very important to understand the legacy GUI system (which still exists for backwards compatibility) and all it has to offer, so you can fully understand just how powerful and useful the new system is. It's crucial to have this understanding, especially since most tutorials will still speak of the legacy GUI system (so you know what on earth they are talking about).

With an understanding of the legacy system, you will then peer over the diving board and walk through a 10,000-foot view of the new system, so you get a feel of what to expect from the rest of this book.

The following is the list of topics that will be covered in this chapter:

- A look back into what legacy Unity GUI is
- Tips, tricks, and an understanding of legacy GUI and what it has done for us
- A high level overview of the new UI features

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you. Additionally, the author has provided a support forum for the book. This forum provides direct support from the author on your queries and any forthcoming announcements regarding the title. You can find this forum at `http://bit.ly/Unity3DUIEssentialsForums`.

State of play

You may not expect it, but the legacy Unity GUI has evolved over time, adding new features and improving performance. However, because it has kept evolving based on the its original implementation, it has been hampered with many constraints and the ever pressing need to remain backwards compatible (just look at Windows, which even today has to cater for programs written in BASIC (`http://en.wikipedia.org/wiki/BASIC`)). Not to say the old system is bad, it's just not as evolved as some of the newer features being added to the Unity 4.x and Unity 5.x series, which are based on newer and more enhanced designs, and more importantly, a new core.

The main drawback of the legacy GUI system is that it is only drawn in screen space (drawn on the screen instead of within it) on top of any 3D elements or drawing in your scenes. This is fine if you want menus or overlays in your title but if you want to integrate it further within your 3D scene, then it is a lot more difficult.

For more information about world space and screen space, see this Unity Answers article (`http://answers.unity3d.com/questions/256817/about-world-space-and-local-space.html`).

So before we can understand how good the new system is, we first need to get to grips with where we are coming from. (If you are already familiar with the legacy GUI system, feel free to skip over this section.)

A point of reference

Throughout this book, we will refer to the **legacy GUI** simply as **GUI**.

When we talk about the new system, it will be referred to as **UI** or **Unity UI**, just so you don't get mixed-up when reading.

When looking around the Web (or even in the Unity support forums), you may hear about or see references to **uGUI**, which was the development codename for the new **Unity UI** system.

GUI controls

The legacy GUI controls provide basic and stylized controls for use in your titles.

All legacy **GUI** controls are drawn during the GUI rendering phase from the built-in `OnGUI` method. In the sample that accompanies this title, there are examples of all the controls in the `Assets\BasicGUI.cs` script.

For GUI controls to function, a camera in the scene must have the GUILayer component attached to it. It is there by default on any **Camera** in a scene, so for most of the time you won't notice it. However, if you have removed it, then you will have to add it back for GUI to work.

The component is just the hook for the `OnGUI` delegate handler, to ensure it has called each frame.

Like the `Update` method in scripts, the `OnGUI` method can be called several times per frame if rendering is slowing things down. Keep this in mind when building your own legacy GUI scripts.

The controls that are available in the legacy GUI are:

- **Label**
- **Texture**
- **Button**
- **Text fields (single/multiline and password variant)**
- **Box**
- **Toolbars**
- **Sliders**
- **ScrollView**
- **Window**

So let's go through them in more detail:

All the following code is implemented in the sample project in the basic GUI script located in the `Assets\Scripts` folder of the downloadable code.

To experiment yourself, create a new project, scene, and script, placing the code for each control in the script and attach the script to the camera (by dragging it from the project view on to the **Main Camera** GameObject in the scene hierarchy). You can then either run the project or adorn the class in the script with the `[ExecuteInEditMode]` attribute to see it in the game view.

The Label control

Most GUI systems start with a **Label** control; this simply provides a stylized control to display read-only text on the screen, it is initiated by including the following `OnGUI` method in your script:

```
void OnGUI() {
  GUI.Label(new Rect(25, 15, 100, 30), "Label");
}
```

This results in the following on-screen display:

The **Label** control supports altering its font settings through the use of the `guiText` GameObject property (`guiText.font`) or `GUIStyles`. (See the following section on *GUIStyles* for more detail.)

To set `guiText.font` in your script, you would simply apply the following in your script, either in the `Awake/Start` functions or before drawing the next section of text you want drawn in another font:

```
public Font myFont = new Font("arial");
guiText.font = myFont;
```

You can also set the `myFont` property in **Inspector** using an imported font.

The **Label** control forms the basis for all controls to display text, and as such, all other controls inherit from it and have the same behaviors for styling the displayed text.

The **Label** control also supports using a Texture for its contents, but not both text and a texture at the same time. However, you can layer Labels and other controls on top of each other to achieve the same effect (controls are drawn implicitly in the order they are called), for example:

```
public Texture2D myTexture;
void Start() {
  myTexture = new Texture2D(125, 15);
}
void OnGUI() {
  //Draw a texture
  GUI.Label(new Rect(125, 15, 100, 30), myTexture);
  //Draw some text on top of the texture using a label
  GUI.Label(new Rect(125, 15, 100, 30), "Text overlay");
}
```

You can override the order in which controls are drawn by setting `GUI.depth = /*<depth number>*/;` in between calls; however, I would advise against this unless you have a desperate need.

The texture will then be drawn to fit the dimensions of the **Label** field, By default it scales on the shortest dimension appropriately. This too can be altered using `GUIStyle` to alter the fixed width and height or even its stretch characteristics.

`GUIStyles` and `GUISkins` are explained in the later *GUI styles and skins* section.

Texture drawing

Not specifically a control in itself, the GUI framework also gives you the ability to simply draw a Texture to the screen Granted there is little difference to using `DrawTexture` function instead of a Label with a texture or any other control. (Just another facet of the evolution of the legacy GUI). This is, in effect, the same as the previous `Label` control but instead of text it only draws a texture, for example:

```
public Texture2D myTexture;
void Start() {
```