How to get started developing your first VR/AR app

Lou Pushelberg Follow
Nov 21, 2018 · 6 min read

Before you go from idea to prototype, you need to figure out what that idea is, the platform you'll build it on, and which device your app is for. That's a lot. But I'll breakdown those options and show you how to get from idea to prototype.

First I'll give a quick overview of Unity and the programming language C#, then dive into figuring out what to build, and how to get your project rolling.

What is Unity?

Unity has been on the market since 2005, and in 13 years it has grown to become one of the most stable and powerful game engines available. It supports the creation of 3D and 2D games, applications, and simulations.

Unity supports a variety of software development kits (SDKs) and integrations for VR and AR development, including: Oculus, HTC Vive, Microsoft Hololens, Magic Leap, Vuforia, ARKit (iOS) and ARCore (Android), Windows Mixed Reality, among others.

All of Unity's capabilities come from its use of C# for programming. Like any language, C# can be used to create almost anything, but it really shines when building desktop, mobile and VR/AR apps. It's widely used in game development and VR (over 50% of mobile games and 60% of VR/AR apps are made with Unity), and over 90% of VR/AR development companies rely on C# and Unity. There are online resources that can teach you C#, from Udemy to Youtube and specialized courses, so don't worry if it seems unfamiliar to you now.

Unity vs Unreal

There's two leading game engine options on the market for a prospective VR/AR developer. Unreal and Unity are both capable engines, but I prefer Unity.

Unity has a lower point of entry for new developers: its intuitive design, large developer community and resources mean you'll have an easier time programming on your own. The expansive selection in Unity's asset store provides a lot of variety and options, including being able to learn and grow into the program by adding and building upon more complex assets. Though Unreal has long been a leader in prioritizing high-resolution graphics, Unity is closing that gap quickly.

I've already preached my love of C# for programming in Unity. Unreal relies on C++, which is a fine language, but it doesn't have as logical or consistent syntax as C#. C# is statically-typed, meaning the code is checked before being 'turned on,' making mistakes easier to identify, learn from, and correct. Stack Overflow found 60% of developers love working with C#, as opposed to 46% loving C++ (53% of developers dread using C++).

How do you start building?

With Unity downloaded and your device set up, you can start developing your project. 81% of professional developers started programming as a hobby and turned that passion into a career. Learning on your own is doable, but difficult. If you run into a creative block or wall, Unity, online forums, and VR/AR meetups are great places to find solutions.

While half of all professional developers have at least a bachelor's degree—most in computer science or software engineering—48% expanded on that education through online courses. You'll get the knowledge, but what you do with it is up to you. And that can be a very daunting proposition.

Before you start building, you want to ask yourself: what exactly is the project? Who is your target audience? How interactive will it be? How accessible?

Developing for AR

Let's posit you want to build an AR app to simulate home renovations before they happen. (Karndean Design Flooring has an app like this if you want to see this idea in action). Your target audience would be home designers, renovation companies, and more. The app would need to be fairly accessible for homeowners, so you'd develop for ARkit for iOS or ARCore for Android, rather than an AR headset like Microsoft Hololens or Magic Leap. And, you'd need to build for a lot of interactivity, depending on the extent of the renovations. Now that you've narrowed the project's scope to a mobile AR app, you'll know Unity is an ideal engine to start using.

Or, let's suppose you were designing an instructional AR app for use during manufacturing and repairs. Your users would be the workers on the manufacturing floor, and they'd need a device that's mobile, small, and convenient. A phone would work great for this purpose. The user could hold their phone up to the machinery to be repaired, receive instructions, and input data into the device.

Vuforia makes a great platform for this purpose thanks to its image-tracking capabilities and accessibility for users. Workers could download the app onto their phone with no expertise needed. Vuforia would track the specific machine part in view, and instructions would be given on screen in real-time. Circuit Stream has how-to instructions for setting up Vuforia and enabling AR image-tracking; I can confidently say Vuforia is a great platform for beginners and experts in AR development.

Developing for VR

There are many options for a VR app, and they all exist on a spectrum of interactivity. On one end, you have 360° videos or VR films that place the user right into the scene but are a passive viewer. On the other end, you have training simulations where the user must interact in order to continue with the program, *and* must learn from the simulator to perfect the training. In the middle sits apps along the lines of Beat Saber where the user only reacts to what is presented to them.

Passive User
Reactive User
Active User

Range of Interactivity for VR/AR Users

Let's say your project is a VR app that teaches users to identify dinosaur fossils at a virtual archaeological dig site. Your target audience would be institutions like universities or museums. The level of interactivity would be fairly minimal, but you'd want to allow for manipulation and analysis of objects. The app itself should be intuitive and accessible for new users unfamiliar with VR.

With this example, you'd build your VR app for the HTC Vive because its base stations allow for a large virtual space. Because users are analyzing fossils Unreal might be the better engine for its high-res graphics, but Unity is fully capable of handling these demands as well.

Circuit Stream's 7-step Vive tutorial would be quite helpful in this scenario. There's an outline how to get setup, design interactions with objects, and Vive's raycasting options. And, because users would be digging fossils out of the ground, their post on using occlusion to hide objects and create x-ray tools would also be of use here.

Getting from Idea to Prototype

You can learn C#, Unity, and building for VR and AR platforms on your own. But it's an uphill battle, especially if you're completely new to computer programming. Online tutorials and YouTube videos have their limitations in terms getting real-time answers to your questions.

Unity's online user manual is hefty but searchable, so it's a great resource for troubleshooting. There's also many online blogs with how-to and other technical posts that can help you get started.

Jenn Duong (@JennDefer on Twitter) has put together an amazing resource of VR/AR community resources that you can check out here. From meet-ups to blogs, podcasts and education programs, you'll want to bookmark her google doc and refer to it frequently.

Circuit Stream offers courses for VR/AR Development with Unity you can check out. The courses include one-on-one mentorship with instructors and project-based learning, so this hands-on education might be what you're looking for! In any case, I wish you luck on your VR/AR journey!