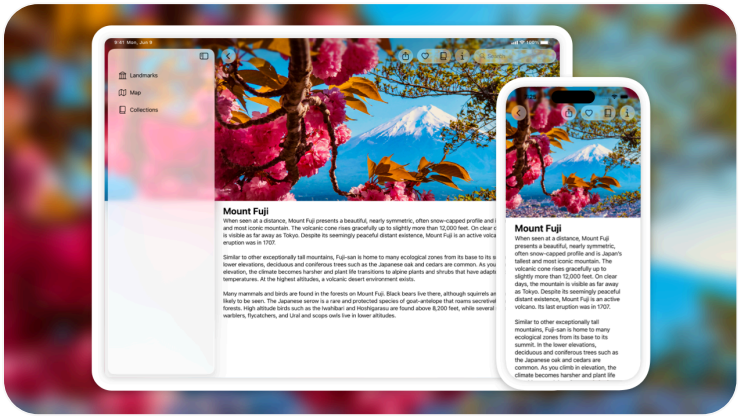
Sample Code Library

Enhance and expand your knowledge of Apple technologies by exploring the full library of sample code projects.

Featured at WWDC25

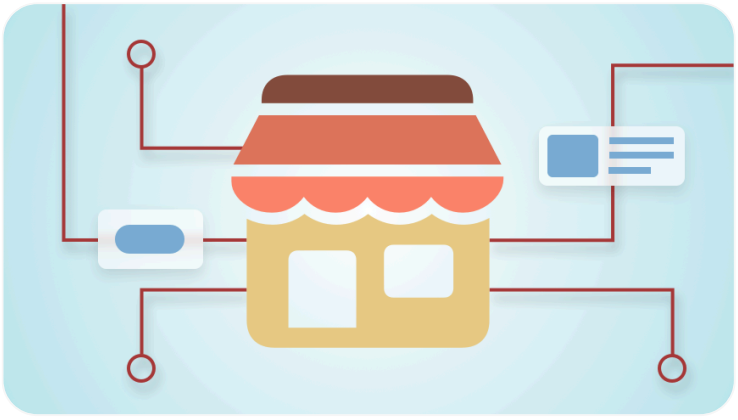
Explore samples that highlight new APIs featured at this year's conference.



Landmarks: Building an app with Liquid Glass

Enhance your app experience with system-provided and custom Liquid Glass.

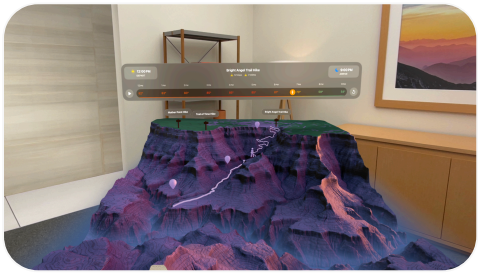
View sample code ›



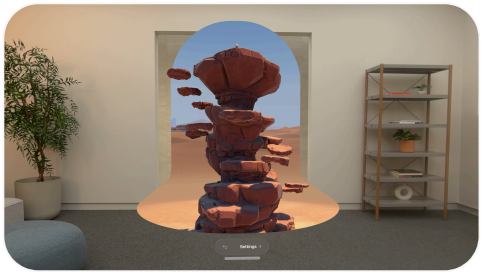
Understanding StoreKit workflows

Implement an in-app store with several product types, using StoreKit views.

View sample code ›



Canyon Crosser: Building a volumetric hike-planning app

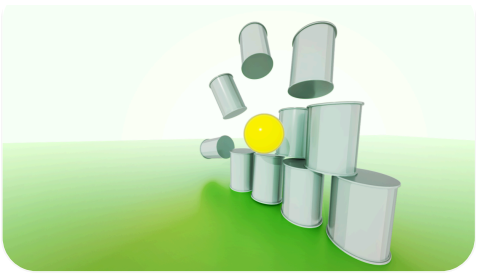


Petite Asteroids: Building a volumetric visionOS game

Synchronizing group gameplay with TabletopKit

Adding intelligent app features with generative models

Building rich SwiftUI text experiences



Tracking accessories in volumetric windows

Topics

WWDC25

{} Adding intelligent app features with generative models

Build robust apps with guided generation and tool calling by adopting the Foundation Models framework.

{} Adopting App Intents to support system experiences

Create app intents and entities to incorporate system experiences such as Spotlight, visual intelligence, and Shortcuts.

{} Adopting SwiftData for a Core Data app

Persist data in your app intuitively with the Swift native persistence framework.

{} Authoring Apple Immersive Video

Prepare and package immersive video content for delivery.

{} AVCam: Building a camera app

Capture photos and record video using the front and rear iPhone and iPad cameras.

{} Bringing advanced speech-to-text capabilities to your app

Learn how to incorporate live speech-to-text transcription into your app with SpeechAnalyzer.

{} Building a workout app for iPhone and iPad

Start a workout in iOS, control it from the Lock Screen with App Intents, and present the workout status with Live Activities.

{} Building peer-to-peer apps

Communicate with nearby devices over a secure, high-throughput, low-latency connection by using Wi-Fi Aware.

{} Building rich SwiftUI text experiences

Build an editor for formatted text using SwiftUI text editor views and attributed strings.

{} Canyon Crosser: Building a volumetric hike-planning app

Create a hike planning app using SwiftUI and RealityKit.

{} Capturing Cinematic video

Capture video with an adjustable depth of field and focus points.

{} Capturing Spatial Audio in your iOS app

Enhance your app's audio recording capabilities by supporting Spatial Audio capture.

{} Code-along: Elevating an app with Swift concurrency

Code along with the WWDC presenter to elevate a SwiftUI app with Swift concurrency.

{} Converting projected video to Apple Projected Media Profile

Convert content with equirectangular or half-equirectangular projection to APMP.

{} Creating a seamless multiview playback experience

Build advanced multiview playback experiences with the AVFoundation and AVRouting frameworks.

{} Drawing a triangle with Metal 4

Render a colorful, rotating 2D triangle by running draw commands with a render pipeline on a GPU.

{} Editing Spatial Audio with an audio mix

Add Spatial Audio editing capabilities with the Audio Mix API in the Cinematic framework.

{} Enhancing your app's privacy and security with quantum-secure workflows

Use quantum-secure cryptography to protect your app from quantum attacks.

{} Enhancing your app with machine learning-based video effects

Add powerful effects to your videos using the VideoToolbox VTFrameProcessor API.

{} Enhancing your custom text engine with Writing Tools

Add Writing Tools support to your custom text engine to enhance the text editing experience.

{} Filtering traffic by URL

Perform fast and robust filtering of full URLs by managing URL filtering configurations.

{} Generate dynamic game content with guided generation and tools

Make gameplay more lively with AI generated dialog and encounters personalized to the player.

{} Implementing a background delivery extension

Receive up-to-date financial data in your app and its extensions by adding a background delivery extension.

{} Implementing a store in your app using the StoreKit API

Offer In-App Purchases and manage entitlements using signed transactions and status information.

{} Landmarks: Applying a background extension effect

Configure an image to blur and extend under a sidebar or inspector panel.

{} Landmarks: Building an app with Liquid Glass

Enhance your app experience with system-provided and custom Liquid Glass.

{} Landmarks: Displaying custom activity badges

Provide people with a way to mark their adventures by displaying animated custom activity badges.

{} Landmarks: Extending horizontal scrolling under a sidebar or inspector

Improve your horizontal scrollbar's appearance by extending it under a sidebar or inspector.

{} Landmarks: Refining the system provided Liquid Glass effect in toolbars

Organize toolbars into related groupings to improve their appearance and utility.

{} Language Introspector

Converts data into human-readable text using formatters and locales.

{} Localizing Landmarks

Add localizations to the Landmarks sample code project.

{} Logging symptoms associated with a medication

Fetch medications and dose events from the HealthKit store, and create symptom samples to associate with them.

{} Managing location-based reminders

Access reminders set up with geofence-enabled alarms on a person's calendars.

{} Optimizing home electricity usage

Shift electric vehicle charging schedules to times when the grid is cleaner and potentially less expensive.

{} Performing fast account creation with passkeys

Allow people to quickly create an account with passkeys and associated domains.

{} Petite Asteroids: Building a volumetric visionOS game

Use the latest RealityKit APIs to create a beautiful video game for visionOS.

{} Playing immersive media with AVKit

Adopt the system playback interface to provide an immersive video watching experience.

{} Playing immersive media with RealityKit

Create an immersive video playback experience with RealityKit.

{} Presenting images in RealityKit

Create and display spatial scenes in RealityKit

{} Processing a texture in a compute function

Create textures by running copy and dispatch commands in a compute pass on a GPU.

{} Recognizing tables within a document

Scan a document containing a contact table and extract the content within the table in a formatted way.

{} Rendering hover effects in Metal immersive apps

Change the appearance of a rendered onscreen element when a player gazes at it.

{} Scheduling an alarm with AlarmKit

Create prominent alerts at specified dates for your iOS app.

{} Searching, displaying, and navigating to places

Convert place information between coordinates and user-friendly place names, get cycling directions, and conveniently display formatted addresses.

{} Supporting real-time ML inference on the CPU

Add real-time digital signal processing to apps like Logic Pro X and GarageBand with the BNNS Graph API.

{} Synchronizing group gameplay with TabletopKit

Maintain game state across multiple players in a race to capture all the coins.

{} Tracking accessories in volumetric windows

Translate the position and velocity of tracked handheld accessories to throw virtual balls at a stack of cans.

{} Tracking a handheld accessory as a virtual sculpting tool

Use a tracked accessory with Apple Vision Pro to create a virtual sculpture.

{} Understanding StoreKit workflows

Implement an in-app store with several product types, using StoreKit views.

Accelerate

{} Adding a bokeh effect to images

Simulate a bokeh effect by applying dilation.

{} Adjusting saturation and applying tone mapping

Convert an RGB image to discrete luminance and chrominance channels, and apply color and contrast treatments.

{} Adjusting the brightness and contrast of an image

Use a gamma function to apply a linear or exponential curve.

{} Adjusting the hue of an image

Convert an image to L*a*b* color space and apply hue adjustment.

{} Applying biquadratic filters to a music loop

Change the frequency response of an audio signal using a cascaded biquadratic filter.

{} Applying tone curve adjustments to images

Use the vImage library's polynomial transform to apply tone curve adjustments to images.

{} Applying transformations to selected colors in an image

Desaturate a range of colors in an image with a multidimensional lookup table.

{} Applying vImage operations to video sample buffers

Use the vImage convert-any-to-any functionality to perform real-time image processing of video frames streamed from your device's camera.

{} Blurring an image

Filter an image by convolving it with custom and high-speed kernels.

{} Calculating the dominant colors in an image

Find the main colors in an image by implementing k-means clustering using the Accelerate framework.

{} Compressing and decompressing files with stream compression

Perform compression for all files and decompression for files with supported extension types.

{} Compressing an image using linear algebra

Reduce the storage size of an image using singular value decomposition (SVD).

{} Converting color images to grayscale

Convert an RGB image to grayscale using matrix multiplication.

{} Converting luminance and chrominance planes to an ARGB image

Create a displayable ARGB image using the luminance and chrominance information from your device's camera.

{} Creating an audio unit extension using the vDSP library

Add biquadratic filter audio-effect processing to apps like Logic Pro X and GarageBand with the Accelerate framework.

{} Cropping to the subject in a chroma-keyed image

Convert a chroma-key color to alpha values and trim transparent pixels using Accelerate.

{} Equalizing audio with discrete cosine transforms (DCTs)

Change the frequency response of an audio signal by manipulating frequency-domain data.

{} Finding the sharpest image in a sequence of captured images

Share image data between vDSP and vImage to compute the sharpest image from a bracketed photo sequence.

{} Halftone descreening with 2D fast Fourier transform

Reduce or remove periodic artifacts from images.

{} Improving the quality of quantized images with dithering

Apply dithering to simulate colors that are unavailable in reduced bit depths.

{} Integrating vImage pixel buffers into a Core Image workflow

Share image data between Core Video pixel buffers and vImage buffers to integrate vImage operations into a Core Image workflow.

{} Reducing artifacts with custom resampling filters

Implement custom linear interpolation to prevent the ringing effects associated with scaling an image with the default Lanczos algorithm.

{} Rotating a cube by transforming its vertices

Rotate a cube through a series of keyframes using quaternion interpolation to transition between them.

{} Sharing texture data between the Model I/O framework and the vImage library

Use Model I/O and vImage to composite a photograph over a computer-generated sky.

{} Signal extraction from noise

Use Accelerate's discrete cosine transform to remove noise from a signal.

{} Solving systems of linear equations with LAPACK

Select the optimal LAPACK routine to solve a system of linear equations.

{} Specifying histograms with vImage

Calculate the histogram of one image, and apply it to a second image.

{} Training a neural network to recognize digits

Build a simple neural network and train it to recognize randomly generated numbers.

{} Using vImage pixel buffers to generate video effects

Render real-time video effects with the vImage Pixel Buffer.

{} Visualizing sound as an audio spectrogram

Share image data between vDSP and vImage to visualize audio that a device microphone captures.

Accessibility

{} Accessibility design for Mac Catalyst

Improve navigation in your app by using keyboard shortcuts and accessibility containers.

{} Creating Accessible Views

Make your app accessible to everyone by applying accessibility modifiers to your SwiftUI views.

{} Delivering an exceptional accessibility experience

Make improvements to your app's interaction model to support assistive technologies such as VoiceOver.

{} Enhancing the accessibility of your SwiftUI app

Support advancements in SwiftUI accessibility to make your app accessible to everyone.

{} Integrating accessibility into your app

Make your app more accessible to users with disabilities by adding accessibility features.

{} Responding to changes in the flashing lights setting

Adjust your UI when a person chooses to dim flashing lights on their Apple device.

{} Translating text within your app

Display simple system translations and create custom translation experiences.

{} WWDC21 Challenge: Large Text Challenge

Design for large text sizes by modifying the user interface.

{} WWDC21 Challenge: Speech Synthesizer Simulator

Simulate a conversation using speech synthesis.

{} WWDC21 Challenge: VoiceOver Maze

Navigate to the end of a dark maze using VoiceOver as your guide.

{} WWDC22 Challenge: Learn Switch Control through gaming

Play a card-matching game using Switch Control.

App frameworks

{} Building a Localized Food-Ordering App

Format, style, and localize your app's text for use in multiple languages with string formatting, attributed strings, and automatic grammar agreement.

{} Building a resumable upload server with SwiftNIO

Support HTTP resumable upload protocol in SwiftNIO by translating resumable uploads to regular uploads.

{} Continuing User Activities with Handoff

Define and manage which of your app's activities can be continued between devices.

{} Displaying Human-Friendly Content

Convert data into readable strings or Swift objects using formatters.

{} Fruta: Building a feature-rich app with SwiftUI

Create a shared codebase to build a multiplatform app that offers widgets and an App Clip.

{} Increasing App Usage with Suggestions Based on User Activities

Provide a continuous user experience by capturing information from your app and displaying this information as proactive suggestions across the system.

{} Interacting with App Clip Codes in AR

Display content and provide services in an AR experience with App Clip Codes.

{} Synchronizing App Preferences with iCloud

Store app preferences in iCloud and share them among instances of your app running on a user's connected devices.

{} Using JSON with Custom Types

Encode and decode JSON data, regardless of its structure, using Swift's JSON support.

App Intents and SiriKit

{} Accelerating app interactions with App Intents

Enable people to use your app's features quickly through Siri, Spotlight, and Shortcuts.

{} Adding Shortcuts for Wind Down

Reveal your app's shortcuts inside the Health app.

{} Booking Rides with SiriKit

Add Intents extensions to your app to handle requests to book rides using Siri and Maps.

{} Defining your app's Focus filter

Customize your app's behavior to reflect the device's current Focus.

{} Handling Payment Requests with SiriKit

Add an Intent Extension to your app to handle money transfer requests with Siri.

{} Handling Workout Requests with SiriKit

Add an Intent Extension to your app that handles requests to control workouts with Siri.

{} Integrating Your App with Siri Event Suggestions

Donate reservations and provide quick access to event details throughout the system.

{} Making your app's functionality available to Siri

Add app intent schemas to your app so Siri can complete requests, and integrate your app with Apple Intelligence, Spotlight, and other system experiences.

{} Managing Audio with SiriKit

Control audio playback and handle requests to add media using SiriKit Media Intents.

{} Providing Hands-Free App Control with Intents

Resolve, confirm, and handle intents without an extension.

{} Soup Chef: Accelerating App Interactions with Shortcuts

Make it easy for people to use Siri with your app by providing shortcuts to your app's actions.

{} Soup Chef with App Intents: Migrating custom intents

Integrating App Intents to provide your app's actions to Siri and Shortcuts.

AppKit

{} Add Functionality to Finder with Action Extensions

Implement Action Extensions to provide quick access to commonly used features of your app.

{} Creating and Customizing the Touch Bar

Adopt Touch Bar support by displaying interactive content and controls for your macOS apps.

{} Developing a Document-Based App

Write an app that creates, manages, edits, and saves text documents.

{} Integrating a Toolbar and Touch Bar into Your App

Provide users quick access to your app's features from a toolbar and corresponding Touch Bar.

{} Navigating Hierarchical Data Using Outline and Split Views

Build a structured user interface that simplifies navigation in your app.

{} Organize Your User Interface with a Stack View

Group individual views in your app's user interface into a scrollable stack view.

{} Supporting Collection View Drag and Drop Through File Promises

Share data between macOS apps during drag and drop by using an item provider.

{} Supporting Drag and Drop Through File Promises

Receive and provide file promises to support dragged app files and pasteboard operations.

{} Supporting Table View Drag and Drop Through File Promises

Share data between macOS apps during drag and drop by using an item provider.

App services

{} Accessing a person's contact data using Contacts and ContactsUI

Allow people to grant your app access to contact data by adding the Contact access button and Contact access picker to your app.

{} Accessing Calendar using EventKit and EventKitUI

Choose and implement the appropriate Calendar access level in your app.

{} Adopting SwiftData for a Core Data app

Persist data in your app intuitively with the Swift native persistence framework.

{} Build an Educational Assessment App

Ensure the academic integrity of your assessment app by using Automatic Assessment Configuration.

{} Build Mail App Extensions

Create app extensions that block content, perform message and composing actions, and help message security.

{} Checking IDs with the Verifier API

Read and verify mobile driver's license information without any additional hardware.

{} Configuring a home automation device

Give users a familiar experience when they manage HomeKit accessories.

{} Configuring the PencilKit tool picker

Incorporate a custom PencilKit tool picker with a variety of system and custom tools into a drawing app.

{} Creating a data visualization dashboard with Swift Charts

Visualize an entire data collection efficiently by instantiating a single vectorized plot in Swift Charts.

{} Creating a Sticker App with a Custom Layout

Expand on the Messages sticker app template to create an app with a customized user interface.

{} Customizing Scribble with Interactions

Enable writing on a non-text-input view by adding interactions.

{} Downloading essential assets in the background

Fetch the assets your app requires before its first launch using an app extension and the Background Assets framework.

{} Drawing with PencilKit

Add expressive, low-latency drawing to your app using PencilKit.

{} Example Order Packages

Edit, build, and add example order packages to Wallet.

{} Fetching weather forecasts with WeatherKit

Request and display weather data for destination airports in a flight-planning app.

{} Handling Communication Notifications and Focus Status Updates

Create a richer calling and messaging experience in your app by implementing communication notifications and Focus status updates.

{} Handling Different Data Types in Core Data

Create, store, and present records for a variety of data types.

{} Highlighting app features with TipKit

Bring attention to new features in your app by using tips.

{} IceCreamBuilder: Building an iMessage Extension

Allow users to collaborate on the design of ice cream sundae stickers.

{} Implementing Alert Push Notifications

Add visible alert notifications to your app by using the UserNotifications framework.

{} Implementing Background Push Notifications

Add background notifications to your app by using the UserNotifications framework.

{} Implementing Wallet Extensions

Support adding an issued card to Apple Pay from directly within Apple Wallet using Wallet Extensions.

{} Incorporating ClassKit into an Educational App

Walk through the process of setting up assignments and recording student progress.

{} Inspecting, Modifying, and Constructing PencilKit Drawings

Score users' ability to match PencilKit drawings generated from text, by accessing the strokes and points inside PencilKit drawings.

{} Integrating the Apple Maps Server API into Java server applications

Streamline your app's API by moving georelated searches from inside your app to your server.

{} Interacting with a home automation network

Find all the automation accessories in the primary home and control their state.

{} Linking Data Between Two Core Data Stores

Organize data in two different stores and implement a link between them.

{} Loading and Displaying a Large Data Feed

Consume data in the background, and lower memory use by batching imports and preventing duplicate records.

{} Managing location-based reminders

Access reminders set up with geofence-enabled alarms on a person's calendars.

{} Offering Apple Pay in Your App

Collect payments with iPhone and Apple Watch using Apple Pay.

{} Refreshing and Maintaining Your App Using Background Tasks

Use scheduled background tasks for refreshing your app content and for performing maintenance.

{} Retrieve Power and Performance Metrics and Log Insights

Use the App Store Connect API to collect and parse diagnostic logs and metrics for your apps.

{} Sharing CloudKit Data with Other iCloud Users

Create and share private CloudKit data with other users by implementing the sharing UI.

{} Sharing Core Data objects between iCloud users

Use Core Data and CloudKit to synchronize data between devices of an iCloud user and share data between different iCloud users.

{} Showcase App Data in Spotlight

Index app data so users can find it by using Spotlight search.

{} Synchronizing a local store to the cloud

Share data between a user's devices and other iCloud users.

{} Synchronizing files using file provider extensions

Make remote files available in macOS and iOS, and synchronize their states by using file provider extensions.

{} Updating your app package installer to use the new Service Management API

Learn about the Service Management API with a GUI-less agent app.

{} Uploading App Previews

Upload your app previews, including video files, to App Store Connect by using the asset upload APIs.

{} Visualizing your app's data

Build complex and interactive charts using Swift Charts.

{} VoIP calling with CallKit

Use the CallKit framework to integrate native VoIP calling.

ARKit

{} Adding realistic reflections to an AR experience

Use ARKit to generate environment probe textures from camera imagery and render reflective virtual objects.

{} Building local experiences with room tracking

Use room tracking in visionOS to provide custom interactions with physical spaces.

{} Capturing Body Motion in 3D

Track a person in the physical environment and visualize their motion by applying the same body movements to a virtual character.

{} Combining user face-tracking and world tracking

Track the user's face in an app that displays an AR experience with the rear camera.

{} Creating a collaborative session

Enable nearby devices to share an AR experience by using a peer-to-peer multiuser strategy.

{} Creating a fog effect using scene depth

Apply virtual fog to the physical environment.

{} Creating a multiuser AR experience

Enable nearby devices to share an AR experience by using a host-guest multiuser strategy.

{} Creating an immersive ar experience with audio

Use sound effects and environmental sound layers to create an engaging AR experience.

{} Creating screen annotations for objects in an AR experience

Annotate an AR experience with virtual sticky notes that you display onscreen over real and virtual objects.

{} Detecting Images in an AR Experience

React to known 2D images in the user's environment, and use their positions to place AR content.

{} Displaying a point cloud using scene depth

Present a visualization of the physical environment by placing points based a scene's depth data.

{} Effecting People Occlusion in Custom Renderers

Occlude your app's virtual content where ARKit recognizes people in the camera feed by using matte generator.

{} Occluding virtual content with people

Cover your app's virtual content with people that ARKit perceives in the camera feed.

{} Placing objects and handling 3D interaction

Place virtual content at tracked, real-world locations, and enable the user to interact with virtual content by using gestures.

{} Recognizing and Labeling Arbitrary Objects

Create anchors that track objects you recognize in the camera feed, using a custom optical-recognition algorithm.

{} Saving and loading world data

Serialize a world-tracking session to resume it later on.

{} Scanning and Detecting 3D Objects

Record spatial features of real-world objects, then use the results to find those objects in the user's environment and trigger AR content.

{} Streaming an AR experience

Control an AR experience remotely by transferring sensor and user input over the network.

{} Tracking and altering images

Create images from rectangular shapes found in the user's environment, and augment their appearance.

{} Tracking and visualizing faces

Detect faces in a front-camera AR experience, overlay virtual content, and animate facial expressions in real-time.

{} Tracking and visualizing planes

Detect surfaces in the physical environment and visualize their shape and location in 3D space.

{} Tracking geographic locations in AR

Track specific geographic areas of interest and render them in an AR experience.

{} Visualizing and interacting with a reconstructed scene

Estimate the shape of the physical environment using a polygonal mesh.

Audio and music

{} Adding synthesized speech to calls

Provide a more accessible experience by adding your app's audio to a call.

{} Becoming a now playable app

Ensure your app is eligible to become the Now Playing app by adopting best practices for providing Now Playing info and registering for remote command center actions.

{} Building a Custom Catalog and Matching Audio

Display lesson content that's synchronized to a learning video by matching the audio to a custom reference signature and associated metadata.

{} Building an Audio Server Plug-in and Driver Extension

Create a plug-in and driver extension to support an audio device in macOS.

{} Building a signal generator

Generate audio signals using an audio source node and a custom render callback.

{} Capturing stereo audio from built-In microphones

Configure an iOS device's built-in microphones to add stereo recording capabilities to your app.

{} Capturing system audio with Core Audio taps

Use a Core Audio tap to capture outgoing audio from a process or group of processes.

{} Classifying Live Audio Input with a Built-in Sound Classifier

Detect and identify hundreds of sounds by using a trained classifier.

{} Creating a custom speech synthesizer

Use your custom voices to synthesize speech by building a speech synthesis provider.

{} Creating an audio device driver

Implement a configurable audio input source as a driver extension that runs in user space in macOS and iPadOS.

{} Creating an Audio Server Driver Plug-in

Build a virtual audio device by creating a custom driver plug-in.

{} Creating custom audio effects

Add custom audio-effect processing to apps like Logic Pro X and GarageBand by creating Audio Unit (AU) plug-ins.

{} Delivering Rich App Experiences with Haptics

Enhance your app's experience by incorporating haptic and sound feedback into key interactive moments.

{} Discovering a third-party media-streaming device

Build an extension that streams media to a server app in iOS or macOS.

{} Encoding and decoding audio

Convert audio formats to efficiently manage data and quality.

{} Explore more content with MusicKit

Track your outdoor runs with access to the Apple Music catalog, personal recommendations, and your own personal music library.

{} Generating spatial audio from a multichannel audio stream

Convert 8-channel audio to 2-channel spatial audio by using a spatial mixer audio unit.

{} Getting motion-activity data from headphones

Configure your app to listen for motion-activity changes from headphones.

{} Incorporating Audio Effects and Instruments

Add custom audio processing and MIDI instruments to your app by hosting Audio Unit (AU) plug-ins.

{} Incorporating MIDI 2 into your apps

Add precision and improve musical control for your MIDI apps.

{} Integrating CarPlay with Your Music App

Configure your music app to work with CarPlay by displaying a custom UI.

{} Integrating CarPlay with Your Navigation App

Configure your navigation app to work with CarPlay by displaying your custom map and directions.

{} Integrating CarPlay with your quick-ordering app

Configure your food-ordering app to work with CarPlay.

{} Performing offline audio processing

Add offline audio processing features to your app by enabling offline manual rendering mode.

{} Playing a Custom Haptic Pattern from a File

Sample predesigned Apple Haptic Audio Pattern files, and learn how to play your own.

{} Playing Collision-Based Haptic Patterns

Play a custom haptic pattern whose strength depends on an object's collision speed.

{} Playing custom audio with your own player

Construct an audio player to play your custom audio data, and optionally take advantage of the advanced features of AirPlay 2.

{} Playing Haptics on Game Controllers

Add haptic feedback to supported game controllers by using Core Haptics.

{} Recognizing speech in live audio

Perform speech recognition on audio coming from the microphone of an iOS device.

{} ShazamKit Dance Finder with Managed Session

Find a video of dance moves for a specific song by matching the audio to a custom catalog, and show a history of recognized songs.

{} Transferring Data Between Bluetooth Low Energy Devices

Create a Bluetooth low energy central and peripheral device, and allow them to discover each other and exchange data.

{} Updating Continuous and Transient Haptic Parameters in Real Time

Generate continuous and transient haptic patterns in response to user touch.

{} Using Core Bluetooth Classic

Discover and communicate with a Bluetooth Classic device by using the Core Bluetooth APIs.

{} Using MusicKit to Integrate with Apple Music

Find an album in Apple Music that corresponds to a CD in a user's collection, and present the information for the album.

{} Using voice processing

Add voice-processing capabilities to your app by using audio engine.

Authentication

{} Accessing Keychain Items with Face ID or Touch ID

Protect a keychain item with biometric authentication.

{} Connecting to a service with passkeys

Allow users to sign in to a service without typing a password.

{} Implementing User Authentication with Sign in with Apple

Provide a way for users of your app to set up an account and start using your services.

{} Logging a User into Your App with Face ID or Touch ID

Supplement your own authentication scheme with biometric authentication, making it easy for users to access sensitive parts of your app.

{} Simplifying User Authentication in a tvOS App

Build a fluid sign-in experience for your tvOS apps using AuthenticationServices.

AVFoundation

{} AVCamBarcode: detecting barcodes and faces

Identify machine readable codes or faces by using the camera.

{} AVCam: Building a camera app

Capture photos and record video using the front and rear iPhone and iPad cameras.

{} AVCamFilter: Applying filters to a capture stream

Render a capture stream with rose-colored filtering and depth effects.

{} AVMultiCamPiP: Capturing from Multiple Cameras

Simultaneously record the output from the front and back cameras into a single movie file by using a multi-camera capture session.

{} Capturing consistent color images

Add the power of a photography studio and lighting rig to your app with the new Constant Color API.

{} Capturing depth using the LiDAR camera

Access the LiDAR camera on supporting devices to capture precise depth data.

{} Converting side-by-side 3D video to multiview HEVC and spatial video

Create video content for visionOS by converting an existing 3D HEVC file to a multiview HEVC format, optionally adding spatial metadata to create a spatial video.

{} Debugging AVFoundation audio mixes, compositions, and video compositions

Resolve common problems when creating compositions, video compositions, and audio mixes.

{} Editing and playing HDR video

Support high-dynamic-range (HDR) video content in your app by using the HDR editing and playback capabilities of AVFoundation.

{} Enhancing live video by leveraging TrueDepth camera data

Apply your own background to a live capture feed streamed from the front-facing TrueDepth camera.

{} Integrating AirPlay for long-form video apps

Integrate AirPlay features and implement a dedicated external playback experience by preparing the routing system for long-form video playback.

{} Processing spatial video with a custom video compositor

Create a custom video compositor to edit spatial video for playback and export.

{} Providing an integrated view of your timeline when playing HLS interstitials

Go beyond simple ad insertion with point and fill occupancy HLS interstitials.

{} Reading multiview 3D video files

Render single images for the left eye and right eye from a multiview High Efficiency Video Coding format file by reading individual video frames.

{} Streaming depth data from the TrueDepth camera

Visualize depth data in 2D and 3D from the TrueDepth camera.

{} Supporting Continuity Camera in your macOS app

Enable high-quality photo and video capture by using an iPhone camera as an external capture device.

{} Supporting coordinated media playback

Create synchronized media experiences that enable users to watch and listen across devices.

{} Supporting remote interactions in tvOS

Set up your app to support remote commands and events in a variety of scenarios by using the relevant approach.

{} Using AVFoundation to play and persist HTTP live streams

Play HTTP Live Streams and persist streams on disk for offline playback using AVFoundation.

{} Using HEVC video with alpha

Play, write, and export HEVC video with an alpha channel to add overlay effects to your video processing.

{} Writing fragmented MPEG-4 files for HTTP Live Streaming

Create an HTTP Live Streaming presentation by turning a movie file into a sequence of fragmented MPEG-4 files.

CoreML and CreateML

{} Classifying Images with Vision and Core ML

Crop and scale photos using the Vision framework and classify them with a Core ML model.

{} Counting human body action repetitions in a live video feed

Use Create ML Components to analyze a series of video frames and count a person's repetitive or periodic body movements.

{} Creating a model from tabular data

Train a machine learning model by using Core ML to import and manage tabular data.

{} Detecting human actions in a live video feed

Identify body movements by sending a person's pose data from a series of video frames to an action-classification model.

{} Detecting human body poses in an image

Locate people and the stance of their bodies by analyzing an image with a PoseNet model.

{} Finding answers to questions in a text document

Locate relevant passages in a document by asking the Bidirectional Encoder Representations from Transformers (BERT) model a question.

{} Integrating a Core ML Model into Your App

Add a simple model to an app, pass input data to the model, and process the model's predictions.

{} Personalizing a Model with On-Device Updates

Modify an updatable Core ML model by running an update task with labeled data.

{} Understanding a Dice Roll with Vision and Object Detection

Detect dice position and values shown in a camera frame, and determine the end of a roll by leveraging a dice detection model.

{} Using Core ML for semantic image segmentation

Identify multiple objects in an image by using the DEtection TRansformer image-segmentation model.

Developer tools

{} Autosizing views for localization in iOS

Add auto layout constraints to your app to achieve localizable views.

{} Configuring your app to use alternate app icons

Add alternate app icons to your app, and let people choose which icon to display.

{} Creating custom modelers for intelligent instruments

Create Custom Modelers with the CLIPS language and learn how the embedded rules engine works.

{} Localization-friendly layouts in macOS

This project demonstrates localization-friendly auto layout constraints.

{} Providing an edge-to-edge, full-screen experience in your iPad app running on a Mac

Take advantage of the true native resolution of a Mac display when running your iPad app in full-screen mode on a Mac.

{} Providing touch gesture equivalents using Touch Alternatives

Enable Touch Alternatives to provide keyboard, mouse, and trackpad equivalents to your iOS app when it runs on a Mac with Apple silicon.

{} SlothCreator: Building DocC documentation in Xcode

Build DocC documentation for a Swift package that contains a DocC Catalog.

Games

{} Adding Recurring Leaderboards to Your Game

Encourage competition in your games by adding leaderboards that have a duration and repeat.

{} Creating real-time games

Develop games where multiple players interact in real time.

{} Creating tabletop games

Develop a spatial board game where multiple players interact with pieces on a table.

{} Creating turn-based games

Develop games where multiple players take turns and can exchange data while waiting for their turn.

{} Implementing playing card overlap and physical characteristics

Add interactive card game behavior for a pile of playing cards with physically realistic stacking and overlapping.

{} Interacting with virtual content blended with passthrough

Present a mixed immersion style space to draw content in a person's surroundings, and choose how upper limbs appear with respect to rendered content.

{} Simulating dice rolls as a component for your game

Create a physically realistic dice game by adding interactive rolling and scoring.

{} Supporting Game Controllers

Support a physical controller or add a virtual controller to enhance how people interact with your game through haptics, lighting, and motion sensing.

Graphics

{} Building Widgets Using WidgetKit and SwiftUI

Create widgets to show your app's content on the Home screen, with custom intents for user-customizable settings.

{} Create a 3D model of an interior room by guiding the user through an AR experience

Highlight physical structures and display text that guides a user to scan the shape of their physical environment using a framework-provided view.

{} Custom Graphics

Demonstrates adding a watermark to a PDF page.

{} Emoji Rangers: Supporting Live Activities, interactivity, and animations

Offer Live Activities, controls, animate data updates, and add interactivity to widgets.

{} Generating an animation with a Core Image Render Destination

Animate a filtered image to a Metal view in a SwiftUI app using a Core Image Render Destination.

{} Merging multiple scans into a single structure

Export a 3D model that consists of multiple rooms captured in the same physical vicinity.

{} PDF Widgets

Demonstrates adding widgets—interactive form elements—to a PDF document.

{} Postprocessing a Scene With Custom Symbols

Create visual effects in a scene by defining a rendering technique with custom symbols.

{} Providing custom models for captured rooms and structure exports

Enhance the look of an exported 3D model by substituting object bounding boxes with detailed 3D renditions.

{} Schema definitions for third-party DCCs

Update your local USD library to add interactive and augmented reality features.

{} Writing spatial photos

Create spatial photos for visionOS by packaging a pair of left- and right-eye images as a stereo HEIC file with related spatial metadata.

Health

{} Accessing a User's Clinical Records

Request authorization to query HealthKit for a user's clinical records and display them in your app.

{} Accessing Data from a SMART Health Card

Query for and validate a verifiable clinical record.

{} Build a workout app for Apple Watch

Create your own workout app, quickly and easily, with HealthKit and SwiftUI.

{} Building a multidevice workout app

Mirror a workout from a watchOS app to its companion iOS app, and perform bidirectional communication between them.

{} Building an App to Notify Users of COVID-19 Exposure

Inform people when they may have been exposed to COVID-19.

{} Creating a Mobility Health App

Create a health app that allows a clinical care team to send and receive mobility data.

{} Reading and Writing HealthKit Series Data

Share and read heartbeat and quantity series data using series builders and queries.

{} Visualizing HealthKit State of Mind in visionOS

Incorporate HealthKit State of Mind into your app and visualize the data in visionOS.

Location and MapKit

{} Adopting live updates in Core Location

Simplify location delivery using asynchronous events in Swift.

{} Annotating a Map with Custom Data

Annotate a map with location-specific data using default and customized annotation views and callouts.

{} Decluttering a Map with MapKit Annotation Clustering

Enhance the readability of a map by replacing overlapping annotations with a clustering annotation view.

{} Displaying an Indoor Map

Use the Indoor Mapping Data Format (IMDF) to show an indoor map with custom overlays and points of interest.

{} Displaying an updating path of a user's location history

Continually update a MapKit overlay displaying the path a user travels.

{} Displaying Indoor Maps with MapKit JS

Use the Indoor Mapping Data Format (IMDF) to show an indoor map with custom overlays and points of interest in your browser.

{} Displaying overlays on a map

Add regions of layered content to a map view.

{} Explore a location with a highly detailed map and Look Around

Display a richly detailed map, and use Look Around to experience an interactive view of landmarks.

{} Finding devices with precision

Leverage the spatial awareness of ARKit and Apple Ultra Wideband Chips in your app to guide users to a nearby device.

{} Implementing Interactions Between Users in Close Proximity

Enable devices to access relative positioning information.

{} Implementing proximity-based interactions between a phone and watch

Interact with a nearby Apple Watch by measuring its distance to a paired iPhone.

{} Implementing spatial interactions with third-party accessories

Establish a connection with a nearby accessory to receive periodic measurements of its distance from the user.

{} Interacting with nearby points of interest

Provide automatic search completions for a partial search query, search the map for relevant locations nearby, and retrieve details for selected points of interest.

{} Monitoring location changes with Core Location

Define boundaries and act on user location updates.

{} Optimizing Map Views with Filtering and Camera Constraints

Display a map that is relevant to the user by filtering points of interest and search results, and constraining the visible region.

{} Ranging for Beacons

Configure a device to act as a beacon and to detect surrounding beacons.

{} Sharing Your Location to Find a Park

Ask for location access using a customizable location button.

Metal

{} Accelerating ray tracing and motion blur using Metal

Generate ray-traced images with motion blur using GPU-based parallel processing.

{} Accelerating ray tracing using Metal

Implement ray-traced rendering using GPU-based parallel processing.

{} Achieving smooth frame rates with a Metal display link

Pace rendering with minimal input latency while providing essential information to the operating system for power-efficient rendering, thermal mitigation, and the scheduling of sustainable workloads.

{} Adding custom functions to a shader graph

Run your own graph functions on the GPU by building the function programmatically.

{} Adjusting the level of detail using Metal mesh shaders

Choose and render meshes with several levels of detail using object and mesh shaders.

{} Applying temporal antialiasing and upscaling using MetalFX

Reduce render workloads while increasing image detail with MetalFX.

{} Calculating primitive visibility using depth testing

Determine which pixels are visible in a scene by using a depth texture.

{} Capturing Metal commands programmatically

Invoke a Metal frame capture from your app, then save the resulting GPU trace to a file or view it in Xcode.

{} Control the ray tracing process using intersection queries

Explicitly enumerate a ray's intersections with acceleration structures by creating an intersection query object.

{} Creating a 3D application with hydra rendering

Build a 3D application that integrates with Hydra and USD.

{} Creating a custom Metal view

Implement a lightweight view for Metal rendering that's customized to your app's needs.

{} Creating a Metal dynamic library

Compile a library of shaders and write it to a file as a dynamically linked library.

{} Creating and sampling textures

Load image data into a texture and apply it to a quadrangle.

{} Culling occluded geometry using the visibility result buffer

Draw a scene without rendering hidden geometry by checking whether each object in the scene is visible.

{} Customizing a PyTorch operation

Implement a custom operation in PyTorch that uses Metal kernels to improve performance.

{} Customizing a TensorFlow operation

Implement a custom operation that uses Metal kernels to accelerate neural-network training performance.

{} Customizing render pass setup

Render into an offscreen texture by creating a custom render pass.

{} Customizing shaders using function pointers and stitching

Define custom shader behavior at runtime by creating functions from existing ones and preferentially linking to others in a dynamic library.

{} Drawing a triangle with Metal 4

Render a colorful, rotating 2D triangle by running draw commands with a render pipeline on a GPU.

{} Encoding argument buffers on the GPU

Use a compute pass to encode an argument buffer and access its arguments in a subsequent render pass.

{} Encoding indirect command buffers on the CPU

Reduce CPU overhead and simplify your command execution by reusing commands.

{} Encoding indirect command buffers on the GPU

Maximize CPU to GPU parallelization by generating render commands on the GPU.

{} Filtering images with MPSGraph FFT operations

Filter an image with MPSGraph fast Fourier transforms using the convolutional theorem.

{} Implementing a multistage image filter using heaps and events

Use events to synchronize access to resources allocated on a heap.

{} Implementing a multistage image filter using heaps and fences

Use fences to synchronize access to resources allocated on a heap.

{} Implementing order-independent transparency with image blocks

Draw overlapping, transparent surfaces in any order by using tile shaders and image blocks.

{} Improving edge-rendering quality with multisample antialiasing (MSAA)

Apply MSAA to enhance the rendering of edges with custom resolve options and immediate and tile-based resolve paths.

{} Loading textures and models using Metal fast resource loading

Stream texture and buffer data directly from disk into Metal resources using fast resource loading.

{} Managing groups of resources with argument buffers

Create argument buffers to organize related resources.

{} Migrating OpenGL code to Metal

Replace your app's deprecated OpenGL code with Metal.

{} Mixing Metal and OpenGL rendering in a view

Draw with Metal and OpenGL in the same view using an interoperable texture.

{} Modern rendering with Metal

Use advanced Metal features such as indirect command buffers, sparse textures, and variable rate rasterization to implement complex rendering techniques.

{} Performing calculations on a GPU

Use Metal to find GPUs and perform calculations on them.

{} Processing a texture in a compute function

Create textures by running copy and dispatch commands in a compute pass on a GPU.

{} Processing HDR images with Metal

Implement a post-processing pipeline using the latest features on Apple GPUs.

{} Reading pixel data from a drawable texture

Access texture data from the CPU by copying it to a buffer.

{} Rendering a curve primitive in a ray tracing scene

Implement ray traced rendering using GPU-based parallel processing.

{} Rendering a scene with deferred lighting in C++

Avoid expensive lighting calculations by implementing a deferred lighting renderer optimized for immediate mode and tile-based deferred renderer GPUs.

{} Rendering a scene with deferred lighting in Objective-C

Avoid expensive lighting calculations by implementing a deferred lighting renderer optimized for immediate mode and tile-based deferred renderer GPUs.

{} Rendering a scene with deferred lighting in Swift

Avoid expensive lighting calculations by implementing a deferred lighting renderer optimized for immediate mode and tile-based deferred renderer GPUs.

{} Rendering a scene with forward plus lighting using tile shaders

Implement a forward plus renderer using the latest features on Apple GPUs.

{} Rendering reflections in real time using ray tracing

Implement realistic real-time lighting by dynamically generating reflection maps by encoding a ray-tracing compute pass.

{} Rendering reflections with fewer render passes

Use layer selection to reduce the number of render passes needed to generate an environment map.

{} Rendering terrain dynamically with argument buffers

Use argument buffers to render terrain in real time with a GPU-driven pipeline.

{} Selecting device objects for compute processing

Switch dynamically between multiple GPUs to efficiently execute a compute-intensive simulation.

{} Selecting device objects for graphics rendering

Switch dynamically between multiple GPUs to efficiently render to a display.

{} Streaming large images with Metal sparse textures

Limit texture memory usage for large textures by loading or unloading image detail on the basis of MIP and tile region.

{} Supporting Simulator in a Metal app

Configure alternative render paths in your Metal app to enable running your app in Simulator.

{} Synchronizing CPU and GPU work

Avoid stalls between CPU and GPU work by using multiple instances of a resource.

{} Training a neural network using MPSGraph

Train a simple neural network digit classifier.

{} Training a Neural Network with Metal Performance Shaders

Use an MPS neural network graph to train a simple neural network digit classifier.

{} Using argument buffers with resource heaps

Reduce CPU overhead by using arrays inside argument buffers and combining them with resource heaps.

{} Using function specialization to build pipeline variants

Create pipelines for different levels of detail from a common shader source.

{} Using Metal to draw a view's contents

Create a MetalKit view and a render pass to draw the view's contents.

Photos and video

{} Bringing Photos picker to your SwiftUI app

Select media assets by using a Photos picker view that SwiftUI provides.

{} Browsing and Modifying Photo Albums

Help users organize their photos into albums and browse photo collections in a grid-based layout using PhotoKit.

{} Building a guessing game for visionOS

Create a team-based guessing game for visionOS using Group Activities.

{} Capturing screen content in macOS

Stream desktop content like displays, apps, and windows by adopting screen capture in your app.

{} Controlling a DockKit accessory using your camera app

Follow subjects in real time using an iPhone that you mount on a DockKit accessory.

{} Creating a Slideshow Project Extension for Photos

Augment the macOS Photos app with extensions that support project creation.

{} Drawing content in a group session

Invite your friends to draw on a shared canvas while on a FaceTime call.

{} Encoding video for live streaming

Configure a compression session to encode video for live streaming.

{} Encoding video for low-latency conferencing

Configure a compression session to optimize encoding for video-conferencing apps.

{} Encoding video for offline transcoding

Configure a compression session to transcode video in offline workflows.

{} Implementing an inline Photos picker

Embed a system-provided, half-height Photos picker into your app's view.

{} Playing and editing Cinematic mode video

Play and edit Cinematic mode video with an adjustable depth of field and focus points.

{} Recording and Streaming Your macOS App

Share screen recordings, or broadcast live audio and video of your app, by adding ReplayKit to your macOS apps and games.

{} Selecting Photos and Videos in iOS

Improve the user experience of finding and selecting assets by using the Photos picker.

RealityKit and Reality Composer Pro

{} Altering RealityKit Rendering with Shader Functions

Create rendering effects by writing surface shaders and geometry modifiers.

{} Animating entity rotation with a system

Rotate an entity around an axis using a Component and a System.

{} Bringing your SceneKit projects to RealityKit

Adapt a platformer game for RealityKit's powerful ECS and modularity.

{} Building an immersive experience with RealityKit

Use systems and postprocessing effects to create a realistic underwater scene.

{} Building an object reconstruction app

Reconstruct objects from user-selected input images by using photogrammetry.

{} Combining 2D and 3D views in an immersive app

Use attachments to place 2D content relative to 3D content in your visionOS app.

{} Composing interactive 3D content with RealityKit and Reality Composer Pro

Build an interactive scene using an animation timeline.

{} Configuring Collision in RealityKit

Use collision groups and collision filters to control which objects collide.

{} Construct an immersive environment for visionOS

Build efficient custom worlds for your app.

{} Controlling Entity Collisions in RealityKit

Create collision filters to control which objects collide.

{} Creating a game with scene understanding

Create AR games and experiences that interact with real-world objects on LiDAR-equipped iOS devices.

{} Creating an App for Face-Painting in AR

Combine RealityKit's face detection with PencilKit to implement virtual face-painting.

{} Creating a photogrammetry command-line app

Generate 3D objects from images using RealityKit Object Capture.

{} Creating a Spaceship game

Build an immersive game using RealityKit audio, simulation, and rendering features.

{} Creating a spatial drawing app with RealityKit

Use low-level mesh and texture APIs to achieve fast updates to a person's brush strokes by integrating RealityKit with ARKit and SwiftUI.

{} Docking a video player in an immersive scene

Secure a video player in an immersive scene with a docking region you can specify.

{} Generating interactive geometry with RealityKit

Create an interactive mesh with low-level mesh and low-level texture.

{} Implementing special rendering effects with RealityKit postprocessing

Implement a variety of postprocessing techniques to alter RealityKit rendering.

{} Loading entities with ShaderGraph materials

Bring entities that contain materials created with Reality Composer Pro for use in your visionOS app.

{} Presenting an artist's scene

Display a scene from Reality Composer Pro in visionOS.

{} Rendering a windowed game in stereo

Bring an iOS or iPadOS game to visionOS and enhance it.

{} Rendering stereoscopic video with RealityKit

Render stereoscopic video in visionOS with RealityKit.

{} Responding to gestures on an entity

Respond to gestures performed on RealityKit entities using input target and collision components.

{} Scanning objects using Object Capture

Implement a full scanning workflow for capturing objects on iOS devices.

{} Simulating particles in your visionOS app

Add a range of visual effects to a RealityKit view by attaching a particle emitter component to an entity.

{} Simulating physics joints in your RealityKit app

Create realistic, connected motion using physics joints.

{} Simulating physics with collisions in your visionOS app

Create entities that behave and react like physical objects in a RealityKit view.

{} Tracking a handheld accessory as a virtual sculpting tool

Use a tracked accessory with Apple Vision Pro to create a virtual sculpture.

{} Transforming entities between RealityKit coordinate spaces

Move an entity between a volumetric window and an immersive space using coordinate space transformations.

{} Transforming RealityKit entities using gestures

Build a RealityKit component to support standard visionOS gestures on any entity.

{} Using object capture assets in RealityKit

Create a chess game using RealityKit and assets created using Object Capture.

{} WWDC21 Challenge: Framework Freestyle

An AR experience that randomly selects a programming framework and maps it onto the user's face.

StoreKit

{} Determining service entitlement on the server

Identify a customer's entitlement to your service, offers, and messaging by analyzing a validated receipt and the state of their subscription.

{} Generating a Promotional Offer Signature on the Server

Generate a signature using your private key and lightweight cryptography libraries.

{} Implementing a store in your app using the StoreKit API

Offer In-App Purchases and manage entitlements using signed transactions and status information.

{} Offering, completing, and restoring in-app purchases

Fetch, display, purchase, validate, and finish transactions in your app.

{} Offering media for sale in your app

Allow users to purchase media in the App Store from within your app.

{} Requesting App Store reviews

Implement best practices for prompting users to review your app in the App Store.

{} Testing and validating ad impression signatures and postbacks for SKAdNetwork

Validate your ad impressions and test your postbacks by creating unit tests using the StoreKit Test framework.

Swift and SwiftData

{} Adding and editing persistent data in your app

Create a data entry form for collecting and changing data managed by SwiftData.

{} Calling APIs Across Language Boundaries

Use a variety of C++ APIs in Swift – and vice-versa – across multiple targets and frameworks in an Xcode project.

{} Defining data relationships with enumerations and model classes

Create relationships for static and dynamic data stored in your app.

{} Deleting persistent data from your app

Explore different ways to use SwiftData to delete persistent data.

{} Filtering and sorting persistent data

Manage data store presentation using predicates and dynamic queries.

{} Maintaining a local copy of server data

Create and update a persistent store to cache read-only network data.

{} Mixing Languages in an Xcode project

Use C++ APIs in Swift – and Swift APIs in C++ – in a single framework target, and consume the framework's APIs in a separate app target.

{} TicTacFish: Implementing a game using distributed actors

Use distributed actors to take your Swift concurrency and actor-based apps beyond a single process.

{} Updating an app to use strict concurrency

Use this code to follow along with a guide to migrating your code to take advantage of the full concurrency protection that the Swift 6 language mode provides.

{} Updating an App to Use Swift Concurrency

Improve your app's performance by refactoring your code to take advantage of asynchronous functions in Swift.

SwiftUI

{} Add rich graphics to your SwiftUI app

Make your apps stand out by adding background materials, vibrancy, custom graphics, and animations.

{} Adopting drag and drop using SwiftUI

Enable drag-and-drop interactions in lists, tables and custom views.

{} Backyard Birds: Building an app with SwiftData and widgets

Create an app with persistent data, interactive widgets, and an all new in-app purchase experience.

{} Bringing multiple windows to your SwiftUI app

Compose rich views by reacting to state changes and customize your app's scene presentation and behavior on iPadOS and macOS.

{} Bringing robust navigation structure to your SwiftUI app

Use navigation links, stacks, destinations, and paths to provide a streamlined experience for all platforms, as well as behaviors such as deep linking and state restoration.

{} Building a document-based app using SwiftData

Code along with the WWDC presenter to transform an app with SwiftData.

{} Building a great Mac app with SwiftUI

Create engaging SwiftUI Mac apps by incorporating side bars, tables, toolbars, and several other popular user interface elements.

{} Building custom views in SwiftUI

Create a custom view with data-driven transitions and animations in SwiftUI.

{} Composing custom layouts with SwiftUI

Arrange views in your app's interface using layout tools that SwiftUI provides.

{} Controlling the timing and movements of your animations

Build sophisticated animations that you control using phase and keyframe animators.

{} Creating accessible views

Make your app accessible to everyone by applying accessibility modifiers to your SwiftUI views.

{} Creating a tvOS media catalog app in SwiftUI

Build standard content lockups and rows of content shelves for your tvOS app.

{} Creating custom container views

Access individual subviews to compose flexible container views.

{} Creating visual effects with SwiftUI

Add scroll effects, rich color treatments, custom transitions, and advanced effects using shaders and a text renderer.

{} Customizing window styles and state-restoration behavior in macOS

Configure how your app's windows look and function in macOS to provide an engaging and more coherent experience.

{} Enhancing your app's content with tab navigation

Keep your app content front and center while providing quick access to navigation using the tab bar.

{} Focus Cookbook: Supporting and enhancing focus-driven interactions in your SwiftUI app

Create custom focusable views with key-press handlers that accelerate keyboard input and support movement, and control focus programmatically.

{} Food Truck: Building a SwiftUI multiplatform app

Create a single codebase and app target for Mac, iPad, and iPhone.

{} Loading and displaying a large data feed

Consume data in the background, and lower memory use by batching imports and preventing duplicate records.

{} Managing model data in your app

Create connections between your app's data model and views.

{} Migrating from the Observable Object protocol to the Observable macro

Update your existing app to leverage the benefits of Observation in Swift.

{} Monitoring data changes in your app

Show changes to data in your app's user interface by using observable objects.

{} Restoring your app's state with SwiftUI

Provide app continuity for users by preserving their current activities.

System

{} Building a custom peer-to-peer protocol

Use networking frameworks to create a custom protocol for playing a game across iOS, iPadOS, watchOS, and tvOS devices.

{} Building an NFC Tag-Reader App

Read NFC tags with NDEF messages in your app.

{} Building a Simple USB Driver

Set up and load a driver that logs output to the Console app.

{} Collecting Network Connection Metrics

Use reports to understand how DNS and protocol handshakes impact connection establishment.

{} Communicating between a DriverKit extension and a client app

Send and receive different kinds of data securely by validating inputs and asynchronously by storing and using a callback.

{} Communicating with a Modem on a Serial Port

Find and connect to a modem attached to a serial port using IOKit.

{} Configuring a Wi-Fi accessory to join a network

Associate an iOS device with an accessory's network to deliver network configuration information.

{} Connecting a network driver

Create an Ethernet driver that interfaces with the system's network protocol stack.

{} Constraining a tool's launch environment

Improve the security of your macOS app by limiting the ways its components can run.

{} Creating a MIDI device driver

Implement a configurable virtual MIDI driver as a driver extension that runs in user space in macOS and iPadOS.

{} Creating NFC Tags from Your iPhone

Save data to tags, and interact with them using native tag protocols.

{} Encrypting and Decrypting a Single File

Encrypt a single file and save the result to the file system, then decrypt and recreate the original file from the archive file using Apple Encrypted Archive.

{} Encrypting and Decrypting a String

Encrypt the contents of a string and save the result to the file system, then decrypt and recreate the string from the archive file using Apple Encrypted Archive.

{} Encrypting and Decrypting Directories

Compress and encrypt the contents of an entire directory or decompress and decrypt an archived directory using Apple Encrypted Archive.

{} Filtering Network Traffic

Use the Network Extension framework to allow or deny network connections.

{} Handling Keyboard Events from a Human Interface Device

Process keyboard-related data from a human interface device and dispatch events to the system.

{} Handling Stylus Input from a Human Interface Device

Process stylus-related input from a human interface device and dispatch events to the system.

{} Implementing netcat with Network Framework

Build a simple `netcat` tool that establishes network connections and transfers data.

{} Monitoring System Events with Endpoint Security

Receive notifications and authorization requests for sensitive operations by creating an Endpoint Security client for your app.

{} Performing Common Cryptographic Operations

Use CryptoKit to carry out operations like hashing, key generation, and encryption.

{} Receiving Voice and Text Communications on a Local Network

Provide voice and text communication on a local network isolated from Apple Push Notification service by adopting Local Push Connectivity.

{} Running GUI Linux in a virtual machine on a Mac

Install and run GUI Linux in a virtual machine using the Virtualization framework.

{} Running Linux in a Virtual Machine

Run a Linux operating system on your Mac using the Virtualization framework.

{} Running macOS in a virtual machine on Apple silicon

Install and run macOS in a virtual machine using the Virtualization framework.

{} Setting up and authorizing a Bluetooth accessory

Discover, select, and set up a specific Bluetooth accessory without requesting permission to use Bluetooth.

{} Storing CryptoKit Keys in the Keychain

Convert between strongly typed cryptographic keys and native keychain types.

TV

{} Adopting Picture in Picture Playback in tvOS

Add advanced multitasking capabilities to your video apps by using Picture in Picture playback in tvOS.

{} Binding JSON data to TVML documents

Create full-fledged TVML documents by using data binding and queries on simplified TVML files.

{} Building a Full Screen Top Shelf Extension

Highlight content from your Apple TV application by building a full screen Top Shelf extension.

{} Creating a Client-Server TVML App

Display and navigate between TVML documents on Apple TV by retrieving and parsing information from a remote server.

{} Creating a multiview video playback experience in visionOS

Build an interface that plays multiple videos simultaneously and handles transitions to different experience types gracefully.

{} Creating immersive experiences using a full-screen layout

Display content with a collection view that maximizes the tvOS experience.

{} Displaying a Product or Bundle in a Full-Page Template

Specify scrollable and fixed regions in a product page.

{} Implementing a Hybrid TV App with TVMLKit

Display content options with document view controllers and fetch and populate content with TVMLKit JS.

{} Mapping Apple TV users to app profiles

Adapt the content of your app for the current viewer by using an entitlement and simplifying sign-in flows.

{} Playing Media in a Client-Server App

Play media items in a client-server app using the built-in media player for TVMLKit JS.

{} Playing video content in a standard user interface

Play media full screen, embedded inline, or in a floating Picture in Picture (PiP) window using a player view controller.

{} Responding to User Interaction

Update onscreen information by adding event listeners to your Apple TV app.

{} Supporting Continuity Camera in your tvOS app

Capture high-quality photos, video, and audio in your Apple TV app by connecting an iPhone or iPad as a continuity device.

{} Supporting Multiple Users in Your tvOS App

Store separate data for each user with the new Runs as Current User capability.

{} Working with Overlays and Parental Controls in tvOS

Add interactive overlays, parental controls, and livestream channel flipping using a player view controller.

UIKit

{} Add Home Screen quick actions

Expose commonly used functionality with static or dynamic 3D Touch Home Screen quick actions.

{} Adding context menus in your app

Provide quick access to useful actions by adding context menus to your iOS app.

{} Adding hardware keyboard support to your app

Enhance interactions with your app by handling raw keyboard events, writing custom keyboard shortcuts, and working with gesture recognizers.

{} Adding menus and shortcuts to the menu bar and user interface

Provide quick access to useful actions by adding menus and keyboard shortcuts to your Mac app built with Mac Catalyst.

{} Adjusting your layout with keyboard layout guide

Respond dynamically to keyboard movement by using the tracking features of the keyboard layout guide.

{} Adopting drag and drop in a custom view

Demonstrates how to enable drag and drop for a `UIImageView` instance.

{} Adopting drag and drop in a table view

Demonstrates how to enable and implement drag and drop for a table view.

{} Adopting hover support for Apple Pencil

Enhance user feedback for your iPadOS app with a hover preview for Apple Pencil input.

{} Adopting iOS Dark Mode

Adopt Dark Mode in your iOS app by using dynamic colors and visual effects.

{} Adopting menus and UIActions in your user interface

Add menus to your user interface, with built-in button support and bar-button items, and create custom menu experiences.

{} Asynchronously loading images into table and collection views

Store and fetch images asynchronously to make your app more responsive.

{} Building a document browser app for custom file formats

Implement a custom document file format to manage user interactions with files on different cloud storage providers.

{} Building a document browser-based app

Use a document browser to provide access to the user's text files.

{} Building and improving your app with Mac Catalyst

Improve your iPadOS app with Mac Catalyst by supporting native controls, multiple windows, sharing, printing, menus and keyboard shortcuts.

{} Building high-performance lists and collection views

Improve the performance of lists and collections in your app with prefetching and image preparation.

{} Changing the appearance of selected and highlighted cells

Provide visual feedback to the user about the state of a cell and the transition between states.

{} Creating self-sizing table view cells

Create table view cells that support Dynamic Type and use system spacing constraints to adjust the spacing surrounding text labels.

{} Customizing and resizing sheets in UIKit

Discover how to create a layered and customized sheet experience in UIKit.

{} Customizing an image picker controller

Manage user interactions and present custom information when taking pictures by adding an overlay view to your image picker.

{} Customizing collection view layouts

Customize a view layout by changing the size of cells in the flow or implementing a mosaic style.

{} Customizing your app's navigation bar

Create custom titles, prompts, and buttons in your app's navigation bar.

{} Data delivery with drag and drop

Share data between iPad apps during a drag and drop operation using an item provider.

{} Detecting changes in the preferences window

Listen for and respond to a user's preference changes in your Mac app built with Mac Catalyst using Combine.

{} Disabling the pull-down gesture for a sheet

Ensure a positive user experience when presenting a view controller as a sheet.

{} Displaying searchable content by using a search controller

Create a user interface with searchable content in a table view.

{} Display text with a custom layout

Lay out text in a custom-shaped container and apply glyph substitutions.

{} Enhancing your iPad app with pointer interactions

Provide a great user experience with pointing devices, by incorporating pointer content effects and shape customizations.

{} Enriching your text in text views

Add exclusion paths, text attachments, and text lists to your text, and render it with text views.

{} Illustrating the force, altitude, and azimuth properties of touch input

Capture Apple Pencil and touch input in views.

{} Implementing modern collection views

Bring compositional layouts to your app and simplify updating your user interface with diffable data sources.

{} Implementing Peek and Pop

Accelerate actions in your app by providing shortcuts to preview content in detail view controllers.

{} Integrating pointer interactions into your iPad app

Support touch interactions in your iPad app by adding pointer interactions to your views.

{} Leveraging touch input for drawing apps

Capture touches as a series of strokes and render them efficiently on a drawing canvas.

{} Navigating an app's user interface using a keyboard

Navigate between user interface elements using a keyboard and focusable UI elements in iPad apps and apps built with Mac Catalyst.

{} Prefetching collection view data

Load data for collection view cells before they display.

{} Restoring your app's state

Provide continuity for the user by preserving current activities.

{} Selecting multiple items with a two-finger pan gesture

Accelerate user selection of multiple items using the multiselect gesture on table and collection views.

{} Showing help tags for views and controls using tooltip interactions

Explain the purpose of interface elements by showing a tooltip when a person positions the pointer over the element.

{} Supporting desktop-class features in your iPad app

Enhance your iPad app by adding desktop-class features and document support.

{} Supporting gesture interaction in your apps

Enrich your app's user experience by supporting standard and custom gesture interaction.

{} Supporting HDR images in your app

Load, display, edit, and save HDR images using SwiftUI and Core Image.

{} Supporting multiple windows on iPad

Support side-by-side instances of your app's interface and create new windows.

{} Synchronizing documents in the iCloud environment

Manage documents across multiple devices to create a seamless editing and collaboration experience.

{} UIKit Catalog: Creating and customizing views and controls

Customize your app's user interface with views and controls.

{} Updating collection views using diffable data sources

Streamline the display and update of data in a collection view using a diffable data source that contains identifiers.

{} Using suggested searches with a search controller

Create a search interface with a table view of suggested searches.

{} Using SwiftUI with UIKit

Learn how to incorporate SwiftUI views into a UIKit app.

{} Using TextKit 2 to interact with text

Interact with text by managing text selection and inserting custom text elements.

Vision

{} Aligning Similar Images

Construct a composite image from images that capture the same scene.

{} Analyzing a selfie and visualizing its content

Calculate face-capture quality and visualize facial features for a collection of images using the Vision framework.

{} Analyzing Image Similarity with Feature Print

Generate a feature print to compute distance between images.

{} Applying Matte Effects to People in Images and Video

Generate image masks for people automatically by using semantic person-segmentation.

{} Applying visual effects to foreground subjects

Segment the foreground subjects of an image and composite them to a new background with visual effects.

{} Building a feature-rich app for sports analysis

Detect and classify human activity in real time using computer vision and machine learning.

{} Classifying images for categorization and search

Analyze and label images using a Vision classification request.

{} Detecting animal body poses with Vision

Draw the skeleton of an animal by using Vision's capability to detect animal body poses.

{} Detecting Hand Poses with Vision

Create a virtual drawing app by using Vision's capability to detect hand poses.

{} Detecting human body poses in 3D with Vision

Render skeletons of 3D body pose points in a scene overlaying the input image.

{} Detecting moving objects in a video

Identify the trajectory of a thrown object by using Vision.

{} Detecting Objects in Still Images

Locate and demarcate rectangles, faces, barcodes, and text in images using the Vision framework.

{} Extracting phone numbers from text in images

Analyze and filter phone numbers from text in live capture by using Vision.

{} Generating high-quality thumbnails from videos

Identify the most visually pleasing frames in a video by using the image-aesthetics scores request.

{} Highlighting Areas of Interest in an Image Using Saliency

Quantify and visualize where people are likely to look in an image.

{} Locating and displaying recognized text

Perform text recognition on a photo using the Vision framework's text-recognition request.

{} Recognizing Objects in Live Capture

Apply Vision algorithms to identify objects in real-time video.

{} Segmenting and colorizing individuals from a surrounding scene

Use the Vision framework to isolate and apply colors to people in an image.

{} Selecting a selfie based on capture quality

Compare face-capture quality in a set of images by using Vision.

{} Structuring Recognized Text on a Document

Detect, recognize, and structure text on a business card or receipt using Vision and VisionKit.

{} Tracking Multiple Objects or Rectangles in Video

Apply Vision algorithms to track objects or rectangles throughout a video.

{} Tracking the User's Face in Real Time

Detect and track faces from the selfie cam feed in real time.

{} Training a Create ML Model to Classify Flowers

Train a flower classifier using Create ML in Swift Playgrounds, and apply the resulting model to real-time image classification using Vision.

visionOS

{} Accessing the main camera

Add camera-based features to enterprise apps.

{} Adding a depth effect to text in visionOS

Create text that expands out of a window using stacked SwiftUI text views.

{} Applying mesh to real-world surroundings

Add a layer of mesh to objects in the real world, using scene reconstruction in ARKit.

{} BOT-anist

Build a multiplatform app that uses windows, volumes, and animations to create a robot botanist's greenhouse.

{} Building an immersive media viewing experience

Add a deeper level of immersion to media playback in your app with RealityKit and Reality Composer Pro.

{} Building local experiences with room tracking

Use room tracking in visionOS to provide custom interactions with physical spaces.

{} Creating 2D shapes with SwiftUI

Draw two-dimensional shapes in your visionOS app with SwiftUI shapes or with your custom shapes.

{} Creating 3D entities with RealityKit

Display a horizontal row of three-dimensional shapes in your visionOS app, using predefined mesh and white material.

{} Creating 3D models as movable windows

Display 3D content with a volumetric window that people can move.

{} Creating a 3D painting space

Implement a painting canvas entity, and update its mesh to represent a stroke.

{} Creating an immersive space in visionOS

Enhance your visionOS app by adding an immersive space using RealityKit.

{} Creating an interactive 3D model in visionOS

Display an interactive car model using gestures in a reality view.

{} Creating SwiftUI windows in visionOS

Display and manage multiple SwiftUI windows in your visionOS app.

{} Destination Video

Leverage SwiftUI to build an immersive media experience in a multiplatform app.

{} Diorama

Design scenes for your visionOS app using Reality Composer Pro.

{} Displaying a 3D environment through a portal

Implement a portal window that displays a 3D environment and simulates entering a portal by using RealityKit.

{} Displaying an entity that follows a person's view

Create an entity that tracks and follows head movement in an immersive scene.

{} Displaying a stereoscopic image

Build a stereoscopic image by applying textures to the left and right eye in a shader graph material.

{} Displaying text in visionOS

Create styled text in a window using SwiftUI.

{} Displaying video from connected devices

Show video from devices connected with the Developer Strap in your visionOS app.

{} Enabling video reflections in an immersive environment

Create a more immersive experience by adding video reflections in a custom environment.

{} Exploring object tracking with ARKit

Find and track real-world objects in visionOS using reference objects trained with Create ML.

{} Generating procedural textures

Display a 3D model that generates procedural textures in a reality view.

{} Happy Beam

Leverage a Full Space to create a fun game using ARKit.

{} Hello World

Use windows, volumes, and immersive spaces to teach people about the Earth.

{} Implementing adjustable material

Update the adjustable parameters of a 3D model in visionOS.

{} Incorporating real-world surroundings in an immersive experience

Create an immersive experience by making your app's content respond to the local shape of the world.

{} Locating and decoding barcodes in 3D space

Create engaging, hands-free experiences based on barcodes in a person's surroundings.

{} Object tracking with Reality Composer Pro experiences

Use object tracking in visionOS to attach digital content to real objects to create engaging experiences.

{} Obscuring virtual items in a scene behind real-world items

Increase the realism of an immersive experience by adding entities with invisible materials real-world objects.

{} Placing content on detected planes

Detect horizontal surfaces like tables and floors, as well as vertical planes like walls and doors.

{} Placing entities using head and device transform

Query and react to changes in the position and rotation of Apple Vision Pro.

{} Playing spatial audio

Create and adjust spatial audio in visionOS with RealityKit.

{} Swift Splash

Use RealityKit to create an interactive ride in visionOS.

{} Tracking and visualizing hand movement

Use hand-tracking anchors to display a visual representation of hand transforms in visionOS.

{} Tracking specific points in world space

Retrieve the position and orientation of anchors your app stores in ARKit.

Watch

{} Building a productivity app for Apple Watch

Create a watch app to manage and share a task list and visualize the status with a chart.

{} Create accessible experiences for watchOS

Learn how to make your watchOS app more accessible.

{} Creating and updating a complication's timeline

Create complications that batch-load a timeline of future entries and run periodic background sessions to update the timeline.

{} Customizing workouts with WorkoutKit

Create, preview, and sync workouts for use in the Workout app on Apple Watch.

{} Displaying essential information on a watch face

Implement complications in a watch app to display essential information on a watch face.

{} Interacting with Bluetooth peripherals during background app refresh

Keep your complications up-to-date by reading values from a Bluetooth peripheral while your app is running in the background.

{} Providing Multiple Complications

Present multiple complications for a single complication family using descriptors.

{} Transferring data with Watch Connectivity

Transfer data between a watchOS app and its companion iOS app.

{} Updating your app and widgets for watchOS 10

Integrate SwiftUI elements and watch-specific features, and build widgets for the Smart Stack.

Web

{} Adopting Declarative Content Blocking in Safari Web Extensions

Block web content with your web extension using the declarative net request API.

{} Adopting New Safari Web Extension APIs

Improve your web extension in Safari with a non-persistent background page and new tab-override customization.

{} Creating Safari Web Inspector extensions

Learn how to make custom Safari Web Inspector extensions.

{} Developing a browser app that uses an alternative browser engine

Create a web browser app and associated extensions.

{} Developing a Safari Web Extension

Customize and enhance web pages by building a Safari web extension.

{} Messaging a Web Extension's Native App

Communicate between your Safari web extension and its containing app.

{} Modernizing Safari Web Extensions

Learn about enhancements to Safari Web Extensions.

{} Previewing Metadata using Open Graph

Build a Safari Extension that displays metadata using Open Graph.

{} Viewing Desktop or Mobile Web Content Using a Web View

Implement a simple iPad web browser that can view either the desktop or mobile version of a website.