

[Metal](#) / [Metal sample code library](#) / Loading textures and models using Metal fast resource loading

Sample Code

# Loading textures and models using Metal fast resource loading

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

[Download](#)

macOS 13.0+ | Xcode 14.0+



## Overview

### Note

This sample code project is associated with WWDC22 session [10104: Load resources faster with Metal 3](#).

## Configure the sample code project

This sample code project requires the following:

- macOS 13 or later, and a Mac with Apple silicon
- Xcode 14 or later

## See Also

# Render workflows

- { } Using Metal to draw a view's contents

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

- { } Drawing a triangle with Metal 4

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

- { } Selecting device objects for graphics rendering

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

- { } Customizing render pass setup

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

- { } Creating a custom Metal view

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

- { } Calculating primitive visibility using depth testing

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

- { } Encoding indirect command buffers on the CPU

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

- { } Implementing order-independent transparency with image blocks

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

- { } Adjusting the level of detail using Metal mesh shaders

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

- { } Creating a 3D application with hydra rendering

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

- { } 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.

- { } 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.

{ } 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.