As there is a manual for Flash users (https://defold.com/manuals/flash/), so as well, there might be manuals for users of other game engines/frameworks, but this demands a good knowledge and understanding of the given engine. If you know anything about them and would like to contribute to such a manual, please, feel free to write something down here.

Unity

This guide presents Defold as an alternative for Unity game developers. It covers some of the key concepts used in Unity game development, and explains the corresponding tools and methods in Defold.

Introduction

Some of the key advantages of Unity are the vast amount of features, tutorials and a lot of configuration options. New users can learn Unity quickly and are enabled to create simple or advanced games in 2D or 3D. Defold is definitely not comparable to Unity in terms of popularity, age, size and amount of community support, but in its focus area can offer seamless, reliable and comfortable game development workflow by providing a suite of tools dedicated to game design, while empowering advanced developers to create solutions for more sophisticated requirements (for instance by allowing developers to edit the default render script). At the core Defold is also a 3D engine, yet it shines in 2D, cross-platform game development and engine size and performance.

Scripting languages

Unity games are programmed primarily in C# through Unity Scripting API, while Defold scripting is done in Lua with Defold API. We will not go into a detailed comparison of Lua and C#. The Defold manual provides a good introduction to Lua programming in Defold, and references the tremendously useful Programming in Lua (first edition) which is freely available online.

Unity also has community support for other programming languages, but all official support (like UnityScript (Javascript-like), Boo(Python-like)) are deprecated. There is also a possibility for visual programming in Unity using Unity Visual Scripting (earlier known as Bolt). Defold has unofficial community support for other languages like TypeScript, Haxe and C#. Defold has no visual programming option.

It is a common opinion that Lua is faster to learn and easier to use, while C# may offer more features and scalability. Both languages are widely adapted and used in game development. Lua is very different from C# in areas like types, simplicity, inheritance and paradigms used, etc., but this guide does not focus further on differences between them.

Defold does not have classes, nor inheritance. It includes the concept of a *game object* which can contain audiovisual representation, behavior and data. Operations on game objects are done with *functions* available in the Defold APIs. Furthermore, Defold encourages the use of *messages* to communicate between objects. Messages are a higher level construct than method calls and are not intended to be used as such. These differences are important and take a while to get used to, but will not be covered in detail in this guide.

This guide explores some of the key concepts of game development in Unity and outlines what the closest Defold equivalents are. Similarities and differences are discussed, along with common pitfalls, to enable you to get off to a running start in transitioning from Unity to Defold.

Game Objects

Let's start with similarities. Both engines operate on a concept of a Game Object, being the most fundamental building block. In Unity, Game Objects always have a name and Transform attached that can not be removed. The same in Defold - Game Objects have a unique id and a transform (position, rotation, scale). Transform is in the case of Unity also called a component. Transforms are the exact equivalents in each of the engines. The same with parent-child relationships between game objects.

Components

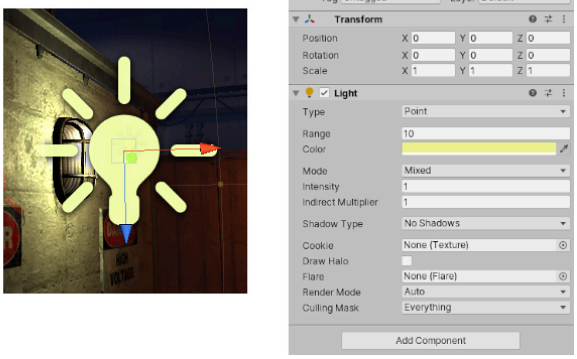
Both engines enhance the Game Objects with Components. Components modifies and enhances game objects behavior and functionalities. Each Game Object in both engines can have none, one or many different components.

Unity has a lot of different purpose components and Defold too, but there are some differences. Direct comparison of components might not be very descriptive, because many things Unity components might offer out of the box could be accomplished in Defold in different ways, like for example with Unity's Light component. Lighting is handled in Defold by using a different rendering pipeline and there are some example assets for making lighting.

On the other hand, there are components in Defold that are missing for now in Unity like the extension for Rive runtime for Defold.

Unity components are sometimes very enhanced and bloated with a lot of configuration options covering a lot of functionalities by default, while Defold components in comparison are barebone and focused on fundamental settings.

Unity allows you to make your own components via Unity Scripting API, where scripts become components when saved to the project and Defold offers the same possibility with Script components. Moreover, Defold's source code is available online, so you can modify the engine to suit your needs, for example adding or modifying components and releasing the game with that modified version.



Scenes - Collections

Unity uses a concept of a Scene to describe the game's world - with all its game objects and their components. Defold uses the analogical abstraction called Collection. Collection in Defold is a file describing included Game Objects with their components and the hierarchy between them. Collections in Defold can also contain other collections within.

Defold starts the game with loading all Game Objects and their hierarchies from a collection specified as Bootstrap collection in a single (and only) game settings file called game.project. This is by default the "main.collection" and it can include other collections. This might be useful for loading other collections, loading levels, cutscenes, introducing menu screens, mini games or any separate abstraction you would like to have in your game.

It is worth noting that Defold's collection does not exist in runtime - it is a file for instantiating the game objects and the hierarchy in the runtime. All Defold files within your project are text based, so are easily traceable in a version control system.

For the details on described above Defold building blocks (collections, game objects and components), please refer to: https://defold.com/manuals/building-blocks/

Prefabs - Factories

Unity comes with an abstraction of Prefabs - templates/prototypes of Game Objects, that can be included in a Scene or instantiated in runtime.

Defold also offers such a possibility - you can make a prototype of a Game Object as a file (it will be in the Assets pane) and when you include it in any Collection it will be instantiated there with all its components. Moreover you can create the Game Object from a file in a script - using a component called Factory. The Factory component is here to specify which Game Object prototype you are to instantiate using this factory.

Moreover, you can instantly instantiate Collections using the Collection Factory component, the same as the Factory component.

Unity offers overwriting instances of Prefabs, and the same in Defold - you can modify components and properties of the game objects referenced to its file prototypes in the collections - the modified values will be highlighted in blue.

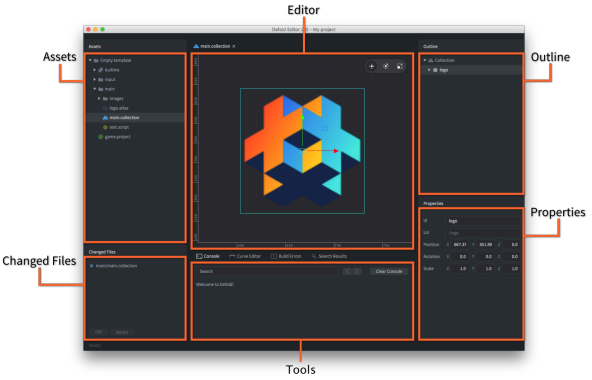
Unity offers to deactivate Game objects in Editor. Defold is yet to get such a feature in the future.

Tags - collision groups

Unity offers an abstraction of Tags. This is to group objects, yet mainly used for Colliders, to e.g. specify collisions between player and enemies or collectables. In Defold, when we come to collisions, there are also Groups for each collision object and Mask property - specifying with which groups this object should collide.

Editor

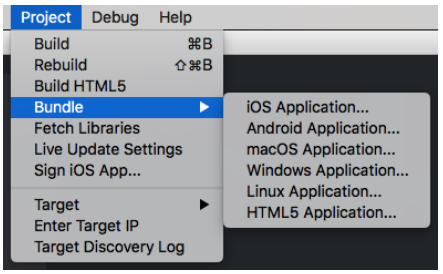
Both Defold and Unity come with the Editors. They differ though. Unity offers more flexibility for the Editor and windows can be rearranged and modified. Defold focuses on minimalism and the given panes are always in the same place, but you can modify the visibility and size of each. You can also split the editor view in half.



For the details on Editor overview, please refer to: https://defold.com/manuals/editor/

Builds and bundling

The game is run inside Unity Editor, while Defold opens the build in a separate window as a separate process using the Defold engine. You can build your project with a shortcut (Ctrl/Cmd+B) or from the top menu Project->Build. You can also bundle your project for one of the supported platforms: Windows, Linux, MacOS, Facebook Instant Games, Android, iOS, Nintendo Switch. For Apple and Nintendo products and especially for releasing on them, you need some additional setup. For the details on bundling, please refer to: https://defold.com/manuals/bundling/



Mobile development

Unity supports mobile game development with a lot of its features, like Cloud Builds or Unity Collaborate and you can build your projects for mobile devices for testing.
Defold is also making it easier to develop on a live mobile device, because you can push your project's content to a physical device over Wi-Fi using the mobile development app, which together with hot reloading speeds up the mobile development. For more on mobile app development, please refer to: https://defold.com/manuals/dev-app/.

Monetization

Unity offers a lot of monetization, analytics tools and in-app purchases.
Defold also has extensions for monetization (including Unity Ads: https://github.com/AGulev/DefVideoAds), analytics tool and official support for in-app purchases for stores of Apple, Amazon, Facebook and Google. For more on IAPs, please refer to: https://defold.com/extension-iap/

Additionally, Defold offers official support for Web Monetization for web builds (there is an unofficial repository with WM support for Unity's WebGL). For more on Web Monetization in Defold, please refer to: https://defold.com/extension-webmonetization/

Hot reloading

Defold allows and encourages hot reloading of your projects, especially for mobile platform development, where you can run your project and modify it live on the mobile device. Unity has a possibility to recompile scripts, but there are a lot of issues with this, like unexpected calls of onEnable/onDisable. The best is also to use only singletons. There are some assets that make it easier, but you still need to be intentional with what you are reloading.

Hot reloading in Defold is way more convenient. Defold reloads not only Lua scripts, but all resources, so you can modify for example animations in runtime. For hot reloading Lua modules (kind of libraries for Lua scripts) and for more on hot reloading in general, please refer to: https://defold.com/manuals/hot-reload/

Unity PlayerPrefs

Unity uses a PlayerPrefs class to allow storing Preferences between game sessions. If you want to update your game with a Defold game, but want to preserve previously stored progress for current players, there is a native extension for Defold to import Unity PlayerPrefs files as Lua tables: https://github.com/AGulev/ppreader

Rendering pipelines

Defold uses OpenGL Shading Language (GLSL) for shader scripting and offers some built-in shaders and rendering pipeline. Unity uses GLSL too and additionally offers Microsoft's HLSL, declarative ShaderLab and Shader Graphs. Defold focuses on making the solid barebone easily adaptable for any game, while Unity is focused on high definition 3D pipelines with AAA graphics.

—

Solar2D (Corona)

This guide presents Defold as an alternative for Solar2D game developers. It covers some of the key concepts used in Solar2D game development, and explains the corresponding tools and methods in Defold.

Unreal

This guide presents Defold as an alternative for Unreal game developers. It covers some of the key concepts used in Unreal game development, and explains the corresponding tools and methods in Defold.

Godot

This guide presents Defold as an alternative for Godot game developers. It covers some of the key concepts used in Godot game development, and explains the corresponding tools and methods in Defold.

Löve

This guide presents Defold as an alternative for Löve game developers. It covers some of the key concepts used in Löve game development, and explains the corresponding tools and methods in Defold

Phaser

This guide presents Defold as an alternative for Phaser game developers. It covers some of the key concepts used in Phaser game development, and explains the corresponding tools and methods in Defold