WWDC State of the Union

Company Participants

- Brandon Corey, Director, Camera Software
- Chris Fleizach, Senior Manager, Accessibility
- Chris Markiewicz, Senior Program Manager, App Store
- Darin Adler, Vice President, Internet Technologies and User Privacy
- Edwin Iskandar, Senior Engineering Manager, visionOS
- Enrica Casucci, Director, visionOS Apps
- Geoff Stahl, Senior Director, visionOS
- Holly Borla, Engineering Manager, Swift Language
- Jason Cahill, Senior Engineering Manager, visionOS Tools
- Jeff Norris, Senior Director, visionOS Apps
- Jonathan Thomassian, Senior Manager, System Experience
- Josh Shaffer, Senior Director, Swift Frameworks
- Katie Skinner, Senior Manager, User Privacy Software
- Ken Orr, Senior Manager, Xcode and Swift Playgrounds
- Linda Dong, Evangelist, Worldwide Developer Relations
- Lori Hylan-Cho, Senior Manager, watchOS System Experience
- Mike Rockwell, Vice President, Technology Development Group
- Ralph Hauwert, Senior Vice President
- Thessa Buscar-Alegria, Senior Engineering Program Manager, visionOS

Presentation

Darin Adler

Welcome to Platform State of the Union 2023. We're excited to share more about how you can take advantage of what's coming in our new releases. Like interactive widgets on iOS and iPadOS, which give you more ways to surface your apps across the system, a new widget experience on macOS, the beautiful design of watchOS 10, Continuity Camera on tvOS, and of course, an entirely new platform with visionOS. Platform State of the Union is a chance to learn about some of the most exciting, impactful, and important developments from this year's releases. And what we have for you delivers on all those fronts.

Before we dive in, I want to talk for a moment about what makes a great platform and great apps. A great platform is more than a collection of frameworks and technologies. It's key ingredients that come together to make something truly special. It's the combination of a language, frameworks, tools, and services created to work seamlessly together. So, the most natural way to write your code is also the

best; a rich set of APIs to help you create amazing experiences that deeply integrate with the system, whether your app is open or not; thoughtful ways for you to safely take advantage of cutting-edge hardware technologies across a wide range of products; and finally, a great platform evolves to meet new challenges so you're able to meet the needs of users right when they emerge.

The more than 300 frameworks on our platforms are building blocks, enabling you to get right to what's innovative, creative, and different about your app. They go further than just saving you some time or helping you add a specific feature. When you use these platform foundations with purpose, your app fits in beautifully and extends the system itself, and that's the path to a great app.

A great app elevates the right information to the user at the right time in the right place. That place could be the Share Sheet, right on the lock screen, Spotlight, Siri, or any number of other places. A great app takes advantage of hardware like a new sensor or a powerful chip with help from frameworks that put those capabilities at your fingertips. And a great app does things that are so important to get right, like creating safe, accessible products that respect a user's privacy.

Our platforms and your apps come together to deliver all of this with the consistency and reliability that people value and expect, and this is a big deal. We're incredibly passionate about great platforms and great apps. Users can tell the difference they make. The shared foundations of iOS, iPadOS, macOS, tvOS, watchOS, and now visionOS form a cohesive ecosystem that allow you to deliver your apps across the full range of Apple platforms and products. Your apps are key to this ecosystem. They empower people as they go about their day, with easy-to-use experiences that move seamlessly from one device to another.

At Apple, we often talk about building the things we want to use ourselves, and it couldn't be more true than with our approach to our platforms, because we're developers, too. We use the same language tools and frameworks here at Apple.

So, today, we have a lot to cover. We'll start with the latest developments in Swift and SwiftUI and how you can use them across all of our platforms to create amazing experiences. We'll talk about how you can leverage hardware features and also how you can prioritize values in the experiences you build. And we'll walk you through the improvements to the powerful tools we all use every day. Finally, we'll take a deep dive into the technologies of our latest platform, visionOS, and show you how you can build the next generation of apps for Apple Vision Pro.

To kick things off, I'll turn it over to Holly to talk about what's new in Swift.

Holly Borla

The foundation for so many of the APIs that you use every day is Swift. It's fast, modern, and safe, guiding you toward code that's efficient, expressive, and correct. APIs allow you to take advantage of libraries and platform features and create experiences that are unique to your app. But some APIs can be hard to use,

requiring you to write a lot of boilerplate code just to get started. That's why Swift is unlocking a new kind of API that's easier to use and easier to get right with the introduction of macros, and these macros are done the Swift way.

A macro is an annotation that uses the structure of your code to generate new code that's built with your project. Macros can either be attached as attributes to your code, or they can be freestanding, spelled with the hash sign. Macros make APIs feel like they're part of the language, and there are so many ways to start using a new API with just an annotation. Macros come to life in Xcode, where the generated code feels like it's part of your project. Let me show you.

I created a URL macro that checks for valid URL strings. Because macros generate regular Swift code, I can use the expand macro feature in Xcode to see exactly what they do, this macro looks really simple. It calls the URL initializer and force unwraps the result, but the URL macro does more than that. It checks that the string is a valid URL at compile time. For example, URLs can't have spaces. So, if I add one, the macro provides a custom error message and a fix-it to automatically remove the space from the string. This moves what would have been an error when my app is running into feedback while I'm writing my code. So, I can fix mistakes right away. All macros have the ability to provide custom feedback to help you write correct code.

Attached macros go even further because they can add new functionality to code you already wrote. Let's take a look at what attached macros can do. I have a function called fetchContent that performs asynchronous work and uses completion handlers, but I want to use async/await. Instead of implementing another fetchContent function, I can attach the AddAsync macro, and that's it. Now, instead of passing in a completion handler, when I call fetchContent, I can simply await the result.

Just like any other API, I can use quick-help on the macro to see its documentation. But if I want to understand how this works under the hood, I can set a breakpoint, run my code, step into the async function, and Xcode will expand the macro right in my source editor.

This is really cool, because there are no secrets. Stepping into an expanded macro while debugging is a natural way to peek behind the scenes and help me understand what's happening in my code. You can write your own macros, so you can extend the language in ways that were previously only possible by implementing features in the Swift Compiler. Because Swift is developed in open source, developers are already using macros to build some of the most commonly requested language features, including automatically generating rich descriptions of assertion failures, providing customizable default protocol conformances, and the list continues to grow.

You'll also see macros used throughout many of this year's new APIs. You can benefit from community-authored macros or share your own macros with others through

Swift packages. Language advancements like macros enable Swift to be adopted in more projects than ever before.

Let's talk about all of your code that isn't written in Swift. Since its inception, Swift has provided bidirectional interoperability with C and Objective-C. You can introduce Swift into your projects a single file or module at a time, without having to rewrite entire codebases. Swift is extending that interoperability to C++. Swift C++ interoperability allows you to use both languages in the same project without an intermediate bridging layer. You can share your classes, functions, and even template specializations like vector across both languages by setting a compiler flag.

Using C++ from Swift will eliminate many sources of undefined behavior in your code, such as using variables before they're initialized and bridging is efficient. Calls are made natively between your Swift and C++ code with minimal overhead. Swift C++ interoperability is already in use in open source, helping to transition the Swift compiler itself from C++ to Swift. And I am so excited to use Swift to implement Swift. Swift's fundamental goals of safe and expressive code are ingrained in the libraries that are built with it, and nowhere is that more apparent than with SwiftUI.

Here's Josh to tell you more.

Josh Shaffer {BIO 20854625 <GO>}

The best way to build modern user interfaces across all Apple platforms is with SwiftUI. SwiftUI lets you write better apps with less code, so you can focus on what's unique to your app. It also maximizes your ability to reuse interface code.

The healthy habits app, Streaks, by Crunchy Bagel is a great example. They first started using SwiftUI when building widgets for iOS. Then they shared that code with their watchOS app and augmented it with beautiful animations. And now they've incorporated the same code back into their original iOS app. Incremental adoption enabled them to share UI code at their own pace and see benefits from SwiftUI in their existing apps.

SwiftUI is also incredibly capable, helping to power sophisticated interfaces in apps like the new Logic Pro and Final Cut Pro for iPad. Adopting SwiftUI in your apps prepares you to build the next generation of user interfaces. And what you can build with SwiftUI continues to expand. As always, its new capabilities in this year's releases focus on areas of high impact and top developer requests, like new support for Pie Charts and selection in Swift Charts, a brand new inspector API, expanded MapKit support, including Overlays, Look Around, and more.

These improvements cover a wide range of functionality, but many of them are focused on improving animations. This is a key area, because well-designed animations make your interface easy to use. They can provide feedback so your users know that something is happening, or they can confirm that a task has completed successfully. And that's why SwiftUI has had advanced features like reversible, interruptible, and cancelable animations from the beginning.

This year, SwiftUI is taking animations even further. It starts with the way that they move. Animations are often triggered by a user's gesture. So, now SwiftUI can automatically transfer the velocity of your gesture into your animation, providing a smooth transition. And animations now default to a spring-based motion that can be configured with just two simple parameters, duration, and bounce. SwiftUI can also take advantage of new animated effects in SF Symbols, bringing the iconography in your apps to life.

Finally, when you need to build multi-part animations, SwiftUI has a new API called Animation Phase that helps you build sophisticated animations with just a few lines of code. I've created a sample app to illustrate new APIs like AnimationPhase, inspired by the beautiful landscape of Apple Park. It provides a calming experience, where users can create outdoor spaces and provide food and drink for visiting birds. To make sure that I never miss a new bird, I've added a reminder to the top of the list. Let's switch over to a preview focused on just that view.

I really want to make this SF Symbol stand out by adding some animation. First, I'll add an enum to define the different states in a custom animation. Next, I'll wrap my view in a PhaseAnimator. This is a new view that automatically animates between a set of states. And finally, I'll scale and rotate the icon in the dot-highlighted animation phase.

This is looking good, but I think it would look even better if the background animated along with the icon. So I'll add a couple more modifiers.

Now, for even more advanced animations, this year SwiftUI adds full support for keyframing. This powerful API lets me animate anything, including the properties of the new SwiftUI-based MapKit API. Keyframes let you define the values of multiple properties at specific times within an animation, and then let SwiftUI interpolate the intermediate values. Here, I've used keyframes to animate the map camera visiting different landmarks in Cupertino. Because keyframes let you define the motion for different properties, we're able to independently animate the pitch, heading, and position of the camera to achieve this smooth, continuous motion across Cupertino and over to Apple Park. Crafting the feel of your app using animations in SwiftUI has never been easier.

Next, let's talk about data flow. When building an app with SwiftUI, you have a few choices to bring data into your views. SwiftUI offers a set of property wrappers for managing data local to your view and for referencing data owned elsewhere. This year, data flow is getting simpler, letting you focus on just state and environment. Now you may have written code like this when exposing your model to SwiftUI, conforming to observable object and adding the published property wrapper to each property.

Swift's new Macro support makes this much simpler with a new Observable macro taking its place. Just annotate a class with @Observable and you're done. All publicly visible properties are published automatically. And when using an Observable in

your SwiftUI views, there's no need to use a property wrapper to trigger view updates. You just reference your variables directly, you'll write less code and get fast and correct behavior by default.

Observable lets SwiftUI track access at a per-field level. So, your views body is only reevaluated when the specific properties used by your view change. If you modify a field not used by your view, no invalidation will happen at all. So, now when you connect your model to your views, just like with the rest of SwiftUI, the most natural code to write is also the correct code to write.

SwiftUI was the start of a new generation of frameworks designed for Swift. But the benefits of a Swift-native framework don't stop with your UI code. Core Data has long provided tools for data management, but its design was born of the era of Objective-C, and it doesn't take full advantage of everything that Swift has to offer. Many of you have been asking for a Swift-native solution to data management, designed with first-class support for all of Swift's features.

Let me introduce you to SwiftData. SwiftData is a framework for data modeling and management. It's built on top of Core Data's proven persistence layer, but with an API completely redesigned and reimagined for Swift. Like SwiftUI, it focuses entirely on code, with no external file formats. Instead, it uses Swift's new macro system to offer a streamlined API. If you were defining a model in Swift, you might write code like this, using regular Swift types. To manage this with Core Data, you'd then need to redefine the same model using the model editor built into Xcode. But with Swift data, you just annotate your class with the @Model macro.

This single line of code packs a lot of functionality, like automatically enabling persistence, iCloud synchronization, undo and redo, and more. You can then refine these automatic behaviors by annotating properties with additional attributes, like indicating the value must be unique across all instances. And Swift data uses the Codable protocol to understand structs and enums, so you can model your data with the tools that you already know. These types are fully modeled in the underlying data store, enabling you to perform fast and efficient queries, even on complex structured data. And, of course, it's simple to integrate SwiftData with SwiftUI.

Now, we started to build our demo app using the new Observable macro to drive SwiftUI view updates. So, my interface updates as I see new birds, but my edits aren't saved across runs. By importing SwiftData, I'll be able to add support for persistence really easily. I'll just replace the two Observable macros on my existing classes with SwiftData's Model macro, and my model's ready. At the root of my app, I'll add a modifier to set up the SwiftData container. And when I create a new backyard, I'll insert it into the model context so that it gets persisted. And finally, I'll hook up my BackyardList view to the persisted data. That's really easy to do with the new @Query property wrapper. Because we're now loading saved data, I can remove the default sample data that I used when I was prototyping.

Before we test it out, let's also update a widget that I've been working on. I'll just set up the container and query the same way. With the app's shared container enabled, SwiftData automatically makes my data directly accessible by the widget using the same API. Backyards are now persisted by SwiftData and delivered to the view by @Query. Before we started, I filled the database with some initial data, so you can already see some bird visitors. And if I add a new backyard object, when I return to the list, it will appear with no additional work and SwiftData provides more than just persistence, including things like support for undo and redo, which just works automatically. And as you can see, my widget is already showing the backyard that we just created. And that's a look at how easy it is to save and restore data using SwiftData.

SwiftUI and SwiftData work hand in hand to help you build engaging and powerful apps. They form a foundation of a new approach to development enabled by Swift that helps you spend less time on boilerplate and more time building your ideas.

Next, Jonathan is going to show you how to apply these technologies to elevate your app in new ways across the system.

Jonathan Thomassian

I'm really excited to tell you about four new app experiences. With WidgetKit, you can surface your content in many places across the system. With App Intents, your app's functionality is integrated into the system even more seamlessly. With TipKit, you can surface beautiful tips directly in your app. And with AirDrop, you can make it even more convenient for your users to share content with those nearby.

Let's start with WidgetKit. Widgets elevate important information from your apps, making your app experience more glanceable and available to your users as they go about their day. Widgets started on the home screen, but now they're in many more places to bring your experiences to users right where and when they need them. And this year, widgets are becoming even more powerful.

Once you rebuild for iOS-17, with just a few simple changes, your existing widgets will look gorgeous in StandBy on iPhone. They'll appear scaled up and drawn out to the edges with the backgrounds removed, so they look stunning side by side. They'll also be available on the lock screen on iPad, again, drawn to their edges and with their backgrounds removed. Here, they'll have a uniform visual appearance, so they blend perfectly with the lock screen.

And on macOS Sonoma, they'll be available on the desktop in full color and then recede into the background when a window has focus. And you can now make your widgets more useful with new support for interactivity. Simple actions that can be accomplished with a tap, like checking off a reminder, can now be handled directly from your widget. All of these new widget updates were enabled by their SwiftUI-based architecture.

Your widget's code is run asynchronously to generate content, and the SwiftUI Views it builds are then saved to an archive. Later, when the widget needs to be drawn, the archive can be loaded, rendered in the background, and then displayed as part of the system UI. When a user taps a button, its extension is run again to handle the action and update the UI. This architecture also enables iPhone widgets to be seamlessly displayed on your Mac. Thanks to the magic of continuity, the widget archive can be sent across the network to your Mac, and user interactions can be sent back to be handled on iPhone. Getting your existing widgets ready couldn't be easier. Just identify your widget's background, and update its padding to use the default provided by WidgetKit.

SwiftUI's stack-based layouts let the system adapt your widget's color and spacing based on context. Interactivity is also easy to adopt by adding SwiftUI buttons or Toggles to your widget. New support for triggering an App Intent from these controls launches your extension on demand. When your widget's content updates, the system triggers a transition animation that works just like a Magic Move in Keynote. Moved elements slide to their new locations, and added or removed elements fade and blur softly in and out. You can use standard SwiftUI transition APIs to further customize and get exactly the effects you want.

Let's take a look at how easy it is to include these new features in an existing widget. Our sample app has a widget that tracks notable events in our backyard. The first thing I need to do is identify its background. Here is our widget view, and we can see our background is right here in the ZStack. All I need to do is move it into the new container background view modifier. And starting in iOS 17, widget padding is provided for me, so I'll go ahead and remove the padding I was adding before.

Now thanks to Swift macros, I can see my entire widget timeline directly in Xcode previews. Here at the bottom of the preview canvas are all of my timeline entries, and clicking through them shows how my widget will animate when it updates. This transition is great, but let's see if we can do something that feels a bit more like a bird arriving. Just like in my app, I can make use of standard SwiftUI transitions.

Here in the editor, I'll add a transition modifier to my BirdView. Let's try a leading-edge push. That looks great. The updated preview canvas also makes it easy to see how my widget looks in different contexts like the new iPad Lock Screen or StandBy on iPhone.

Let's look at StandBy. When the background is removed, I want my bird to stand out, and its feet are a bit lost here, so let's add a birdbath. SwiftUI provides a new shows widget container background variable that lets me customize how my widget looks when the background is removed. Nice. Many of you have asked for the ability to interact with your app from your widget, and now you can. Let's add a button to refill my bird's water. Already defined an App Intent for this action, so all I have to do is associate it with a button in my widget.

Let's try it out in the simulator. And just like that, I've refilled my bird's water, and that's WidgetKit. It's never been easier to surface your content in so many places across the system. We're excited about all the ways you can use interactivity and animated transitions to enhance your widgets.

Now App Intents are about more than just interactivity and widgets. They elevate your app's functionality across the system in Spotlight, Shortcuts, and Siri. Now when you wrap your intent in an app shortcut, it will appear right next to your app icon in Spotlight results with a richer, more interactive presentation.

You just need to provide a few things. In your App Shortcut, a short title and an image or symbol, and in your app's info.plist a background color that's complementary to your app icon. And your app shortcuts will be surfaced in the updated shortcuts app, where users can set them up to run automatically, add them to their home screen, or use them to create their own shortcuts. Siri has gotten even better, too. Users have more natural language flexibility to invoke a shortcut. With WidgetKit and App Intents, users can access features of your app in whole new ways.

Next, let's talk about feature discovery in your app. Every year, we developers spend time building features that we think users will love. But sometimes, the users who would benefit most don't know the feature exists. A new framework called TipKit helps address this by intelligently educating users about the right features at the right time.

TipKit includes templates to match what users are accustomed to seeing in system apps and are easily customizable to match the look and feel of your app. Power of TipKit is its turnkey nature. Simply customize a template and add targeting to educate users on functionality related to their current context. You can also manage the overall frequency to avoid showing tips users have already seen, even if they saw them on another device.

Lastly, we have a fun new way to help users share content from your app with AirDrop. On iOS 17, your users can skip the share sheet and quickly send content to another device nearby. You can use ShareLink in SwiftUI or adopt activity items configuration on your app's UIKit view controller. And system-provided view controllers like Share Sheet and Quick Look already work by default.

WidgetKit, App Intents, TipKit, and AirDrop will help you build experiences that make your app's content available to your users in the right place and at the right time. We're looking forward to seeing how you'll use these to take your apps even further.

And now, back to Darin.

Darin Adler

Our platforms are built to give you access to the unique hardware capabilities of our products right out of the gate. Whether you're tapping into a neural engine to process a photograph, or using an accelerometer to measure sleep movements, our APIs give you the ability to harness the power of hardware technologies with ease.

Next, we'll look at this year's improvements for gaming, cameras, displays, and Apple Watch.

Here's Brandon to tell us how this all plays out.

Brandon Corey {BIO 19968990 <GO>}

The blazing performance, long battery life, and stunning graphics of Apple silicon Macs make them the perfect platform for running high-end games. And now, Game Mode and macOS Sonoma makes gaming on Mac even better. With Macs more popular than ever before, there's never been a better time to bring your games to millions of new players.

If you're a developer who builds games for Windows or gaming consoles, you'll find it is now easier than ever to bring your games to Mac by using the new Game Porting Toolkit, which significantly accelerates your development using three simple steps. First, you can evaluate just how well your existing Windows game could run on Mac using the provided emulation environment. This lets you analyze your game's potential performance immediately, eliminating months of upfront work.

In the second step, you'll convert and compile your shaders. The Game Porting Toolkit includes a new Metal Shader Converter, which you can use to automatically convert all of your existing HLSL GPU shaders to Metal, including all of your game's advanced shading pipelines like geometry, tessellation, mesh, and ray tracing stages.

You can use this tool while you're building your game in Xcode or in a custom Windows-based toolchain for shader compilation, which brings us to step three, converting your graphics code and optimizing your game. Metal provides all of the it's all of the advanced graphics and compute features used by modern high-end games, which makes converting your graphics code incredibly straightforward. And the powerful graphics performance and debugging tools, integrated right into Xcode, give you all the guidance you need to fully optimize your game. We can't wait to see you make use of these tools to take advantage of the incredible performance of Apple silicon, and powerful graphics, display, audio, input, and gaming technologies to bring your games to Mac dramatically faster than ever before.

Speaking of faster performance, we also have news to share about camera. Apple continues to push mobile photography forward and expand APIs that let you do incredible things with the world's most popular camera. AVCapture is used by over a third of iOS apps across all categories, from creativity to productivity, social media, and even health. And it's getting some powerful performance improvements this

year. With zero shutter lag, you capture the exact moment when the shutter is pressed.

With overlapping captures, the camera will dynamically adjust image quality when the shutter is pressed rapidly. And with deferred processing, high-quality images, including deep fusion, can be processed in the background. These improvements will make shot-to-shot time up to 3x faster between shutter presses in your iOS or iPadOS app, even while capturing high-quality images. And later this year, you'll also be able to use the volume up and down buttons to trigger the camera shutter.

There is also big news on displaying photos in your apps. In recent years, HDR-capable displays have become widely available, including on iPhone, iPad, and Mac. HDR photography takes advantage by capturing additional dynamic range to display bright highlights and dark shadows with more fidelity and realism than ever before, just like you experience the scene in real life.

Apple has been the leader in bringing HDR photography to the mainstream with trillions of HDR photos captured on iPhone 12 and later. To do this, the camera app intelligently analyzes the scene and stores additional highlight and shadow data at capture time. But while there are industry standards for HDR video, there has been no standard for storing and displaying HDR photos, which has made them difficult to work with, but that's all changing this year.

Apple has driven an industry-wide effort resulting in a specification for encoding and displaying HDR photos, recently ratified by the International Standards Organization. It's now available as an API in iOS, iPadOS, and macOS, so that you can display compatible photos in your apps with just a couple of lines of code. This is great news for all apps that display images, and not just for photos. Generative content apps can use it to make images pop with extra dynamic range. So, now a complete HDR workflow will be possible from capture to edit to sharing and everyone will be able to enjoy its full HDR glory just like the photographer experienced it.

Next, let's talk about video. We all spend a lot of time with our cameras on these days, whether for video conferencing or in creative apps. We've continued to invest in video lighting and effects like these new reactions, gestures and presenter overlay effects. They're built into the camera feed you receive so they'll just work in your app. Your app can observe when these effects are invoked and you can perform additional actions like promoting that user's tile so everyone can see that they reacted. If you have a video conferencing app, there are a number of improvements to screen sharing and camera functionality in Screen Capture Kit. The new Screen Capture Kit picker makes it simpler for users to start screen sharing in a more private and secure way. The picker also makes it simple for your app to capture multiple windows or even multiple apps all at once. Your users can start sharing right from the app they're in, and they'll appreciate that they see a preview of what's being shared in the new Video Effects menu.

Another benefit of Screen Capture Kit is higher resolution content for better-looking screen shares when sharing a single window. We've also brought external camera support to iPad. Any USB camera can now be connected and used within your iPad app. And we are thrilled to add camera and microphone capabilities to an entirely new platform, tvOS.

With Continuity Camera on tvOS, you can take advantage of iPhone and even iPad to integrate video and audio in tvOS apps for the first time. And with a living room as your stage, there's so much potential to do amazing things with your apps. For example, conferencing apps can use Center Stage, which makes group video calls more dynamic on the biggest screen in the home. Games can incorporate camera and audio feeds directly into the action, and creative apps can stream or record video while applying portrait mode and other fun effects. If you already have an app on Apple TV, you can make the experience even more connected and social by leveraging frameworks you're likely already familiar with.

And for those who want to take advantage of Continuity Camera, but haven't yet developed for tvOS, it's super easy to get started with SwiftUI. Let me show you how it works. So, here's my example iPad app, PartyCam, that takes selfies with fun filters. First, we'll add Apple TV as a destination. Next, we need to add the picker so the user knows which phone to choose from. Then we'll add the code to present the picker, and then we run. And now, I just pick up the remote and snap a photo.

Hey, it's Susan and Serenity. Awesome. We can't wait to see how you make the living room even more entertaining.

Now, over to Lori with what's new on watchOS.

Lori Hylan-Cho

As our first wearable device, Apple Watch opened up new possibilities for developers to bring timely notifications, health and fitness insights, and handy app experiences to people's wrists. This year, watchOS 10 is getting a huge design refresh that takes advantage of the larger, brighter displays, and more capable connected hardware of modern Apple Watch models. SwiftUI is at the very heart of this redesign. We employed it across the system to update virtually every app, to be more dynamic, more colorful, and more glanceable.

Let me show you a few examples. There's a renewed focus on pagination in watchOS 10, and vertical tab views allow for variable page sizes, making better use of the crown for scrolling within apps without sacrificing space for all your valuable content. The new container background modifier lets you use color with gradient fills for readability to help users understand where they are within your app or make data more glanceable. For app designs that fill the screen, new toolbar item placements make better use of the corners and allow the time to automatically shift to the center if necessary.

And for layouts with a strong source list/detail view association, NavigationSplitView is more compelling than ever on Apple Watch, using the same code you'd use on other platforms. And the pivot in and out of the detail view on Apple Watch even uses SwiftUI's new interactive spring animation. This animation is composed of 12 separate tracks for an animation plan that animates both the size and position of the views, driven by the velocity of the user's finger. And of course, all the major UI components have been updated with material treatments to be legible on any background, and they size automatically for the hardware.

If you've already adopted SwiftUI in your app, you'll get these updates automatically, when you build with the watchOS 10 SDK. Let me show you what I mean with the Backyard Birds app that I originally designed for watchOS 9. It's a list detail app built using a navigation stack, and even without any code changes, it already works well when recompiled with watchOS 10. The views automatically adjust to accommodate the new navigation bar height, and the large title shrinks and animates to a position under the time as I scroll.

Now let's make some edits to make the app feel even more at home on watchOS 10. I'd like to see the current status of my primary yard whenever I launch the app, rather than always having to choose it from the list. NavigationSplitView is designed specifically for apps with a strong source list/detail relationship. So I'll swap out my navigation stack for a NavigationSplitView. I'll remove the navigation title from the source list, since it's no longer the entry point for the app, and indicate the detail view that should open by default.

The source list is now tucked away behind the detail view, and I'm focused on the details, but my detail view is pretty long. It's a scrolling list with three clear sections, so I'll break up the content by converting the list to a vertical tab view with a separate page for each section.

Each of these pages will be the height of the screen by default, and because that final section contained a ForEach loop, each bird is getting its own page. I'd rather have a list of all the birds on a single tab that scrolls if it exceeds the height of the screen, so I'll change this ForEach to a list, which will make it a scrollable view.

Next, I'd like to add some color to differentiate between the tabs, so I'll add a container background modifier to each tab, and to quickly spot when food or water are running low, I'll make the sustenance background dynamic. It'll be green, yellow, or red based on the supply levels. Looking at the changes in live previews, you can see how the background color gives a sense of place within the app, and makes it easier to spot when I need to refill food or water.

The final tab lets contents scroll beyond the bounds of the screen, and I can still reach the list of backyards by tapping the source list button. It took just a few changes to make my app fit right in on watchOS 10. SwiftUI made our lives easier as we worked to make every app on the system look amazing, and as we just saw, it'll

make updating your apps for these new design paradigms easy too. If you haven't moved to SwiftUI in your app yet, now is the time.

In addition to these SwiftUI-based app updates, watchOS 10 also has a brand new system space, where app intense relevance helps your widgets move up the smart stack at just the right time. And you can use the same container background modifier mentioned earlier to make those widgets really shine with beautiful custom backgrounds.

For those of you building workout apps, you can help your users take advantage of the fitness tracking capabilities of Apple Watch with the new custom workout API, which lets you share fitness plans from your app, and a new core motion API that will enable higher fidelity capture of motion and accelerometer data to help improve swing analysis for sports like golf and tennis. And because we know how creative you all can be, this data is available when running any kind of workout. We can't wait to see what kinds of experiences you'll build with it. And that's a glimpse of some of the big changes coming to watchOS this year.

And now, back to Darin.

Darin Adler

A platform is also about the people who use it and the values that are built into it from the ground up. These core values mean a lot to us at Apple, and we know they mean a lot to you, too. Together, we can make your app adapt to people's needs so anyone can use it and create apps that show respect for each user's need to protect themselves and protect their data. Starting with what our platforms provide can make this important work easier for you.

To tell you more, here's Chris.

Chris Fleizach

We believe that everyone should be able to use our devices to do what they love. That's why making our products accessible is a core value at Apple. According to the World Health Organization, there are over 1 billion people around the world with disabilities. Building accessible technology means giving more people the best tools to create, learn, stay connected, and live on their terms. That's why Apple products have accessibility features available out of the box for users, who are blind or low vision, deaf or hard of hearing, non-speaking or at risk of speech loss, those with physical or motor disabilities, and those with cognitive disabilities.

Our frameworks come with built-in support for these accessibility features, along with tools and APIs to help you ensure that your apps are accessible to everyone. Often, accessibility comes down to taking small steps that have a big impact. As an example, last year, we added the ability to detect and describe doors in the Magnifier app, helping people independently navigate their environment. This year, we're expanding our support for users, who are sensitive to animations and flashing

lights. These users often face a difficult choice between avoiding potentially risky content, such as movies that begin with a flashing lights warning, or consuming it without enough information, which can mean exposing themselves to health risks like seizures.

Our frameworks now include APIs for two features that can make content in your apps more accessible to these users. The first feature is pause animated images, which will stop the motion in animated GIFs in Safari, Messages, and more. And it's easy for you to add it to your app. Suppose you're animating a sequence of images with TimelineView and SwiftUI, you can use a new environment property to understand if the user prefers to pause animated images, and if they do, replace animated images with a static image to respect their choice.

The second feature is dim flashing lights, which automatically darkens the display of video during sequences of bright flashing lights. There's a new phase in AVFoundation's video rendering pipeline to identify and dim flashing elements. If you use AVFoundation to play media in your app, there's no extra work to support this feature. However, if your app uses a custom media player, you can use a new API to identify video content with flashing lights, and automatically darken it. We've even open-sourced the algorithm so you can understand the science behind this feature.

Our commitment to building accessibility to each product and framework extends to our newest platform, visionOS, because we believe the world's best technology should adapt to everyone's needs. VisionOS comes with dozens of accessibility features built into its foundation from the very start. And you, as a developer, will play a critical role by making your visionOS apps accessible for everyone, using the same tools and APIs that you use on iPadOS and iOS today. We are thrilled to work with you to bring spatial computing to diverse sets of users, and you'll hear more about accessibility on Vision Pro in a bit.

Now, over to Katie to talk about privacy.

Katie Skinner

At Apple, we believe privacy is a fundamental human right. We hear from our users how much they care about it, and we're here to help you deliver great privacy along with your features. Over the years, we've built many ways to help you build privacy into your apps. This year, we're making it even easier for you to offer robust privacy experiences to your users in a number of areas, starting with improvements to privacy prompts.

Let's start with calendar permissions. We know that in many cases, your apps don't need read access to Calendar. They just need to write new events. For these cases, we've created a new add-only permission. This will help you get the access that you need, and the user will get a prompt that makes sense to them.

And in photos, we've offered two choices for users to provide photos to your app. They can select which photos to share, or they can provide access to their full library.

We wanted to make it even easier to help users select photos to share with your app so you get exactly what you need and users share only what they intend. So, we're adding a new photo picker that you can embed into your app so users can easily select photos to share from inside your experience.

Next, app privacy; it's important to help users understand how you protect their data, so we've built features to help you do just that. For example, Privacy Nutrition Labels help users understand what data you collect and how you use it. Many of you include third-party SDKs in your apps, which can offer great functionality. But including them can make it harder to get your labels right, because you might not understand how they handle user data. And they can introduce security challenges to your software supply chain.

So, this year, we're introducing two updates. First, to help you understand how third-party SDKs use data, we've introduced privacy manifests. These are files that outline the privacy practices of the third-party code in your app in a standard format. When you prepare to distribute your app, Xcode will combine all the manifests across all the third-party SDKs you're using into a single, easy-to-use report. With one summary report for all of your third-party SDKs, it has never been easier to create accurate labels in App Store Connect.

We also want to help you improve the integrity of your software supply chain. When using third-party SDKs, it can be hard to know the code you've downloaded was written by the developer you expect. To address that, we're introducing signatures for third-party SDKs. Now when you adopt a new version of a third-party SDK in your app, Xcode will validate that it was signed by the same developer to give you more peace of mind. With privacy manifest and signatures for third-party SDKs, we're making it even easier for you to protect users and help them understand how their data will be used.

And finally, communication safety, which uses privacy-preserving technology to protect children on our platforms. Our platforms and the apps you build play an important role in the lives of many families across the world, especially in how they communicate. In iOS 15, we introduced the Communication Safety feature in messages to offer more protections for children using Apple devices.

Many of you build apps, where users share content and also want to protect children in vulnerable situations. We know that training and implementing a model to classify unsafe content can be a difficult problem to solve and a lot of work to undertake. And so we're bringing Communication Safety to the entire platform with the Sensitive Content Analysis Framework.

With just a few lines of code, the framework helps you detect images and videos that contain nudity. And this happens entirely on device, so you can build positive experiences in your app for children who have Communication Safety-enabled. The Sensitive Content Analysis Framework uses the same technology that underpins

Communication Safety. The framework protects users' privacy by processing images and videos entirely on device.

In addition, we wanted to give everyone the ability to blur sensitive content with Sensitive Content Warning. The Sensitive Content Analysis framework will let you know, if a user has enabled Communication Safety or Sensitive Content Warning, so you can tailor your app experiences based on which feature is enabled. These new features make it easier than ever to offer privacy and peace of mind to your users.

Next, Chris will tell us what's new in the App Store.

Chris Markiewicz

There is a trusted place for users to discover apps that meet our high standards for safety, privacy and performance, and that's the App Store. The App Store empowers you to scale your app distribution worldwide using a variety of business models like in-app purchase.

StoreKit is foundational for safe and trusted in-app purchase. We provide a robust data model to connect your products to your app's UI. If you offer in-app purchase, you know the importance of presenting key details like price and subscription duration so users can make an informed purchase. And we know many developers struggle to get this right.

This year, we're taking StoreKit further with a new collection of views to power your app's merchandising UI across all platforms using the best practices from the Human Interface Guidelines. Using the declarative syntax of SwiftUI, you can craft your merchandising experience and StoreKit takes care of the rest. Let's view some examples.

The ProductView enables you to display your products using data you defined in App Store Connect. You can easily customize it to match the look and feel of your app. The Subscription Store View is a purpose-built view for subscriptions. With as little as one line of code, the description, price, and duration for each level of service are clearly presented to the user. These views aren't just for iPhone. The same code creates a view to match the platform experience across all Apple devices, and it even scales to fit the compact display of Apple Watch.

And of course, they were also created with accessibility and localization in mind for all users. Integrating the Subscription Store View into my app is super easy. Let's work on the Backyard Birds Pass subscription offering. I'll go into Xcode to add some declarations to my Subscription Store view to match the look and feel of Backyard Birds. I'll customize the marketing content with the view, add a container background, and change the style of the subscription options. Now we have something that looks great and matches the app's branding.

Let's also take a quick look at the simulator, where I can get the same full experience that my users would. Here I can select an offer, click subscribe, and complete the purchase. That's all it takes to build a customized purchase experience to show users all the information they need to make an informed decision.

We also know how hard it can be to customize the offers you present based on the user's purchase history or subscription status. That's why the Subscription Store view will automatically determine user eligibility and display the right offer. The new StoreKit views are the best way to merchandise your in-app purchases.

Another important aspect of growing your business is understanding how advertising helps users discover your app. That's why SKAdNetwork helps ad networks measure how successfully ad campaigns drive downloads of your app, all while preserving user privacy.

In addition to measuring downloads, we know it's important to understand how advertising can bring users back into your app. SKAdNetwork 5 will support measuring re-engagement. In addition to measuring conversions after a user downloads your app, you'll also be able to measure conversions after a user opens your app by tapping on an ad. Version 5 will be available in an iOS release coming later this year. Our new features for in-app purchase and SKAdNetwork will help you grow your business responsibly while respecting user privacy, giving users transparency and peace of mind.

Now back to Darin.

Darin Adler

Like our programming language and frameworks, our tools were made for each other. Xcode brings together everything you need to develop, test, and distribute apps across all our platforms. TestFlight and Xcode Cloud make the experience seamless, from your first line of code to the first download in the App Store.

Now, Ken will take us through a look at the latest developments.

Ken Orr {BIO 17574204 <GO>}

For all Apple platforms, one tool is at the center of your developer experience, helping you build all your great apps, Xcode; it brings together powerful features like an editor that blends crafting code and designing user interfaces with interactive previews and live animations and built-in source control for making, reviewing, and sharing changes with your team, debugging, profiling and testing tools to help you evaluate and refine your app. All of it is connected in Xcode Cloud, a continuous integration and delivery service that simplifies distribution to TestFlight and the App Store.

For Xcode 15, there are some big updates. Let's start where you spend most of your time, the source editor. Code completion helps you get the code you want faster and with fewer mistakes. With Xcode 15, you'll start with the most relevant completions, with the editor using the surrounding code for prioritization, whether that's the most appropriate modifier for a specific view you're using, or when you're chaining modifiers as you customize a view. And Xcode will automatically generate symbols for your asset catalog resources, which means they show up in code completion and are easy-to-use in your code.

Now this gives you type safety and peace of mind that your assets are available at runtime. A great place to use those assets is in your SwiftUI views. Xcode's previews are a seamless way to iterate on your UI with nearly instant feedback as you design right alongside your app's code.

In Xcode 15, previews are easier to use and available in even more places. It starts with a new syntax built using Swift macros that is simple to write and easy to remember. Working with different platforms and devices is now easier. You can choose them right in the canvas to make sure your views look great everywhere.

And you can now use previews across all UI frameworks, wherever you're using SwiftUI, UIKit or AppKit, you can iterate on your UI with a consistent experience from anywhere in your code.

Now to keep pace with quickly making and previewing changes, Xcode streamlines how you review and commit those changes. Many of you use Git staging in your workflows and now it's integrated directly into Xcode. You can stage or unstage any change with just a click, crafting your next commit without leaving your code. From the Source Control Navigator, you can review all your changes in a single view, helping you put the finishing touches on your commit. And Xcode shows unpushed commits too, so you can perfectly time getting all your commits together into a pull request.

Now let's talk about testing. It's an essential part of creating a high-quality app. As your code grows across many platforms, devices, languages, and user features, navigating your test results and knowing where to focus, well, that can be challenging. Xcode 15 includes a complete redesign of the test report, giving you new tools to better understand your results and take action.

And it starts with a beautiful results overview. It shows me top insights, including common failure patterns, like those with the same assertion message. I also get a comprehensive summary of how my tests ran across configurations, like device and language, and that helps me spot patterns. And there's a heat map to quickly show me where to focus.

Now, I've been working on a feature in the Backyard Birds app, and looking at my latest test run, I see I've got some issues. I'll start by clicking on this insight. Now, it looks like a tap gesture is failing across my new test. I'll click this test that failed,

which gives me a detailed view of the test activities here on the left, along with a full recording of the app's UI over on the right. I can select a specific activity, or I can use the new timeline to jump to any moment. I can see marks for interesting points, like when the test scrolled this view here, or over here where it tapped the search field. And this is a full video recording that gives me much more detail, showing me exactly what happened before the test failed.

I'll play it to see what happened. Now here the test is selecting some food to feed the bird. The bird starts flying around and then it disappears before my action is complete, which is what caused my test to fail. Now this new test detail with a timeline and video recording makes reviewing tests for animations, gestures, and other UI experiences so much faster. And when I need more than video, I can also see accessibility frames, giving me even more insight into my views.

All combined, the new report experience gives you deep insights, finer details, and more interaction to get the most out of your testing. And it works seamlessly in Xcode and Xcode Cloud. Xcode Cloud is a continuous integration and delivery service that helps you build, test and share your app across all Apple platforms. It's deeply integrated into Xcode, saving you time by keeping you focused in one place. It leverages Apple's cloud infrastructure to offload your builds, tests, and even code signing for distribution. It connects with Apple services like TestFlight and App Store Connect, and it's built with advanced security to protect you and your projects.

Last year, we made Xcode Cloud available to every member of the Apple Developer Program. Tens of thousands of teams onboarded their projects directly from Xcode, and the feedback has been great. And we continue to make Xcode Cloud even better.

We really care about performance. Since last year, we've made significant investments so you can do even more with your compute time. And we've also added your most important requests. When distributing to TestFlight, you can now create and share tester notes, helping keep all your users up to date on your latest improvements. Xcode Cloud also supports macOS notarization when distributing with Developer ID. So, you can automatically check your app for malicious components before sharing it with your users. With just a few minutes to set up your workflows, you can take advantage of hours of compute time to build, test and deploy your apps. And we are excited about the future of Xcode Cloud as we continue to make it faster, safer, and more flexible.

Over the last year, we've optimized the compiler for the multi-core architecture of Apple silicon to make all of your builds faster and more scalable. The linker has been redesigned from the ground up, bringing massive improvements in link speed. Linking is up to 5x faster. The new linker also reduces the size of debug binaries by up to 30%. And for apps that embed a lot of frameworks, there's a new framework type that delivers faster builds during development and reduced app size and faster launch time for production.

Finally, we continue to make Xcode faster and easier to just get started. On the Mac App Store, Xcode is now 50% smaller and all simulators are downloadable on demand. And that means you can get started quickly and install the platforms you need at just the right time.

And now back to Darin.

Darin Adler

Our singular approach of building a language, frameworks, tools, and services that all work together truly comes to life when we introduce a new platform. The use of shared foundations help make the platform seem familiar while ensuring its individual strengths shine. visionOS takes advantage of the investments we've been making in our platforms for many years and showcases the benefits of this approach.

Here's Mike to tell you all about it.

Mike Rockwell {BIO 3482166 <GO>}

The launch of Apple Vision Pro marks the beginning of a journey together in spatial computing. You can now push beyond what you previously understood to be possible, and reimagine what it means to be connected, productive, and entertained. Vision Pro has many groundbreaking technologies. Your apps will make use of new elements to interact with the user's space and blend seamlessly with their room. This is all possible using powerful technologies you already know: SwiftUI, RealityKit, and ARKit, now extended for visionOS. With this solid foundation and an updated set of developer tools, you're going to feel right at home creating an entirely new universe of apps for Apple Vision Pro.

Today, we're going to cover everything you need to get started with visionOS, from the fundamental building blocks to critical frameworks, tools, and technologies. You'll learn about the activities and programs we are launching to support you as you bring your apps to life on Vision Pro, apps that take advantage of the infinite canvas of Vision Pro or that turn a room into an immersive media environment. So, let's roll our sleeves up.

Vision Pro allows you to rethink what is possible for your app experience. Regardless of the kind of app you're building, you need to understand how it will exist in 3D in your user space. This will inform the decisions you make as a developer. That understanding will help you take advantage of the capabilities of Vision Pro, bringing focus and immersion to your apps in ways that, until now, were just not possible. By default, apps launch into the shared space. The shared space is where apps exist side-by-side, much like multiple apps on a Mac desktop. The user has full agency to reposition apps wherever they would like.

Let's talk about the elements you can use inside a space, starting with the familiar window. On visionOS, your app can open one or more windows, which are SwiftUI scenes, and behave just as you would expect as planes in space. They can contain

traditional views and controls, and they even support 3D objects and reality views, allowing 3D content to sit alongside 2D content.

Additionally, your app can create three-dimensional volumes, which are also SwiftUI scenes, and showcase 3D objects like a game board or a globe. Volumes can be moved around the space and viewed from all angles.

In some cases, you may want more control over the level of immersion in your in your app, maybe for the user to watch a video, play a game, or rehearse a presentation. You can do this by opening a dedicated full space, in which your apps, windows, volumes, and 3D objects appear across the user's view.

So, those are the foundational elements of spatial computing: windows, volumes, and spaces. They give you a flexible toolset to build apps that can span the continuum of immersion.

Now to tell you more about how to get started building apps, here's Geoff.

Geoff Stahl

Vision Pro supports several different kinds of apps. Existing iPad and iPhone apps are supported each as a single, scalable 2D window with their original look and feel, but that's only the beginning of what's possible. At its core, visionOS is similar to iPadOS and iOS, and includes many of the same foundational frameworks. You use SwiftUI and UIKit to build your user experience, RealityKit to present 3D content animations and visual effects and ARKit to understand the space around the user. These are all part of the visionOS SDK.

So what does it take to bring your app to Vision Pro? First, in Xcode, add the visionOS destination to your project. Now, when you rebuild, your app automatically gets some really cool improvements: materials with the visionOS look and feel, fully resizable windows with spacing tune for eyes and hands input, and access to highlighting adjustments for your custom controls. Then, you can add visionOS-specific code to expand your app into a collection of windows, volumes, or spaces. From here, you can begin to take advantage of the extended capabilities of SwiftUI, RealityKit and ARKit.

Here's Enrica to tell you how.

Enrica Casucci

On visionOS, many of our frameworks have been extended to support spatial experiences. With SwiftUI, you can now add depth or add a 3D object inside a window. On iOS and macOS, a ZStack is typically used for layering views. VisionOS goes further, and you can separate them with depth. This gives your two-dimensional apps a three-dimensional feel.

You can add subtle changes in depth to your UI elements by using the new z-offset view modifier, a higher offset value presents the view in front of views with lower values. And you can use this to show emphasis or to indicate a change in modality, for example, when showing details for a selection. And with additional view modifiers, you can have more control over width, height, and depth.

On Vision Pro, even gestures are aware of the additional space. We are all used to dragging objects around the screen. Now people will be able to move or rotate objects anywhere in their physical space.

You can also create a volume with SwiftUI. It can exist alongside your app windows, and when running in the shared space, sits side-by-side with other apps. And SwiftUI windows and volumes can also be inside a full space, where you can place 3D objects and SwiftUI elements anywhere in the user's room. Not only can SwiftUI quickly and easily bring your apps onto visionOS, it can also render fully immersive experiences.

And SwiftUI now renders through RealityKit, so you can easily mix SwiftUI and RealityKit APIs. You can utilize UI enhancements like ornaments, materials, and hover effects. Ornaments allow you to affix UI components to the edges of your windows and volumes. They're great for things like toolbars and menus. Hover effects highlight UI elements in response to where a user looks. While system controls automatically get these effects, you have options to decide whether your custom control shows a highlight or a glow, and materials adapt to the world around your user. Dynamic blurs and vibrancy make your app readable no matter the conditions.

Just like on our other platforms, we recommend you use SwiftUI to build your visionOS apps. Your existing apps built with UIKit can also be recompiled for visionOS, and will get access to ornaments, hover effects, materials, and the native look and feel. SwiftUI is just one of the frameworks that you'll use to build spatial experiences on visionOS.

Edwin will walk you through a few more that will help you on your journey.

Edwin Iskandar

When you're ready to extend your apps with full scenes of dynamic 3D models, animations, and visual effects, you'll want to use RealityKit, Apple's 3D rendering engine built from the ground up for rendering spatial experiences. We introduced RealityKit and SwiftUI together in 2019 as independent frameworks. On Apple Vision Pro, they are deeply integrated so you can build sharp, responsive, and volumetric interfaces.

3D content can bind to SwiftUI views and states, seamlessly coupling 2D and 3D visual elements together. RealityKit automatically adjusts the physical lighting conditions and grounds the experience in reality by casting shadows on floors and tables. This makes your app look like it belongs in the room.

RealityKit also has significant new capabilities, including the ability to create portals into 3D scenes, like the dinosaur we saw in the keynote, to render incredibly sharp text so you can read comfortably, and a customizable material system to create stunning visual effects.

Additionally, rendering is even more efficient on Apple Vision Pro by using a technique called dynamic foveation. RealityKit leverages eye tracking to selectively render regions the user is focusing on at very high fidelity, reducing the rendering cost of content in the periphery and enabling your apps to maximize the processing power of the device.

RealityKit renders your 3D models with stunning photorealistic results by allowing you to specify its physical properties, such as how reflective or metallic it is. By default, RealityKit renders virtual content so its lighting is connected to the lighting conditions of the physical room. If you want to customize or even extend the realism based on your creative intent, you can provide an image-based lighting asset, or IBL, to individualize the look.

For authoring your materials, RealityKit has adopted MaterialX, an open standard for specifying surface and geometry shaders used by leading film, visual effects, entertainment, and gaming companies. The broad support for MaterialX across a variety of industry-leading creation tools makes it much easier to write shaders. MaterialX lets you design and rapidly iterate on the look of individual materials by allowing you to define its shading properties. There are repositories of materials you can choose from and helpful tools like Houdini and Maya that can be used to author your own custom shaders.

RealityKit provides a new SwiftUI view called RealityView. RealityView can be used within windows, volumes, and spaces letting you place 3D content anywhere you want within the scenes that you control. It also supports attachments, which allow you to embed 2D SwiftUI views with your 3D content.

Alongside SwiftUI and RealityKit, a third powerful framework enables you to take full advantage of the user's surroundings, ARKit. ARKit understands the space around the user, allowing app content to interact with the room, whether it's a ball bouncing off the floor or water splashing on the wall. ARKit hosts the real-time algorithms on visionOS that power a host of core system capabilities. These algorithms are always running, which means your apps automatically benefit from ARKit in the shared space, no matter how they are built. Persistence and world mapping are handled by the system. Segmentation, matting, and environment lighting are automatically applied to the entire space. And with the user's permission, ARKit features can be made directly available to your full space apps.

Inside a full space, your apps can interact realistically with the room by leveraging ARKit's plane estimation, scene reconstruction, image anchoring, and world tracking, which uses the same anchor concepts as ARKit on iOS. This makes migrating your existing ARKit app to visionOS easy.

And on visionOS, you now also get access to skeletal hand tracking. You can use hand tracking to create custom gestures for your app experience or even augment the user's hands with digital information. ARKit provides positioning and orientation of joints in the same skeletal model introduced on iOS.

With the integrated approach across SwiftUI, RealityKit, and ARKit, you can leverage the learning and investment you've already put into Apple platforms over the years. And that includes efforts you have put into making your apps accessible. This is especially meaningful because spatial computing opens up great new opportunities for accessibility. On Apple Vision Pro, users with physical and motor disabilities can interact with their device entirely with their eyes or their voice or a combination of both. Users can also select with their index finger, their wrist, or their head as an alternative pointer for navigating Vision Pro. You'll make your apps accessible on visionOS with the same techniques you used for years on our other platforms.

Accessibility Inspector analyzes and reports accessibility issues in your visionOS apps and gives you suggestions on how to make them more accessible. And with the accessibility support built into our frameworks, it's easy to bring your visionOS apps and experiences to users with disabilities.

For example, you can help VoiceOver describe objects in your apps to users who are blind or have low vision by adding labels and descriptions that convey information like an object's physical appearance. You can also incorporate user preferences to make your apps more accessible by design, such as dynamic type for larger text sizes, reduced transparency for better legibility, and alternatives to headlock content. By making use of these familiar APIs and tools, your apps will play a huge role in making spatial computing accessible to users around the world.

So, that's an introduction to the powerful frameworks you can use to build amazing visionOS apps. Now here's Thessa to talk about all the tools available to you.

Thessa Buscar-Alegria

As you start building apps for Vision Pro, you'll recognize more than just the frameworks. Our developer tools have also been updated to support visionOS. Just like for all our other platforms, Xcode is at the center of your development experience for Vision Pro. You can add a visionOS destination to your existing projects, or you can build an entirely new app dedicated to the unique capabilities of this platform.

The first step in building your app is getting the look right across both 2D and 3D content. Xcode Previews lets you rapidly iterate on appearance without leaving the editor. When it's time to start testing your entire app, you can move to the simulator, which provides a powerful environment to run and debug your app. You can move and look around the scene using a keyboard, trackpad, or a compatible game controller. And you can interact with your app by simulating the system gestures. The simulator also provides three different simulated scenes with day and nighttime lighting conditions to help you visualize your app.

When it comes to developing with your Vision Pro, you'll be building and running your app on the device to make sure it's working. I'm super excited to share this additional feature that will change the way you work.

Mac Virtual Display lets you bring a high-fidelity 4K virtual monitor of your Mac right into your Vision Pro just by looking at it. You can use the full capabilities of your Mac from within your device, and that includes Xcode. This is an incredible end-to-end development experience: coding, testing, and debugging your app on your Vision Pro the entire time without any interruptions or loss of context.

Sometimes you need to dig into the details of how your app's content relates to the real world. When your app interacts with the user's room, it can present unique challenges, such as 3D content visually clipping through walls. Xcode features practical debug visualizations to help you explore collisions, occlusions, and scene understanding as it relates to your 3D content. These visualizations work both in the simulator and on Vision Pro.

As you evolve your visionOS apps, they will become more spatial, breaking outside the boundaries of flat windows and bringing 3D experiences to users unlike ever before. Doing this right requires a new visual tool. That's why we made Reality Composer Pro. Reality Composer Pro is an application that lets you preview and prepare 3D content for your visionOS apps. You can import and organize assets such as 3D models, animations, images, and sounds. It integrates tightly with the Xcode build process to optimize your assets for use on Vision Pro. You can easily send your content to Vision Pro to preview, size, and rotate it, push new changes at any time and see how your content looks before ever typing a line of code.

Here's Jason to give you a tour.

Jason Cahill {BIO 20122291 <GO>}

I've been working on one of our sample projects that takes you through the solar system. I want to make sure my 3D content looks great on Vision Pro. And I'm going to use Reality Composer Pro to update it now. I'll start by importing the satellite model and images that I made in my 3D modeling application. This model has a lot of details, but presently no materials. I'll create a custom material in Reality Composer Pro and then I'll jump to the shader graph, where we can customize it. It's easy and fun to add some simple images and function nodes to change the look of an object, all without needing to get into code. You can try things and see the results instantly, and you'll never get compiler errors from having to hand type shaders.

I'll repeat this for the other nodes, too. At any time, I can preview my work right on my Vision Pro. I can scale it, move it, or rotate it, and make sure it looks and feels how I want. I like these changes. And then, jump back into Xcode where my changes have been reflected in my project and are ready for use in Xcode Previews or the simulator, just like that, ready for me to load in code and use in my app.

With Reality Composer Pro, your visionOS development process has a powerful new tool that works side-by-side with Xcode, making previewing and preparing 3D content as easy as building and running your code.

Thessa Buscar-Alegria

When you're ready to beta test your app, TestFlight will be available to use on Vision Pro from the start. Submit new builds to TestFlight just as you would expect, through App Store Connect, Xcode, or Xcode Cloud. And testers can install them and provide valuable feedback right from the native TestFlight app. They can also access beta builds of your compatible iPad and iPhone apps on Vision Pro.

With this family of development tools and services from Xcode to TestFlight, and now Reality Composer Pro, you'll be well on your way to making amazing apps for Vision Pro.

And now back to Mike.

Mike Rockwell {BIO 3482166 <GO>}

Many of you have invested years building 3D apps and games with Unity. To make it easy to bring your apps to Vision Pro, Apple and Unity have been deeply collaborating to layer Unity's real-time engine on top of RealityKit and enable their development tools to target visionOS. This means that Unity-created apps can coexist with other visionOS apps in the shared space and take full advantage of the unique benefits of Apple Vision Pro.

To tell you more, I'd like to introduce Ralph from Unity.

Ralph Hauwert {BIO 21931953 <GO>}

Thank you, Mike.

It's been truly exciting to collaborate with Apple on our deep integration for visionOS. Now you can use Unity's robust and familiar authoring tools to create new visionOS games and apps. And you can bring your existing Unity-created projects to Vision Pro to reimagine your experiences for this new platform. Your apps get access to all the benefits of visionOS, such as high-resolution pass-through and dynamic-foveated rendering, in addition to familiar Unity features like AR Foundation.

And with this deep integration between Unity and visionOS, your apps can even appear alongside other apps in the shared space. By combining Unity's authoring and simulation capabilities with RealityKit's managed app rendering, content created with Unity looks and feels at home on visionOS. With Unity, you'll be able to get started quickly with our community, our tutorials, and templates.

We believe that Apple Vision Pro has launched the new frontier of spatial computing, and we're really excited to see the brand new apps and games that you will create.

Mike Rockwell {BIO 3482166 <GO>}

Thank you so much, Ralph. It's been amazing working with you.

There are a few more fundamental features of visionOS that you should know when you're planning your next steps.

Here's Jeff to give you all the details.

Jeff Norris {BIO 3718164 <GO>}

The frameworks you use to create apps for Vision Pro were designed with two goals in mind: to enable you to leverage the platform's powerful hardware and software with minimal effort; and to preserve user privacy. You can see these principles at work throughout the system from the way apps blend in with their surroundings to the natural user interactions to the new possibilities for collaboration.

visionOS builds a 3D model of the user's surroundings to enable realistic lighting and shadows. As a result, apps can naturally blend in with the world around them without needing access to camera data. VisionOS uses the same 3D model to enable the next generation of spatial audio for your apps.

In the real world, sound reverberates through and around a space, bouncing off walls and furniture before arriving at a person's ears. Your app's virtual sound should behave the same way. This requires a sophisticated understanding of the user's surroundings. The PHASE spatial audio engine in visionOS fuses acoustic sensing with 3D scene understanding to create a detailed model of the sonic characteristics of the space. As sound sources and the user move around the environment, PHASE updates its audio ray-traced simulation in real-time. This model combines with Apple's personalized spatial audio technology to tailor the experience to each individual and their surroundings automatically.

All of this incredible complexity is managed for you, and you don't need access to any details of the user's physical space. Using RealityKit, you simply decide where you want sounds to come from, and visionOS naturally mixes them into the real world. Just like the algorithms that power realistic lighting and audio in your apps, user input on visionOS works automatically and is private by design.

Hover effects render automatically for any UI elements built using SwiftUI or RealityKit. If a user looks at a button, it highlights before they tap on it. And just like a click of a mouse or a tap on an iPhone, your app is notified of a selection when a user taps their fingers. The user gets the visual feedback they need, and you receive the interactions you expect without needing to know where the user looks before they tap.

Sharing and collaboration are a central part of the Vision Pro experience. Just like on macOS, Vision Pro users can share any app window with others on a FaceTime call. And with SharePlay, next generation shared experiences become possible. When your app adopts SharePlay with the GroupActivities API, the sharing control over each window adds an option that launches your app for everyone on the call. This allows them to participate in a truly shared experience, together in real time. You decide how your app can be used collaboratively, just as you have for SharePlay on our other platforms.

A Vision Pro user on a FaceTime call appears to other participants as a persona, a natural representation created with advanced machine learning techniques that dynamically matches their facial and hand movements. Personas appear to other Vision Pro users with three-dimensional volume and depth.

But we want to take FaceTime to the next level on Vision Pro and empower users anywhere to interact like they are actually in a room together. This experience is still in its early form and we're excited to share it with you here for the first time. Spatial Personas allow Vision Pro users to break out of the familiar FaceTime tile and feel more present, like they are gathered in the same physical space. visionOS ensures that users and apps have a shared and consistent spatial context.

Your apps can take advantage of this new context with SharePlay. Now, beyond just synchronizing changes in your app between users, you can respond to a user's physical actions, so that standing together at a Freeform board or moving pieces in a tabletop game feels natural. We want your apps to be a part of this incredible new experience and your feedback will be important as it continues to develop.

That's why we're making a developer preview of Spatial Personas available to you later this year. These are just a few more examples of the rich, extensive technologies you can take advantage of on visionOS.

Now, here's Linda to talk about where you'll go from here.

Linda Dong

Now that you've been introduced to Apple Vision Pro and visionOS, what's next? Your development journey starts today. We encourage you to dive in and learn as much as you can as you begin designing, developing, and testing for Vision Pro. The visionOS SDK, along with an updated Xcode and Simulator, and Reality Composer Pro, will be available later this month, so you can start working on your ideas. You'll also have access to extensive technical documentation, new design kits and tools, and an updated Human Interface Guidelines for visionOS.

In addition to the Vision Pro simulator, we also have options for you to see how your app works on Vision Pro. This summer, we're launching the Apple Vision Pro Developer Labs, where you can test your app on Vision Pro hardware. You'll have hands-on support and an ability to preview your experience before Vision Pro is released to customers. These labs will be located in several locations around the

world, including our Developer Center here in Cupertino. And no matter where you are, you'll be able to submit a request for us to evaluate your app's compatibility with Vision Pro. We'll install your app and share any issues we may find.

To get further updates on these exciting opportunities, visit the developer website. And of course, Vision Pro will launch with a brand new app store, so people can discover and download the incredible apps you create.

We invite you to share your ideas for this platform and we'll continue to support you as you bring them to life.

Now back to Darin.

Darin Adler

What an exciting year, an incredible opportunity with visionOS, so many thoughtful additions to each platform, like more places to surface your app with WidgetKit, an expansion of Continuity Camera, and a new design for watchOS, plus major enhancements to Swift, with macros and Swift data, and a big upgrade to Xcode Previews.

This year's developments across languages, frameworks, tools, and services represent our deep commitment to helping you create unique experiences for billions of devices across our platforms. These platforms span an increasingly diverse range of products with a level of consistency and integration that make a difference to you as a developer and to users.

We successfully scratched the surface of what's new in our platforms, but to go further, we have 175 in-depth video sessions, 40 for visionOS alone, along with over 200 labs staffed by Apple experts, to answer your burning questions as you build great apps. There has never been a better time to be an Apple developer, and I have no doubt what you create will astound and delight your users.

Have a fantastic WWDC.