```
import CoreGraphics
import Foundation
import SceneKit.ModelIO
import SceneKit.SCNAction
import SceneKit.SCNAnimation
import SceneKit.SCNAudioSource
import SceneKit.SCNBoundingVolume
import SceneKit.SCNCAAnimationExtensions
import SceneKit.SCNCamera
import SceneKit.SCNCameraController
import SceneKit.SCNConstraint
import SceneKit.SCNGeometry
import SceneKit.SCNHitTest
import SceneKit.SCNJavascript
import SceneKit.SCNLayer
import SceneKit.SCNLevelOfDetail
import SceneKit.SCNLight
import SceneKit.SCNMaterial
import SceneKit.SCNMaterialProperty
import SceneKit.SCNMorpher
import SceneKit.SCNNode
import SceneKit.SCNParametricGeometry
import SceneKit.SCNParticleSystem
import SceneKit.SCNPhysicsBehavior
import SceneKit.SCNPhysicsBody
import SceneKit.SCNPhysicsContact
import SceneKit.SCNPhysicsField
import SceneKit.SCNPhysicsShape
import SceneKit.SCNPhysicsWorld
import SceneKit.SCNReferenceNode
import SceneKit.SCNRenderer
import SceneKit.SCNScene
import SceneKit.SCNSceneRenderer
```

```
import SceneKit.SCNSceneSource
import SceneKit.SCNShadable
import SceneKit.SCNSkinner
import SceneKit.SCNTechnique
import SceneKit.SCNTransaction
import SceneKit.SCNView
import SceneKit.SceneKitAvailability
import SceneKit.SceneKitDeprecated
import SceneKit.SceneKitTypes
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims
import simd

public typealias SCNFloat = CGFloat

extension SCNVector3 {

    public init(_ x: Float, _ y: Float, _
z: Float)

    public init(_ x: CGFloat, _ y:
CGFloat, _ z: CGFloat)

    public init(_ x: Double, _ y: Double,
_ z: Double)

    public init(_ x: Int, _ y: Int, _ z:
Int)

    public init(_ v: SIMD3<Float>)

    public init(_ v: SIMD3<Double>)
```

```
}

extension SIMD3 where Scalar == Float {

    public init(_ v: SCNVector3)
}

extension SIMD3 where Scalar == Double {

    public init(_ v: SCNVector3)
}

extension SCNVector4 {

    public init(_ x: Float, _ y: Float, _
z: Float, _ w: Float)

    public init(_ x: CGFloat, _ y:
CGFloat, _ z: CGFloat, _ w: CGFloat)

    public init(_ x: Double, _ y: Double,
_ z: Double, _ w: Double)

    public init(_ x: Int, _ y: Int, _ z:
Int, _ w: Int)

    public init(_ v: SIMD4<Float>)

    public init(_ v: SIMD4<Double>)
}

extension SIMD4 where Scalar == Float {
```

```
    public init(_ v: SCNVector4)
}

extension SIMD4 where Scalar == Double {

    public init(_ v: SCNVector4)
}

extension CATransform3D {

    public init(_ m: float4x4)

    public init(_ m: double4x4)
}

extension simd_float4x4 {

    public init(_ m: SCNMatrix4)
}

extension simd_double4x4 {

    public init(_ m: SCNMatrix4)
}

@available(iOS 8.0, macOS 10.8, *)
extension SCNGeometryElement {

    /// Creates an instance from
`indices` for a `primitiveType`
    /// that has a constant number of
indices per primitive
    /// - Precondition: the
```

```
`primitiveType` must be `.triangles`,
`.triangleStrip`, `.line` or `.point`
    public convenience
init<IndexType>(indices: [IndexType],
primitiveType: SCNGeometryPrimitiveType)
where IndexType : FixedWidthInteger
}

@available(iOS 8.0, macOS 10.8, *)
extension SCNGeometrySource {

    @nonobjc public convenience
init(vertices: [SCNVector3])

    @nonobjc public convenience
init(normals: [SCNVector3])

    @nonobjc public convenience
init(textureCoordinates: [CGPoint])
}

@available(iOS 8.0, macOS 10.10, *)
extension SCNBoundingVolume {

    public var boundingBox: (min:
SCNVector3, max: SCNVector3)

    public var boundingSphere: (center:
SCNVector3, radius: Float) { get }
}

@available(iOS 8.0, macOS 10.8, *)
extension SCNSceneSource {
```

```
    public func entryWithIdentifier<T>(_
uid: String, withClass entryClass:
T.Type) -> T? where T : AnyObject
}


// MARK: - SwiftUI Additions

import SwiftUI

// Available when SwiftUI is imported
with SceneKit
/// The view to present your SCNScene
nodes in.
@available(iOS 14.0, macOS 11.0, tvOS
14.0, watchOS 7.0, *)
@MainActor @preconcurrency public struct
SceneView : View {

    /// Option set that's used to
customize how the SceneView behaves
    public struct Options : OptionSet {

        /// The corresponding value of
the raw type.
        ///
        /// A new instance initialized
with `rawValue` will be equivalent to
this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
```

```
        ///         case A4, A5, Letter,
Legal
        ///     }
        ///
        ///     let selectedSize =
PaperSize.Letter
        ///
print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize ==
PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public let rawValue: Int

        /// Creates a new option set from
the given raw value.
        ///
        /// This initializer always
succeeds, even if the value passed as
`rawValue`
        /// exceeds the static properties
declared as part of the option set. This
        /// example creates an instance
of `ShippingOptions` with a raw value
beyond
        /// the highest element, with a
bit mask that effectively contains all
the
        /// declared static members.
        ///
        ///     let extraOptions =
```

```
ShippingOptions(rawValue: 255)
        ///
print(extraOptions.isStrictSuperset(of: .
all))
        ///     // Prints "true"
        ///
        /// - Parameter rawValue: The raw
value of the option set to create. Each
bit
        ///   of `rawValue` potentially
represents an element of the option set,
        ///   though raw values may
include bits that are not defined as
distinct
        ///   values of the `OptionSet`
type.
        public init(rawValue: Int)

        /// Toggles whether the view
allows the user to manipulate the camera
with gestures or mouse
        /// Defaults to false.
        public static let
allowsCameraControl: SceneView.Options

        /// Toggles whether the view
renders continuously at the desired frame
rate.
        /// If set to false the view will
redraw only when necessary.
        /// Defaults to false.
        public static let
rendersContinuously: SceneView.Options
```

```
        /// Specifies whether the
receiver should automatically light up
scenes that have no light source.
        /// The default is false.
        public static let
autoenablesDefaultLighting:
SceneView.Options

        /// Specifies whether the
receiver should jitter the rendered scene
to reduce aliasing artifacts.
        /// The default is false.
        public static let
jitteringEnabled: SceneView.Options

        /// Specifies whether the
receiver should reduce aliasing artifacts
in real time based on temporal coherency.
        /// The default is false.
        public static let
temporalAntialiasingEnabled:
SceneView.Options

        /// The type of the elements of
an array literal.
        @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias
ArrayLiteralElement = SceneView.Options

        /// The element type of the
option set.
```

```
        ///
        /// To inherit all the default
implementations from the `OptionSet`
protocol,
        /// the `Element` type must be
`Self`, the default.
        @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias Element =
SceneView.Options

        /// The raw type that can be used
to represent all values of the conforming
        /// type.
        ///
        /// Every distinct value of the
conforming type has a corresponding
unique
        /// value of the `RawValue` type,
but there may be values of the `RawValue`
        /// type that don't have a
corresponding value of the conforming
type.
        @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
        public typealias RawValue = Int
    }

    /// Creates an instance with the
given `scene`
    ///
    /// - Parameters:
    ///     - scene: SCNScene to present
```

```
    ///     - pointOfView: The point of
view to use to render the scene.
    ///     - options: Various options
(see above) to configure the receiver.
    ///     - preferredFramesPerSecond:
sets fps to define the desired rate for
current SceneView.
    ///     Actual rate maybe limited by
hardware or other software
    ///     - antialiasingMode: desired
level of antialiasing. Defaults to 4X.
    ///     - delegate: The delegate of
the receiver
    ///     - technique: Specifies a
custom post process
    @MainActor @preconcurrency public
init(scene: SCNScene? = nil, pointOfView:
SCNNode? = nil, options:
SceneView.Options = [],
preferredFramesPerSecond: Int = 60,
antialiasingMode: SCNAntialiasingMode
= .multisampling4X, delegate: (any
SCNSceneRendererDelegate)? = nil,
technique: SCNTechnique? = nil)

    /// The content and behavior of the
view.
    ///
    /// When you implement a custom view,
you must implement a computed
    /// `body` property to provide the
content for your view. Return a view
    /// that's composed of built-in views
```

```
that SwiftUI provides, plus other
    /// composite views that you've
already defined:
    ///
    ///     struct MyView: View {
    ///         var body: some View {
    ///             Text("Hello, World!")
    ///         }
    ///     }
    ///
    /// For more information about
composing views and a view hierarchy,
    /// see <doc:Declaring-a-Custom-
View>.
    @MainActor @preconcurrency public var
body: some View { get }

    /// The type of view representing the
body of this view.
    ///
    /// When you create a custom view,
Swift infers this type from your
    /// implementation of the required
``View/body-swift.property`` property.
    @available(iOS 14.0, tvOS 14.0,
watchOS 7.0, macOS 11.0, *)
    public typealias Body = some View
}

// Available when SwiftUI is imported
with SceneKit
@available(iOS 14.0, macOS 11.0, tvOS
14.0, watchOS 7.0, *)
```

```
extension SceneView : Sendable {
}
```