Game technologies

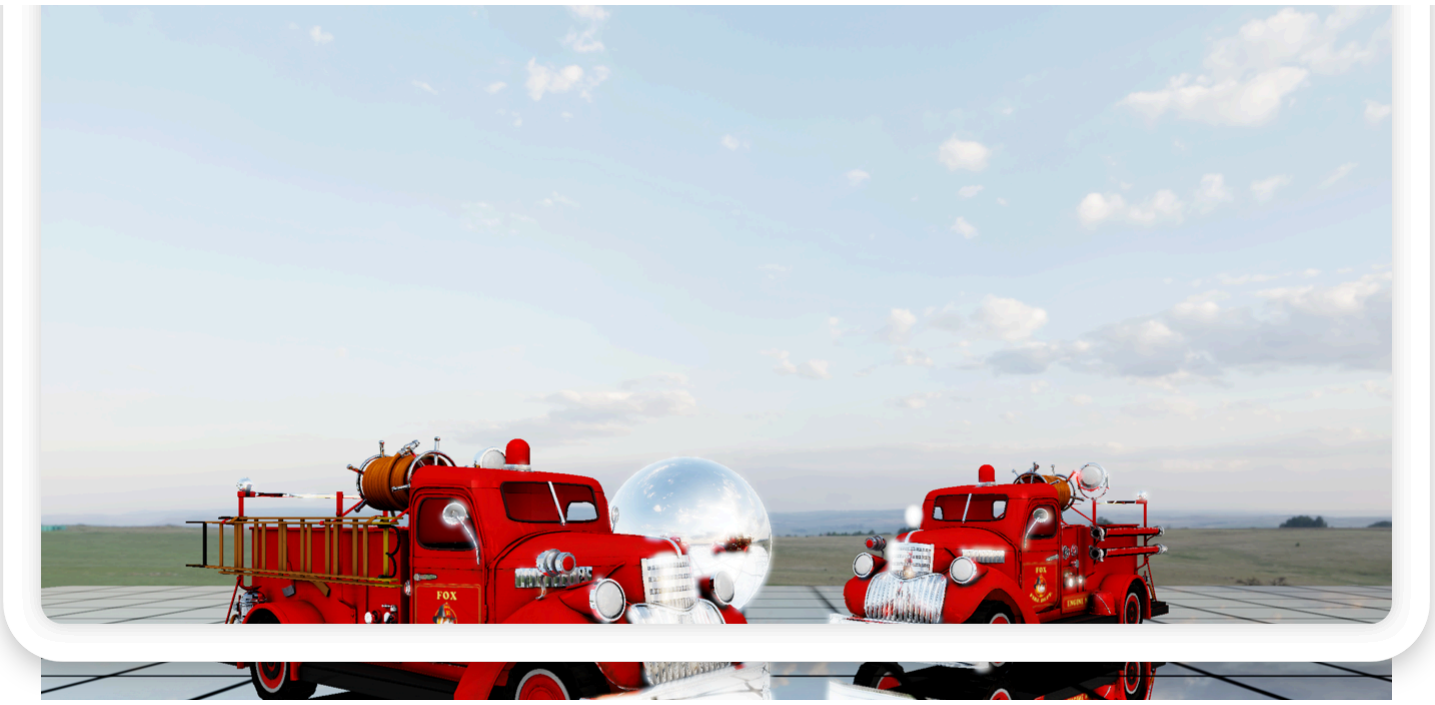
Plan the creation of your game and incorporate the gameplay features people expect.

Game development starts with Xcode, Apple's integrated development environment, which includes code editors, debugging tools, device simulators, graphics performance and analysis tools, and platform SDKs. When you create a project, Xcode lets you choose the initial platforms, templates, and game technologies for your project. If you already have a game, you can also bring that game to Apple platforms.

Great games take advantage of the characteristics and capabilities of the devices that they run on. For example, someone might use their iPhone when travelling or relaxing on the couch, or use their Mac when they're sitting at their desk. Design your game with these factors in mind and take advantage of all available hardware connected to the device.

Create gorgeous graphics

Get the best graphics performance on all Apple platforms using Metal — Apple's API for hardware-accelerated 2D and 3D graphics. With Metal, you have the power of the graphics processing unit (GPU) at your fingertips. Deliver smooth 3D animations with realistic visual effects from your game's rendering code. Tune the performance of that code to maximize frame rates and efficiency on a variety of Apple devices.



Metal offers the features you need to create detailed graphics, with the performance you need to achieve high frame rates:

- Run commands directly on the device's GPU. Encode compute passes to run in parallel with render passes so that computations don't impact graphics rendering.
- Load graphics resources faster by streaming asset data to textures and buffers asynchronously.
- Improve the performance of 3D scenes using high-performance temporal antialiasing or spatial upscaling. Upscale low-resolution images to higher-resolution images in less time using MetalFX.
- Apply high-performance filters to images, multiply matrices, and vectors using the Metal Performance Shaders framework.
- Accelerate ray tracing tasks using Metal's built-in ray tracing support.

Make your game fast and efficient using the powerful suite of Metal development tools.

- Validate your Metal code and catch shader execution errors in Xcode.
- Investigate visual artifacts, and optimize GPU performance, using the Metal debugger.
- Find the cause of low frame rates so you can fix interruptions and stutters.
- Maintain a small memory footprint to improve overall performance.
- Analyze your game's performance in real time with the Metal Performance HUD, which shows CPU and GPU metrics in real time.

Make sure you tune your graphics code specifically for Apple silicon:

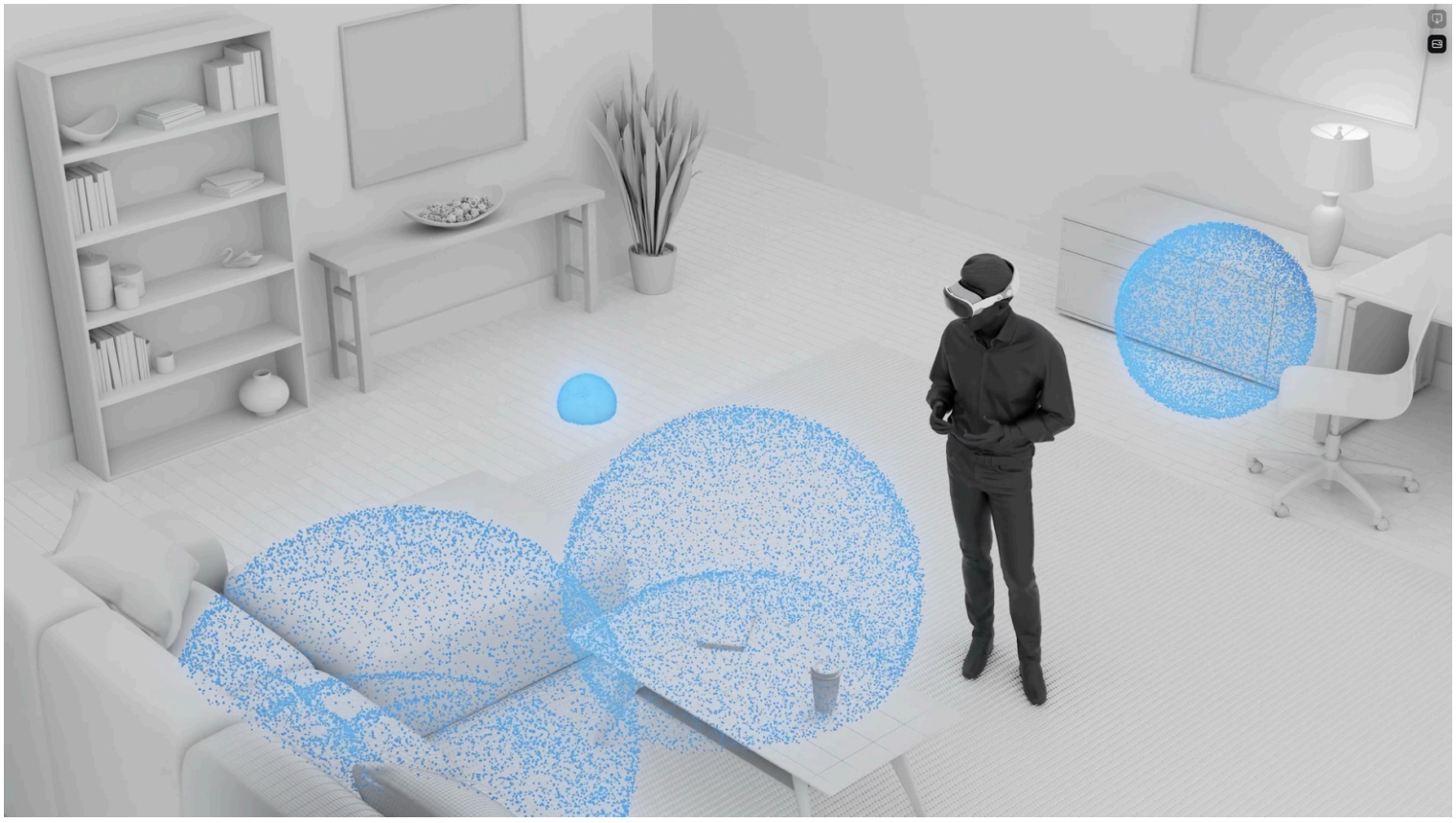
- Fix architectural differences in your code to ensure your code runs well on Apple silicon. Make sure your code supports virtual memory-page sizes, cache line sizes, variadic functions,

memory that's simultaneously writable and executable, and more.

- Manage thread priorities and schedules using Dispatch and Quality of Service (QoS) levels.
- Accelerate your code's performance using the open source SSE2Neon and AVX2Neon libraries. The Neon instruction set provides single instruction multiple data (SIMD) operations to accelerate performance on ARM processors.
- Optimize for Apple GPUs and Tile-Based Deferred Rendering. Write your Metal code to use features like imageblocks, tile shading, and raster order groups.

Build a spatial computing game for visionOS

Apple Vision Pro gives you an opportunity to create immersive 3D experiences. Whether you're creating a fully immersive game or an augmented reality experience, take advantage of visionOS features. And if you're making a game for iPad or iPhone, that game can also run run on Apple Vision Pro in a window and use direct and indirect gestures as input.



- Integrate your game's virtual content into someone's real-world environment using RealityKit, SwiftUI, and ARKit. RealityKit makes your virtual content look more realistic by applying depth mitigation, passthrough, ground shadows, and other effects automatically.
- Create a fully immersive app and render everything yourself using Metal and Compositor Services.

- Design your game for spatial input using ARKit. Support interactions with attached game controllers and keyboards, which passthrough makes visible even in a Full Space.
- Bring your existing Unity VR app to visionOS or create a new immersive Unity app.

Get started by learning about the building blocks that make up spatial computing, and find out how you use these elements to build your game experiences. Adapt your 2D games by adjusting your content to take advantage of 3D. For example, render objects in separate layers to create a parallax effect, and add smoke, sparks, and other visual elements that come out of visual plane.

Create exceptional soundtracks

Increase engagement and realism by adding sound effects and music that complement the action within your game.

- Play your app's music, audio, and sound effects using the AVFAudio framework. Configure your audio session to mix your audio components the way you want, then play back your audio using an audio player object.
- Play music from someone's iTunes library using the MusicKit framework.
- Create an immersive audio experience that reacts in real time to events in your game using the PHASE framework.
- Perform advanced audio manipulation, or work directly with audio hardware, using the Core Audio framework.
- Add audio to RealityKit entities to bring those objects to life. RealityKit automatically applies reverb and real-world acoustics to your entities to give them a realistic sound. Explore other aspects of immersive sound design to further improve the audio experience of your game.

On iPhone and Apple TV, get the player's attention and reinforce actions with tactile and audio feedback using the Core Haptics framework. Play both transient and continues patterns to match the events in your app. For example, play a short vibration pattern when someone toggles a switch, and play a continuous pattern to represent the ringing of a telephone. Generate haptics for an attached game controller using the Game Controller framework instead.

Support input devices

Let players interact with your game in more natural ways using Game Controller. Support the latest

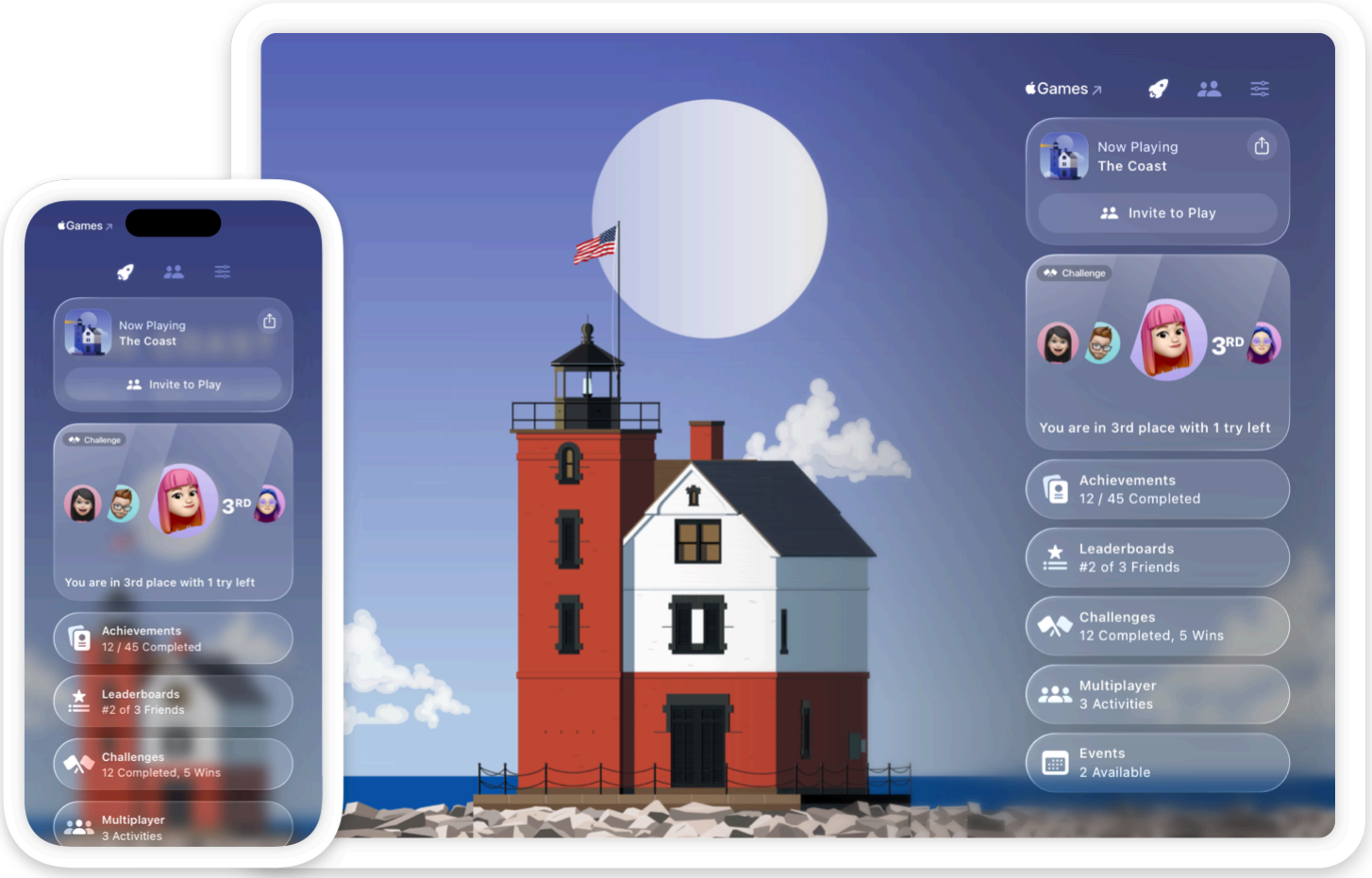
- Support third-party gamepads, arcade sticks, and racing wheels, as well as the mouse and keyboard.
- Overlay a virtual controller on mobile devices to emulate a physical controller.

- Support advanced game controller features such as rumble and haptic feedback from a physical controller.

The App Store adds a game controller badge to your game if it supports controllers.

Build a social network

Take advantage of the services Apple provides for social gaming and storing game data using Game Center. Let Apple help promote your game organically through players sharing and interacting with each other.



- Grow discovery and engagement using Game Center, Apple's social-gaming network.
- Keep players engaged using achievements and leaderboards.
- Connect players to their friends and create new ways for friends to interact with each other.
- Let people continue your game on any of their devices by saving game data in their iCloud storage.
- Present familiar interfaces to match players with other players before the game starts.
- Share data between players to advance the gameplay as each person takes their turn. Exchange messages between players in turn-based games.

- Let people communicate with each other using FaceTime or Messages by turning your game into a shared activity.
- Add a screen-sharing or game-streaming feature to your macOS game using ScreenCaptureKit.

Distribute your game

The App Store lets you easily distribute your game to hundreds of millions of people around the world on all Apple platforms. The App Store processes customer payments, provides secure and reliable downloads, manages releases, and helps promote your game.

- Distribute your game through the App Store. Join the Apple Developer Program and upload builds using Xcode. Use App Store Connect to configure game-oriented features, distribute your game for beta testing, and submit builds to App Review.
- Prepare your app for distribution and let the App Store optimize downloads for each platform. Enable faster downloads of your game by hosting assets on the App Store or your own server, and downloading them separately.
- Create your product page on the App Store with app previews, screenshots, and a description that gets players excited about your game.
- Make your macOS game available on Apple silicon only. Target your game for the “Apple silicon-only apps” section of the Mac App Store by targeting your game to Macs with an M1 chip or later.
- Give players confidence that your macOS game is from you and is free of malicious components by notarizing it before distribution.
- Protect the integrity of your game by enabling the Hardened Runtime capability. Select options to prevent against malicious code injection, dynamic library hijacking, and process memory space tampering.

Contact Apple if you’re working on a groundbreaking, unreleased game and would like Apple to consider it for the Apple Arcade game subscription service.