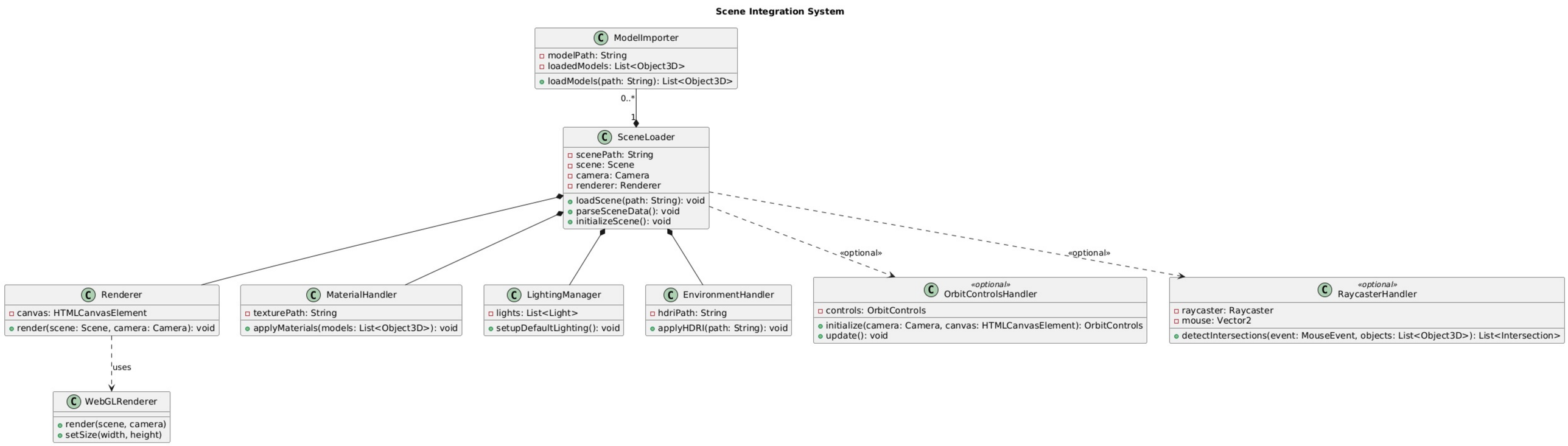
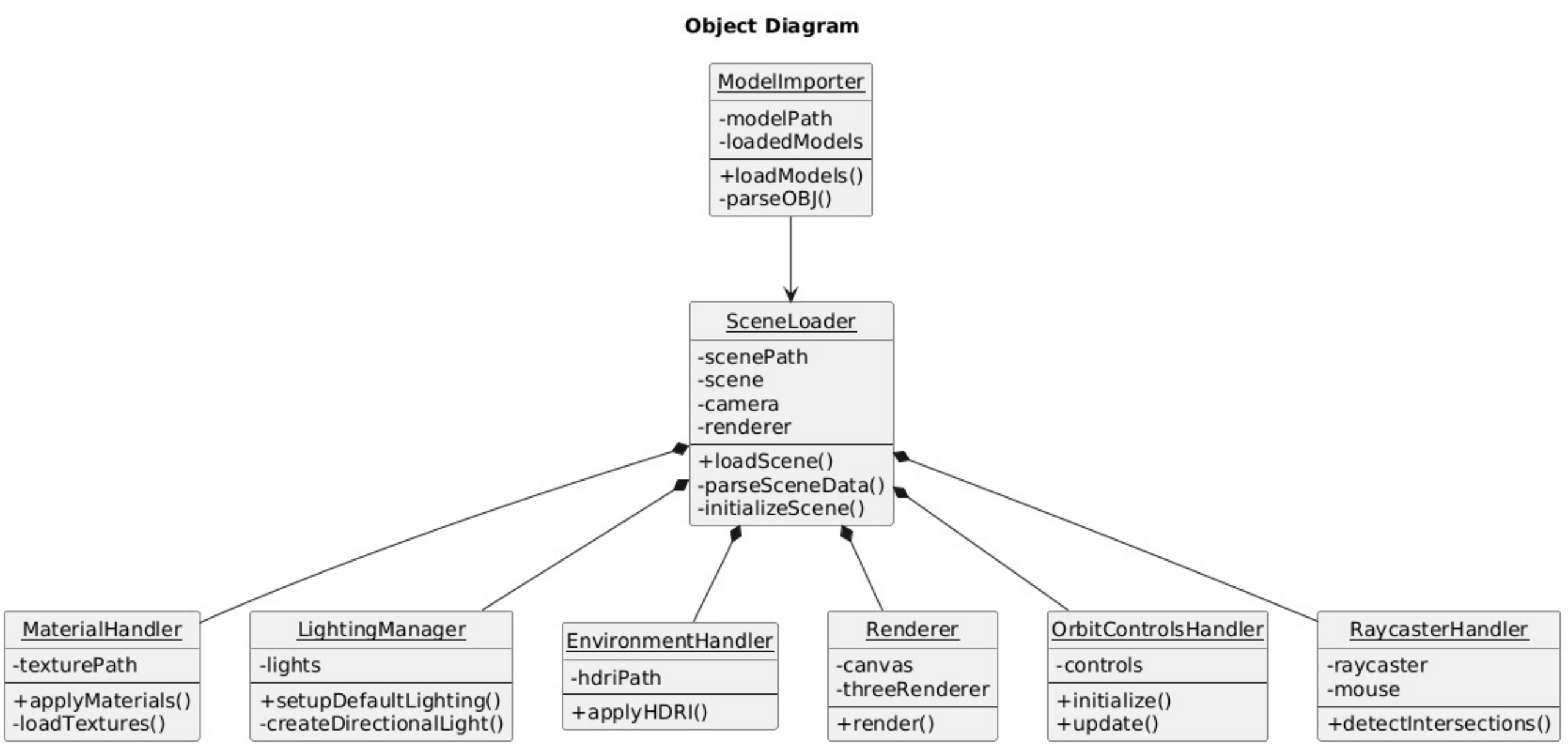
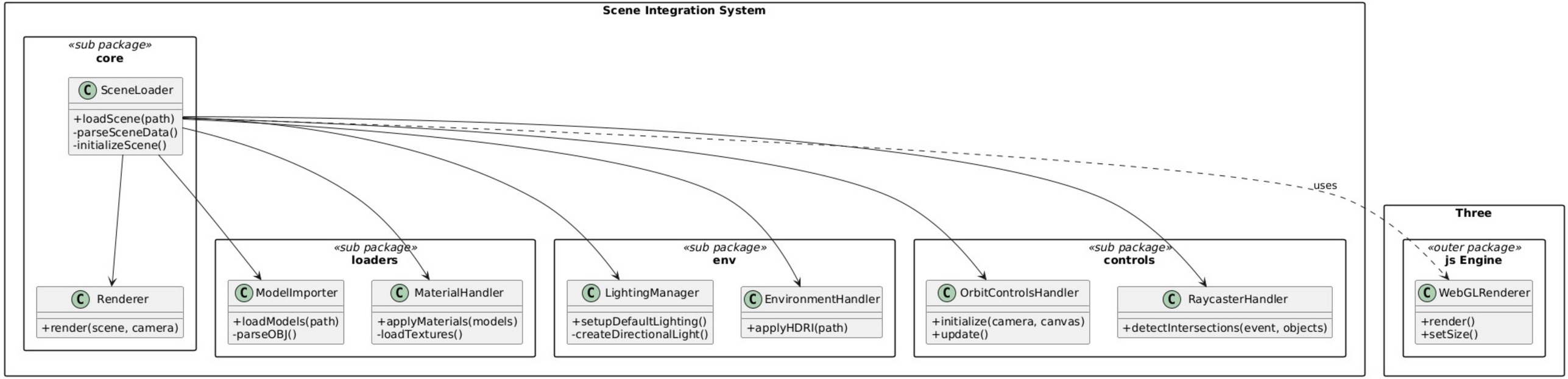
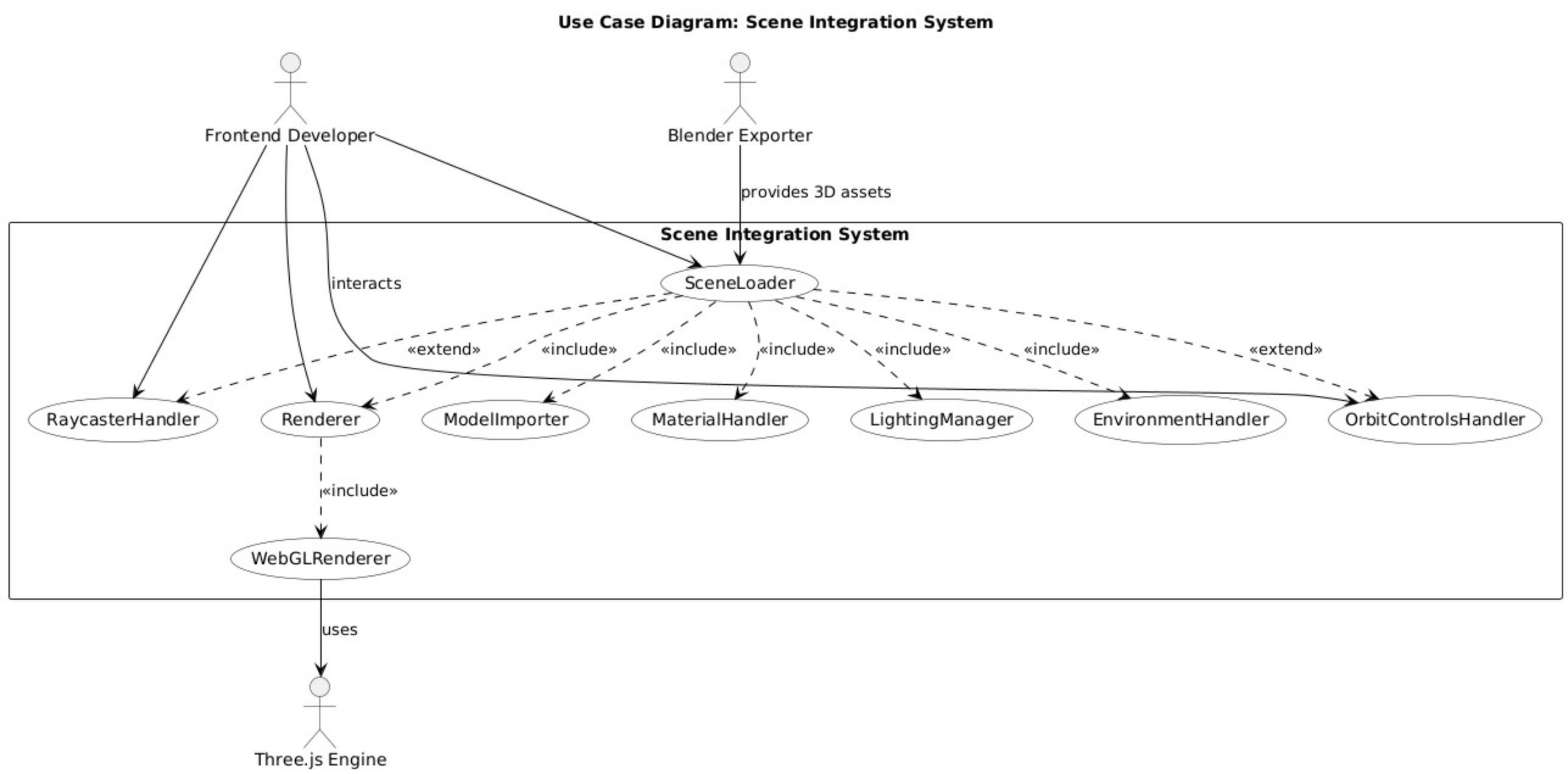
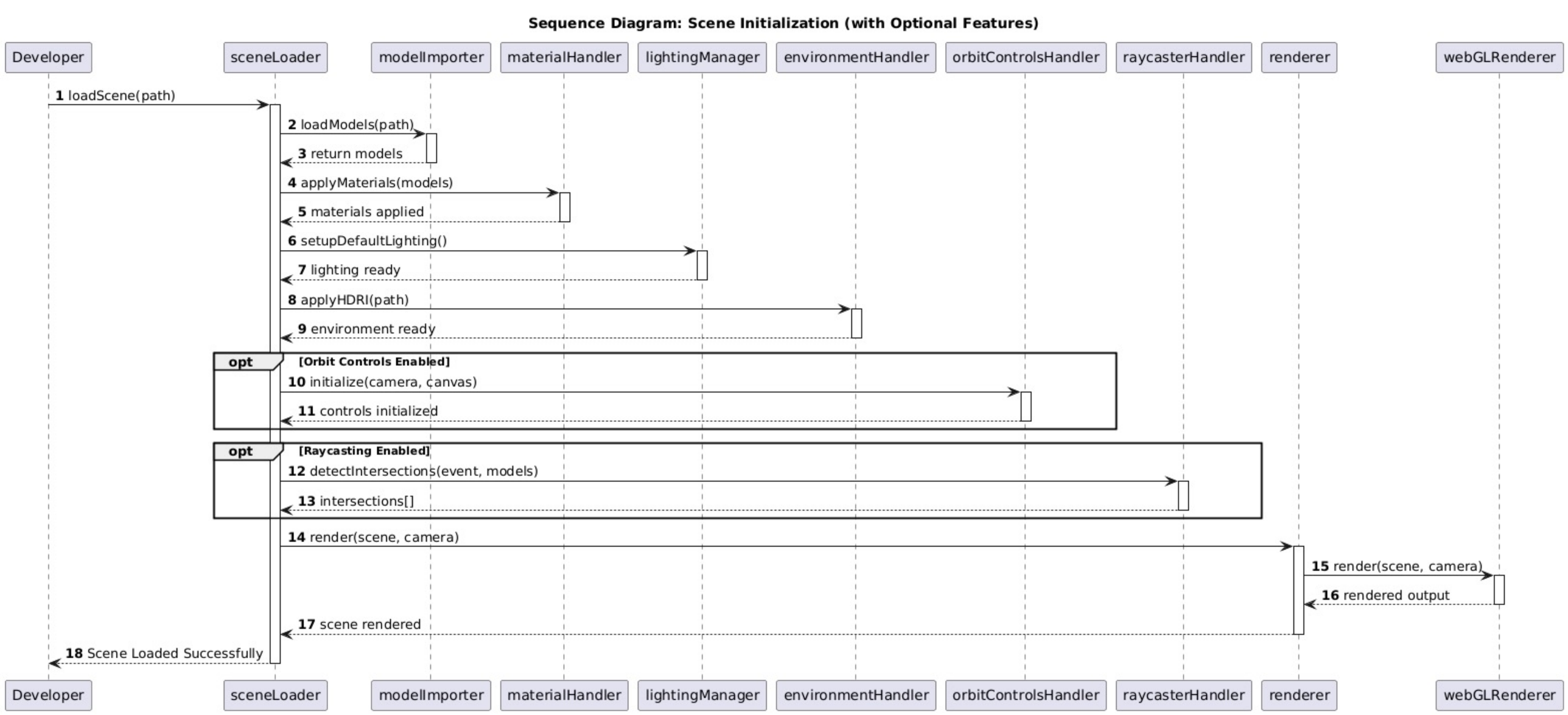

Scene Integration System
ModelImporter
- modelPath: String
- loadedModels: List<Object3D>
+ loadModels(path: String): List<Object3D>
0..*
1
SceneLoader
- scenePath: String
- scene: Scene
- camera: Camera
- renderer: Renderer
+ loadScene(path: String): void
+ parseSceneData(): void
+ initializeScene(): void
«optional»
«optional»
Renderer
- canvas: HTMLCanvasElement
+ render(scene: Scene, camera: Camera): void
uses
WebGLRenderer
+ render(scene, camera)
+ setSize(width, height)
MaterialHandler
- texturePath: String
+ applyMaterials(models: List<Object3D>): void
LightingManager
- lights: List<Light>
+ setupDefaultLighting(): void
EnvironmentHandler
- hdriPath: String
+ applyHDRI(path: String): void
«optional»
OrbitControlsHandler
- controls: OrbitControls
+ initialize(camera: Camera, canvas: HTMLCanvasElement): OrbitControls
+ update(): void
«optional»
RaycasterHandler
- raycaster: Raycaster
- mouse: Vector2
+ detectIntersections(event: MouseEvent, objects: List<Object3D>): List<Intersection>


Object Diagram
ModelImporter
-modelPath
-loadedModels
+loadModels()
-parseOBJ()
SceneLoader
-scenePath
-scene
-camera
-renderer
+loadScene()
-parseSceneData()
-initializeScene()
MaterialHandler
-texturePath
+applyMaterials()
-loadTextures()
LightingManager
-lights
+setupDefaultLighting()
-createDirectionalLight()
EnvironmentHandler
-hdriPath
+applyHDRI()
Renderer
-canvas
-threeRenderer
+render()
OrbitControlsHandler
-controls
+initialize()
+update()
RaycasterHandler
-raycaster
-mouse
+detectIntersections()

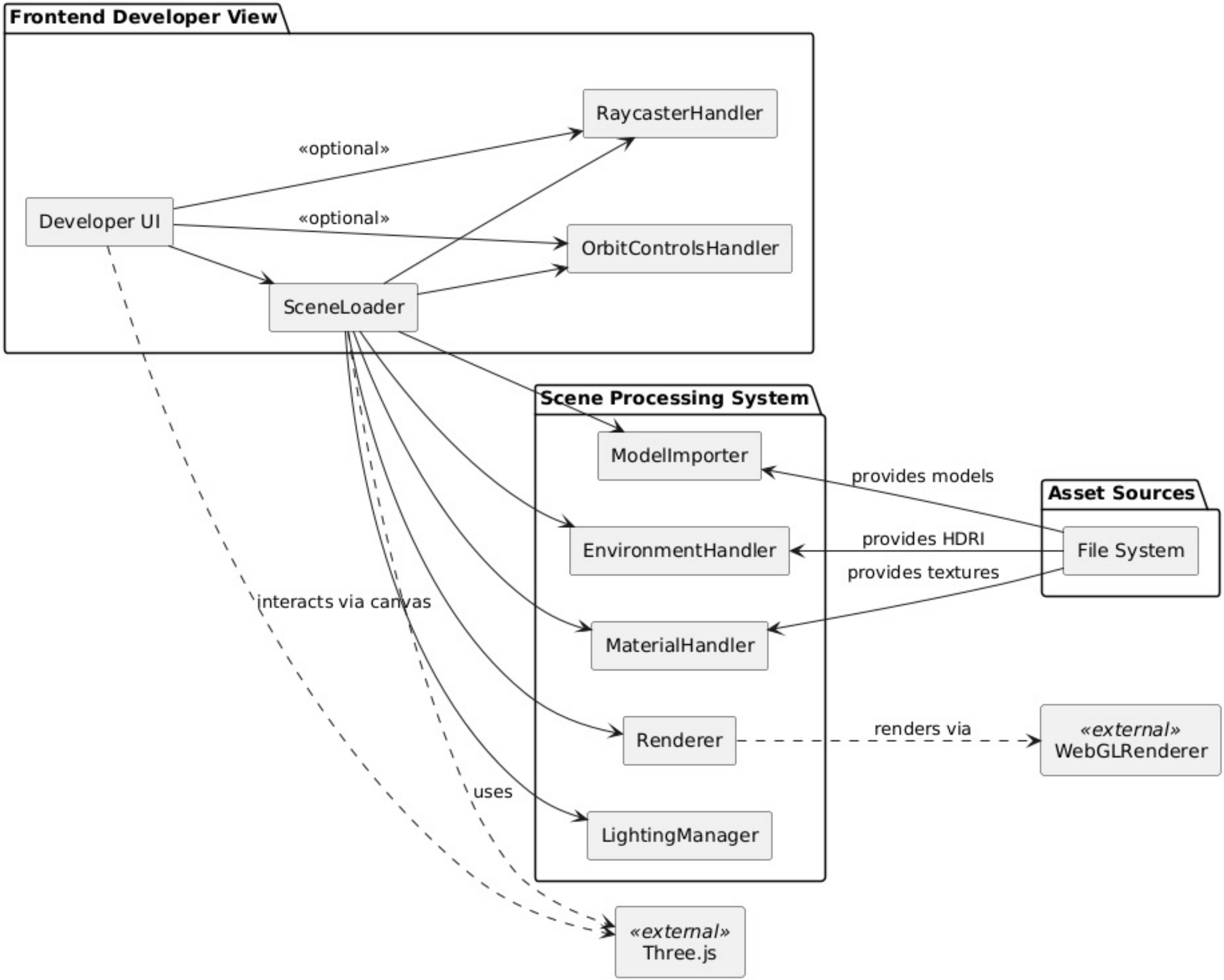
Package Diagram: Scene Integration System

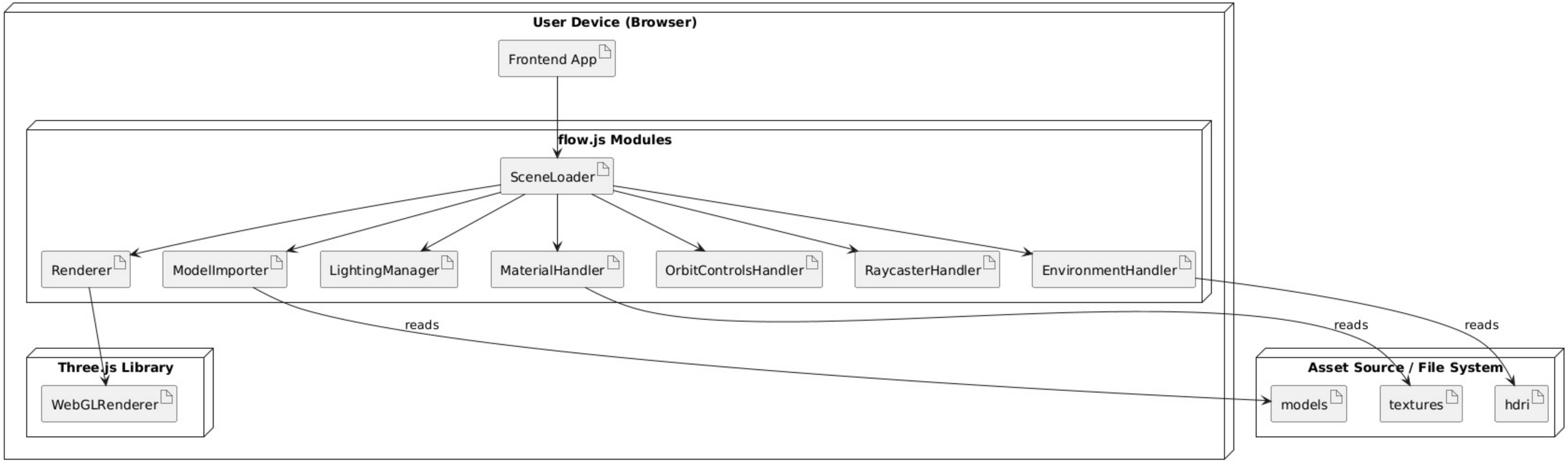



Use Case Diagram: Scene Integration System
Frontend Developer
Blender Exporter
provides 3D assets
Scene Integration System
interacts
SceneLoader
«extend»
«include»
«include»
«include»
«include»
«include»
«extend»
RaycasterHandler
Renderer
ModelImporter
MaterialHandler
LightingManager
EnvironmentHandler
OrbitControlsHandler
«include»
WebGLRenderer
uses
Three.js Engine

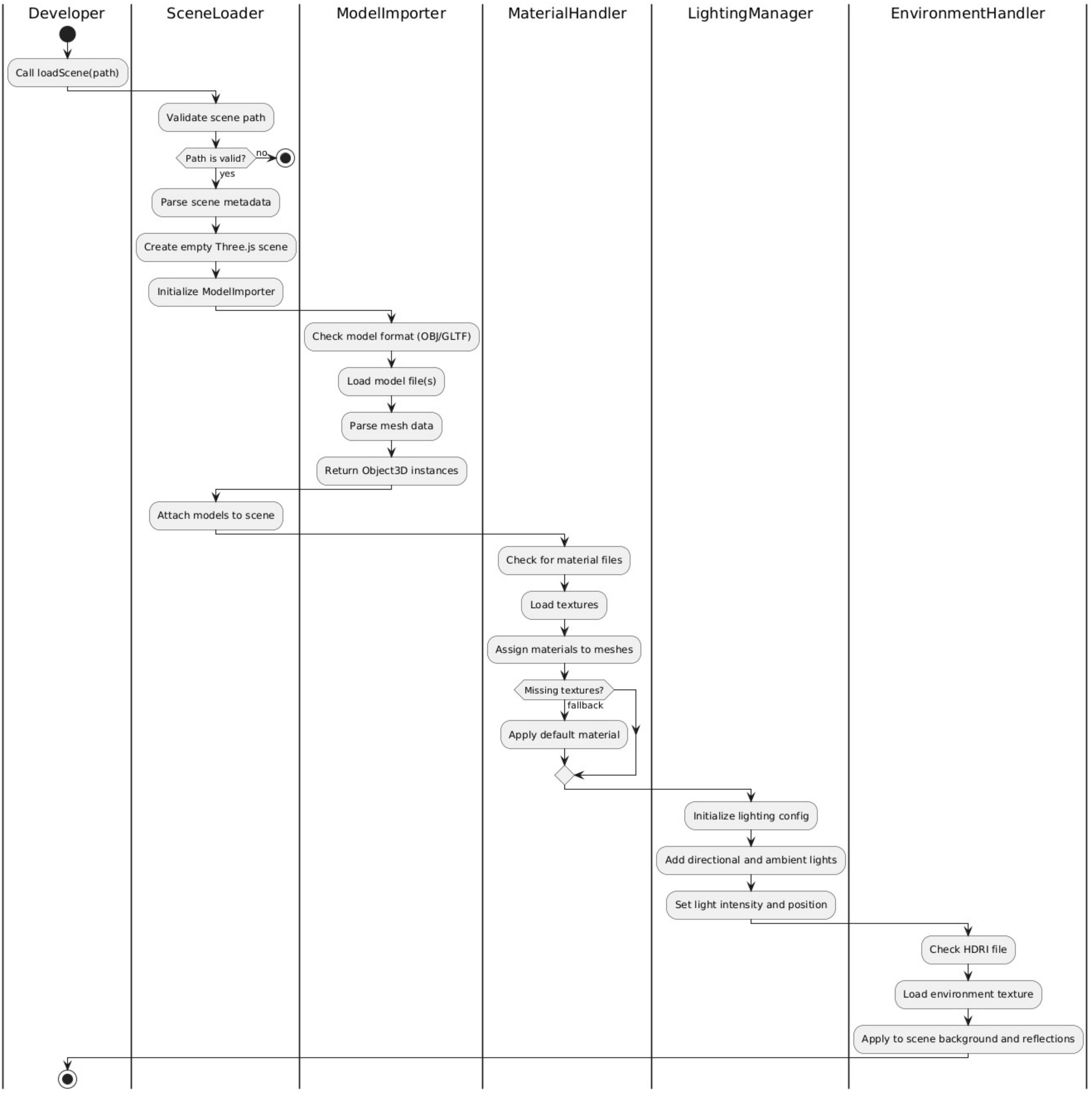

Sequence Diagram: Scene Initialization (with Optional Features)
Developer
sceneLoader
modelImporter
materialHandler
lightingManager
environmentHandler
orbitControlsHandler
raycasterHandler
renderer
webGLRenderer
1 loadScene(path)
2 loadModels(path)
3 return models
4 applyMaterials(models)
5 materials applied
6 setupDefaultLighting()
7 lighting ready
8 applyHDRI(path)
9 environment ready
opt
[Orbit Controls Enabled]
10 initialize(camera, canvas)
11 controls initialized
opt
[Raycasting Enabled]
12 detectIntersections(event, models)
13 intersections[]
14 render(scene, camera)
15 render(scene, camera)
16 rendered output
17 scene rendered
18 Scene Loaded Successfully

Component Diagram: Scene Integration System

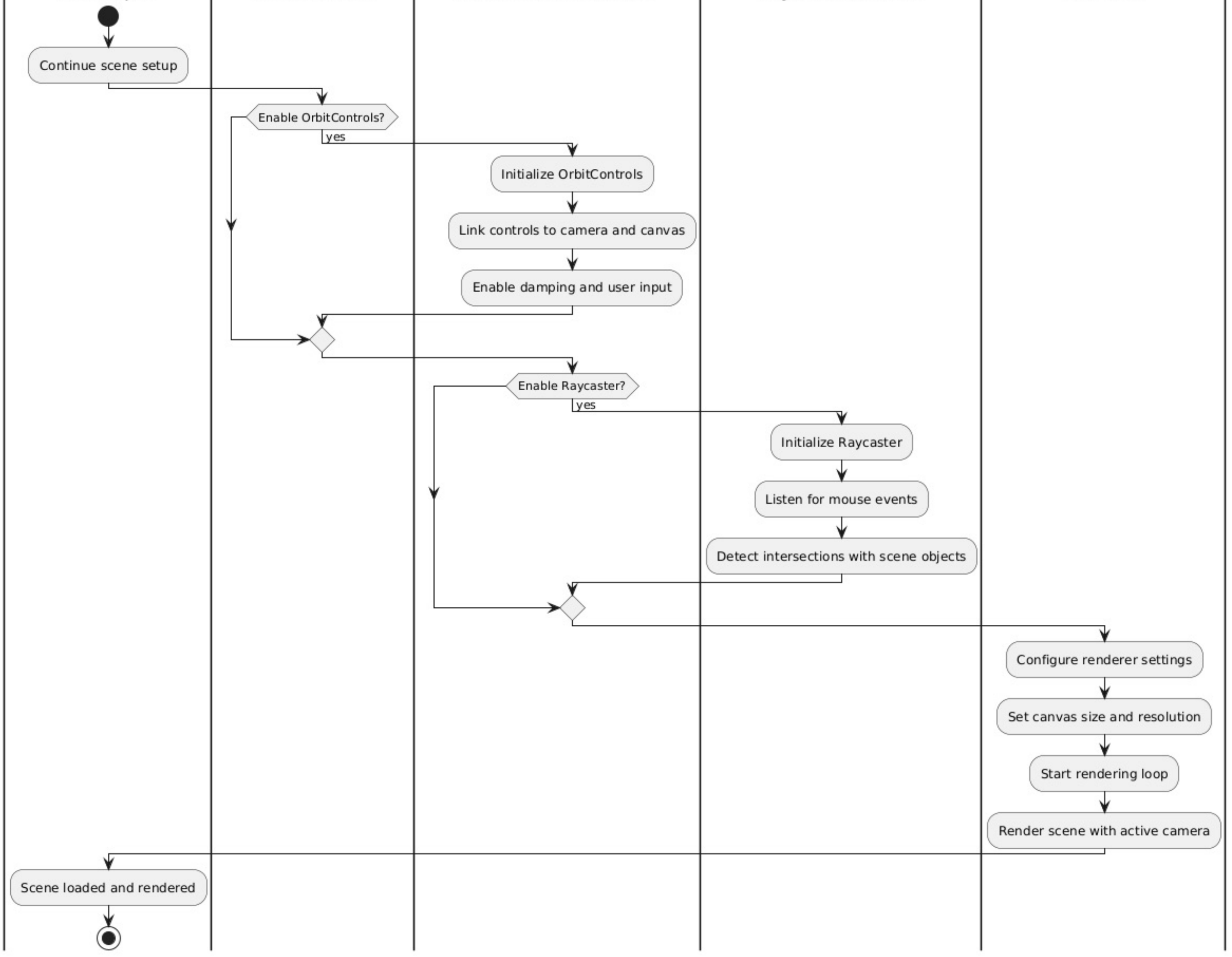



Deployment Diagram: Scene Integration System
User Device (Browser)
Frontend App
flow.js Modules
SceneLoader
Renderer
ModelImporter
LightingManager
MaterialHandler
OrbitControlsHandler
RaycasterHandler
EnvironmentHandler
reads
reads
reads
Three.js Library
WebGLRenderer
Asset Source / File System
models
textures
hdri

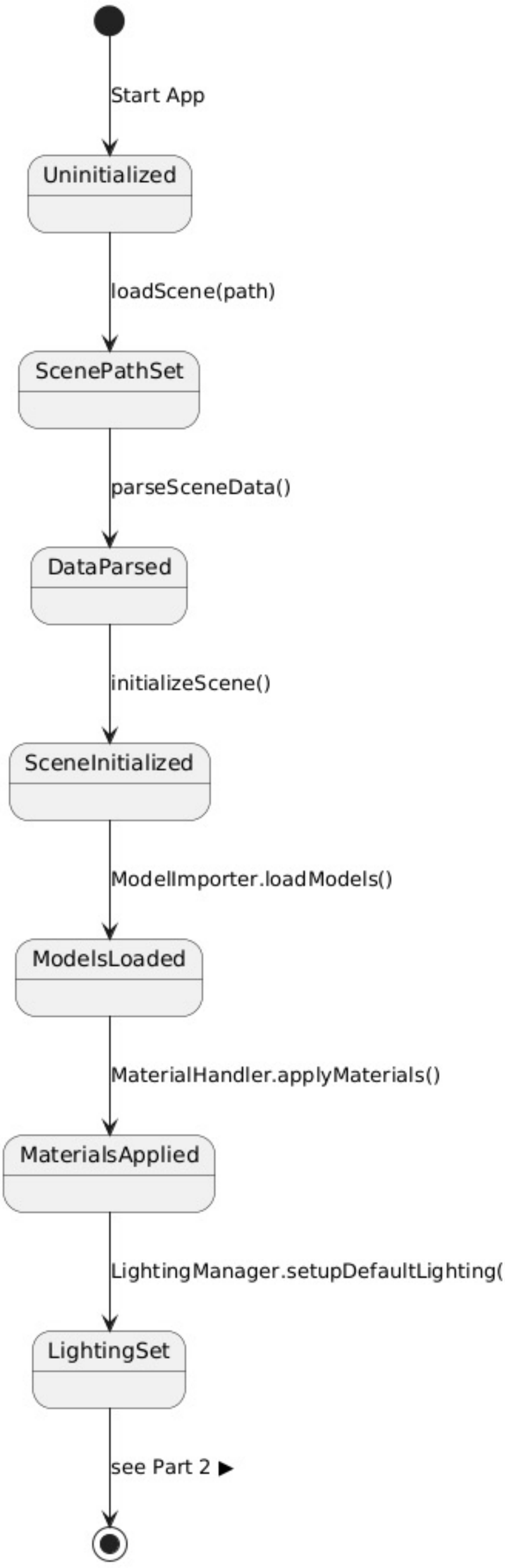
Activity Diagram - Part 1: Scene Setup & Asset Import

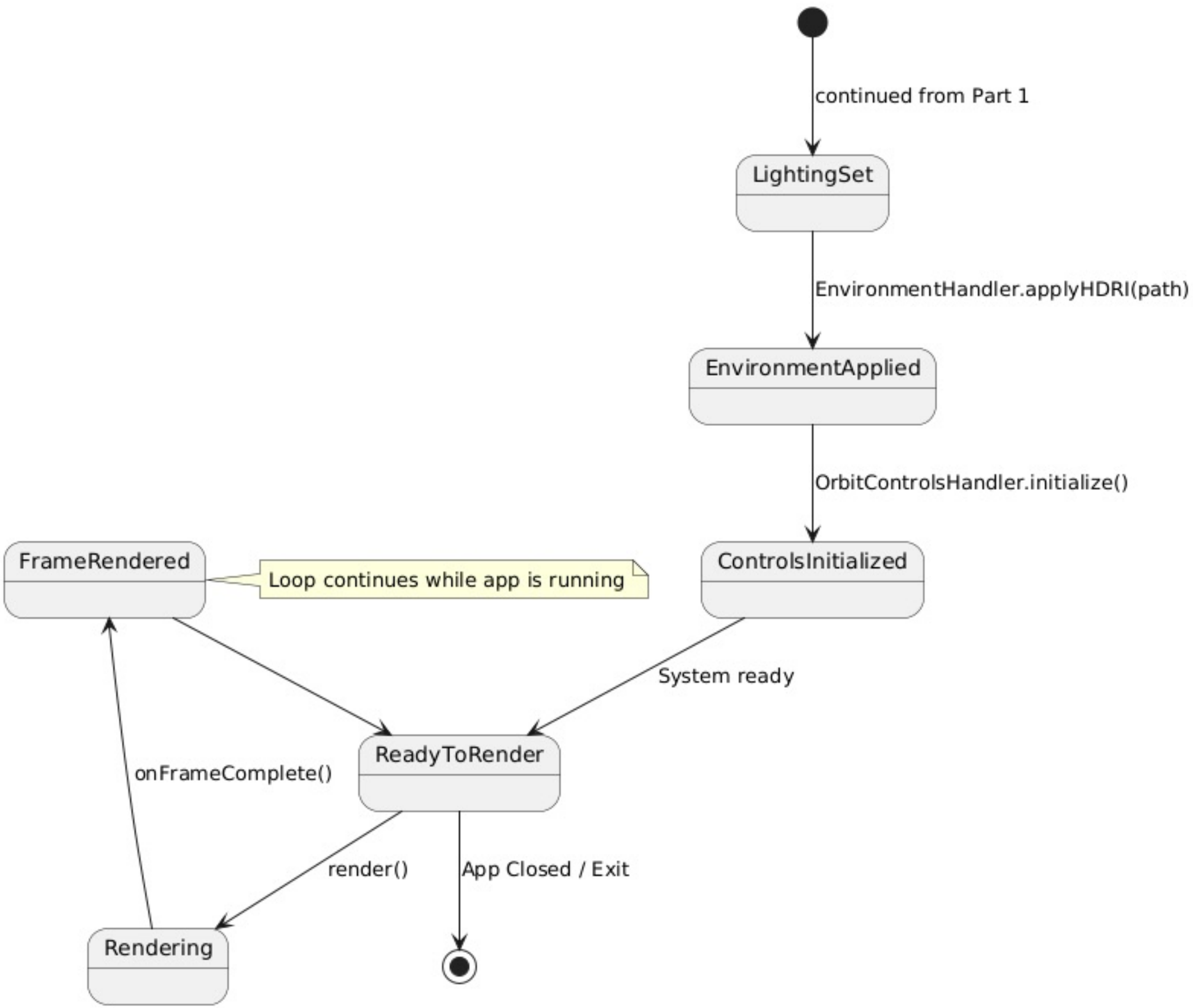


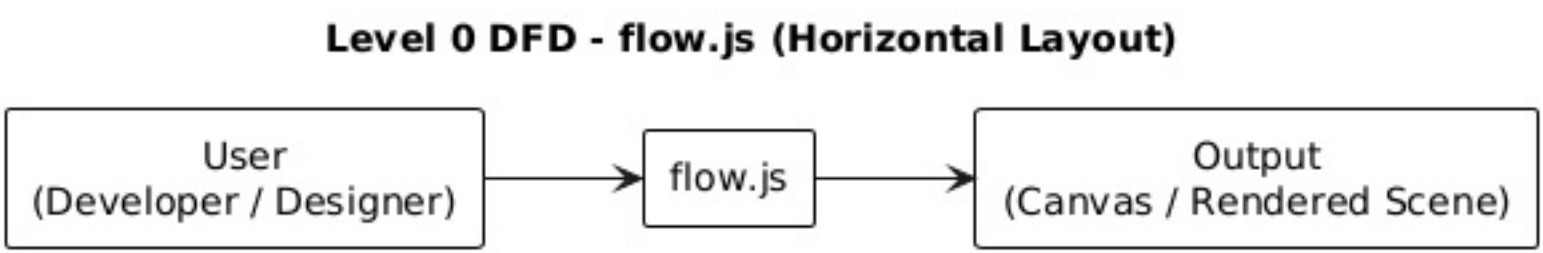
Activity Diagram - Part 2: Optional Features & Rendering

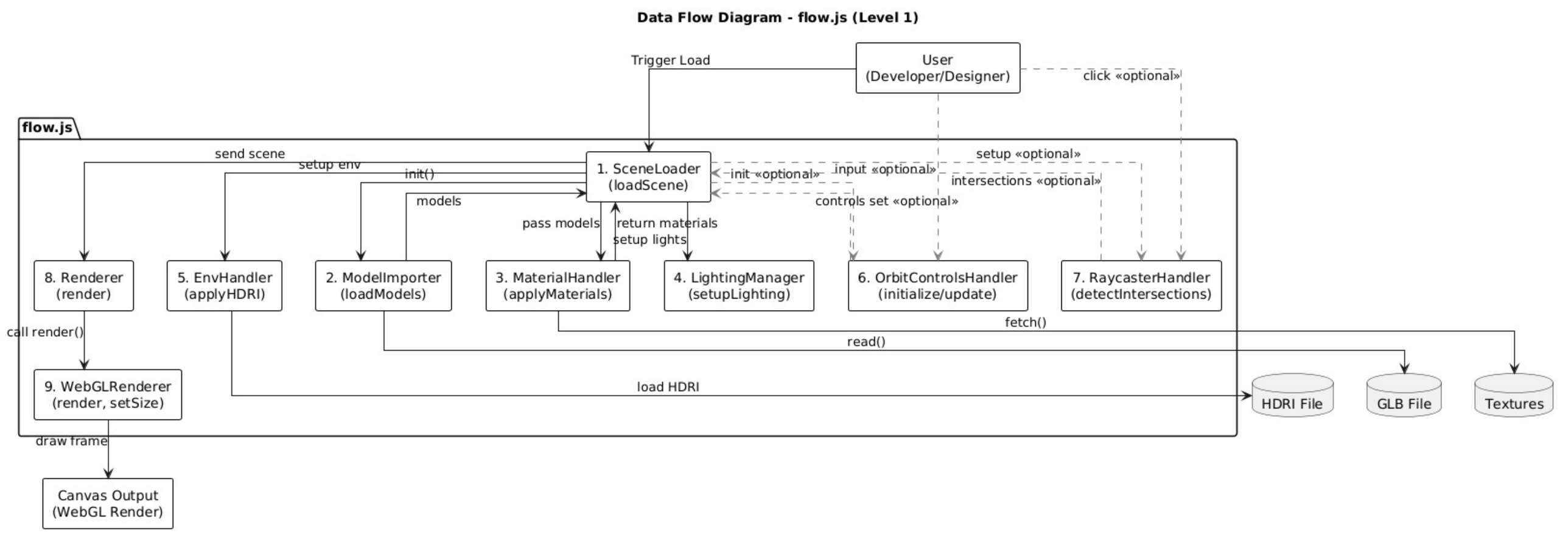


State Diagram (Part 1): Scene Preparation




State Diagram (Part 2): Environment and Rendering
continued from Part 1
LightingSet
EnvironmentHandler.applyHDRI(path)
EnvironmentApplied
OrbitControlsHandler.initialize()
ControlsInitialized
FrameRendered
Loop continues while app is running
System ready
ReadyToRender
onFrameComplete()
render()
App Closed / Exit
Rendering


Level 0 DFD - flow.js (Horizontal Layout)
User
(Developer / Designer)
flow.js
Output
(Canvas / Rendered Scene)


Data Flow Diagram - flow.js (Level 1)
Trigger Load
User
(Developer/Designer)
click «optional»
flow.js
send scene
setup env
init()
models
1. SceneLoader
(loadScene)
init «optional»
input «optional»
setup «optional»
intersections «optional»
controls set «optional»
pass models
return materials
setup lights
8. Renderer
(render)
5. EnvHandler
(applyHDRI)
2. ModelImporter
(loadModels)
3. MaterialHandler
(applyMaterials)
4. LightingManager
(setupLighting)
6. OrbitControlsHandler
(initialize/update)
7. RaycasterHandler
(detectIntersections)
call render()
fetch()
read()
9. WebGLRenderer
(render, setSize)
load HDRI
HDRI File
GLB File
Textures
draw frame
Canvas Output
(WebGL Render)