# Rootex

SDSLabs

Aug 24, 2023

# CONTENTS

Rootex is a Windows based 3D multithreaded game engine written in C++ and powers an in-production game being developed at SDSLabs .

- Issue Tracker: http://github.com/sdslabs/rootex/issues
- Source Code: http://github.com/sdslabs/rootex
- Discord : https://discord.gg/ZDxvjm9dX8

CHAPTER

ONE

# SUPPORT

If you are having issues, please let us know. SDSLabs has a public chat channel at http://chat.sdslabs.co

CHAPTER

TWO

# LICENSE

The project is licensed under the MIT license. See *THIRDPARTY.md* for thirdparty licenses.

## 2.1 Getting Started

Rootex is a pure Entity-Component-System architectured game engine. We use terms like scenes, entities, components and systems analogous to the domain of Scene trees and ECS architecture. Find more information about these here

**Note:** This also means that the Rootex Editor is made with the Rootex Engine itself.

Rootex Editor is structured to work like popular game engines with its simplistic user interface.

Any user coming from ECS based game engines should be able to pick up the interface quickly.

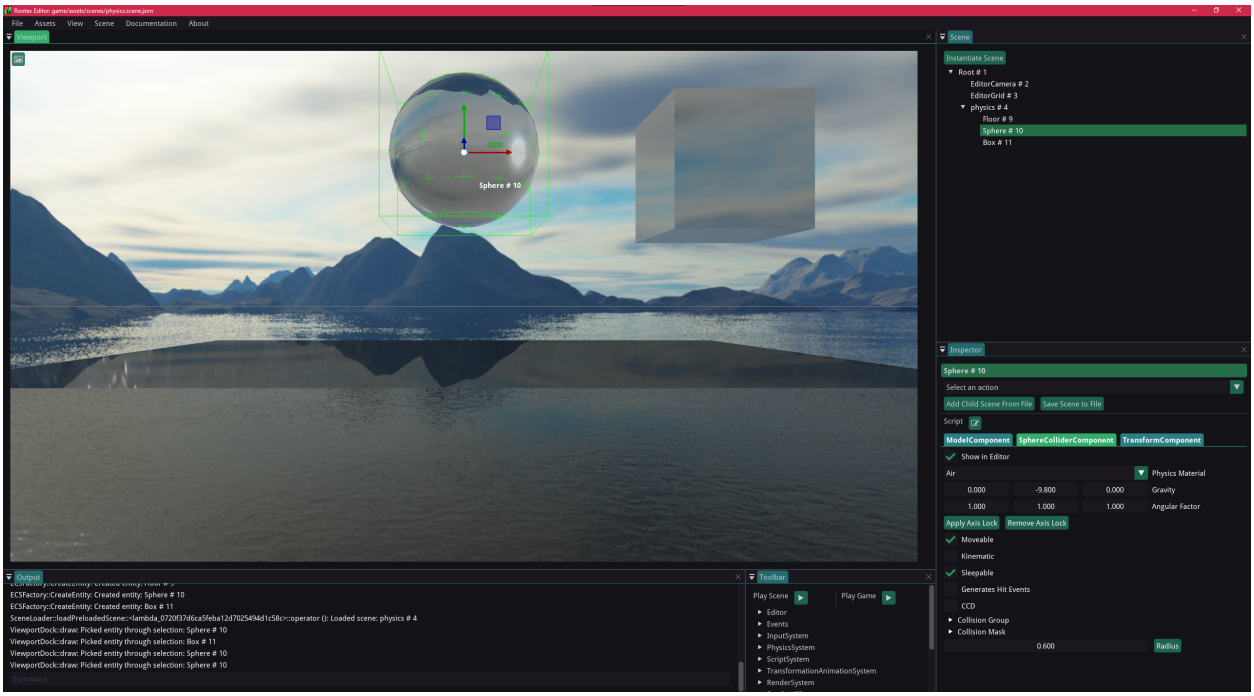
### 2.1.1 Rootex Editor

Rootex comes with a separate editor for making games, called the Rootex Editor.

Rootex Editor is built using Dear ImGui . The editor UI is subject to change but the overall workings shall remain the same.

#### Running the Editor

1. Run the editor by opening the editor executable.
2. Once editor is open you are greeted by the editor UI.
3. Try to open a scene from the File menu > Open Scene option.

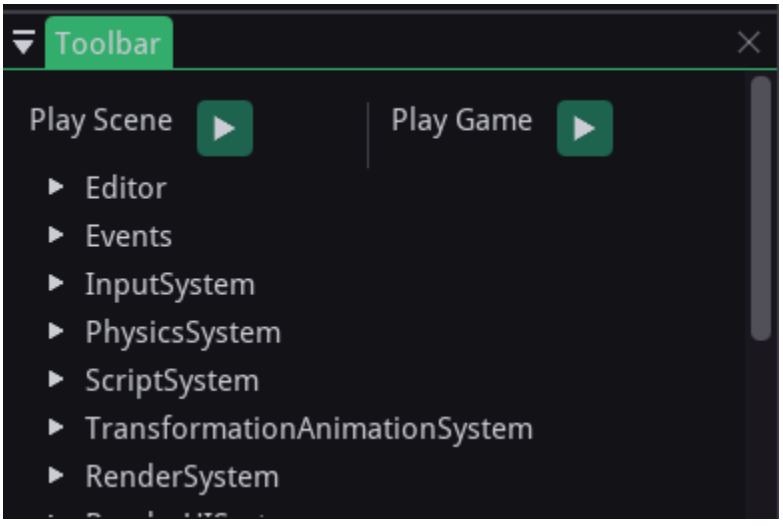



**Note:** If you get an error along the lines of `dxgidebug.dll not loaded` while opening the executable, install **Graphics Tools** by following this guide.

### 2.1.2 Editor Layout

The main editor area is laid out in 6 major sections.

#### Toolbar Dock

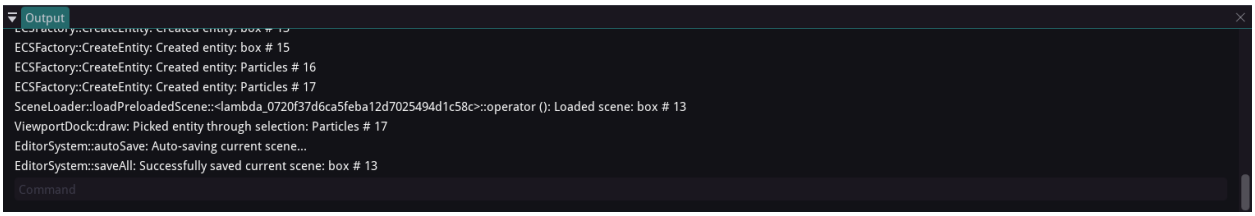



The toolbar dock displays different settings that affect the editor's view of the game world and modify the Rootex Engine's overall state.

You can find data related to the Editor FPS, registered Events in the EventManager, the current camera being used to view the world, etc. You can try fiddling with the settings here to know what each thing does. The toolbar dock also allows playing the currently open level in game or play the game from the original starting level as defined in the game settings.

**Tip:** Try setting the Camera in RenderSystem tab to EditorCamera to allow easy navigation.

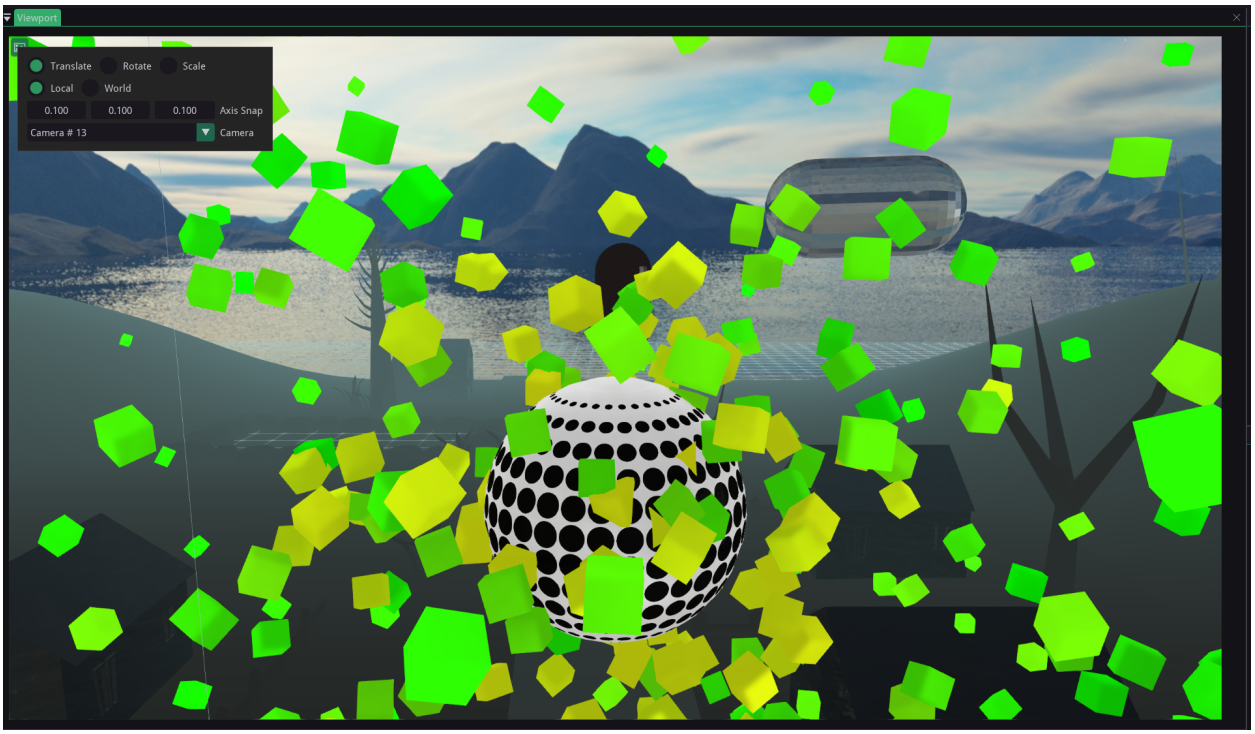
### Output Dock




The output dock is the Rootex Engine's channel to report stuff happening internally in the engine and in the editor. You can expect to see error messages, warnings and plain reports in the output dock. You will also notice a text input bar at the bottom of the output dock.

Use the command input to run Lua code in Rootex's Lua VM. All of Rootex's scripting API is available through this command line.

### Viewport Dock




The viewport dock provides the view into the game world through the default camera, though you want to change that to EditorCamera. The EditorCamera is an editor-only entity. Viewing the game world in the editor also enables a few

perks that are only accessible in the editor and not the game. The view mode for the game world can be changed using the View main menu, usually present at the top of the window, alongside the File main menu and others.

### EditorCamera

The EditorCamera is an editor-only entity which is setup to be the view of the editor into the game world.

To view the world though EditorCamera, select EditorCamera as the current camera entity from *Viewport > Current Camera*. The EditorCamera can be controlled from the editor by holding Right Mouse Button and using WASD/Space/Shift to move. You can tweak the camera turning speed and the moving speed.

### Gizmo

The 3D gizmo is an editor-only tool to let the user change the position, rotation, and scale of selected entity in either local or world space. Scenes can be selected from the scene dock.

The gizmo has 3 separate modes of working.

- Translation

Select an entity and press Q. In this mode the gizmo takes the shape of 3 axes point in orthogonal directions. These axes are selectable with the mouse pointer and position of entities can be altered by dragging.

- Rotation

Select and entity and press W. In this mode the gizmo takes the shape of 3 circles with their axes going in orthogonal directions. These circles denote the rotation of the entities in Euler angles and rotation of the entities can be altered by dragging.

- Scale

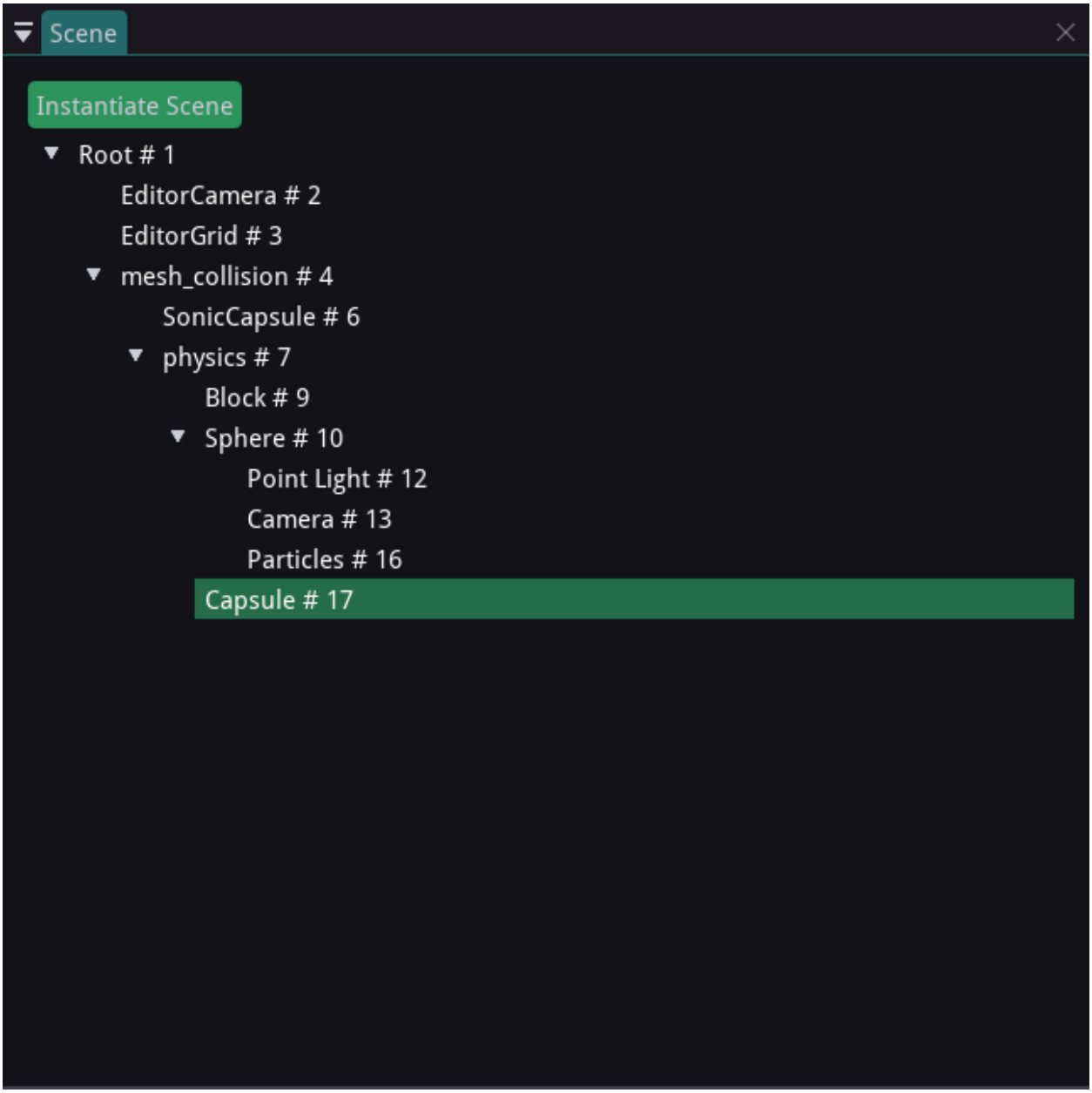
Select an entity and press E. In this mode the gizmo takes the shape of 3 axes in orthogonal directions. These axes denote the scales which can be altered by dragging.

The gizmo has 2 modifiers to each of the modes. The Local modifier will apply changes in the local coordinate system. The World modifier will apply changes in the world coordinate system.

### EditorGrid

There is one more editor-only entity that is helpful to the viewport. The EditorGrid displays the grid defined by the grid cells sizes. You can alter the grid settings by selecting EditorGrid in the scene dock.

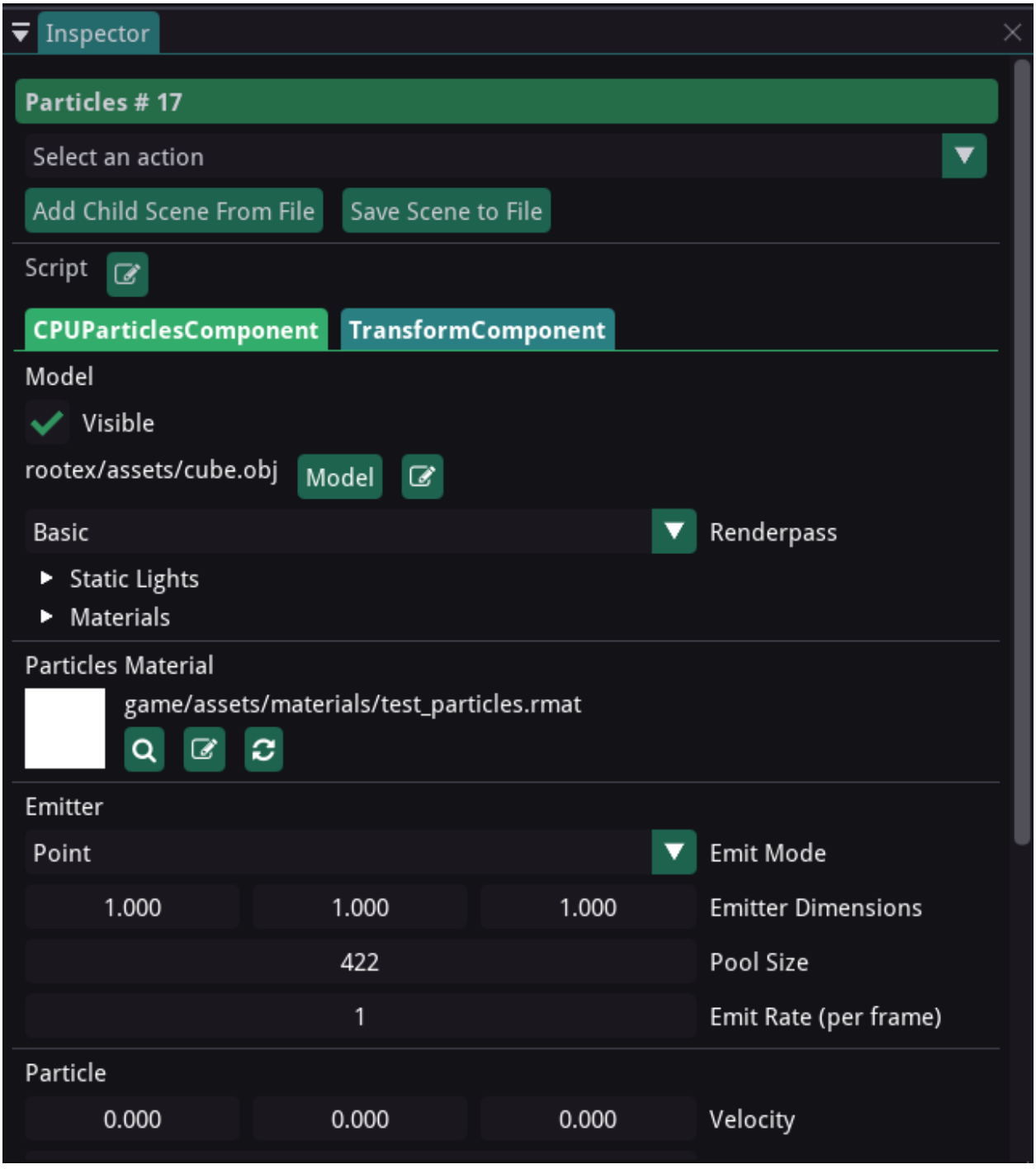
### Scene Dock




The scene dock displays the parent-child hierarchy of scenes in the current game world.

The hierarchy between scenes is defined by the Scene class and its children. Scenes can be selected by clicking on their name in the scene dock or selecting the associated entity in the inspector. You can also change the hierarchy between scenes by dragging and dropping the scene over your chosen parent scene.

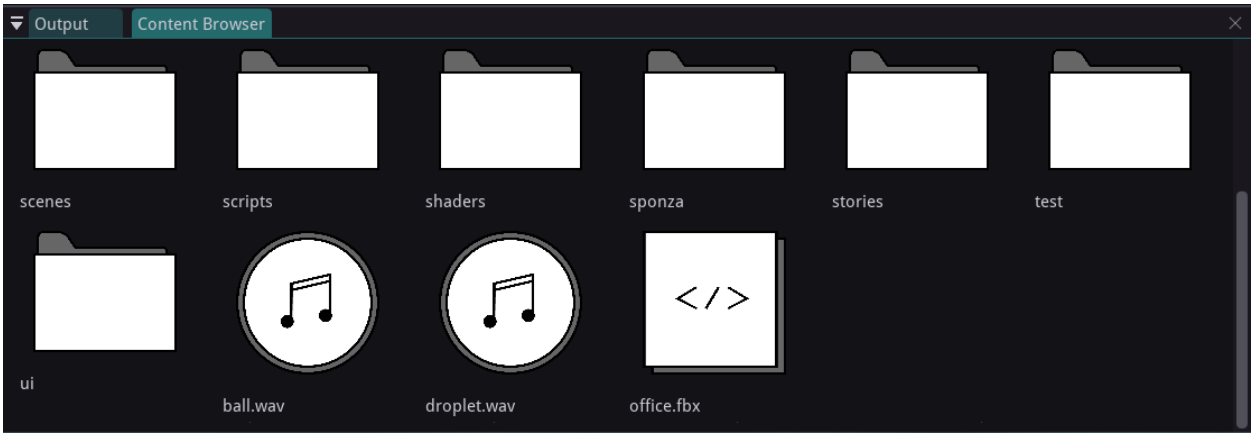
### Inspector Dock




Inspector dock is the main hub of all data related to components in an entity. Data under each component is available for change using the inpector dock. Use the scene dock, or click on the scene in the viewport to select them.

Inspector dock also allows changing the name of the scene, attaching Lua scripts, adding or changing or removing components, resetting inter-component linkages and deleting entities, along with instantiating new scenes as children from files and saving scenes to files.

### Content Browser Dock




The Cotent Browser allows access to the filesystem in the game/ directory withing the engine itself.

Content Browser can recognize supported filetypes and shows special icons for them, which can directly be dragged and dropped into suitable places.

Current drag and drop support:

- Image -> Texture slots in Materials
- Audio -> Music source track for MusicComponent and ShortMusicComponent
- Model -> 3D Mesh for ModelComponent, Rigged skeletal mesh for AnimatedModelComponent, Collision mesh in MeshColliderComponent
- Material -> Custom .rmat file format for RenderableComponent
- Script -> Lua files for entity scripts, RML file for UIComponent

## 2.1.3 Animating Objects using TransformAnimationComponent

TransformAnimationComponent can be used to easily animate scenes which have to be animated endlessly without relying on scripting for the same.

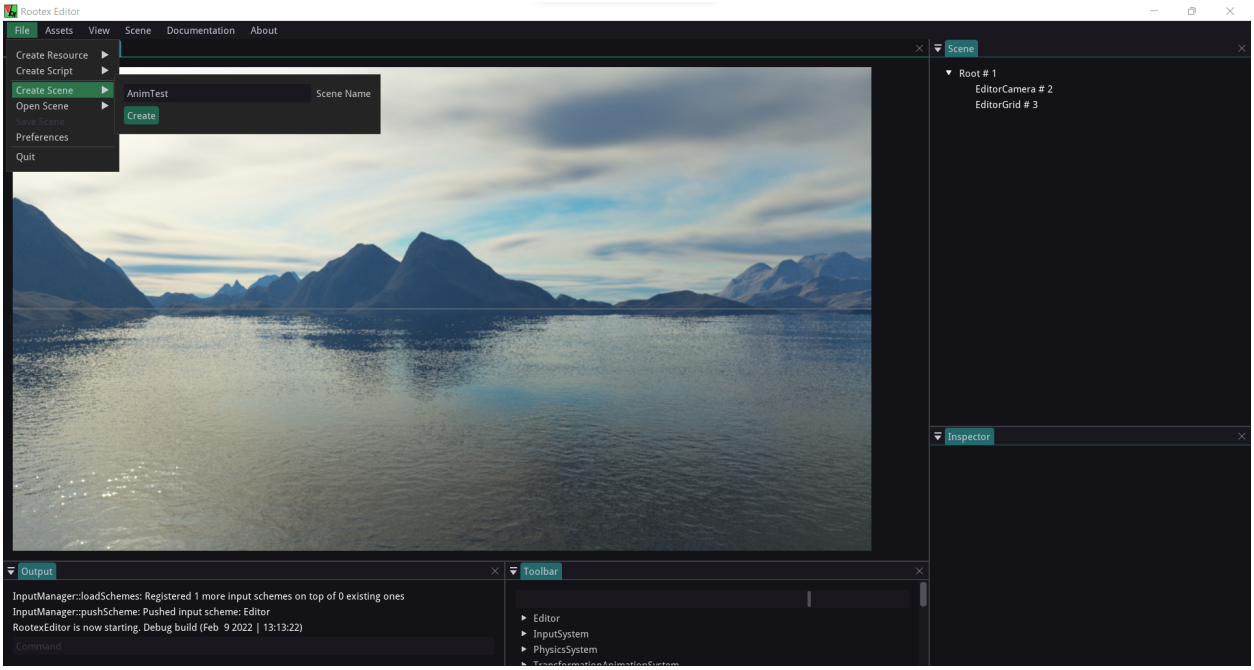
It is quite easy to as it uses a keyframe based interface to achieve this.

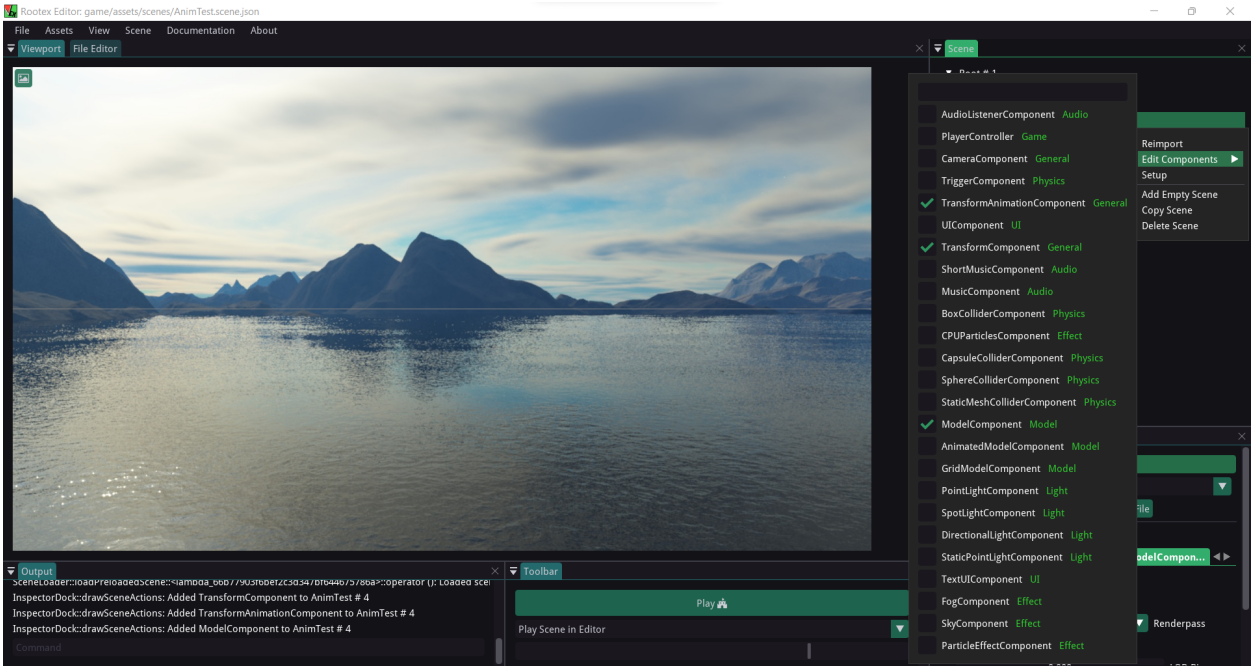
### Setting up

Create a new scene and assign the TransformComponent, ModelComponent and TransformAnimationComponent to it.

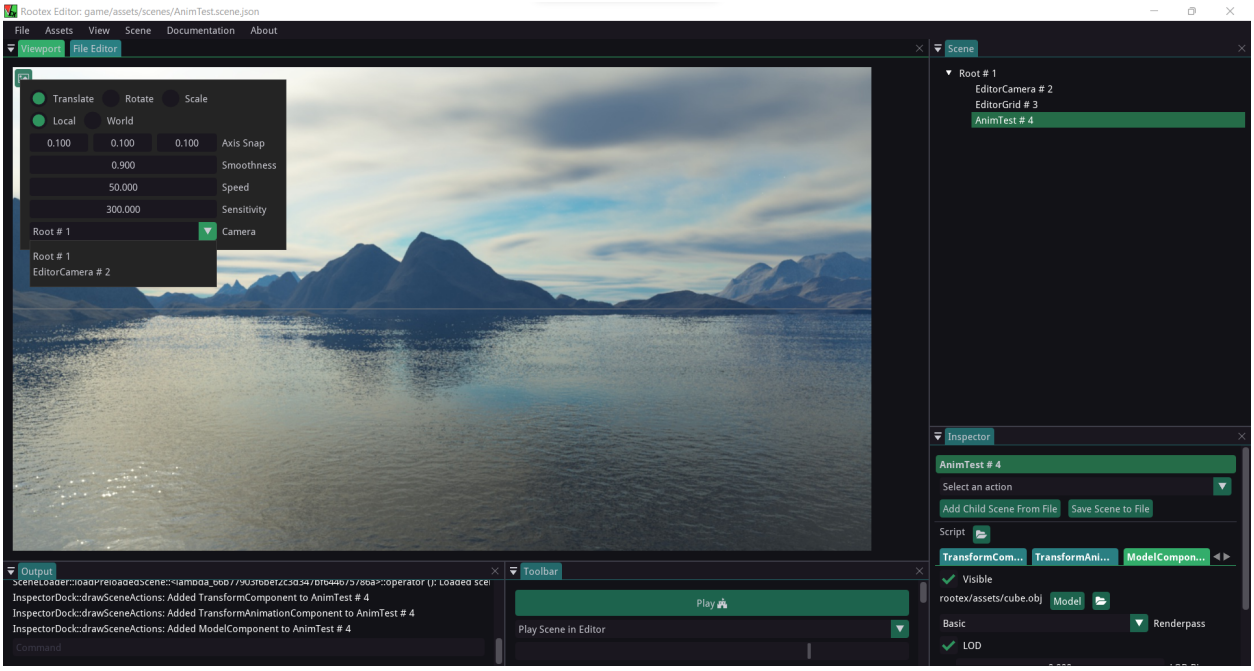
Creating a new scene:

Adding the Components by right clicking on the scene in inspector:


Rootex Editor
File
Assets
View
Scene
Documentation
About
Create Resource
Create Script
Create Scene
Open Scene
Preferences
Quit
AnimTest
Scene Name
Create
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
Inspector
Output
InputManager::loadSchemes: Registered 1 more input schemes on top of 0 existing ones
InputManager::pushScheme: Pushed input scheme: Editor
RootexEditor is now starting. Debug build (Feb 9 2022 | 13:13:22)
Command
Toolbar
Editor
InputSystem
PhysicsSystem


Rootex Editor: game/assets/scenes/AnimTest.scene.json
File
Assets
View
Scene
Documentation
About
Viewport
File Editor
Scene
AudioListenerComponent Audio
PlayerController Game
CameraComponent General
TriggerComponent Physics
TransformAnimationComponent General
UIComponent UI
TransformComponent General
ShortMusicComponent Audio
MusicComponent Audio
BoxColliderComponent Physics
CPUParticlesComponent Effect
CapsuleColliderComponent Physics
SphereColliderComponent Physics
StaticMeshColliderComponent Physics
ModelComponent Model
AnimatedModelComponent Model
GridModelComponent Model
PointLightComponent Light
SpotLightComponent Light
DirectionalLightComponent Light
StaticPointLightComponent Light
TextUIComponent UI
FogComponent Effect
SkyComponent Effect
ParticleEffectComponent Effect
Reimport
Edit Components
Setup
Add Empty Scene
Copy Scene
Delete Scene
Renderpass
Output
InspectorDock::drawSceneActions: Added TransformComponent to AnimTest # 4
InspectorDock::drawSceneActions: Added TransformAnimationComponent to AnimTest # 4
InspectorDock::drawSceneActions: Added ModelComponent to AnimTest # 4
Command
Toolbar
Play
Play Scene in Editor

Select the EditorCamera:

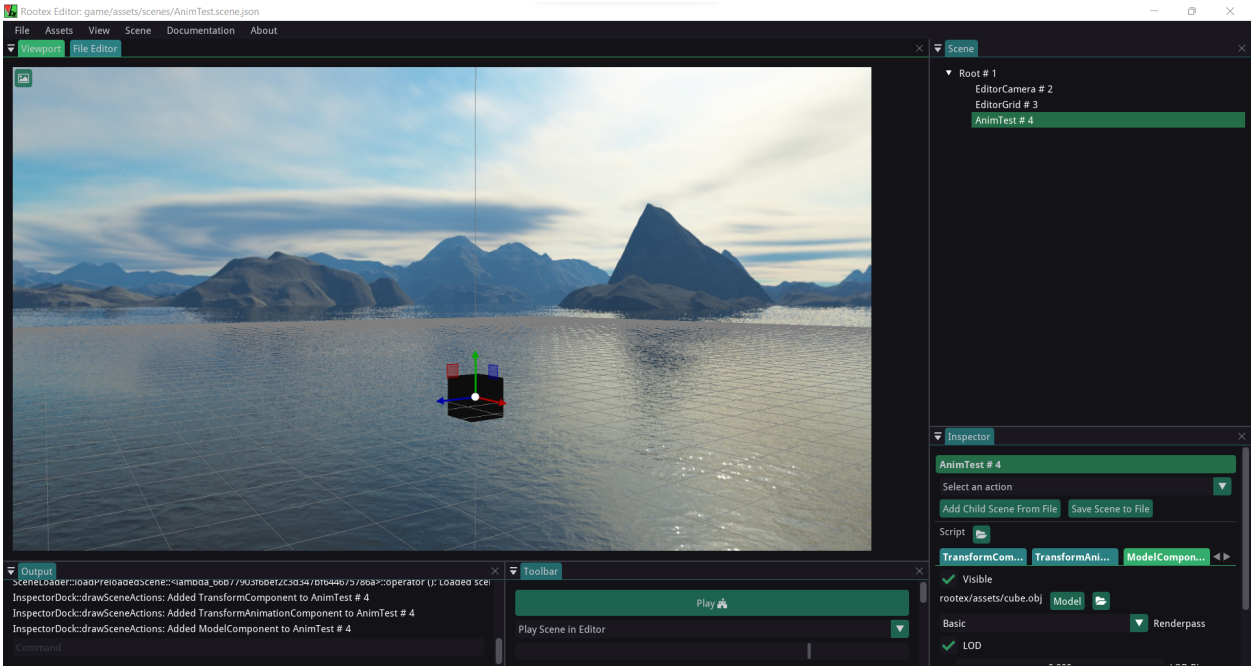


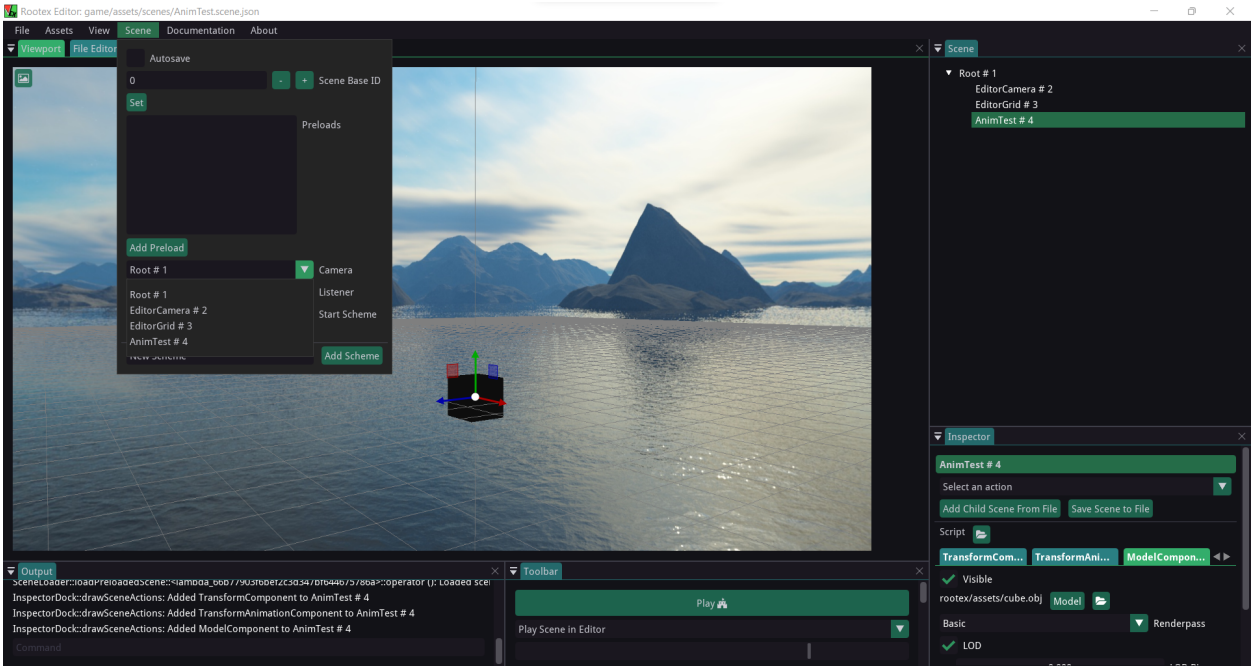

Set the camera to EditorCamera and position it such that it has a good view of the scene.

Select the EditorCamera as default camera for the scene.

## Interface

- **Transition Type** : How the transition must be between keyframes. Following are the possible values for the Transition Type:
  - **SmashSmash** : Abrupt start and end transition
  - **EaseEase** : Smooth start and end transition
  - **SmashEase** : Abrubt start and smooth end transition
  - **EaseSmash** : Smooth start and abrupt end transition
- **Animation Mode** : How the animation should play. Explaining the options:
  - **Looping** : Animation plays from Start time to End time unidirectionally.

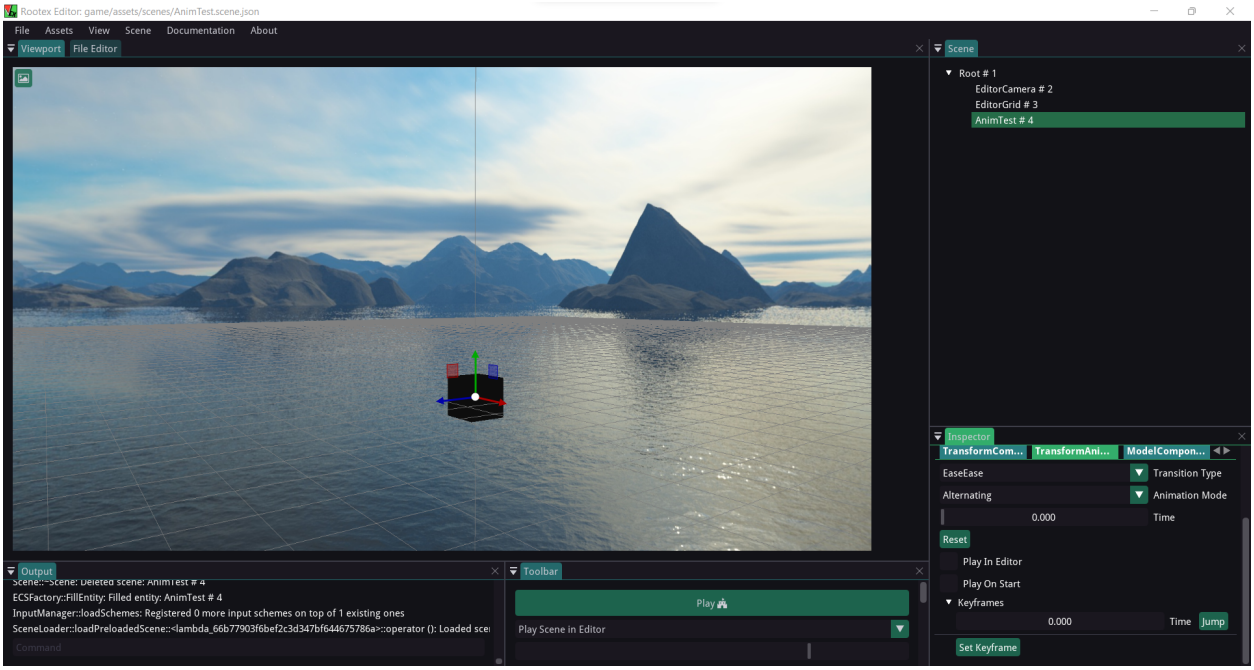

Rootex Editor: game/assets/scenes/AnimTest.scene.json
File
Assets
View
Scene
Documentation
About
Viewport
File Editor
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
AnimTest # 4
Inspector
AnimTest # 4
Select an action
Add Child Scene From File
Save Scene to File
Script
TransformCom...
TransformAni...
ModelCompon...
Visible
rootex/assets/cube.obj
Model
Basic
Renderpass
LOD
Output
InspectorDock::drawSceneActions: Added TransformComponent to AnimTest # 4
InspectorDock::drawSceneActions: Added TransformAnimationComponent to AnimTest # 4
InspectorDock::drawSceneActions: Added ModelComponent to AnimTest # 4
Command
Toolbar
Play
Play Scene in Editor


Rootex Editor: game/assets/scenes/AnimTest.scene.json
File
Assets
View
Scene
Documentation
About
Viewport
Autosave
Scene Base ID
Set
Preloads
Add Preload
Root # 1
Camera
Listener
Start Scene
Add Scheme
Root # 1
EditorCamera # 2
EditorGrid # 3
AnimTest # 4
Scene
Inspector
AnimTest # 4
Select an action
Add Child Scene From File
Save Scene to File
Script
TransformCom...
TransformAni...
ModelCompon...
Visible
rootex/assets/cube.obj
Model
Basic
Renderpass
LOD
Output
InspectorDock::drawSceneActions: Added TransformComponent to AnimTest # 4
InspectorDock::drawSceneActions: Added TransformAnimationComponent to AnimTest # 4
InspectorDock::drawSceneActions: Added ModelComponent to AnimTest # 4
Command
Toolbar
Play
Play Scene in Editor

  - **Alternating** : Animation plays from Start time to EndTime and then reverses from End time to Start time.
- **Time** : Shows the time while animation plays
- **Reset** : Resets the animation
- **Play in Editor** : Play the animation in editor
- **Play on start** : Play the animation when game starts
- **Keyframes** : To set the animation keyframes
  - **Set Keyframe** : To add a new keyframe at the end
  - **Pop Keyframe** : To remove the last keyframe

### Keyframes

Open the keyframes dropdown to get a list of keyframes.




Each keyframe has a timestamp and a jump button.

More keyframes can be added using the Set Keyframe Button. For now We will have 3 Keyframes.

Now, to animate any object, we need to set it to its desired location for a given keyframe and specify the time in the keyframe.

Let's see how.

For example, if I want our object to move up to X,Y = 0,5 and have a certain rotation at 0.5 sec of our rotation:

What I did here is entered a Keyframe using the **Jump** button next to it, gave the object its new Transform values (which I want at that keyframe), set the transform using the **exit jump** button and gave it a timestamp of 0.5 sec.

Setting Another Keyframe.

Playing the Animation:

Popping a Keyframe removes the last keyframe.

Playing after popping the last keyframe:

This way, more keyframes can be added using **Set Keyframe** to add more steps to the animation or keyframes can be removed using **Pop Keyframe**.

### Transition Type Examples

Examples of different transition types for better understanding.

SmashSmash:

EaseEase:

SmashEase:

EaseSmash:

### Animation Mode Examples

Showing demo of each mode for better understanding.

None: Animation plays only once.

Looping: Animation plays unidirectionally and repeats afteer ending.

Alternating: Animation plays back and forth (bidirectionally).

Check the time progressbar for more clarity.

### Reset

Resets the animation to the starting frame (Time 0.00).

### Play in Editor

Plays the Animation in EditorView.

#### Play on Start

Plays the Animation on game start.

### 2.1.4 Exploring the Graphical capabilities of Rootex

This documentation aims to showcase the graphical capabilities of rootex and act as a tutorial for beginners to get started.

Let's start by creating a scene.

#### Create a scene

To Create a Scene

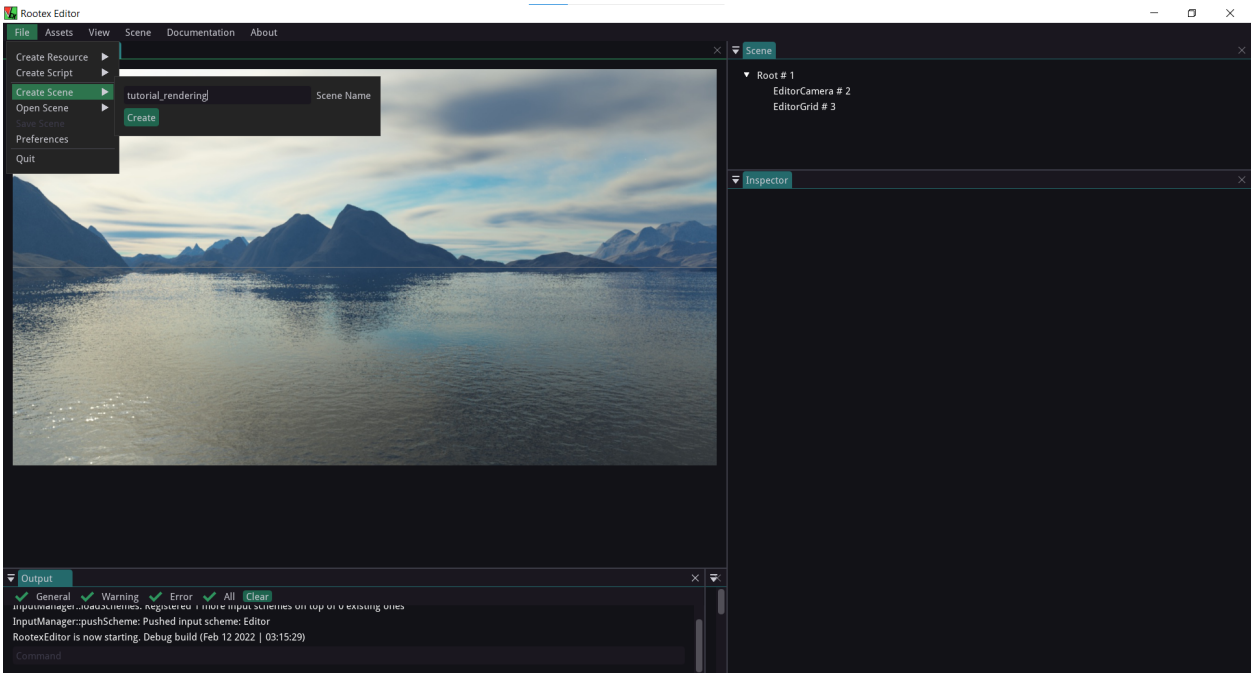
1) Go to *file->CreateScene*.
2) Name the scene and click create.

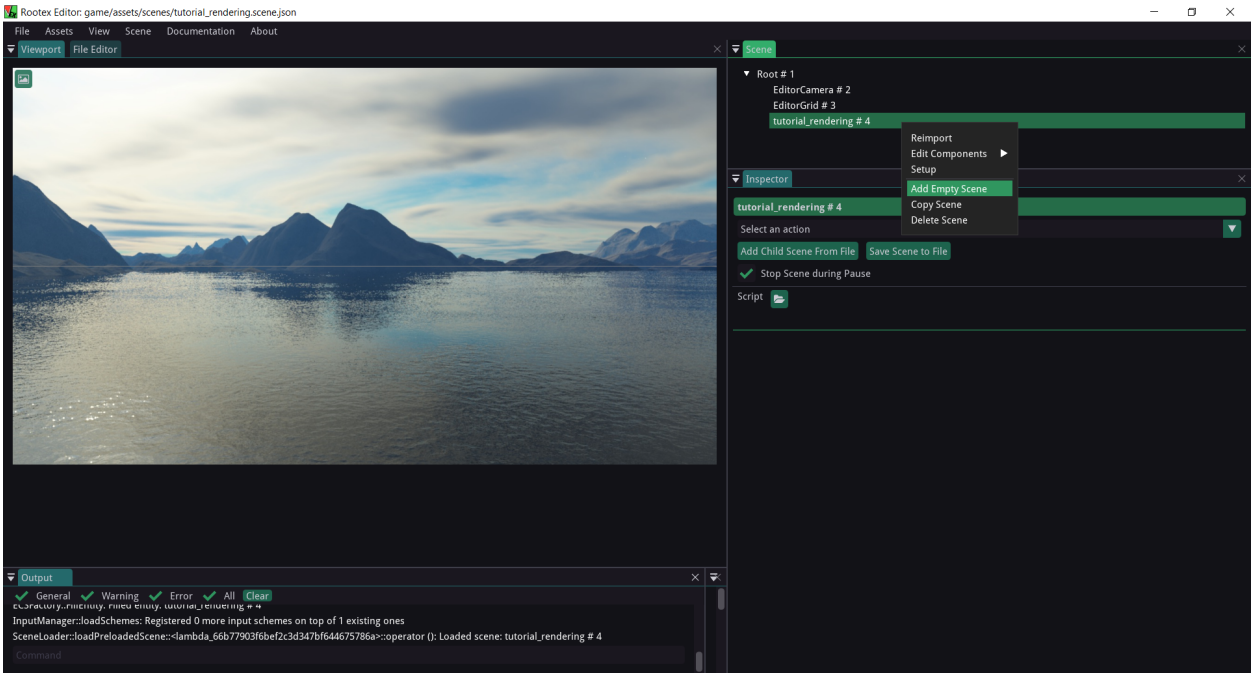
Now we Create an Empty Scene. An empty scene is nothing but objects. You can have different components in it, more on that later.

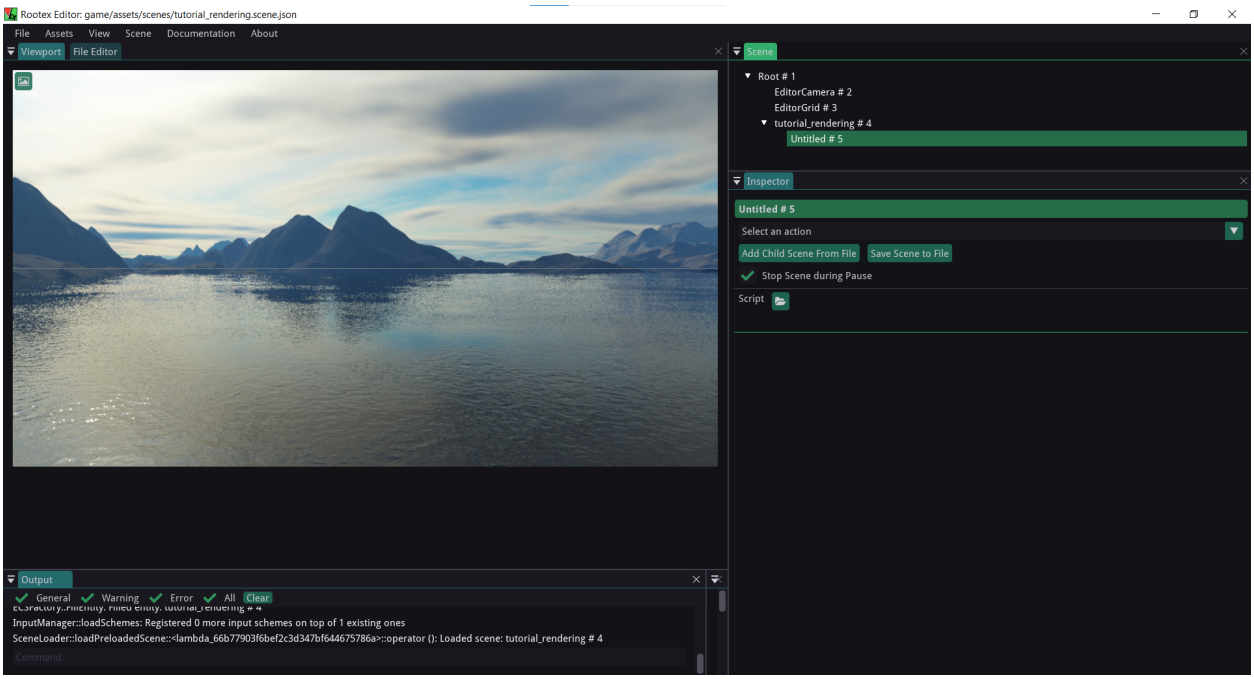
#### Create Empty scene

To create an empty scene.

1) Right-click the root scene.
2) Click Add Empty Scene


Rootex Editor
File
Assets
View
Scene
Documentation
About
Create Resource
Create Script
Create Scene
Open Scene
Save Scene
Preferences
Quit
tutorial_rendering
Scene Name
Create
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
Inspector
Output
General
Warning
Error
All
Clear
InputManager::pushScheme: Pushed input scheme: Editor
RootexEditor is now starting. Debug build (Feb 12 2022 | 03:15:29)
Command


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File
Assets
View
Scene
Documentation
About
Viewport
File Editor
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
Reimport
Edit Components
Setup
Add Empty Scene
Copy Scene
Delete Scene
Inspector
tutorial_rendering # 4
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
Output
General
Warning
Error
All
Clear
InputManager::loadSchemes: Registered 0 more input schemes on top of 1 existing ones
SceneLoader::loadPreloadedScene::<lambda_66b77903f6bef2c3d347bf644675786a>::operator (): Loaded scene: tutorial_rendering # 4
Command




## Giving Components

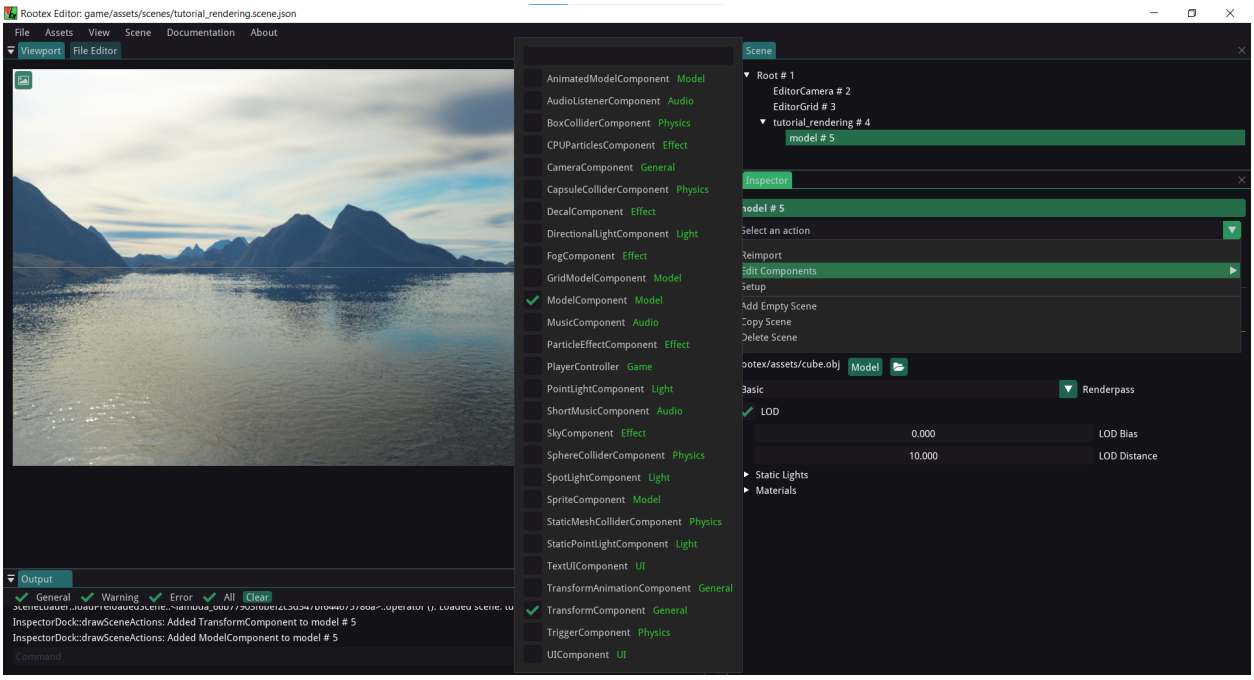
Now we give components to the empty scene.

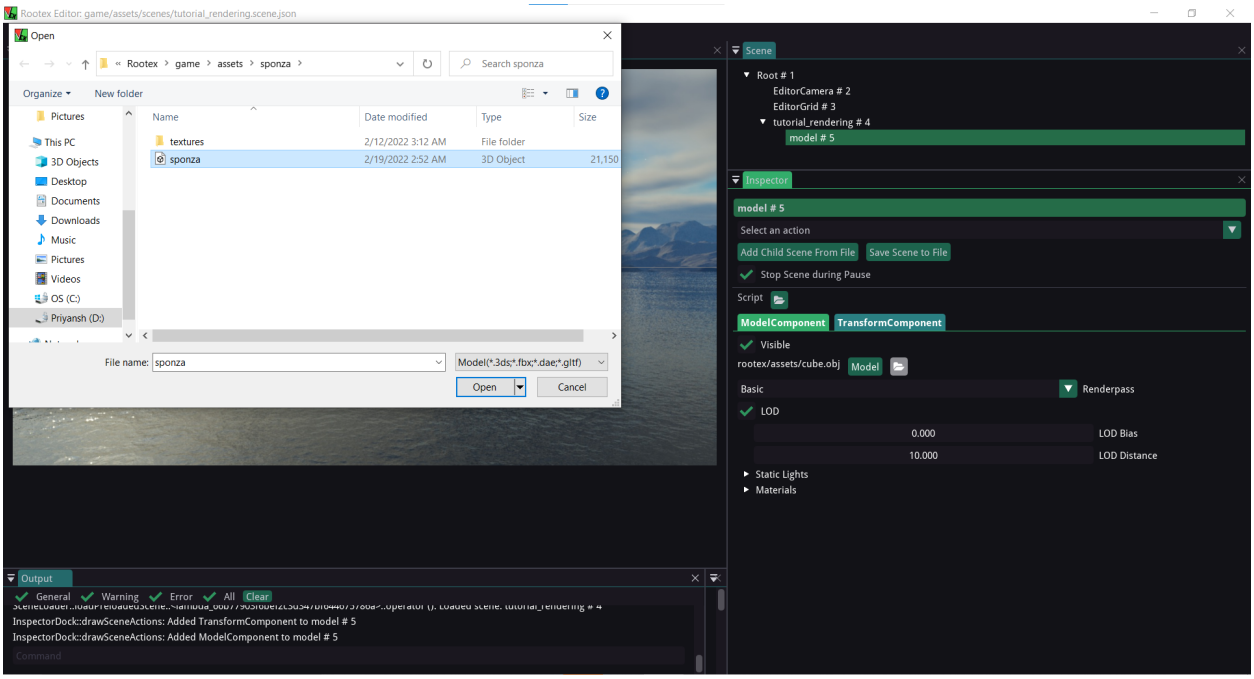
1) Right-click the empty scene.
2) Click Edit Components.
3) Check the appropriate components, in this case, transform and Model. **Note**: Transform Component is a must.

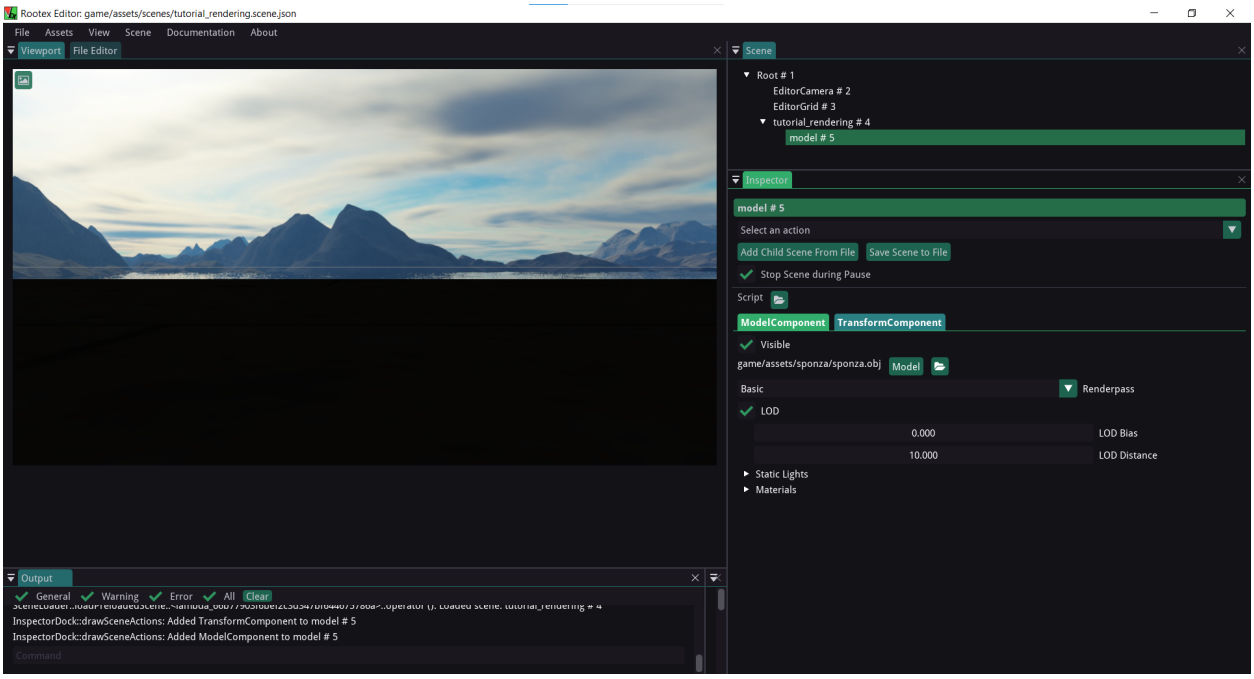
4) Open inspector.
5) Go to the model component in the inspector.
6) Click the folder icon next to Model.
7) Select the sponza 3D model file located at `Rootex\game\assets\sponza\sponza.obj`

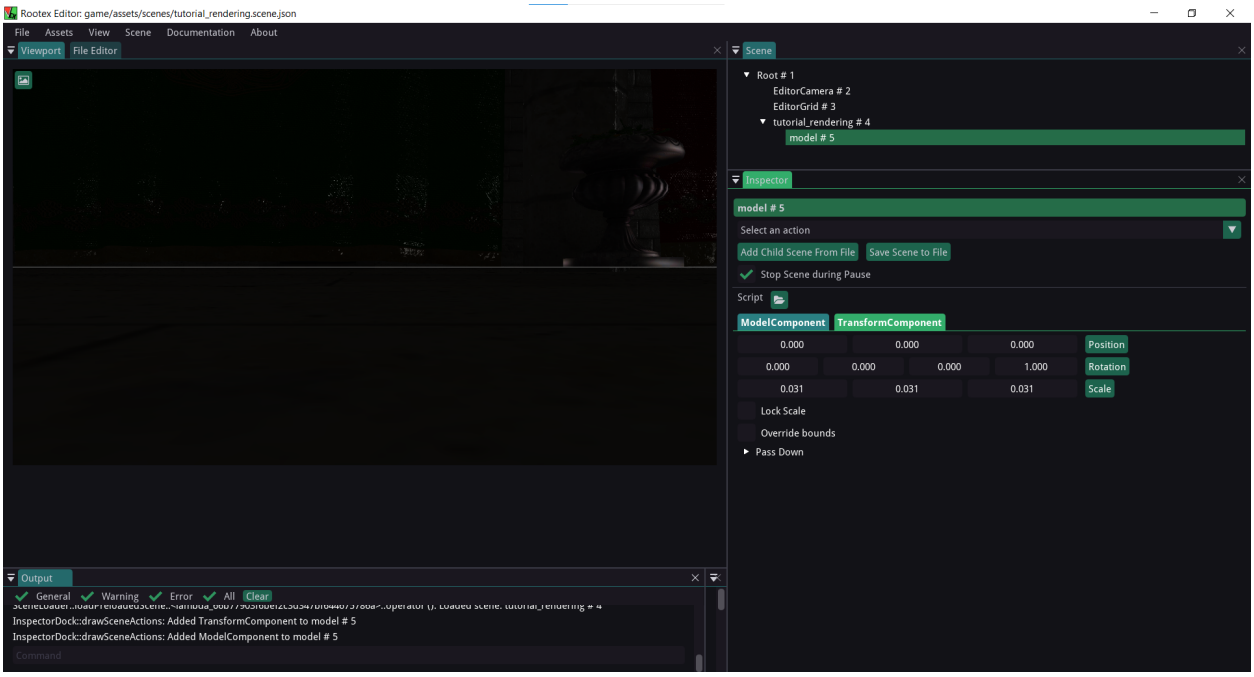
For sponza initially, it would look like this:

This is due to the default settings of the sponza obj file. To get a better view, set the scale to (0.031, 0.031, 0.031) and set the LOD distance to 123:


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
AnimatedModelComponent Model
AudioListenerComponent Audio
BoxColliderComponent Physics
CPUParticlesComponent Effect
CameraComponent General
CapsuleColliderComponent Physics
DecalComponent Effect
DirectionalLightComponent Light
FogComponent Effect
GridModelComponent Model
ModelComponent Model
MusicComponent Audio
ParticleEffectComponent Effect
PlayerController Game
PointLightComponent Light
ShortMusicComponent Audio
SkyComponent Effect
SphereColliderComponent Physics
SpotLightComponent Light
SpriteComponent Model
StaticMeshColliderComponent Physics
StaticPointLightComponent Light
TextUIComponent UI
TransformAnimationComponent General
TransformComponent General
TriggerComponent Physics
UIComponent UI
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
Inspector
Select an action
Reimport
Edit Components
Setup
Add Empty Scene
Copy Scene
Delete Scene
Model
Renderpass
LOD
0.000
LOD Bias
10.000
LOD Distance
Static Lights
Materials
Output
General Warning Error All Clear
InspectorDock::drawSceneActions: Added TransformComponent to model # 5
InspectorDock::drawSceneActions: Added ModelComponent to model # 5
Command


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
Open
Rootex › game › assets › sponza ›
Search sponza
Organize
New folder
Pictures
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
Priyansh (D:)
Name
Date modified
Type
Size
textures
2/12/2022 3:12 AM
File folder
sponza
2/19/2022 2:52 AM
3D Object
21,150
File name: sponza
Model(*.3ds;*.fbx;*.dae;*.gltf)
Open
Cancel
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
Inspector
model # 5
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
ModelComponent
TransformComponent
Visible
rootex/assets/cube.obj
Model
Basic
Renderpass
LOD
0.000
LOD Bias
10.000
LOD Distance
Static Lights
Materials
Output
General Warning Error All Clear
InspectorDock::drawSceneActions: Added TransformComponent to model # 5
InspectorDock::drawSceneActions: Added ModelComponent to model # 5
Command


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
Inspector
model # 5
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
ModelComponent
TransformComponent
Visible
game/assets/sponza/sponza.obj
Model
Basic
Renderpass
LOD
0.000
LOD Bias
10.000
LOD Distance
Static Lights
Materials
Output
General Warning Error All Clear
InspectorDock::drawSceneActions: Added TransformComponent to model # 5
InspectorDock::drawSceneActions: Added ModelComponent to model # 5
Command

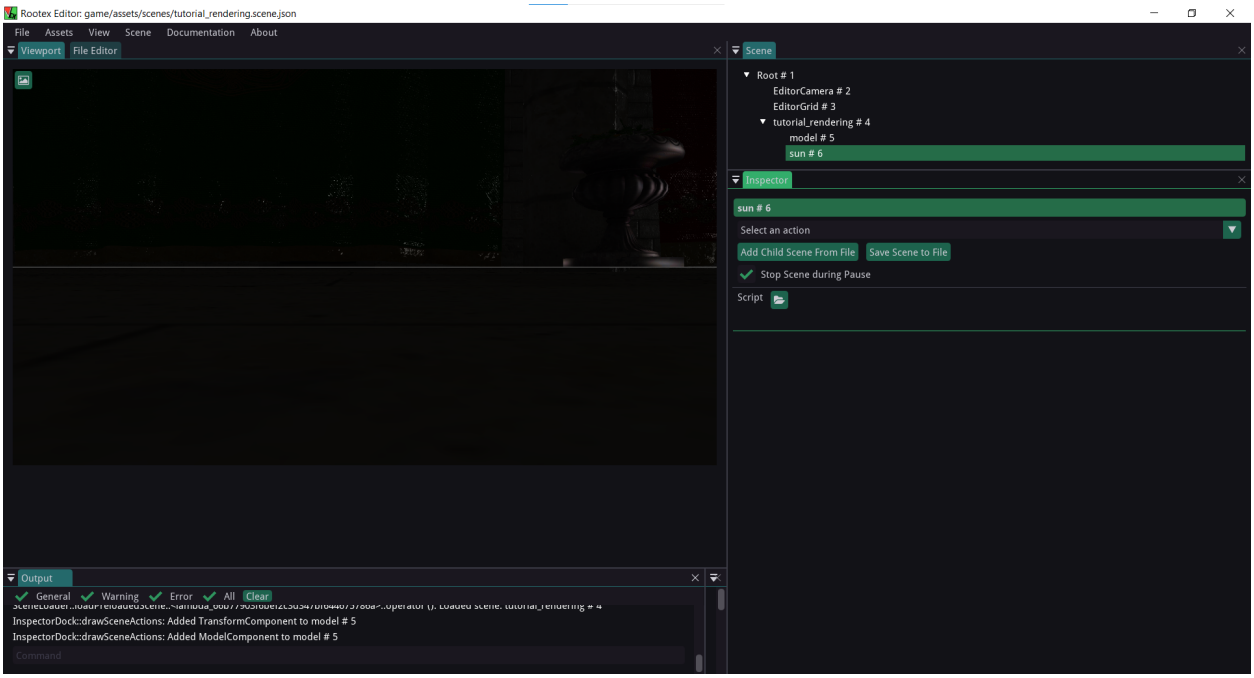

Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
Inspector
model # 5
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
ModelComponent
TransformComponent
0.000
0.000
0.000
Position
0.000
0.000
0.000
1.000
Rotation
0.031
0.031
0.031
Scale
Lock Scale
Override bounds
Pass Down
Output
General Warning Error All Clear
InspectorDock::drawSceneActions: Added TransformComponent to model # 5
InspectorDock::drawSceneActions: Added ModelComponent to model # 5
Command

We need to create an empty scene and add a light component to it to add light.

### Light Component

To add light, we now create an empty scene.

1) Name the scene.
2) Add transform and directional light components.

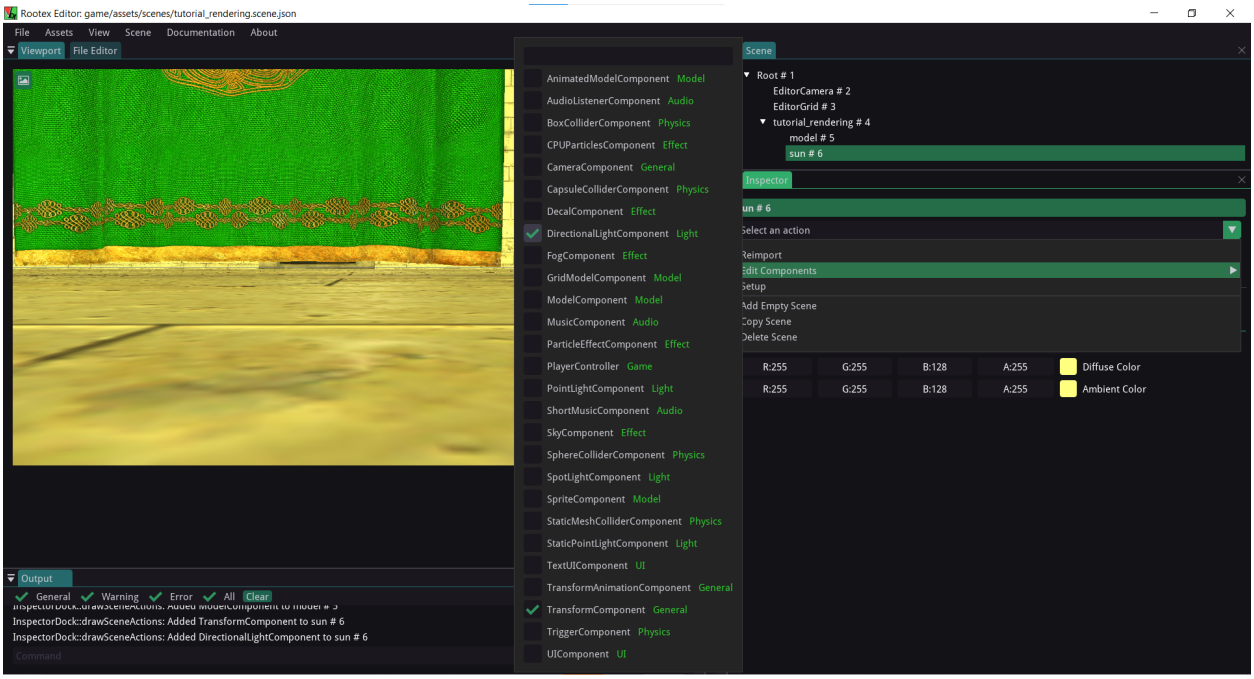


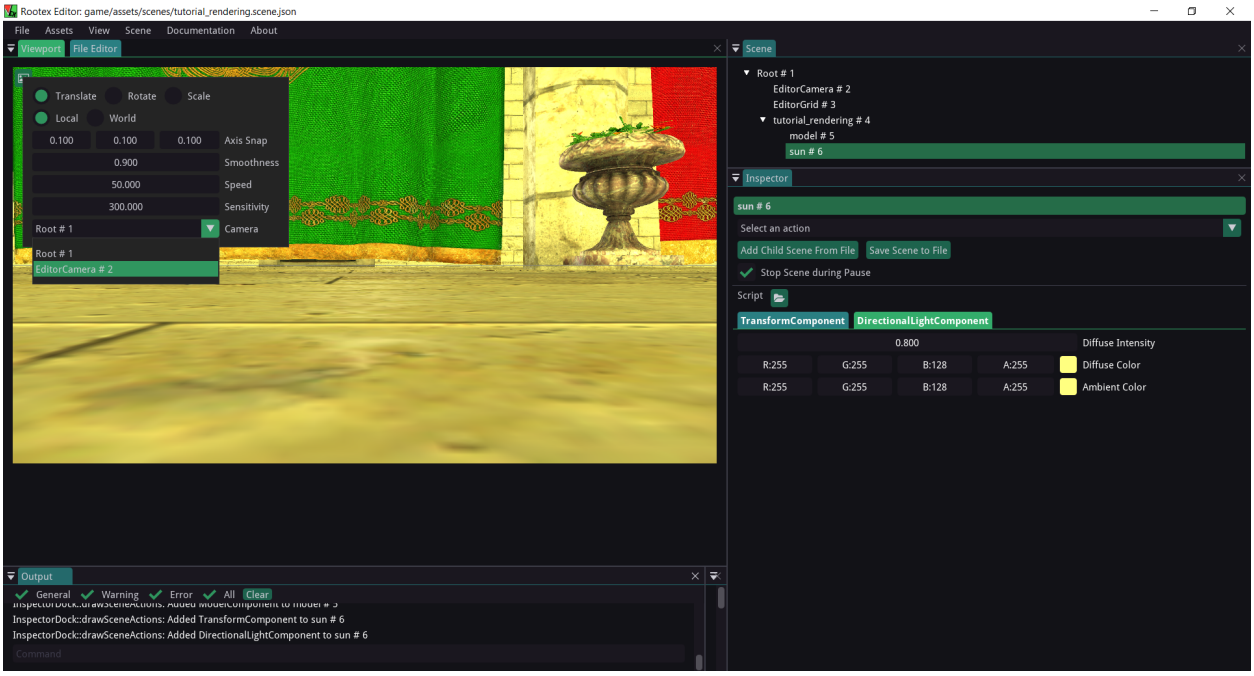

To move freely, we can change our camera mode to Editor Camera. This allows us to move freely.

### Editor Camera

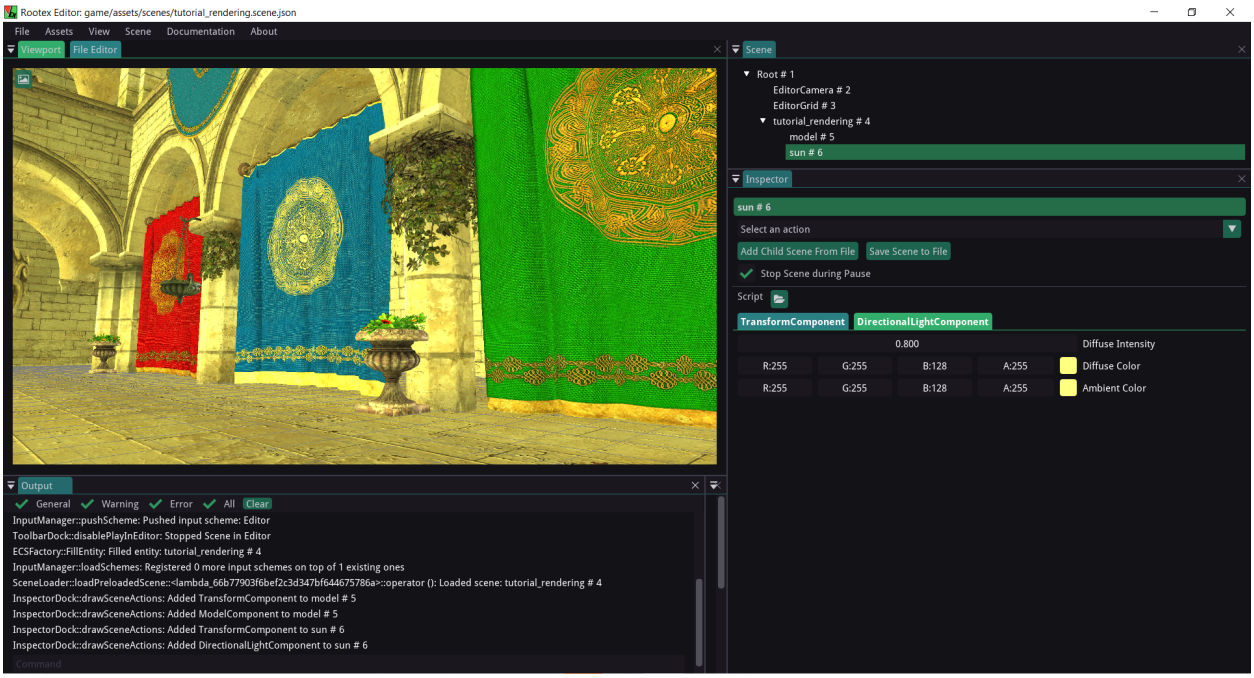
To have complete control of movement, you can use an editor camera.

1) Click the figure icon at the top left of the viewport.
2) Open dropdown of camera.
3) Select editor camera.


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
AnimatedModelComponent Model
AudioListenerComponent Audio
BoxColliderComponent Physics
CPUParticlesComponent Effect
CameraComponent General
CapsuleColliderComponent Physics
DecalComponent Effect
DirectionalLightComponent Light
FogComponent Effect
GridModelComponent Model
ModelComponent Model
MusicComponent Audio
ParticleEffectComponent Effect
PlayerController Game
PointLightComponent Light
ShortMusicComponent Audio
SkyComponent Effect
SphereColliderComponent Physics
SpotLightComponent Light
SpriteComponent Model
StaticMeshColliderComponent Physics
StaticPointLightComponent Light
TextUIComponent UI
TransformAnimationComponent General
TransformComponent General
TriggerComponent Physics
UIComponent UI
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
Inspector
Select an action
Reimport
Edit Components
Setup
Add Empty Scene
Copy Scene
Delete Scene
R:255 G:255 B:128 A:255 Diffuse Color
R:255 G:255 B:128 A:255 Ambient Color
Output
General Warning Error All Clear
InspectorDock::drawSceneActions: Added TransformComponent to sun # 6
InspectorDock::drawSceneActions: Added DirectionalLightComponent to sun # 6
Command


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Translate Rotate Scale
Local World
0.100 0.100 0.100 Axis Snap
0.900 Smoothness
50.000 Speed
300.000 Sensitivity
Root # 1 Camera
Root # 1
EditorCamera # 2
Scene
Root # 1
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
Inspector
sun # 6
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
TransformComponent
DirectionalLightComponent
0.800 Diffuse Intensity
R:255 G:255 B:128 A:255 Diffuse Color
R:255 G:255 B:128 A:255 Ambient Color
Output
General Warning Error All Clear
InspectorDock::drawSceneActions: Added TransformComponent to sun # 6
InspectorDock::drawSceneActions: Added DirectionalLightComponent to sun # 6
Command

To move, you have to hold the right mouse button and then use WASD space and shift keys to move. The cursor for direction. Space to move up and shift to move down.

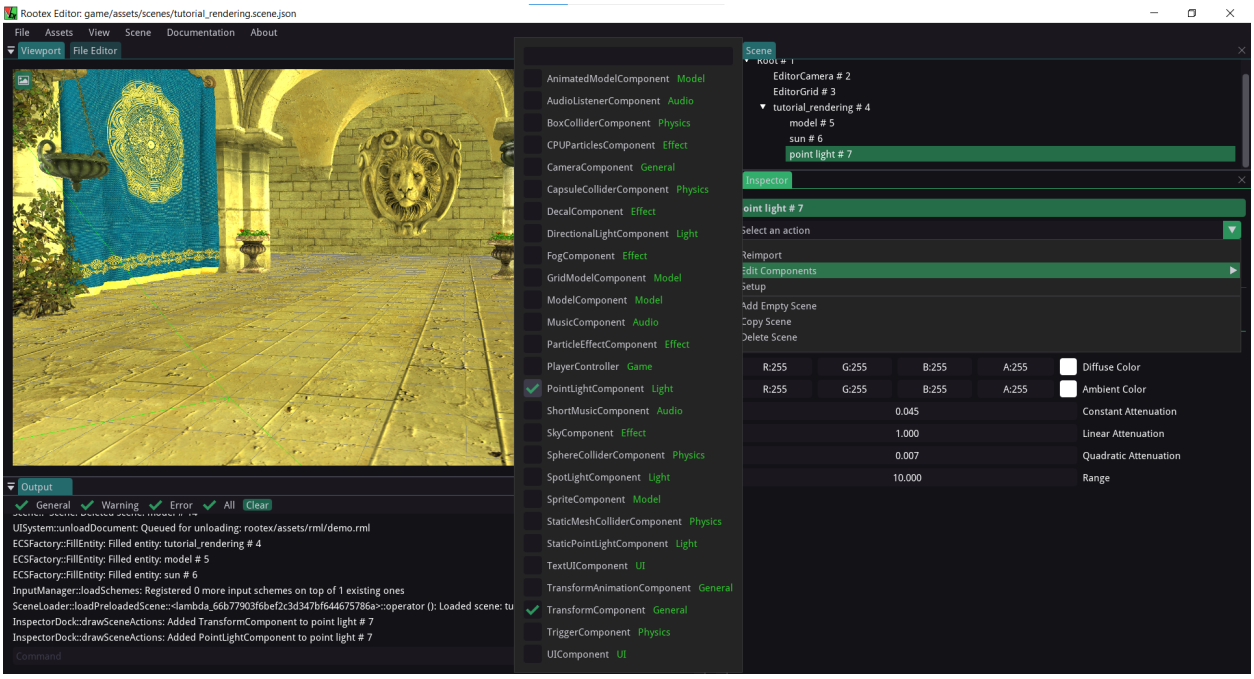


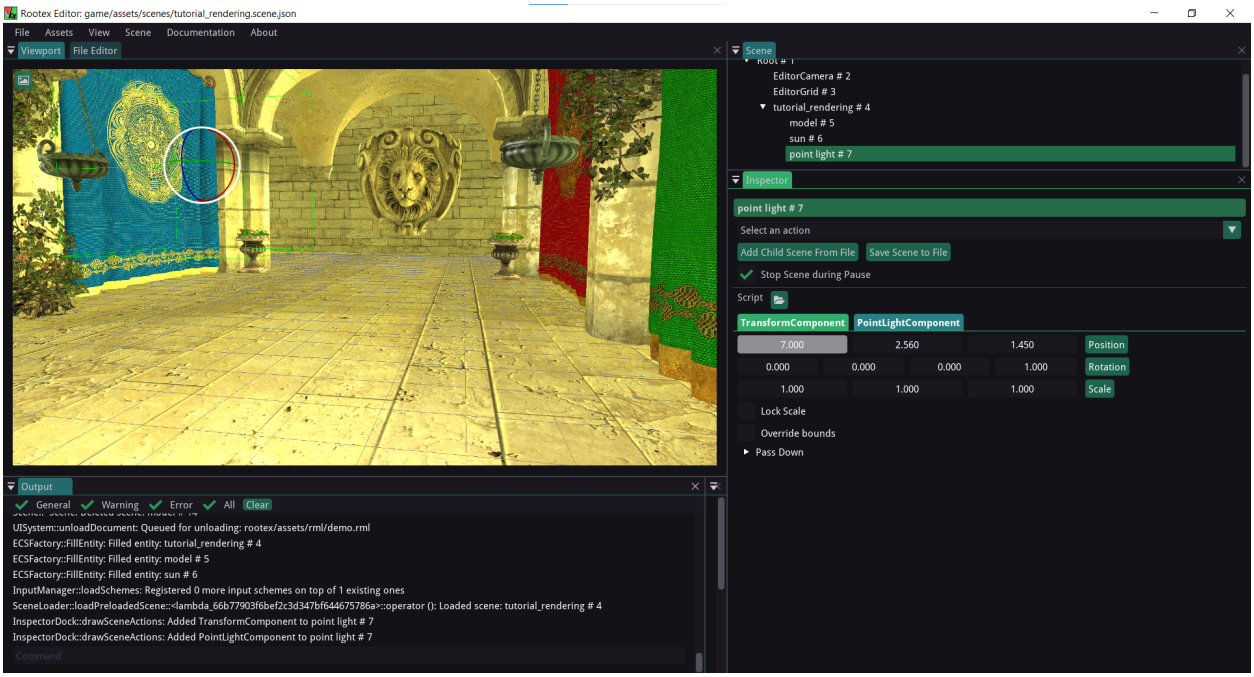

## Point Light

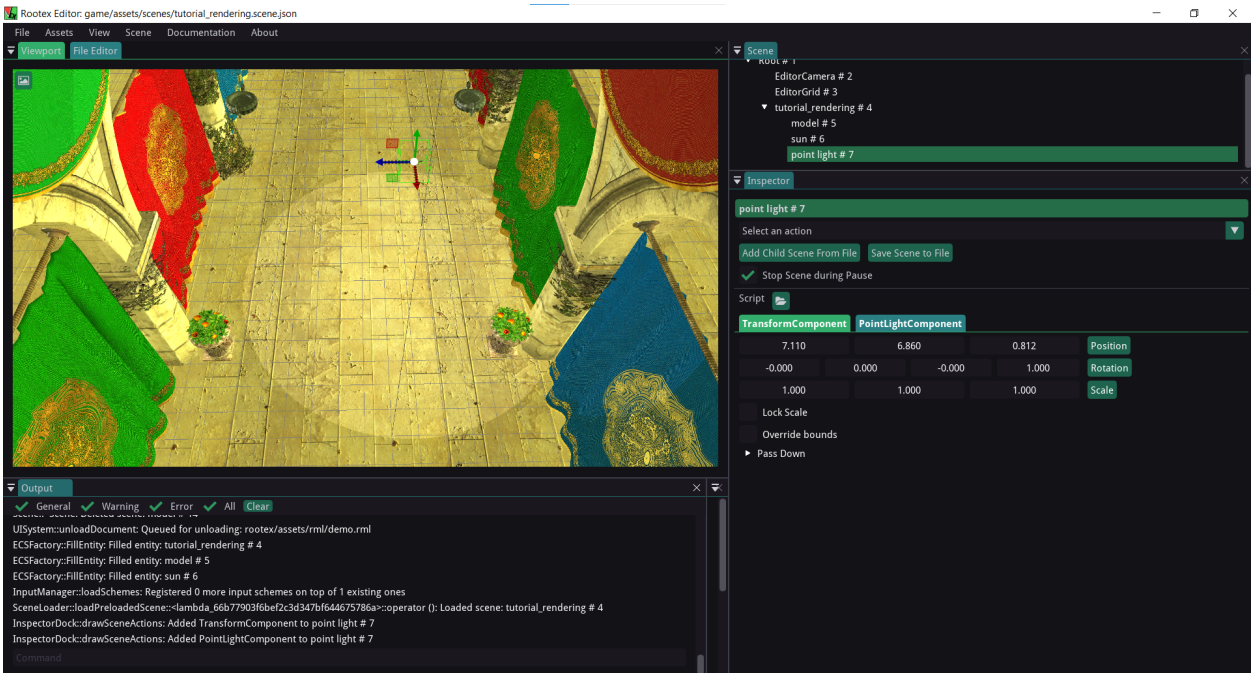
A point light is helpful if you have a source of light, e.g. a candle, bulb etc. To add a point light, follow the given steps.

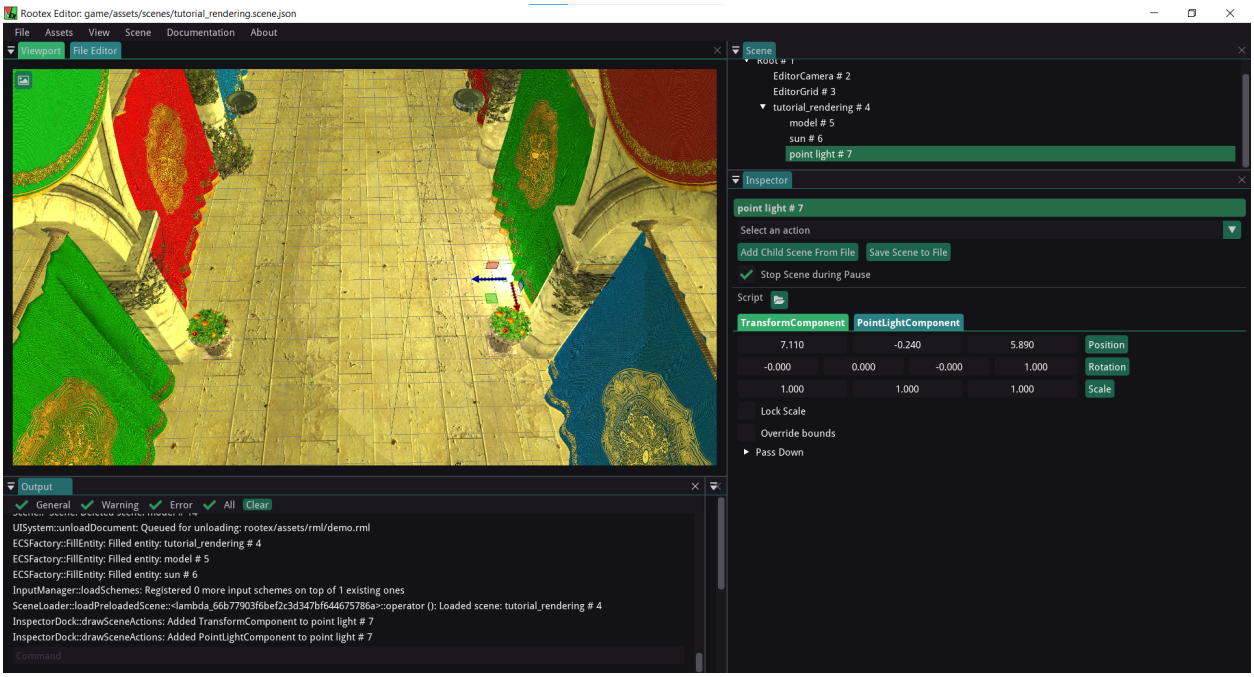
1) Add an empty scene and give it a point light component.
2) You can tweak its transformation value by either inputting it or dragging it left or right.

If you press ‘q’, a transform gizmo will appear on the object you have selected. You can adjust light location through it. For rotation and scaling gizmo, press ‘w’ and ‘e’, respectively.


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Scene
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
point light # 7
Inspector
Select an action
Reimport
Edit Components
Setup
Add Empty Scene
Copy Scene
Delete Scene
R:255 G:255 B:255 A:255 Diffuse Color
R:255 G:255 B:255 A:255 Ambient Color
0.045 Constant Attenuation
1.000 Linear Attenuation
0.007 Quadratic Attenuation
10.000 Range
AnimatedModelComponent Model
AudioListenerComponent Audio
BoxColliderComponent Physics
CPUParticlesComponent Effect
CameraComponent General
CapsuleColliderComponent Physics
DecalComponent Effect
DirectionalLightComponent Light
FogComponent Effect
GridModelComponent Model
ModelComponent Model
MusicComponent Audio
ParticleEffectComponent Effect
PlayerController Game
PointLightComponent Light
ShortMusicComponent Audio
SkyComponent Effect
SphereColliderComponent Physics
SpotLightComponent Light
SpriteComponent Model
StaticMeshColliderComponent Physics
StaticPointLightComponent Light
TextUIComponent UI
TransformAnimationComponent General
TransformComponent General
TriggerComponent Physics
UIComponent UI
Output
General Warning Error All Clear
UISystem::unloadDocument: Queued for unloading: rootex/assets/rml/demo.rml
ECSFactory::FillEntity: Filled entity: tutorial_rendering # 4
ECSFactory::FillEntity: Filled entity: model # 5
ECSFactory::FillEntity: Filled entity: sun # 6
InputManager::loadSchemes: Registered 0 more input schemes on top of 1 existing ones
InspectorDock::drawSceneActions: Added TransformComponent to point light # 7
InspectorDock::drawSceneActions: Added PointLightComponent to point light # 7
Command


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Scene
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
point light # 7
Inspector
point light # 7
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
TransformComponent
PointLightComponent
7.000 2.560 1.450 Position
0.000 0.000 0.000 1.000 Rotation
1.000 1.000 1.000 Scale
Lock Scale
Override bounds
Pass Down
Output
General Warning Error All Clear
UISystem::unloadDocument: Queued for unloading: rootex/assets/rml/demo.rml
ECSFactory::FillEntity: Filled entity: tutorial_rendering # 4
ECSFactory::FillEntity: Filled entity: model # 5
ECSFactory::FillEntity: Filled entity: sun # 6
InputManager::loadSchemes: Registered 0 more input schemes on top of 1 existing ones
SceneLoader::loadPreloadedScene::<lambda_66b77903f6bef2c3d347bf644675786a>::operator (): Loaded scene: tutorial_rendering # 4
InspectorDock::drawSceneActions: Added TransformComponent to point light # 7
InspectorDock::drawSceneActions: Added PointLightComponent to point light # 7
Command


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Scene
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
point light # 7
Inspector
point light # 7
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
TransformComponent
PointLightComponent
7.110
6.860
0.812
Position
-0.000
0.000
-0.000
1.000
Rotation
1.000
1.000
1.000
Scale
Lock Scale
Override bounds
Pass Down
Output
General Warning Error All Clear
UISystem::unloadDocument: Queued for unloading: rootex/assets/rml/demo.rml
ECSFactory::FillEntity: Filled entity: tutorial_rendering # 4
ECSFactory::FillEntity: Filled entity: model # 5
ECSFactory::FillEntity: Filled entity: sun # 6
InputManager::loadSchemes: Registered 0 more input schemes on top of 1 existing ones
SceneLoader::loadPreloadedScene::<lambda_66b77903f6bef2c3d347bf644675786a>::operator (): Loaded scene: tutorial_rendering # 4
InspectorDock::drawSceneActions: Added TransformComponent to point light # 7
InspectorDock::drawSceneActions: Added PointLightComponent to point light # 7
Command

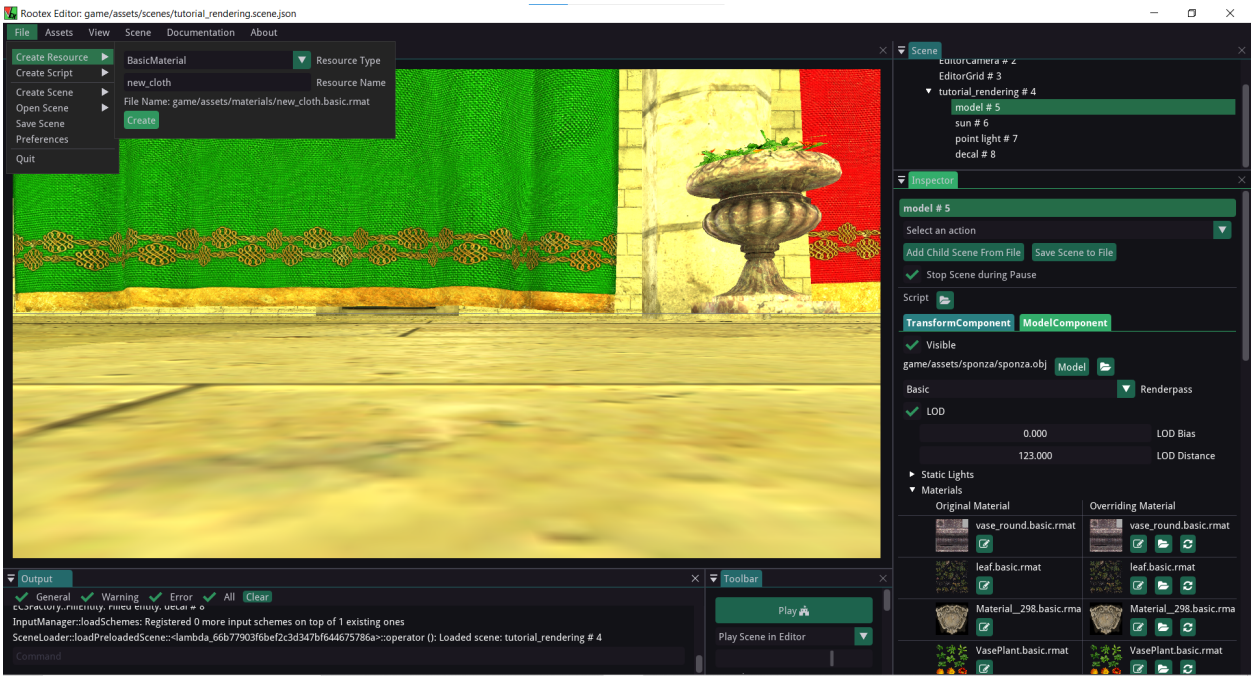

Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Scene
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
point light # 7
Inspector
point light # 7
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
TransformComponent
PointLightComponent
7.110
-0.240
5.890
Position
-0.000
0.000
-0.000
1.000
Rotation
1.000
1.000
1.000
Scale
Lock Scale
Override bounds
Pass Down
Output
General Warning Error All Clear
UISystem::unloadDocument: Queued for unloading: rootex/assets/rml/demo.rml
ECSFactory::FillEntity: Filled entity: tutorial_rendering # 4
ECSFactory::FillEntity: Filled entity: model # 5
ECSFactory::FillEntity: Filled entity: sun # 6
InputManager::loadSchemes: Registered 0 more input schemes on top of 1 existing ones
SceneLoader::loadPreloadedScene::<lambda_66b77903f6bef2c3d347bf644675786a>::operator (): Loaded scene: tutorial_rendering # 4
InspectorDock::drawSceneActions: Added TransformComponent to point light # 7
InspectorDock::drawSceneActions: Added PointLightComponent to point light # 7
Command

### Overriding a material

To change the properties of one object without changing the original material, we can use overriding materials. To override a material:

1) Create a new basic material by going to file -> Create Resource.

2) Name the material and click create.

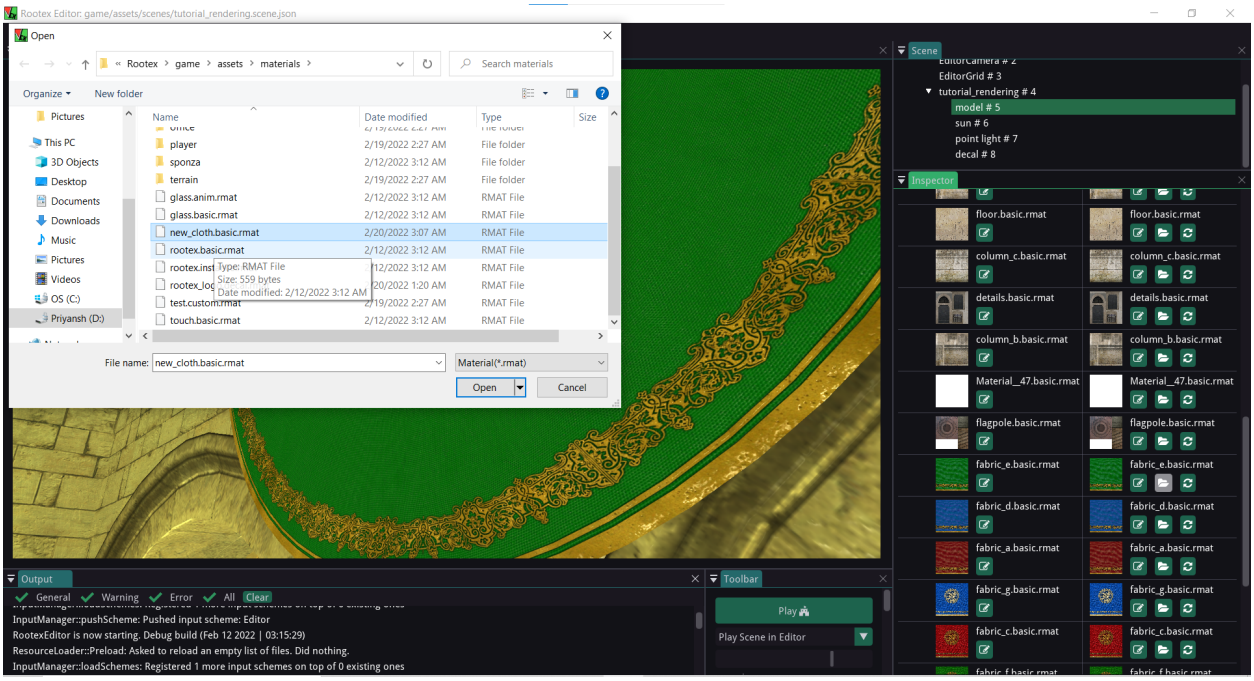


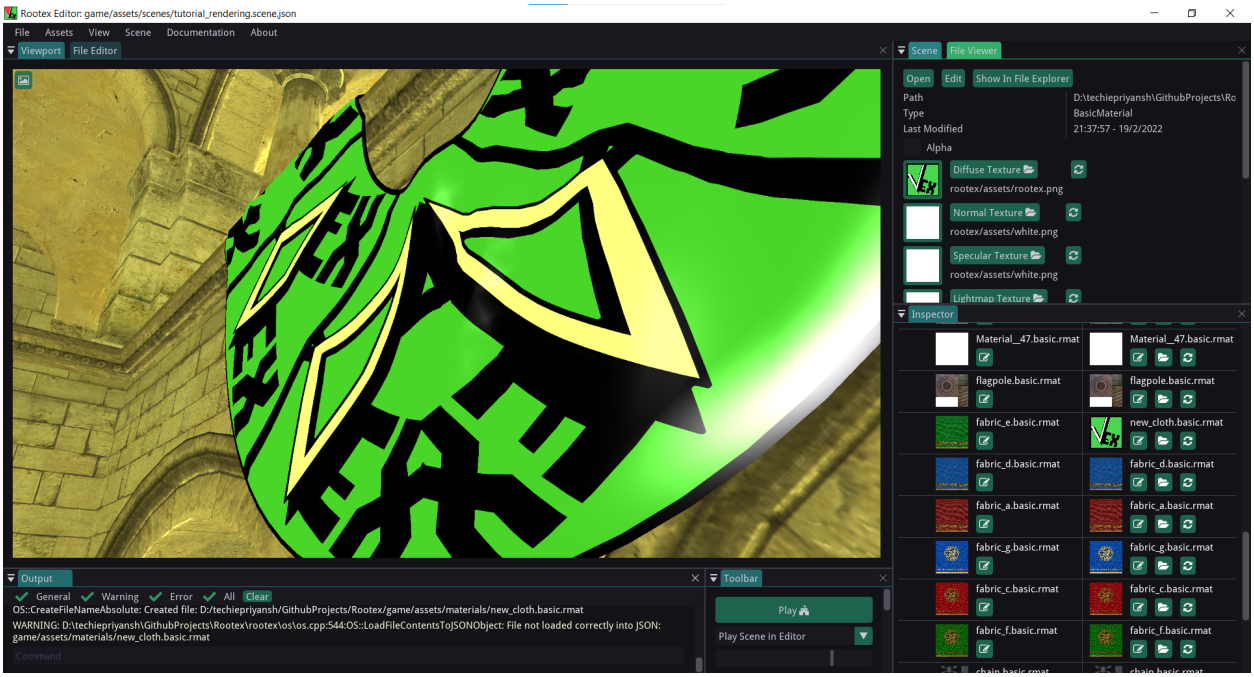

3) Go to the `Inspector-> Model Component->Materials`.

4) Click on the folder icon on the corresponding overriding material.

5) Select the newly created basic material located at `Rootex\game\assets\materials\new_cloth.basic.rmat`

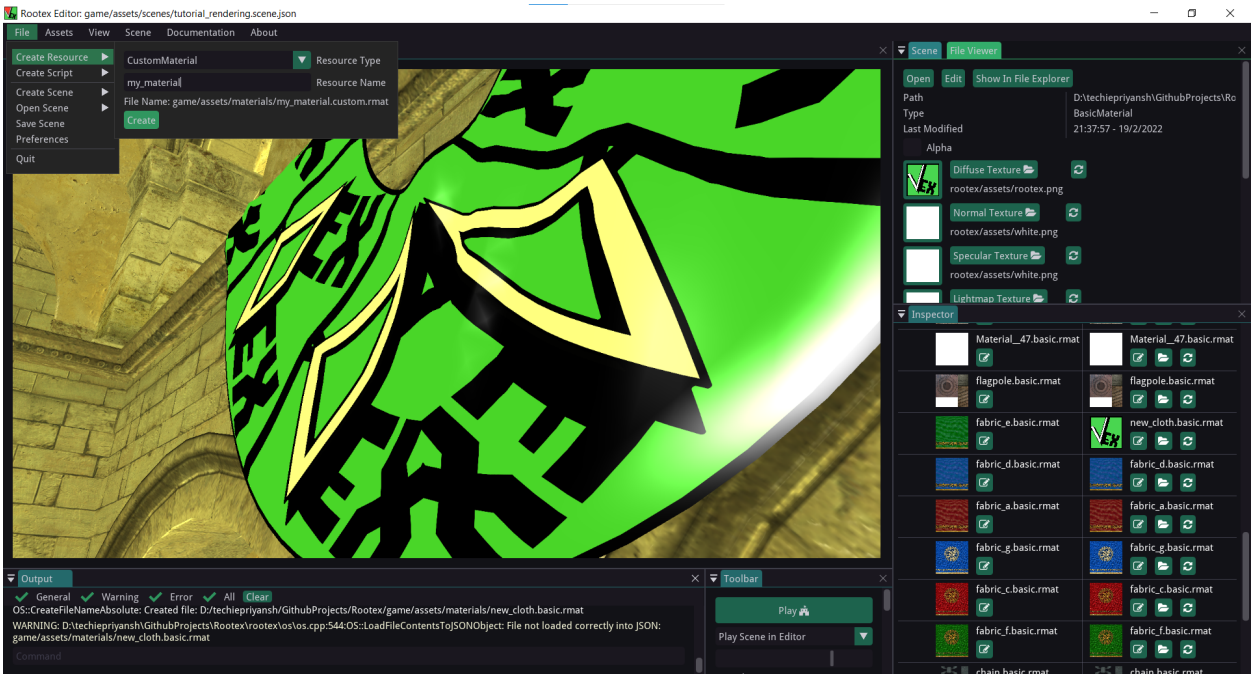
Now you can change its basic textures by 1)clicking on the pencil icon 2)In the file viewer now click on the diffuse texture and select the appropriate diffuse texture.


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
Open
Rootex › game › assets › materials ›
Search materials
Organize
New folder
Pictures
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
Priyansh (D:)
Name
Date modified
Type
Size
player
sponza
terrain
glass.anim.rmat
glass.basic.rmat
new_cloth.basic.rmat
rootex.basic.rmat
test.custom.rmat
touch.basic.rmat
2/12/2022 3:12 AM
2/20/2022 3:07 AM
RMAT File
File folder
Type: RMAT File
Size: 559 bytes
Date modified: 2/12/2022 3:12 AM
File name: new_cloth.basic.rmat
Material(*.rmat)
Open
Cancel
Scene
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
point light # 7
decal # 8
Inspector
floor.basic.rmat
column_c.basic.rmat
details.basic.rmat
column_b.basic.rmat
Material__47.basic.rmat
flagpole.basic.rmat
fabric_e.basic.rmat
fabric_d.basic.rmat
fabric_a.basic.rmat
fabric_g.basic.rmat
fabric_c.basic.rmat
Output
General
Warning
Error
All
Clear
InputManager::pushScheme: Pushed input scheme: Editor
RootexEditor is now starting. Debug build (Feb 12 2022 | 03:15:29)
ResourceLoader::Preload: Asked to reload an empty list of files. Did nothing.
InputManager::loadSchemes: Registered 1 more input schemes on top of 0 existing ones
Toolbar
Play
Play Scene in Editor


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File
Assets
View
Scene
Documentation
About
Viewport
File Editor
Scene
File Viewer
Open
Edit
Show In File Explorer
Path
Type
Last Modified
BasicMaterial
21:37:57 - 19/2/2022
Alpha
Diffuse Texture
rootex/assets/rootex.png
Normal Texture
rootex/assets/white.png
Specular Texture
rootex/assets/white.png
Lightmap Texture
Inspector
Material_47.basic.rmat
flagpole.basic.rmat
fabric_e.basic.rmat
new_cloth.basic.rmat
fabric_d.basic.rmat
fabric_a.basic.rmat
fabric_g.basic.rmat
fabric_c.basic.rmat
fabric_f.basic.rmat
Output
General
Warning
Error
All
Clear
OS::CreateFileNameAbsolute: Created file: D:/techiepriyansh/GithubProjects/Rootex/game/assets/materials/new_cloth.basic.rmat
WARNING: D:\techiepriyansh\GithubProjects\Rootex\rootex\os\os.cpp:544:OS::LoadFileContentsToJSONObject: File not loaded correctly into JSON: game/assets/materials/new_cloth.basic.rmat
Command
Toolbar
Play
Play Scene in Editor

## Custom Material

1) Go to create Resource -> Custom Material.




2) Enter material name.
3) Now go to Inspector -> ModelComponent and then to Materials.
4) Click on the folder icon and choose the material.

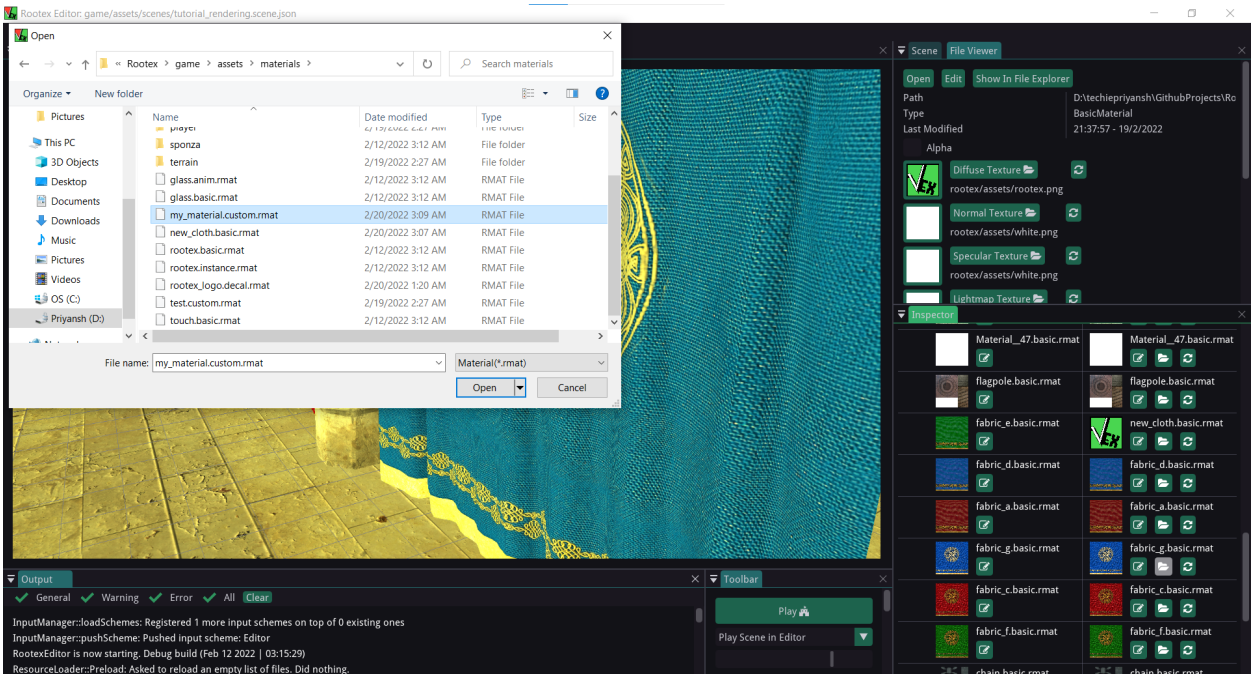
## Adding a shader

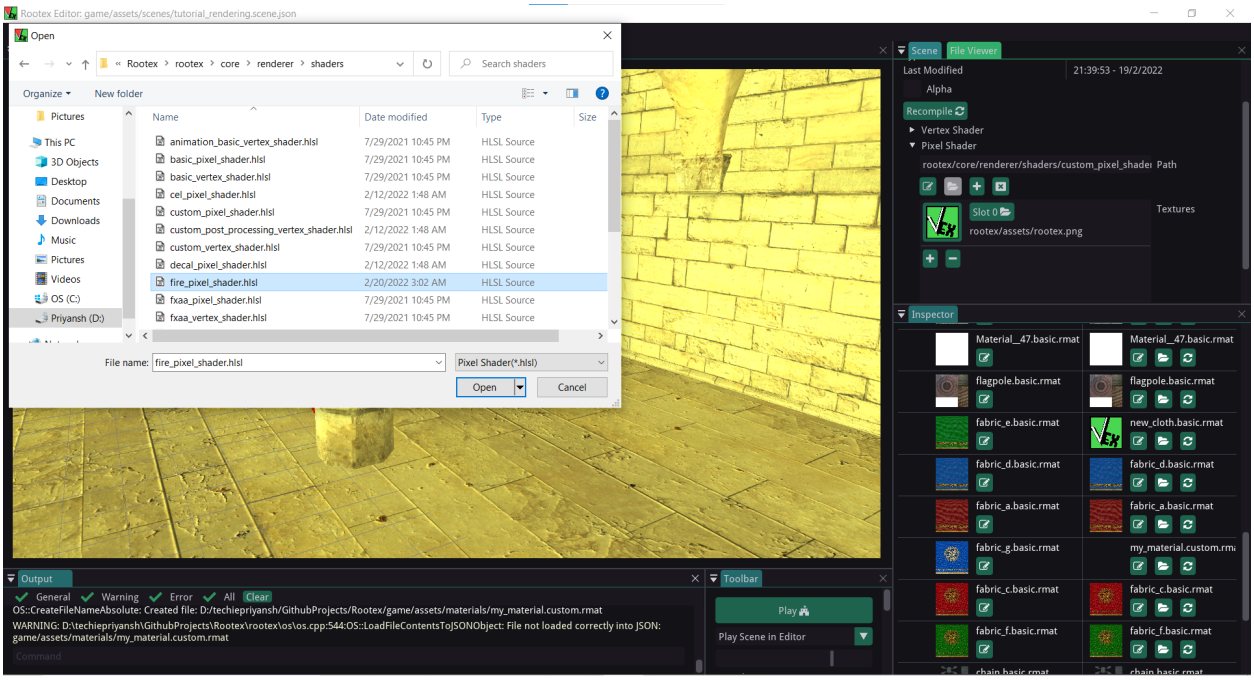
To Add shader:

1) Click on the pencil icon on the overriding custom material.
2) Now, in the file viewer you'll get options to add vertex and pixel shaders.
3) Click on the pixel shader. A dialog box will open now you can just select the shader.

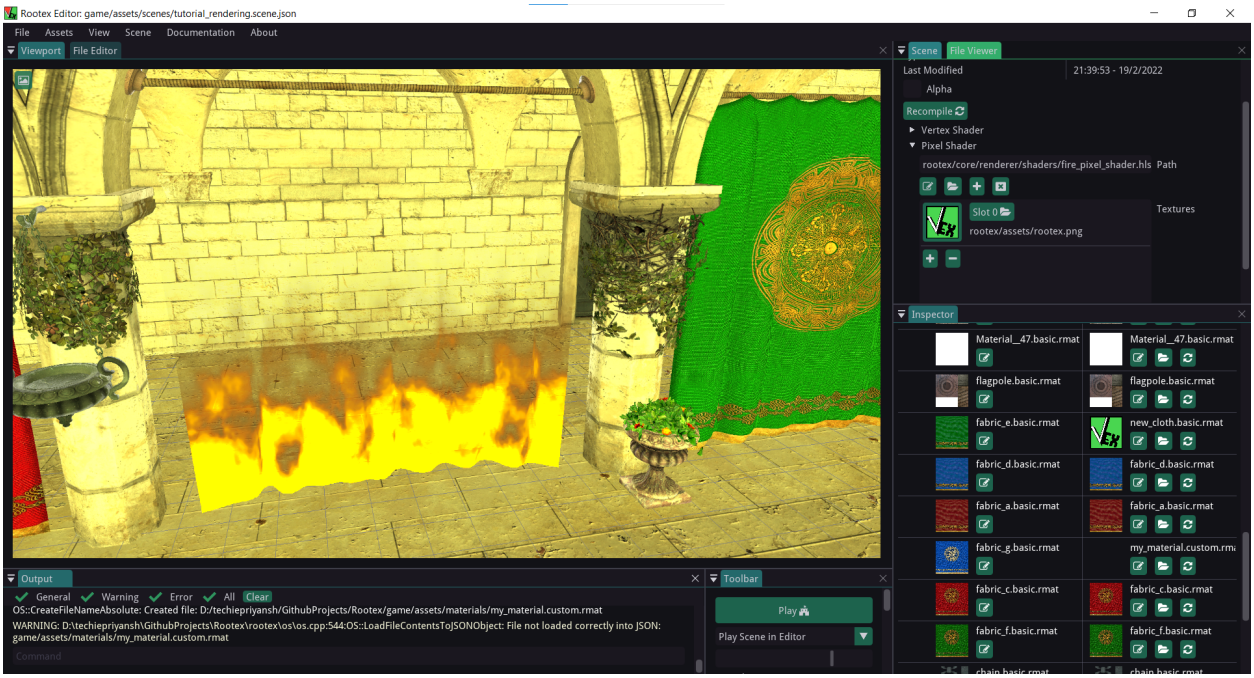
You can use fire_pixel_shader from rootex/core/renderer/shaders

Clicking on the pencil icon opens an editor to customise the shader.


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
Open
Rootex › game › assets › materials ›
Search materials
Organize
New folder
Pictures
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
Priyansh (D:)
Name
Date modified
Type
Size
sponza
terrain
glass.anim.rmat
glass.basic.rmat
my_material.custom.rmat
new_cloth.basic.rmat
rootex.basic.rmat
rootex.instance.rmat
rootex_logo.decal.rmat
test.custom.rmat
touch.basic.rmat
2/12/2022 3:12 AM
2/19/2022 2:27 AM
2/20/2022 3:09 AM
2/20/2022 3:07 AM
2/20/2022 1:20 AM
File name: my_material.custom.rmat
Material(*.rmat)
Open
Cancel
Scene
File Viewer
Open
Edit
Show In File Explorer
Path
Type
BasicMaterial
Last Modified
21:37:57 - 19/2/2022
Alpha
Diffuse Texture
rootex/assets/rootex.png
Normal Texture
rootex/assets/white.png
Specular Texture
rootex/assets/white.png
Lightmap Texture
Inspector
Material__47.basic.rmat
flagpole.basic.rmat
fabric_e.basic.rmat
new_cloth.basic.rmat
fabric_d.basic.rmat
fabric_a.basic.rmat
fabric_g.basic.rmat
fabric_c.basic.rmat
fabric_f.basic.rmat
Output
General
Warning
Error
All
Clear
InputManager::loadSchemes: Registered 1 more input schemes on top of 0 existing ones
InputManager::pushScheme: Pushed input scheme: Editor
RootexEditor is now starting. Debug build (Feb 12 2022 | 03:15:29)
ResourceLoader::Preload: Asked to reload an empty list of files. Did nothing.
Toolbar
Play
Play Scene in Editor


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
Open
Rootex › rootex › core › renderer › shaders
Search shaders
Organize
New folder
Pictures
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
Priyansh (D:)
Name
Date modified
Type
Size
animation_basic_vertex_shader.hlsl
basic_pixel_shader.hlsl
basic_vertex_shader.hlsl
cel_pixel_shader.hlsl
custom_pixel_shader.hlsl
custom_post_processing_vertex_shader.hlsl
custom_vertex_shader.hlsl
decal_pixel_shader.hlsl
fire_pixel_shader.hlsl
fxaa_pixel_shader.hlsl
fxaa_vertex_shader.hlsl
7/29/2021 10:45 PM
2/12/2022 1:48 AM
2/20/2022 3:02 AM
HLSL Source
File name: fire_pixel_shader.hlsl
Pixel Shader(*.hlsl)
Open
Cancel
Scene
File Viewer
Last Modified
21:39:53 - 19/2/2022
Alpha
Recompile
Vertex Shader
Pixel Shader
rootex/core/renderer/shaders/custom_pixel_shade
Path
Slot 0
rootex/assets/rootex.png
Textures
Inspector
Material__47.basic.rmat
flagpole.basic.rmat
fabric_e.basic.rmat
new_cloth.basic.rmat
fabric_d.basic.rmat
fabric_a.basic.rmat
fabric_g.basic.rmat
my_material.custom.rma
fabric_c.basic.rmat
fabric_f.basic.rmat
Output
General
Warning
Error
All
Clear
OS::CreateFileNameAbsolute: Created file: D:/techiepriyansh/GithubProjects/Rootex/game/assets/materials/my_material.custom.rmat
WARNING: D:\techiepriyansh\GithubProjects\Rootex\rootex\os\os.cpp:544:OS::LoadFileContentsToJSONObject: File not loaded correctly into JSON: game/assets/materials/my_material.custom.rmat
Command
Toolbar
Play
Play Scene in Editor

**Note:** You can only add shaders to custom materials. If you want to use default material, override the original default material with custom material and then add a shader to the overriding material. The overriding material does inherit the textures of the original materials.

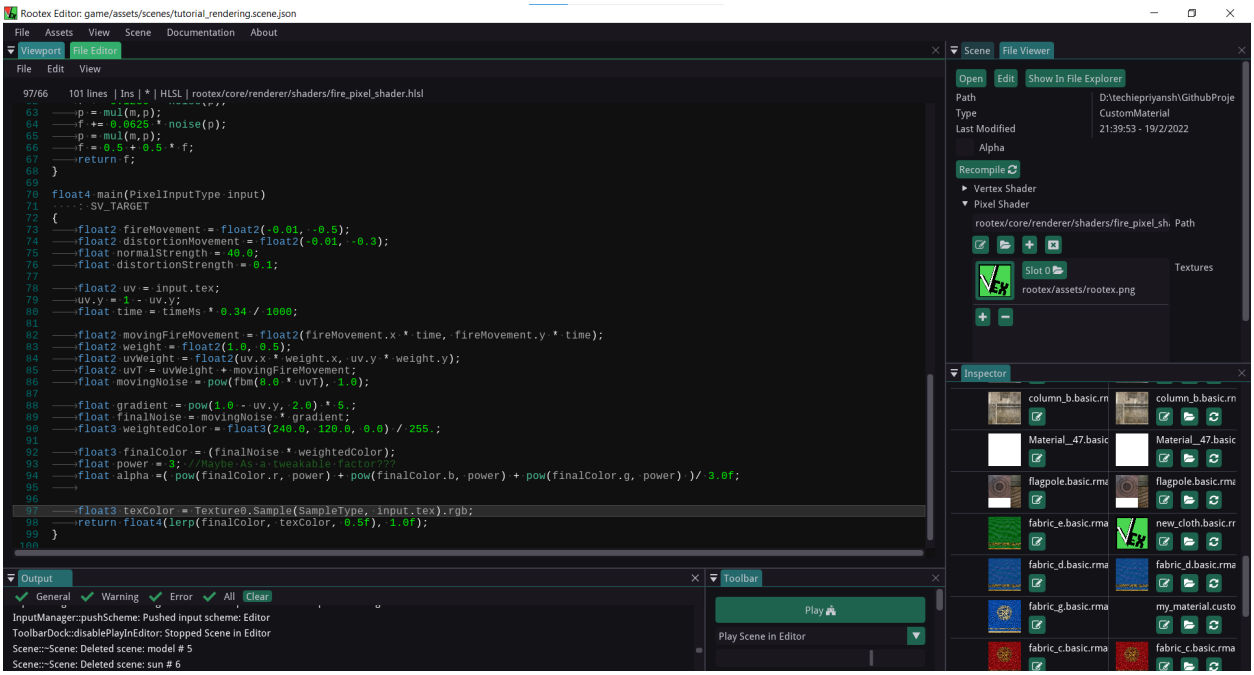


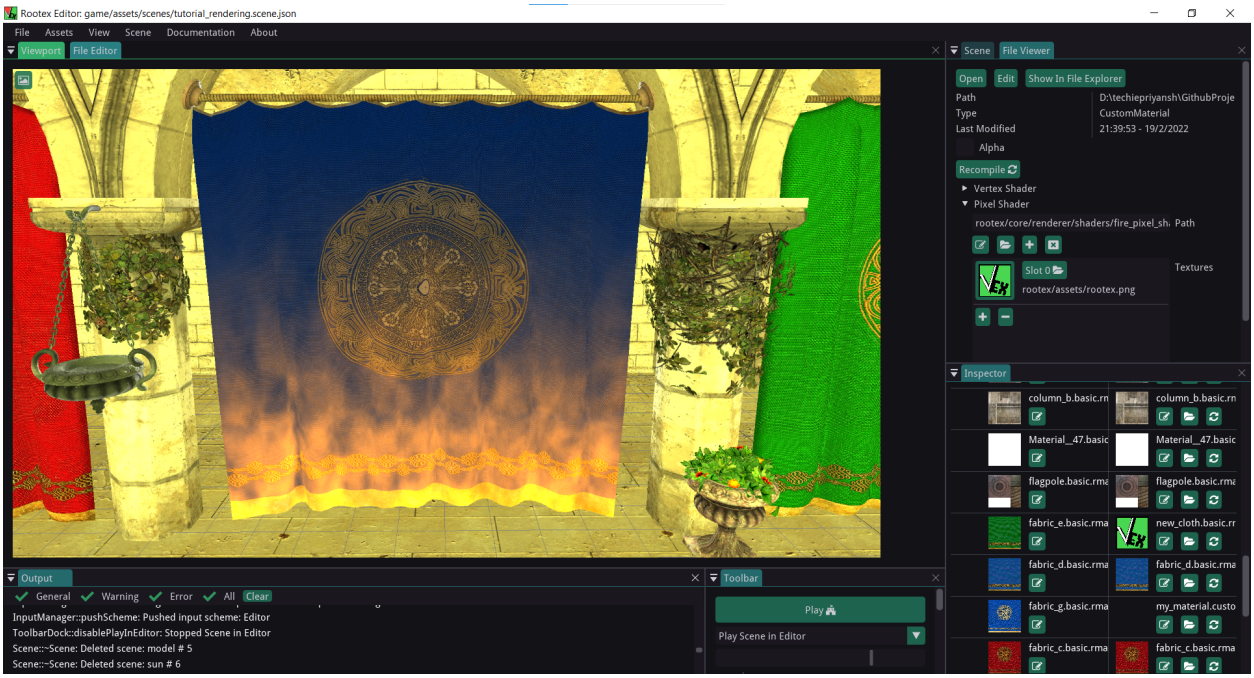

## Decal Component

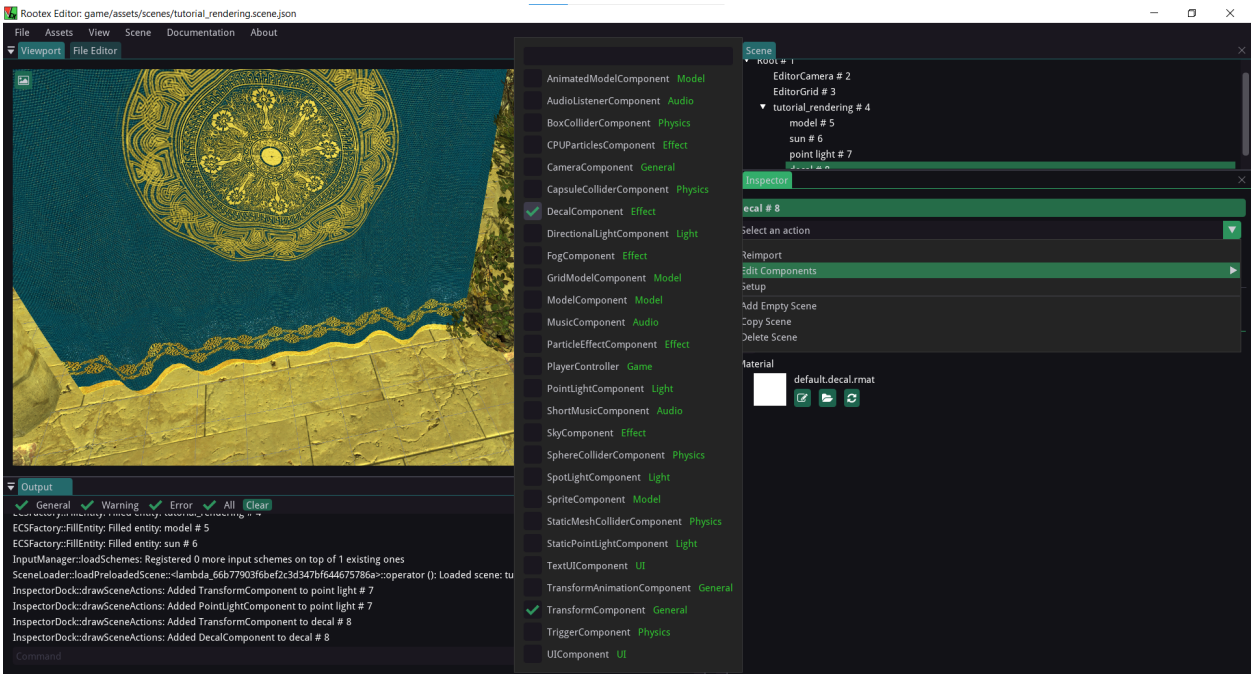
To add a decal component.

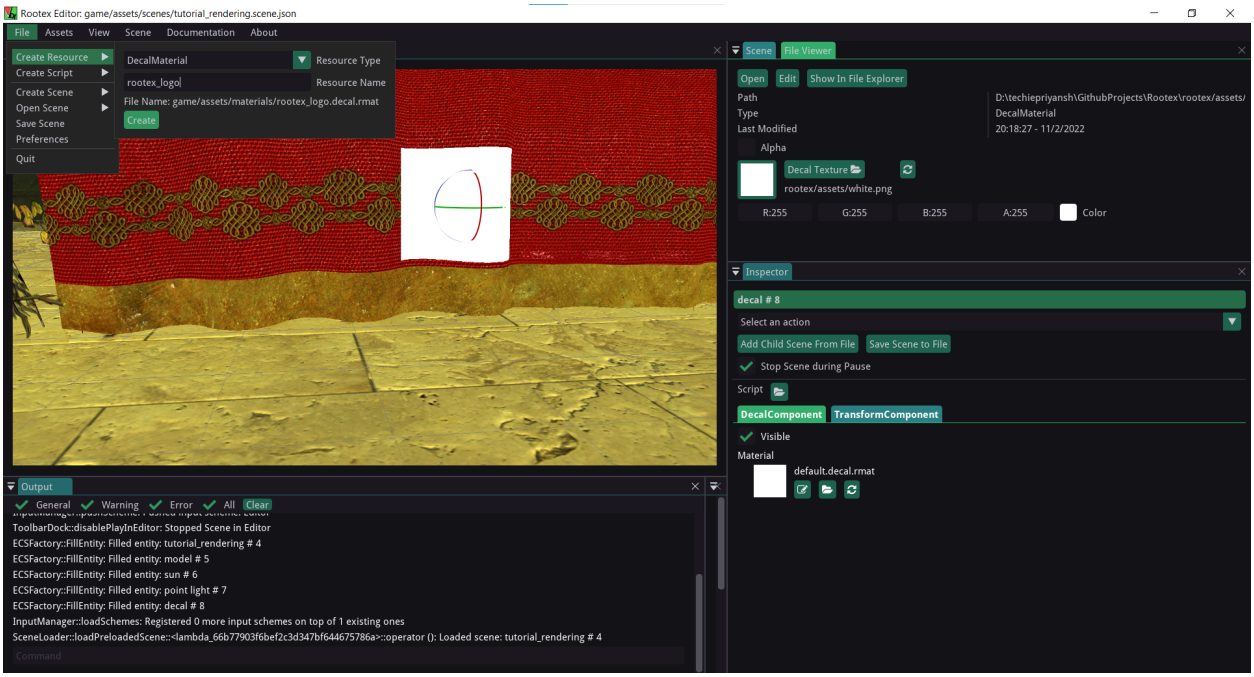
1) Make a scene DECAL and give it transform and Decal Component.

2) Create a decal material. By going to File -> CreateResource. And then slect Decal material in resource type dropdown.


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
File Edit View
97/66 101 lines | Ins | * | HLSL | rootex/core/renderer/shaders/fire_pixel_shader.hlsl
p = mul(m,p);
f += 0.0625 * noise(p);
p = mul(m,p);
f = 0.5 + 0.5 * f;
return f;
}
float4 main(PixelInputType input)
: SV_TARGET
{
float2 fireMovement = float2(-0.01, -0.5);
float2 distortionMovement = float2(-0.01, -0.3);
float normalStrength = 40.0;
float distortionStrength = 0.1;
float2 uv = input.tex;
uv.y = 1 - uv.y;
float time = timeMs * 0.34 / 1000;
float2 movingFireMovement = float2(fireMovement.x * time, fireMovement.y * time);
float2 weight = float2(1.0, 0.5);
float2 uvWeight = float2(uv.x * weight.x, uv.y * weight.y);
float2 uvT = uvWeight + movingFireMovement;
float movingNoise = pow(fbm(8.0 * uvT), 1.0);
float gradient = pow(1.0 - uv.y, 2.0) * 5.;
float finalNoise = movingNoise * gradient;
float3 weightedColor = float3(240.0, 120.0, 0.0) / 255.;
float3 finalColor = (finalNoise * weightedColor);
float power = 3; //Maybe As a tweakable factor???
float alpha =( pow(finalColor.r, power) + pow(finalColor.b, power) + pow(finalColor.g, power) )/ 3.0f;
float3 texColor = Texture0.Sample(SampleType, input.tex).rgb;
return float4(lerp(finalColor, texColor, 0.5f), 1.0f);
}
Output
General Warning Error All Clear
InputManager::pushScheme: Pushed input scheme: Editor
ToolbarDock::disablePlayInEditor: Stopped Scene in Editor
Scene::~Scene: Deleted scene: model # 5
Scene::~Scene: Deleted scene: sun # 6
Toolbar
Play
Play Scene in Editor
Scene File Viewer
Open Edit Show In File Explorer
Path D:\techiepriyansh\GithubProje
Type CustomMaterial
Last Modified 21:39:53 - 19/2/2022
Alpha
Recompile
Vertex Shader
Pixel Shader
rootex/core/renderer/shaders/fire_pixel_sh Path
Slot 0
Textures
rootex/assets/rootex.png
Inspector
column_b.basic.rn
Material_47.basic
flagpole.basic.rma
fabric_e.basic.rma
new_cloth.basic.rr
fabric_d.basic.rma
fabric_g.basic.rma
my_material.custo
fabric_c.basic.rma

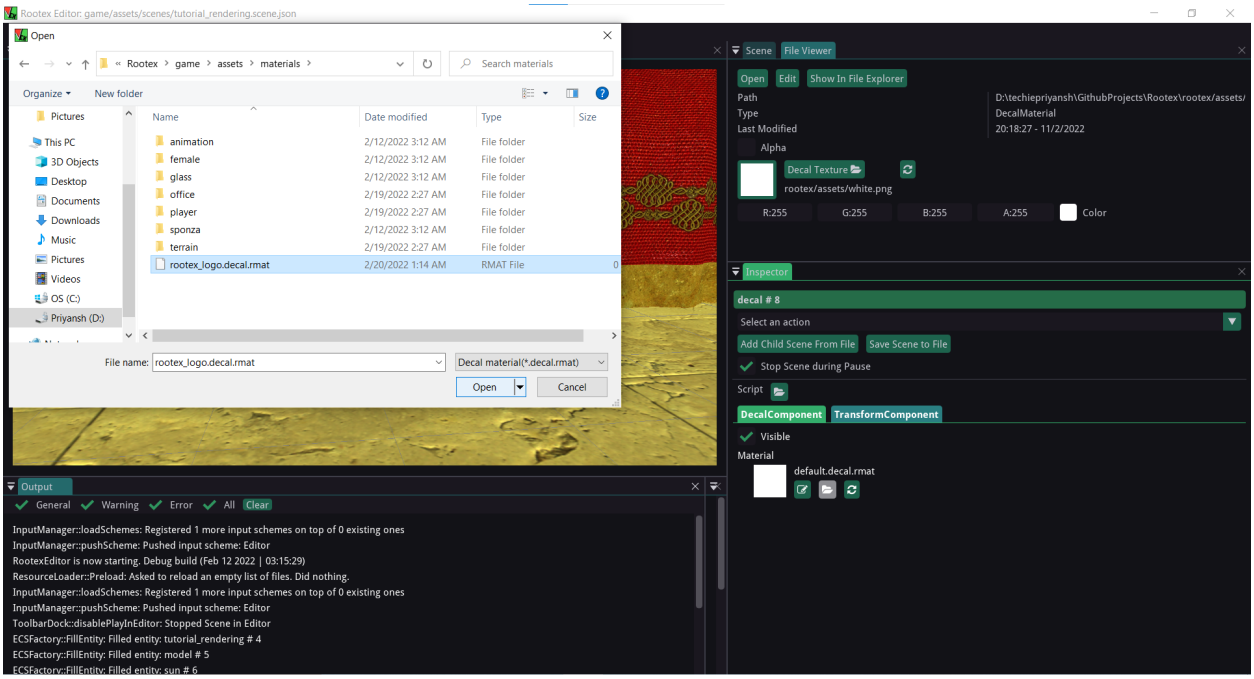

Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
Output
General Warning Error All Clear
InputManager::pushScheme: Pushed input scheme: Editor
ToolbarDock::disablePlayInEditor: Stopped Scene in Editor
Scene::~Scene: Deleted scene: model # 5
Scene::~Scene: Deleted scene: sun # 6
Toolbar
Play
Play Scene in Editor
Scene File Viewer
Open Edit Show In File Explorer
Path D:\techiepriyansh\GithubProje
Type CustomMaterial
Last Modified 21:39:53 - 19/2/2022
Alpha
Recompile
Vertex Shader
Pixel Shader
rootex/core/renderer/shaders/fire_pixel_sh Path
Slot 0
Textures
rootex/assets/rootex.png
Inspector
column_b.basic.rn
Material_47.basic
flagpole.basic.rma
fabric_e.basic.rma
new_cloth.basic.rr
fabric_d.basic.rma
fabric_g.basic.rma
my_material.custo
fabric_c.basic.rma


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Viewport File Editor
AnimatedModelComponent Model
AudioListenerComponent Audio
BoxColliderComponent Physics
CPUParticlesComponent Effect
CameraComponent General
CapsuleColliderComponent Physics
DecalComponent Effect
DirectionalLightComponent Light
FogComponent Effect
GridModelComponent Model
ModelComponent Model
MusicComponent Audio
ParticleEffectComponent Effect
PlayerController Game
PointLightComponent Light
ShortMusicComponent Audio
SkyComponent Effect
SphereColliderComponent Physics
SpotLightComponent Light
SpriteComponent Model
StaticMeshColliderComponent Physics
StaticPointLightComponent Light
TextUIComponent UI
TransformAnimationComponent General
TransformComponent General
TriggerComponent Physics
UIComponent UI
Scene
EditorCamera # 2
EditorGrid # 3
tutorial_rendering # 4
model # 5
sun # 6
point light # 7
Inspector
ecal # 8
elect an action
eimport
dit Components
etup
dd Empty Scene
opy Scene
elete Scene
aterial
default.decal.rmat
Output
General Warning Error All Clear
ECSFactory::FillEntity: Filled entity: model # 5
ECSFactory::FillEntity: Filled entity: sun # 6
InputManager::loadSchemes: Registered 0 more input schemes on top of 1 existing ones
SceneLoader::loadPreloadedScene::<lambda_66b77903f6bef2c3d347bf644675786a>::operator (): Loaded scene: tu
InspectorDock::drawSceneActions: Added TransformComponent to point light # 7
InspectorDock::drawSceneActions: Added PointLightComponent to point light # 7
InspectorDock::drawSceneActions: Added TransformComponent to decal # 8
InspectorDock::drawSceneActions: Added DecalComponent to decal # 8
Command


Rootex Editor: game/assets/scenes/tutorial_rendering.scene.json
File Assets View Scene Documentation About
Create Resource
Create Script
Create Scene
Open Scene
Save Scene
Preferences
Quit
DecalMaterial
Resource Type
rootex_logo
Resource Name
File Name: game/assets/materials/rootex_logo.decal.rmat
Create
Scene File Viewer
Open Edit Show In File Explorer
Path
Type
Last Modified
D:\techiepriyansh\GithubProjects\Rootex\rootex/assets/
DecalMaterial
20:18:27 - 11/2/2022
Alpha
Decal Texture
rootex/assets/white.png
R:255 G:255 B:255 A:255 Color
Inspector
decal # 8
Select an action
Add Child Scene From File
Save Scene to File
Stop Scene during Pause
Script
DecalComponent TransformComponent
Visible
Material
default.decal.rmat
Output
General Warning Error All Clear
ToolbarDock::disablePlayInEditor: Stopped Scene in Editor
ECSFactory::FillEntity: Filled entity: tutorial_rendering # 4
ECSFactory::FillEntity: Filled entity: model # 5
ECSFactory::FillEntity: Filled entity: sun # 6
ECSFactory::FillEntity: Filled entity: point light # 7
ECSFactory::FillEntity: Filled entity: decal # 8
InputManager::loadSchemes: Registered 0 more input schemes on top of 1 existing ones
SceneLoader::loadPreloadedScene::<lambda_66b77903f6bef2c3d347bf644675786a>::operator (): Loaded scene: tutorial_rendering # 4
Command

3) Now go to the inspector and click DecalComponent.

4) Click on the folder icon and select the decal material.



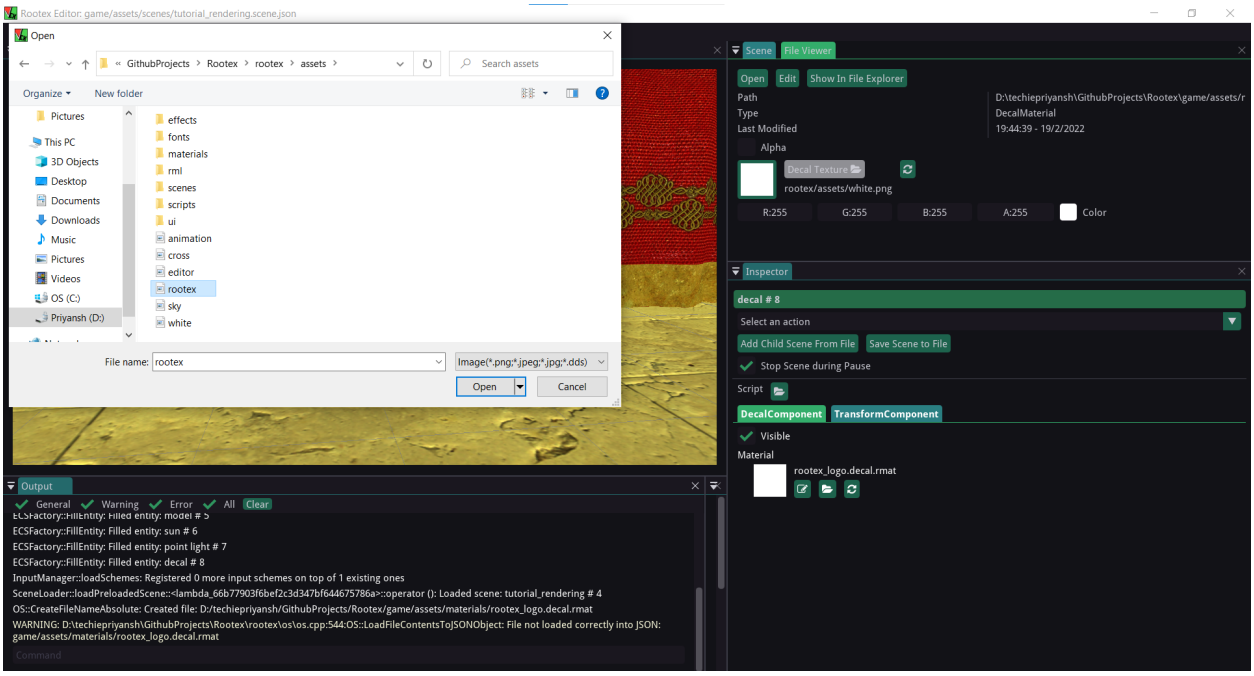

5) Click on the pencil icon and the in the file viewer click on Decal Texture.

6) Shift its position by manipulating the transform component.

By default, the decal shader projects on the negative z-axis. You can rotate it till you get the desired result.

### 2.1.5 Making HUD using UI-component

UI creation is quite easy in Rootex. The UI-component spans the viewport and is not affected by the camera transform, so it can act be used for creating HUD, Menus etc.




## RmlUi

The UI-component makes use of RmlUi. RmlUi is a C++ UI library based on the HTML and CSS standards. It contains of **RML** (based loosely around XHTML 1.0 and HTML 4.01) and **RCSS** (based on CSS2).

RmlUi Documentation

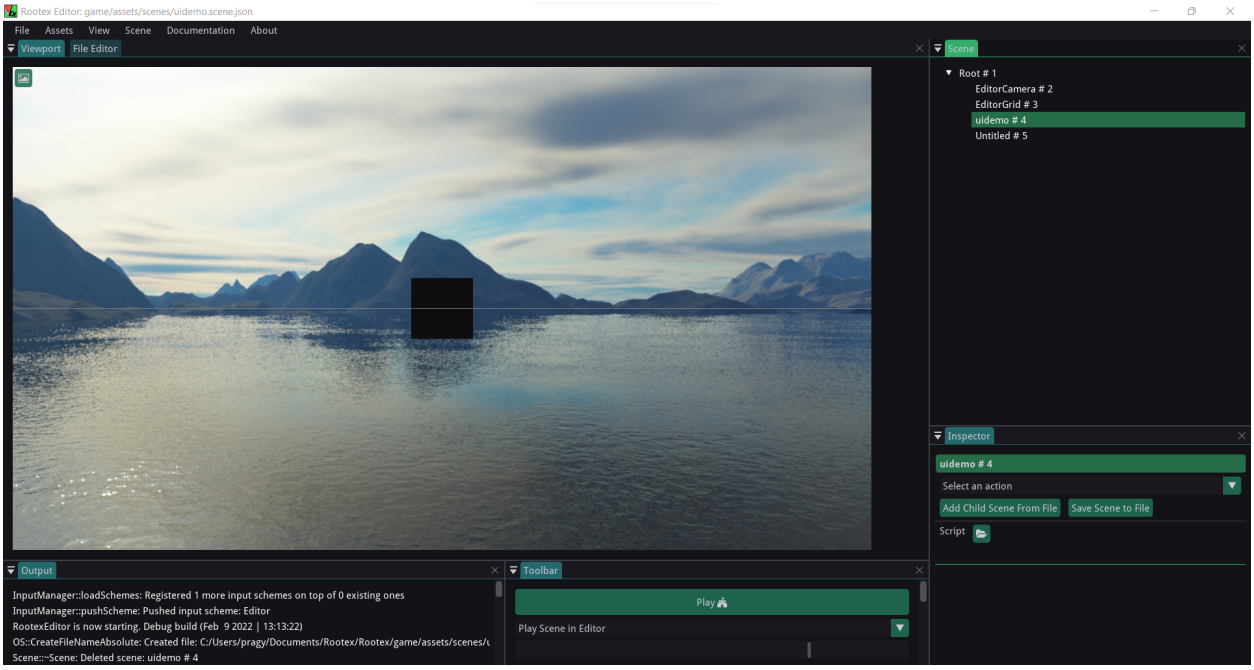
RmlUi Source

## Basic HUD

Let's create an empty scene.

Add a UIComponent to this scene

Make an empty scene with a TransformComponent and ModelComponent (basic cube)




Now add the following RML script to the UIComponent

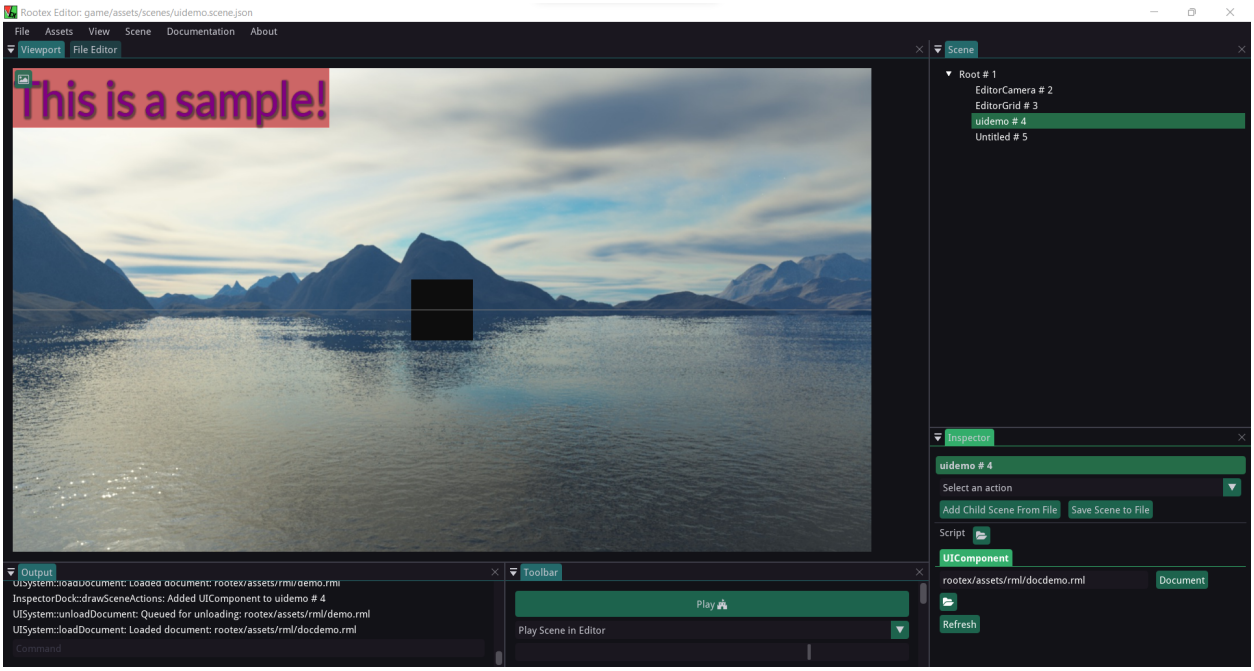
```
    <rml>
            <head>
                    <title>Demo</title>
                    <style>
                            #transition_test {
                                    transition: padding-left background-color transform 1.6s elastic-out;
                                    transform: scale(1.0);
                                    background-color: #c66;
                            }
                            #transition_test:hover {
                                    padding-left: 60px;
                                    transform: scale(1.5);
                                    background-color: #f8b600;
                            }
                            p
                            {
                                    font-family: "Lato";
                                    font-size: 100px;
                                    color: purple;
                                    font-effect: glow(2px 4px 2px 3px #644);
                                    border: 2px #DB6565;
                            }
                    body
                            {
                                    width: 100%;
                                    height: 100%;
                                    margin: auto;
                            }
            </style>
        </head>
        <body>
                    <p id="transition_test">This is a sample!</p>
```

(continues on next page)

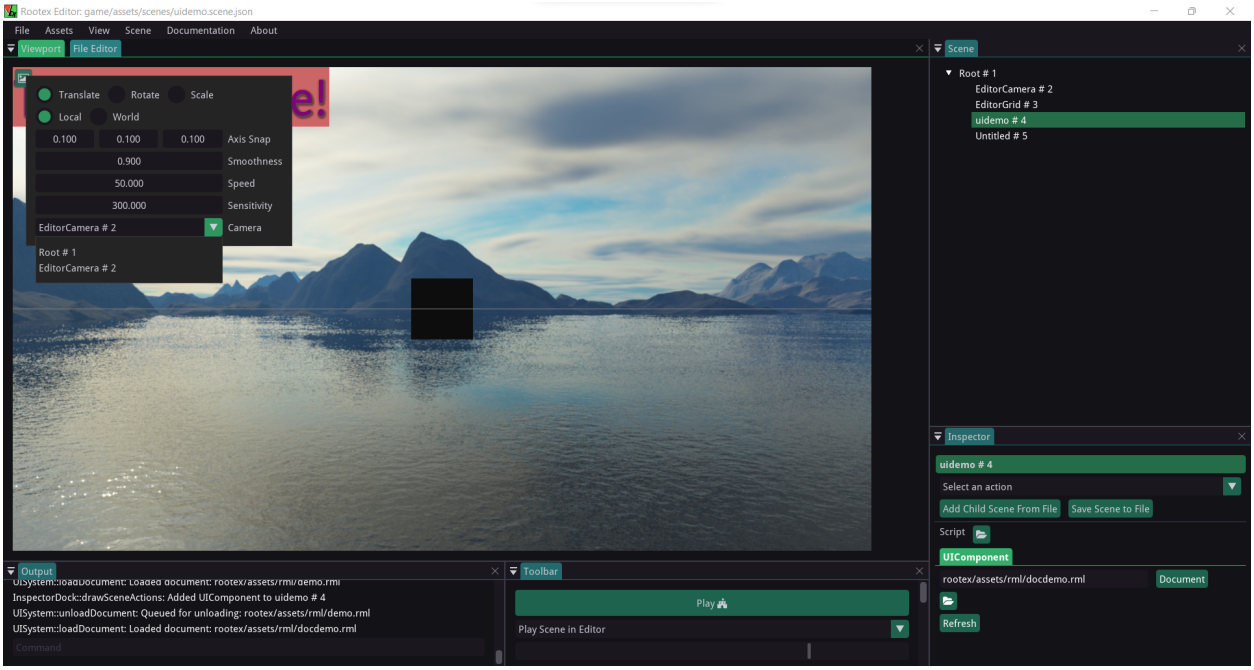
(continued from previous page)

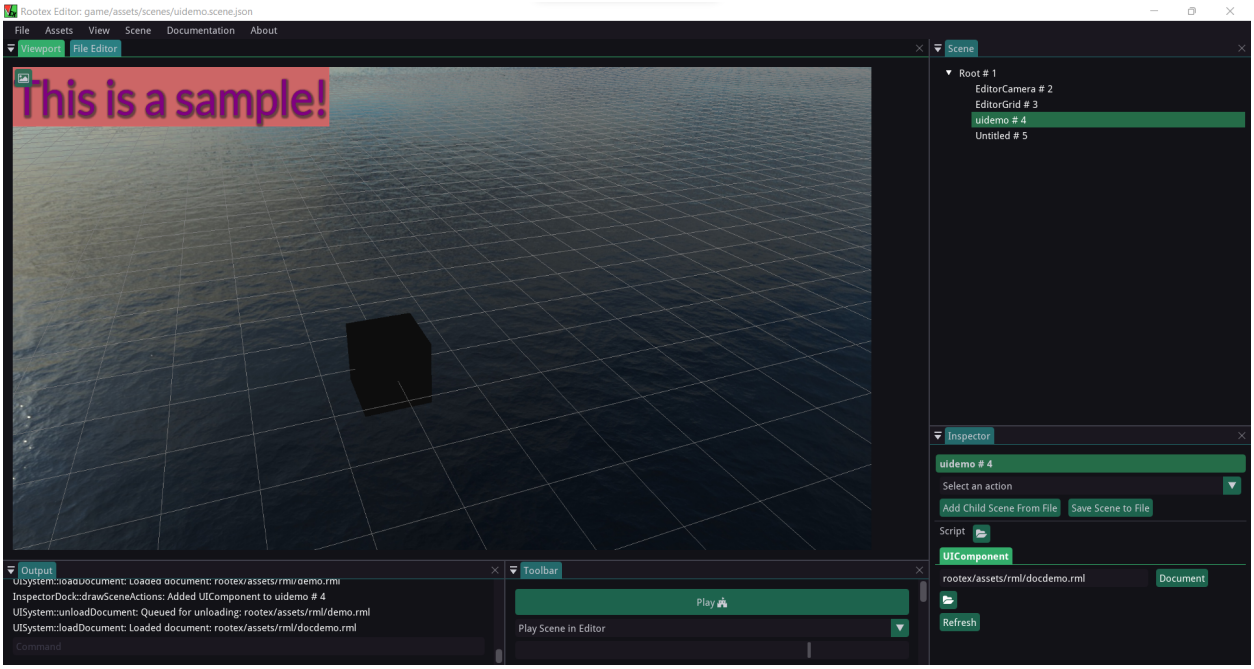
```
        </body>
    </rml>
```

This will add sample text to the top left of our viewport which will react on mouse hover.




Upon playing this scene and switching to EditorCamera, when we move the EditorCamera around, we can see the view of the cube changing, but the HUD stays in place.






## Fade-In effect

We can also make a simple fade-in effect.

RML code:

```
<rml>
    <head>
        <title>Transition</title>
        <style>
            @keyframes fade-in {
                0% {
                    opacity: 0;
                }
                to {
                    opacity: 1;
                }
            }
            body {
                background-color: black;
                width: 100%;
                height: 100%;
                animation: alternate 2s fade-in;
            }
        </style>
    </head>
    <body>
    </body>
</rml>
```

Upon loading this in the UIComponent of our scene, a fade effect will trigger and the scene will go black.

### 2.1.6 Making effects using ParticleEffectComponent

ParticleEffectComponent in Rootex can be used to make awesome particle effects!

#### Effekseer

The ParticleEffectComponent makes use of Effekseer. Effekseer is a particle effect creation tool which can export the effect as an `*.efkefc` file.

This `*.efkefc` file is used by the ParticleEffectComponent in Rootex to import the effects.

You can follow the documentation and tutorial of Effekseer to know how to create effects using it.

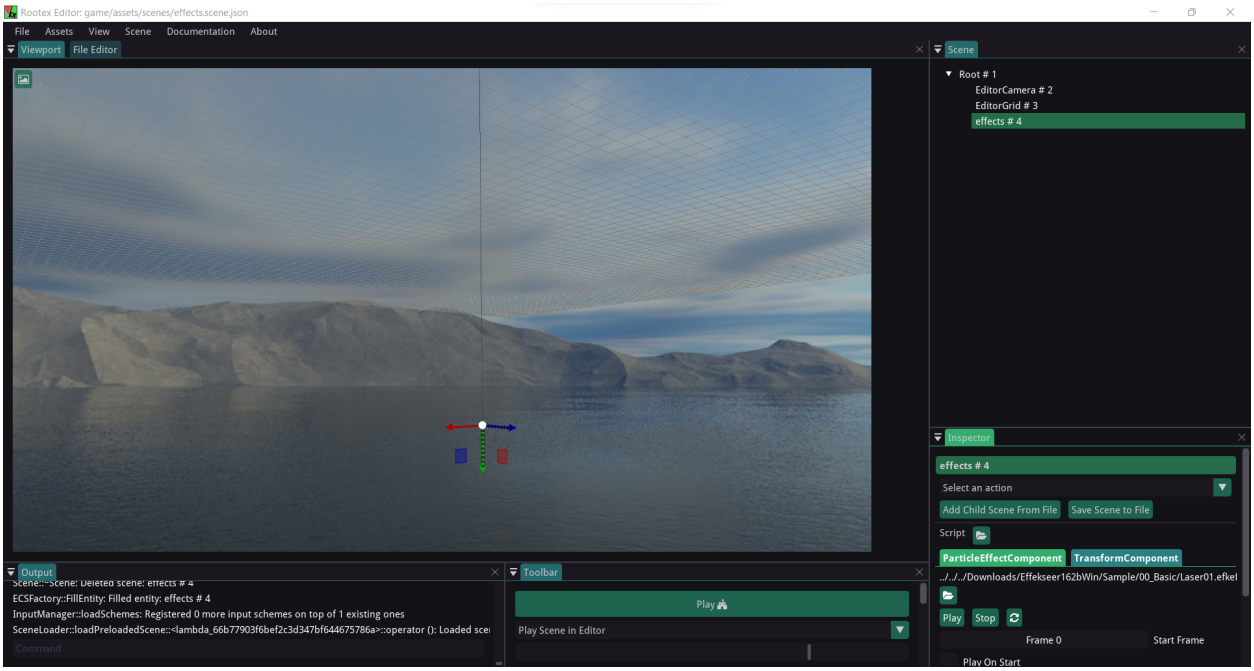
#### Using the Effekseer exports (Demo)

Let's start by creating an empty scene and adding the TransformComponent and ParticleEffectComponent to it.

We have added a demo effect which we get bundled with Effekseer 1.62b (Laser01.efkefc). To get this effect, download Effekseer 1.62b, unzip it and you'll find the effect in Sample/00_Basic/ of the unzipped folder.

Add the effect by going to your scene's ParticleEffectComponent in the Inspector and clicking on the folder icon to select the file.

We have tweaked the EditorCamera to get a better view of the scene.




Explanation of options:

- Play : Plays the loaded effect
- Stop : Stops the playing effect
- Play On Start : Sets whether to play the effect at editor and game start or not.
- Start Frame : Sets the frame from which effect starts.
- Moving : Sets if the effect moves with the transform component once playing.

- Use Speed : Running the effect at a user defined spped. Default speed is 1.

While Play, Stop and Play On Start are pretty self-explanatory, The following explains the other options.

#### Start Frame

An example with Start Frame as Frame 0

An example with Start Frame as Frame 40

In the 2nd example it is visible that the effect starts after skipping the initial part of the first example.

#### Moving

An example with Moving off

An example with Moving on

#### Use Speed

At speed 1

At speed 0.1

### 2.1.7 Getting Help

You can try searching in the documentation to answer your query.

If you are stuck with using Rootex or discover any bug, or if the documentation is not helping you, kindly report that at our issue tracker on Github: http://github.com/sdslabs/rootex/issues

Also you can join SDSLabs' Open Source related Discord server: https://discord.gg/sn4CSvzepP

## 2.2 Architecture

Games using the Rootex Engine are build using the Entity-Component-System architecture.

Rootex is essentally a framework to allow ECS based implementations of games. Details about each of Rootex' important modules is provided below.

### 2.2.1 Framework

Rootex has certain funcitionalities built-in to support an ECS + Godot-like Scene implementation. The main aim of this kind of a architecture is to break down the game into a collection of behaviors and perceiving the game as an interplay of behaviors, rather than hardcoded functionalities inside each game object through object oriented programming. It originates from a place in the programming world that suggests "Composition is better than Inheritance".

ECS just implements a dynamic composition, in the sense that behaviors can be added to game objects at runtime.

One of the popular side benefits of ECS is increased cache coherency.

Scenes allow users to organise and connect their entities into trees, providing powerful features to manipulate entity hierarchies, both at level design time and runtime.

Rootex' ECS architecture has 3 main parts.

### 2.2.2 Component

*Class Component*

A Component is a collection of data for the game to use. Components are analogous to behaviors and components store some peculiar data to maintain their behavior. Component do not do anything else. They may allow changing the data in a certain manner from their public API.

Instances of a type of component are often iterated en masse. Hence all instances of a component type are stored in a special kind of array which allows max MAX_COMPONENT_ARRAY_SIZE instances to exist, but prevents changing location of any instance. The sole ownership of the components lies with this array. Everything else gets raw pointers to elements of this array.

Components can also register dependencies on other components, in form of hard or soft *Define DEPENDENCY*. Hard dependency on a component means that the dependent component cannot function without it and creation of the component is blocked if the dependency isn't fulfilled. Soft dependency is optional dependency, without which a component may be able to function, albeit limitedly and component creation is not blocked if a soft dependency is not fulfilled.

### 2.2.3 Entity

*Class Entity*

An Entity in Rootex is a collection of components. The entity will have a name additionally but all data being used in the game will be stored in one of the components of an entity. Entities provide the component with an identity so that components can be theorized to "belong" to a thing in the game. Entites can be globally identified from their IDs, which is guaranteed to be uniquely generated.

An Entity stores a HashMap of all the components assigned to it. The sole ownership(Ptr) of an entity belongs to it's owner scene. Everything else gets raw pointers.

### 2.2.4 System

*Class System*

A System in Rootex is containing all the logic/algorithms that are needs to make sense of the data that is stored inside a specific type of component. Systems only interact with a certain type of components. In Rootex, all components of similar type are stored in an array and all these arrays containing different types of components are stored in a hash map so that the array having an component type can be indexed and used for processing by a _exhale_class_class_system.

Systems often require iterating over componets of a type, this can only be accomplished by using range based for loops as we have a custom iterator that only returns "valid" elements from our custom component array.

---

#### Scene

A scene is a hierarchical data structure which can optionally store and entity. It stores children nodes of the same type and controls their lifetime based of if the parent is alive. Once the parent has decided to kill itself, it makes sure its children also meet the same fate.

Scene's store a *Typedef Ptr* to an entity and provide structure to out hybrid ECS + Scene architecture.

Rootex uses JSON style serializations to store and load scenes. Entities are also created from these files with all the necessary components.

A scene subtree can be saved to a file/loaded from a file to allow functionality reuse both during level design phase and runtime.

Scene files are recognized by *.scene.json* file extensions.

Scene construction is handled by *Class Scene* itself but it delegates the entity construction using the JSON data to Class ECSFactory.

Each *Class Component* accepts owner entity and constituent json data in its constructor, allowing it to setup its data memebrs from the serialised data. Every Component also defines a getJSON method which serialises its members back to JSON format. The data retention across engine restarts is guarranteed through ensuring that the component use the same data which it generates while saving the scene.

### 2.2.5 Pausing

Pausing is tightly bound to ECS + Scenes. The engine provides a pause UI scene, which is enbled on pressing ESC key. All scenes which have the "Stop Scene during Pause" checkbox checked will have most of their components and scripts being skipped by Systems, effectively bringing the game logic to a pause. Certain scenes can be exempted from being skipped to allow stuff like Music to keep playing.

### 2.2.6 Event Manager

Events are a way to solve the problem of sphagetti code, and an efficient way to implement the publisher-subscriber design pattern in a considerably large codebase.

Events (*Class Event*) in Rootex are the equivalent of broadcast messages of a particular channel and Rootex' Event Manager (*Class EventManager*) is the equivalent of a broadcast company managing multiple channels. Functions can be registered as callbacks to events. When an event is called, all the functions registered to that event are called with the corresponding data related to the cause of origin of that event. Rootex' event manager has the ability to call both global functions and member functions (with the corresponding object that registered its member function, as a parameter into the member function, which is how C++ implements member functions).

*Class EventManager* is a singleton, and all engine level events are passed by it. User events can also be channeled through with no issues. E.g. Input events that are configured by the user are sent through the engine level event manager.

Optionally EventManager allows Typedef Variant data to be passed along with an event call, which are further of 2 types:

- Call: The registered handlers are called immediately. This is useful for non destructive activity.

- Deferred Call: The registered handlers are called at the end of a frame. This is especially useful for destrutive activity related to entities, components and systems, as the engine is very likely to be iterating on them and deletion may lead to corruption.

Rootex editor has been made on top of the rootex engine. The engine is never aware of the existence of editor. This is made possible by the help of events.

See *Struct RootexEvents*, *Struct EditorEvents*, *Class Event*

### 2.2.7 Multithreading

Rootex engine is multithreading ready, however its main focus is on single threaded operations.

Every Rootex application (*Class Application*) has a pool of threads, called simply a threadpool in common CS language. These threads can be assigned work either by the engine or game code.

Rootex uses the concept of Worker threads, a.k.a. Job Based multithreading.

At startup, Rootex' threadpool manager (*Class ThreadPool*) queries the CPU and returns the number of logical CPU cores in the system. The threadpool allocates the same number of threads and uses one of them to be the master thread that distributes "jobs" to different threads. Jobs are implemented as simple overriden virtual functions of *Class Task*.

During testing Rootex was run simply as a single threaded engine. As time went on, certain functions of Rootex were run in separate threads in a controlled multithreading environment.

### 2.2.8 Resources

Resources are first class members in the Rootex Engine. By resources, we refer to any file loaded data, that is processed and ready to be used in the engine.

### 2.2.9 ResourceLoader

Resources are created and owned by the *Class ResourceLoader* and distributed to the user and the engine as pointers to instances of the polymorphic *Class ResourceFile*. *Class ResourceFile* has been subclassed multiple times to store different kinds of data like sounds, music, images, fonts, 3D models, normal text files like Lua files or JSON files, etc. Look up the documentation on the resource loader for more information.

Resources are often the heaviest parts of a game in terms of actual memory that they occupy. *Class ResourceLoader* has been designed in such a manner that stores resources and distributes the earlier cached resource again instead of loading the same resource again to save memory, in case the same resource is instructed to be loaded more than once. Hence, the engine and the users need not worry about not loading the same resources multiple times.

Resources may change in the file system after they have been loaded by the engine. To fix this, all resources can detect if they have been changed by the file system and have the ability to reload their contents and re-process it on demand.

Resource Loading can often be a bottleneck for complex scenes in a simple game engine. Rootex provides 2 mechanisms to make Resoource handling smooth:

- Multithreaded loading: ResourceFiles are loaded to memory in parallel and the create objects are allowed to reference each other once basic setup is done.
- Preloads: A scene can define a list of filepaths as preloads. These resources are loaded into Resource cache during the initial loading of the scene, even if they're not being directly referenced. By specifying preloads, We can prevent frame drops due to disk reads loading mid-gameplay.

### 2.2.10 Audio

Audio in Rootex has been implemented using OpenAL 1.1. Rootex supports both stereo and mono sound effects, as well as longer duration music.

Rootex is aware of the delay that might occur while trying to play large length audio pieces at once. To rectify this, Rootex implements Audio Streaming with *Class MusicComponent* and shorter sound effects that need to be loaded and played as fast as possible are implemented as *Class ShortMusicComponent*.

Rootex also supports audio attenuation models like Linear, Exponential and their respective clamped versions, as offered by OpenAL. However, audio attenuation works only with mono channel audio.

### 2.2.11 Rendering

Rootex uses DirectX 11 to render graphics.

Rendering in Rootex has been implemented with special attention so that it behaves properly with our ECS architecture. The *Class RenderSystem* uses the *Class RenderableComponent* to share common funcitonalities across different components which add to the visuals of the scene.

The *Class RenderSystem* uses the owning *Class Scene* of the renderable component to recursively traverse the object hierarchy, starting from the root entity (which is persistent across levels). Every time the render system recognizes a parent, before processing its children, the render system takes note of the transform (a representation of position, rotation and scale all at once) of the parent and appends it to the transformation stack. The transformation stack is an implementation for inheriting transforms from the parent entity of a child entity, used while performing a Depth-First-Search on the component hierarchy established by hierarchy component instances.

The transformation stack of UI components is kept separate from the transformation stack of 3D world visual components.

Once all transformations are updated, *Class RenderSystem* loops over all the renderable components and does the rendering required to show them.

Rootex also performs sky, fog and related rendering effects and post processing effects.

### 2.2.12 Physics

Physics in Rootex has been implemented using the Bullet Collision Detection and Physics library (https://pybullet.org/Bullet/BulletFull/index.html)

Rootex uses the concept of colliders, adopted from the Bullet Physics library, to represent objects responding to physics and being controlled by it.

*Class PhysicsSystem* is responsible for providing physics to the Rootex engine. The physics system uses Class PhysicsColliderComponent instances to perform physics based calculations on them. Rootex engine currently supports all the collision shapes provides by Bullet. There are also extra features available like ray casting a ray into the world and reporting the colliders that the ray touched.

The Rootex Editor also helps in visualizing the collider shapes enabled by viewing the collider component in the Inspector.

### 2.2.13 Inputs

User inputs in Rootex are handled with polling and/or callback based mechanisms using the event system. Rootex also abstracts the exact button press occurences from the user and forces the user to use keybinding names strings to query for input.

*Class InputManager* handles the input for the entire engine. The input manager is initially feeded with a collection of 'input schemes'. These input schemes are defined in the level JSON files likewise:

```
// flappy_bird.scene.json
{
    "camera": 23,
    "inputSchemes": {
        "FlappyBird": {
            "bools": [
                {
                    "inputEvent": "Jump",
                    "device": 1,
                    "button": 99
                }
            ],
            "floats": []
        },
    },
    "startScheme": "FlappyBird"
}
```

The field `inputSchemes` is the collection of input schemes that the input manager will recognize. In this example, there is only 1 scheme called "FlappyBird", but a game can have multiple input schemes and only one of those input schemes can be active at a time. The field `startScheme` tells the Rootex engine which input scheme should be selected as soon as the level is loaded up.

Each input scheme has a name identified by its key and the following fields:

- `bools`: Array of inputs that are represented as boolean values. Used for buttons that are either held down or not held down.
- `floats`: Array of inputs that are not digital in nature and rather are analogous, like mouse positions and joystick movements.

Inside each input keybinding, there are fields:

- `inputEvent`: The event name that gets emitted as soon as the input changes state. This need not be unique across other keybindings under the same input scheme or even across input schemes.
- `device`: The device enum value that this input keybinding is present on.
- `button`: The button value that is mapped to the keybinding

### 2.2.14 Scripting

Rootex Engine has a fully scriptable interface implemented using Lua and the Sol3 (https://sol2.readthedocs.io/en/latest/) library for creating bindings.

Scripts i.e. Lua files, can be attached to entities and can define functions upon those entities.

See Middleclass (https://github.com/kikito/middleclass) for details on the `class()` based Lua OOP support.

```
EmptyScript = class("EmptyScript")

-- First method called after script initialisation
-- not safe to refer other entity script tables here
-- setup initial data members here that don't refer entities
function EmptyScript:begin(entity)
end
```

(continues on next page)

(continued from previous page)

```
-- Called after all `begin` for the frame have been called
-- safe to assume that all scripts have `begin`ed and have
-- data members
function EmptyScript:enterScene(entity)
    print("Nothing is true")
end

-- called once every frame
function EmptyScript:update(entity, delta)
end

-- called during entity destruction
function EmptyScript:destroy(entity)
    print("Everything is permitted")
end

-- called when Collider of the entity detects a hit
function EmptyScript:hit(hit)
    print("Everything is permitted")
end

-- called when entity enters a TriggerComponent
function EmptyScript:enterTrigger(entity, trigger)
end

-- called when entity exits a TriggerComponent
function EmptyScript:exitTrigger(entity, trigger)
end

return EmptyScript
```

The functions are called into Lua from Rootex on the command of the *Class ScriptSystem*.

The script files are run in a Lua VM and the Rootex functions available are registed by the *Class LuaInterpreter*'s implementation. The Lua scripting interface for Rootex mostly looks the same as the Rootex engine API that the engine uses internally to provide as vast a scripting environment as possible. All Rootex class names and functions are hidden under the `RTX` global Lua variable. An object which has its constructor registered can be constructed from scripts as `RTX.Type.new(args)`.

### 2.2.15 Scripting API

You can find the scripting API and related docs in the editor itself.

## 2.3 Rootex

### 2.3.1 Full API

#### Namespaces

#### Namespace ECSFactory

Contents



### Functions



### Variables



### Namespace nlohmann

Namespace for the JSON library.

Contents



### Classes

### Classes and Structs

### Struct AnimatedVertexData

- Defined in file_rootex_core_renderer_vertex_data.h

### Inheritance Relationships

### Base Type

- `public VertexData` (*Struct VertexData*)

### Struct Documentation

**`struct AnimatedVertexData : public`** *VertexData*

#### Public Members

int **`boneIndices`**[4]

*Vector4* **`boneWeights`**

### Struct BasicMaterialData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**`struct BasicMaterialData`**

#### Public Members

*String* **`diffuseImage`**

*String* **`normalImage`**

*String* **`specularImage`**

*String* **`lightmapImage`**

*PerModelPSCBData* **`pixelBufferData`**

### Struct Component::Category

- Defined in file_rootex_framework_component.h

### Nested Relationships

This struct is a nested type of *Class Component*.

### Struct Documentation

**struct Category**

#### Public Static Attributes

**const** *String* **General** = "General"

**const** *String* **Audio** = "Audio"

**const** *String* **Game** = "Game"

**const** *String* **Physics** = "Physics"

**const** *String* **Model** = "Model"

**const** *String* **Effect** = "Effect"

**const** *String* **Light** = "Light"

**const** *String* **UI** = "UI"

### Struct ContentBrowser::ContentBrowserSettings

- Defined in file_editor_gui_content_browser.h

### Nested Relationships

This struct is a nested type of *Class ContentBrowser*.

### Struct Documentation

**struct ContentBrowserSettings**

#### Public Members

bool **m_IsActive** = true

### Struct CPUParticlesComponent::Particle

- Defined in file_rootex_framework_components_visual_effect_cpu_particles_component.h

### Nested Relationships

This struct is a nested type of *Class CPUParticlesComponent*.

### Struct Documentation

**struct Particle**

#### Public Members

float **sizeBegin**

float **sizeEnd**

float **lifeTime**

float **lifeRemaining**

*Color* **colorBegin**

*Color* **colorEnd**

*Vector3* **velocity**

*Vector3* **angularVelocity**

*Vector3* **position**

*Quaternion* **rotation**

*Vector3* **scale**

### Struct CustomMaterialData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct CustomMaterialData**

#### Public Members

*String* **vertexShaderPath**

*String* **pixelShaderPath**

*Vector*<*Ref*<*ImageResourceFile*>> **vertexShaderTextures**

*Vector*<*Ref*<*ImageResourceFile*>> **pixelShaderTextures**

*Vector*<float> **customConstantBuffers**

*Vector*<*TYPES_OF_BUFFERS*> **typeOfCustomConstantBuffers**

### Struct CustomRenderInterface::GeometryData

- Defined in file_rootex_core_ui_custom_render_interface.h

### Nested Relationships

This struct is a nested type of *Class CustomRenderInterface*.

### Struct Documentation

**struct GeometryData**

#### Public Functions

**GeometryData**(**const** *UIVertexData* *vertices, size_t verticesSize, int *indices, size_t indicesSize, Rml::TextureHandle texture)

#### Public Members

*VertexBuffer* **vertexBuffer**

*IndexBuffer* **indexBuffer**

Rml::TextureHandle **textureHandle**

### Struct DecalMaterialData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct DecalMaterialData**

#### Public Members

*String* **decalImage**

*PerModelDecalPSCBData* **pixelBufferData**

### Struct DirectionalLight

- Defined in file_rootex_core_renderer_directional_light.h

### Struct Documentation

**struct DirectionalLight**

#### Public Members

float **diffuseIntensity**
Diffuse intensity of light.

*Color* **diffuseColor**
Diffuse color of light.

*Color* **ambientColor**
Ambient color of light.

### Struct DirectionalLightInfo

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct DirectionalLightInfo**
Used to bind a directional light to the Pixel shader.

#### Public Members

*Vector3* **direction** = {1.0f, , }

float **diffuseIntensity** = 0.0f

*Color* **ambientColor** = {0.05f, , , }

*Color* **diffuseColor** = {1.0f, , , }

### Struct EditorEvents

- Defined in file_editor_editor_events.h

### Struct Documentation

**struct EditorEvents**

#### Public Functions

**DEFINE_EVENT**(EditorAutoSave)
Auto-saving event.

**DEFINE_EVENT**(EditorSceneIsClosing)
Close any scene opened in Inspector.

**DEFINE_EVENT**(EditorCreateNewScene, *String*)
Create new scene from scene file path.

**DEFINE_EVENT**(EditorCreateNewFile, *String*)
Create new file at path.

**DEFINE_EVENT**(EditorOpenFile, *String*, int)
Open file at the path passed in inside File Viewer with the *ResourceFile::Type* to load it as.

**DEFINE_EVENT**(EditorEditFile, *String*)
Edit file at the path passed in inside File Editor.

**DEFINE_EVENT**(EditorOpenScene, *Scene* *)
Open scene in Inspector.

**DEFINE_EVENT**(EditorSaveBeforeQuit)
Set editor to ask to save before quitting.

**DEFINE_EVENT**(EditorSaveAll)
Set editor to save all.

**DEFINE_EVENT**(EditorReset)
Reset editor window states.

### Struct EditorSystem::Icons

- Defined in file_editor_editor_system.h

### Nested Relationships

This struct is a nested type of *Class EditorSystem*.

### Struct Documentation

**struct Icons**

#### Public Members

**const** char ***lua** = ICON_ROOTEX_FILE_CODE_O

**const** char ***font** = ICON_ROOTEX_FONT

**const** char ***text** = ICON_ROOTEX_FILE_TEXT

**const** char ***audio** = ICON_ROOTEX_FILE_AUDIO_O

**const** char ***model** = ICON_ROOTEX_FORT_AWESOME

**const** char ***image** = ICON_ROOTEX_FILE_IMAGE_O

### Struct FlipbookDecorator::FlipbookElementData

- Defined in file_rootex_core_ui_flipbook_decorator.h

### Nested Relationships

This struct is a nested type of *Class FlipbookDecorator*.

### Struct Documentation

**struct FlipbookElementData**

#### Public Functions

**FlipbookElementData**()

**~FlipbookElementData**()

#### Public Members

float **currentFrame** = 0

Rml::Geometry **geometry**

#### Public Static Attributes

*Vector*<*FlipbookElementData* *> **s_DataInstances**

### Struct FXAAData

- Defined in file_rootex_core_renderer_vertex_data.h

### Struct Documentation

**struct FXAAData**

#### Public Members

*Vector3* **position**

*Vector2* **texturecoord**

### Struct GodRaysData

- Defined in file_rootex_core_renderer_vertex_data.h

### Struct Documentation

**struct GodRaysData**

#### Public Members

*Vector3* **position**

*Vector2* **texturecoord**

### Struct Hit

- Defined in file_rootex_framework_components_physics_hit.h

### Struct Documentation

**struct Hit**

#### Public Functions

**Hit**(*Entity* *left, *Entity* *right)

**Hit**(**const** *Hit*&)

**~Hit**()

#### Public Members

*Entity* ***thisOne**

*Entity* ***thatOne**

### Template Struct IndexTriangleList

- Defined in file_rootex_core_renderer_index_triangle_list.h

### Struct Documentation

template<typename **T**>
**struct IndexTriangleList**
Encapsulated Vertices and Indices.

#### Public Members

std::vector<T> **vertices**

std::vector<unsigned short> **indices**

### Struct InputDescription

- Defined in file_rootex_core_input_input_manager.h

### Struct Documentation

**struct InputDescription**

#### Public Members

*Device* **device**

*DeviceButtonID* **button**

*Event*::*Type* **inputEvent**

### Struct InputScheme

- Defined in file_rootex_core_input_input_manager.h

### Struct Documentation

**struct InputScheme**

#### Public Members

*Vector*<*InputDescription*> **bools**

*Vector*<*InputDescription*> **floats**

bool **isActive**

### Struct InspectorDock::InspectorSettings

- Defined in file_editor_gui_inspector_dock.h

### Nested Relationships

This struct is a nested type of *Class InspectorDock*.

### Struct Documentation

**struct InspectorSettings**

#### Public Members

bool **m_IsActive** = true

### Struct InstanceData

- Defined in file_rootex_core_renderer_vertex_data.h

### Struct Documentation

**struct InstanceData**

#### Public Functions

**InstanceData**()

**InstanceData**(**const** *Matrix* &*matrix*, **const** *Color* &*instanceColor*)

#### Public Members

*Matrix* **transform**

*Matrix* **inverseTransposeTransform**

*Color* **color**

### Struct LightsInfo

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct LightsInfo**

#### Public Members

*Vector3* **cameraPos**

int **pointLightCount** = 0

*PointLightInfo* **pointLightInfos**[**MAX_DYNAMIC_POINT_LIGHTS**]

int **directionalLightPresent** = 0

float **pad2**[3]

*DirectionalLightInfo* **directionalLightInfo**

int **spotLightCount** = 0

float **pad3**[3]

*SpotLightInfo* **spotLightInfos**[**MAX_DYNAMIC_SPOT_LIGHTS**]

### Struct MasterThread

- Defined in file_rootex_os_thread.h

### Struct Documentation

**struct MasterThread**

Master thread to communicate with and maintain worker threads.

#### Public Members

*TaskComplete* **m_TasksComplete**

*TaskReady* **m_TasksReady**

### Struct Mesh

- Defined in file_rootex_core_renderer_mesh.h

### Struct Documentation

**struct Mesh**

#### Public Functions

**Mesh**()

**Mesh**(**const** *Mesh*&)

**~Mesh**()

void **addLOD**(*Ref*<*IndexBuffer*> *ib*, float *lodLevel*)

*Ref*<*IndexBuffer*> **getLOD**(float *lodLevel*) **const**
LOD level can range from 0.0f to 1.0f, returns the appropriate LOD.

*Ref*<*VertexBuffer*> **getVertexBuffer**()

**const** *BoundingBox* &**getBoundingBox**()

#### Public Members

*Ref*<*VertexBuffer*> **m_VertexBuffer**

*BoundingBox* **m_BoundingBox**

*Vector*<*Pair*<*Ref*<*IndexBuffer*>, float>> **m_LODs**

### Template Struct adl_serializer< BoundingBox >

- Defined in file_rootex_common_types.h

### Struct Documentation

template<>
**struct adl_serializer**<*BoundingBox*>

#### Public Static Functions

**static** void **to_json**(json &*j*, **const** *BoundingBox* &*v*)

**static** void **from_json**(**const** json &*j*, *BoundingBox* &*v*)

### Template Struct adl_serializer< Color >

- Defined in file_rootex_common_types.h

### Struct Documentation

template<>
**struct adl_serializer**<*Color*>

#### Public Static Functions

**static** void **to_json**(json &*j*, **const** *Color* &*v*)

**static** void **from_json**(**const** json &*j*, *Color* &*v*)

### Template Struct adl_serializer< Matrix >

- Defined in file_rootex_common_types.h

### Struct Documentation

template<>
**struct adl_serializer**<*Matrix*>

#### Public Static Functions

**static** void **to_json**(json &*j*, **const** *Matrix* &*v*)

**static** void **from_json**(**const** json &*j*, *Matrix* &*v*)

### Template Struct adl_serializer< Quaternion >

- Defined in file_rootex_common_types.h

### Struct Documentation

template<>
**struct adl_serializer**<*Quaternion*>

#### Public Static Functions

**static** void **to_json**(json &*j*, **const** *Quaternion* &*v*)

**static** void **from_json**(**const** json &*j*, *Quaternion* &*v*)

### Template Struct adl_serializer< Vector2 >

- Defined in file_rootex_common_types.h

### Struct Documentation

template<>
**struct adl_serializer**<*Vector2*>

#### Public Static Functions

**static** void **to_json**(json &*j*, **const** *Vector2* &*v*)

**static** void **from_json**(**const** json &*j*, *Vector2* &*v*)

### Template Struct adl_serializer< Vector3 >

- Defined in file_rootex_common_types.h

### Struct Documentation

template<>
**struct adl_serializer**<*Vector3*>

#### Public Static Functions

**static** void **to_json**(json &*j*, **const** *Vector3* &*v*)

**static** void **from_json**(**const** json &*j*, *Vector3* &*v*)

### Template Struct adl_serializer< Vector4 >

- Defined in file_rootex_common_types.h

### Struct Documentation

template<>
**struct adl_serializer**<*Vector4*>

#### Public Static Functions

**static** void **to_json**(json &*j*, **const** *Vector4* &*v*)

**static** void **from_json**(**const** json &*j*, *Vector4* &*v*)

### Struct OutputDock::OutputDockSettings

- Defined in file_editor_gui_output_dock.h

### Nested Relationships

This struct is a nested type of *Class OutputDock*.

### Struct Documentation

**struct OutputDockSettings**

#### Public Members

bool **m_IsActive** = true

### Struct ParticleTemplate

- Defined in file_rootex_framework_components_visual_effect_cpu_particles_component.h

### Struct Documentation

**struct ParticleTemplate**

#### Public Members

*Vector3* **velocity** = {1.0f, , }

*Color* **colorBegin** = ColorPresets::Red

*Color* **colorEnd** = ColorPresets::Blue

float **velocityVariation** = 10.0f

float **rotationVariation** = DirectX::XM_PI

float **angularVelocityVariation** = 0.5f

float **sizeBegin** = 0.1f

float **sizeEnd** = 0.0f

float **sizeVariation** = 0.1f

float **lifeTime** = 1.0f

### Struct PerCameraChangePSCB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PerCameraChangePSCB**

#### Public Members

*Vector2* **DepthUnpackConsts**

*Vector2* **Viewport2xPixelSize**

*Vector2* **CameraTanHalfFOV**

*Vector2* **pad**

### Struct PerDecalPSCB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PerDecalPSCB**

#### Public Members

*Vector3* **decalRight**

float **pad1**

*Vector3* **decalForward**

float **pad2**

*Vector3* **decalUp**

float **pad3**

*Vector3* **decalHalfScale**

float **pad4**

*Vector3* **decalViewspacePosition**

float **pad5**

### Struct PerFrameCustomPSCBData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct PerFrameCustomPSCBData**

#### Public Members

float **timeMs**

float **deltaTimeMs**

*Vector2* **resolution**

*Vector2* **mouse**

float **pad**[2]

### Struct PerFramePSCB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PerFramePSCB**
Constant buffer uploaded once per frame in the PS.

#### Public Members

*LightsInfo* **lights**

*Color* **fogColor**

### Struct PerFrameVSCB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PerFrameVSCB**
Constant buffer uploaded once per frame in the VS.

#### Public Members

*Matrix* **view**

float **fogStart**

float **fogEnd**

*LightsInfo* **light**

float **pad**[2]

### Struct PerModelAnimationVSCBData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct PerModelAnimationVSCBData**

#### Public Functions

**PerModelAnimationVSCBData**()

**PerModelAnimationVSCBData**(**const** *Vector*<*Matrix*> &*transforms*)

#### Public Members

*Matrix* **m_BoneTransforms**[**MAX_BONES**]

### Struct PerModelDecalPSCBData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct PerModelDecalPSCBData**

#### Public Members

*Color* **color**

### Struct PerModelPSCB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PerModelPSCB**

#### Public Members

int **staticPointsLightsAffectingCount** = 0

float **pad**[3]

*StaticLightID* **staticPointsLightsAffecting**[**MAX_STATIC_POINT_LIGHTS_AFFECTING_1_OBJECT**]

### Struct PerModelPSCBData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct PerModelPSCBData**

Kept separate from the main data buffer class because it needs proper packing.

#### Public Members

*Color* **color**

int **isLit** = 0

float **specularIntensity** = 2.0f
Describes brightness of specular spot, high for metallic material.

float **specularPower** = 30.0f
Describes angular fall-off of specular spot, high for metallic material.

float **reflectivity** = 0.0f

float **refractionConstant** = 0.5f

float **refractivity** = 0.0f

int **affectedBySky** = 0

int **hasNormalMap** = 0

float **fresnelPower** = 0.0f

float **fresnelBrightness** = 0.0f

float **pad**[2]

### Struct PerModelVSCBData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct PerModelVSCBData**

#### Public Functions

**PerModelVSCBData**()

**PerModelVSCBData**(**const** *Matrix* &*modelMatrix*, int *hasNormalMap* = 0)

#### Public Members

*Matrix* **model**

*Matrix* **modelInverseTranspose**

int **hasNormalMap**

int **pad**[3]

### Struct PerScenePSCB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PerScenePSCB**
Constant buffer uploaded at least once per scene load in the PS.

#### Public Members

*StaticPointLightsInfo* **staticLights**

### Struct PhysicsMaterialData

- Defined in file_rootex_framework_systems_physics_system.h

### Struct Documentation

**struct PhysicsMaterialData**

#### Public Members

float **restitution** = 1.0f

float **friction** = 1.0f

float **specificGravity** = 1.0f

### Struct PointLight

- Defined in file_rootex_core_renderer_point_light.h

### Struct Documentation

**struct PointLight**

#### Public Members

float **attConst**
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **attLin**
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **attQuad**
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **range**
Lighting effect clipped for distance > range.

float **diffuseIntensity**
Diffuse intensity of light.

*Color* **diffuseColor**
Diffuse color of light.

*Color* **ambientColor**
Ambient color of light.

### Struct PointLightInfo

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PointLightInfo**
Used to bind a point light to the Pixel shader.

#### Public Members

*Color* **ambientColor** = {0.0f, , , }

*Color* **diffuseColor** = {0.0f, , , }

float **diffuseIntensity** = 0.0f

float **attConst** = 1.0f
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **attLin** = 0.045f
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **attQuad** = 0.0075f
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

*Vector3* **lightPos** = {0.0f, , }
Is filled with the *TransformComponent* while rendering.

float **range** = 0.0f
Lighting effect clipped for distance > range.

### Struct PostProcessingDetails

- Defined in file_rootex_core_renderer_post_processor.h

### Struct Documentation

**struct PostProcessingDetails**

#### Public Members

bool **isPostProcessing** = false

bool **isGodRays** = false

bool **isASSAO** = false

bool **isBloom** = false

bool **isSepia** = false

bool **isMonochrome** = false

bool **isGaussianBlur** = false

bool **isToneMap** = false

bool **isFXAA** = false

int **godRaysNumSamples** = 100

float **godRaysDensity** = 1.0f

float **godRaysWeight** = 0.01f

float **godRaysDecay** = 1.0f

float **godRaysExposure** = 1.0f

float **assaoRadius** = 1.2f

float **assaoDetailShadowStrength** = 0.5f

int **assaoBlurPassCount** = 2

float **assaoFadeOutFrom** = 50.0f

float **assaoFadeOutTo** = 300.0f

float **assaoHorizonAngleThreshold** = 0.06f

int **assaoQualityLevel** = 2

float **assaoShadowClamp** = 0.98f

float **assaoShadowMultiplier** = 1.0f

float **assaoShadowPower** = 1.5f

float **assaoSharpness** = 0.98f

float **assaoAdaptiveQualityLimit** = 0.45f

float **bloomThreshold** = 0.8f

float **bloomSize** = 1.0f

float **bloomBrightness** = 1.0f

float **bloomValue** = 1.0f

float **bloomBase** = 1.0f

float **bloomSaturation** = 1.0f

float **bloomBaseSaturation** = 1.0f

float **gaussianBlurMultiplier** = 1.0f

float **toneMapExposure** = 0.0f

int **toneMapOperator** = 0

int **toneMapTransferFunction** = 0

float **toneMapWhiteNits** = 200.0f

*Map*<*String*, bool> **customPostProcessing**

### Struct PSFXAACB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PSFXAACB**
PS constant buffer used during the FXAA post process.

#### Public Members

*Vector4* **rcpFrame**
{ 1.0f / screenWidth, 1.0f / screenHeight, 0.0f, 0.0f }

### Struct PSGodRaysCB

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct PSGodRaysCB**

#### Public Members

*Vector3* **sunScreenSpacePos**

int **numSamples** = 0

float **density**

float **weight**

float **decay**

float **exposure**

### Struct RenderSystem::LineRequests

- Defined in file_rootex_framework_systems_render_system.h

### Nested Relationships

This struct is a nested type of *Class RenderSystem*.

### Struct Documentation

**struct LineRequests**

#### Public Members

*Vector*<float> **m_Endpoints**

*Vector*<unsigned short> **m_Indices**

### Struct RootexEvents

- Defined in file_rootex_core_event.h

### Struct Documentation

**struct RootexEvents**

#### Public Functions

**DEFINE_EVENT**(ApplicationExit)
*Application* exited the main loop.

**DEFINE_EVENT**(DeleteScene, *Scene* *)
Delete the Scene* passed in.

**DEFINE_EVENT**(OpenedScene)
*SceneLoader* opened a new *Scene*.

**DEFINE_EVENT**(OSPrint, *String*)
A string was printed to output.

**DEFINE_EVENT**(UISystemEnableDebugger)
*UISystem* debugger on/off.

**DEFINE_EVENT**(UISystemDisableDebugger)
*UISystem* debugger on/off.

**DEFINE_EVENT**(WindowGetScreenState)
Get bool if window is fullscreen.

**DEFINE_EVENT**(WindowResized, *Vector2*)
*Window* has resized to size passed in.

**DEFINE_EVENT**(WindowToggleFullscreen)
Toggle the application window fullscreen mode.

**DEFINE_EVENT**(QuitWindowRequest)
*Window* has requested to quit.

**DEFINE_EVENT**(QuitEditorWindow)
Editor has requested to quit.

### Struct RotationKeyframe

- Defined in file_rootex_core_animation_animation.h

### Struct Documentation

**struct RotationKeyframe**

#### Public Members

float **m_Time**

*Quaternion* **m_Rotation**

### Struct ScalingKeyframe

- Defined in file_rootex_core_animation_animation.h

### Struct Documentation

**struct ScalingKeyframe**

#### Public Members

float **m_Time**

*Vector3* **m_Scaling**

### Struct SceneDock::SceneDockSettings

- Defined in file_editor_gui_scene_dock.h

### Nested Relationships

This struct is a nested type of *Class SceneDock*.

### Struct Documentation

**struct SceneDockSettings**

#### Public Members

bool **m_IsActive** = true

### Struct SceneSettings

- Defined in file_rootex_framework_scene.h

### Struct Documentation

**struct SceneSettings**

#### Public Functions

void **drawCameraSceneSelectables**(*Scene* *scene, *SceneID* &toSet)

void **drawListenerSceneSelectables**(*Scene* *scene, *SceneID* &toSet)

void **drawInputScheme**(*InputDescription* &floatInput)

void **draw**()

#### Public Members

*ResourceCollection* **preloads**

*SceneID* **camera** = ROOT_SCENE_ID

*SceneID* **listener** = ROOT_SCENE_ID

*HashMap*<*String*, *InputScheme*> **inputSchemes**

*String* **startScheme** = {}

### Struct SkeletonNode

- Defined in file_rootex_core_animation_animation.h

### Struct Documentation

**struct SkeletonNode**

Represents a node in the skeleton heirarchy(a tree like structure) of an animated model.

#### Public Members

*Vector*<*Ptr*<*SkeletonNode*>> **m_Children**

*String* **m_Name**

*Matrix* **m_LocalBindTransform**

### Struct SkyMaterialData

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Struct Documentation

**struct SkyMaterialData**

#### Public Members

*String* **skyImage**

### Struct SpotLight

- Defined in file_rootex_core_renderer_spot_light.h

### Struct Documentation

**struct SpotLight**

#### Public Members

float **attConst**
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **attLin**
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **attQuad**
attenuation = 1/(attConst + attLin * r + attQuad * r * r)

float **range**
Lighting effect clipped for distance > range.

float **diffuseIntensity**
Diffuse intensity of light.

*Color* **diffuseColor**
Diffuse color of light.

*Color* **ambientColor**
Ambient color of light.

float **spot**
Increasing spot increases the angular attenuation wrt axis.

float **angleRange**
Lighting effect clipped for angle > angleRange.

### Struct SpotLightInfo

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct SpotLightInfo**
Used to bind a spot light to the Pixel shader.

#### Public Members

*Color* **ambientColor** = {0.05f, , , }

*Color* **diffuseColor** = {1.0f, , , }

float **diffuseIntensity** = 0.0f

float **attConst** = 1.0f

float **attLin** = 0.045f

float **attQuad** = 0.0075f

*Vector3* **lightPos** = {0.0f, , }

float **range** = 0.0f

*Vector3* **direction**
Direction of axis of light cone.

float **spot**
Increasing spot increases the angular attenuation wrt axis.

float **angleRange**
Lighting effect clipped for angle > angleRange.

float **pad**[3]

### Struct StaticLightID

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct StaticLightID**

#### Public Members

int **id**

*Vector3* **pad**

### Struct StaticPointLightsInfo

- Defined in file_rootex_core_renderer_constant_buffer.h

### Struct Documentation

**struct StaticPointLightsInfo**

#### Public Members

*PointLightInfo* **pointLightInfos**[**MAX_STATIC_POINT_LIGHTS**]

### Struct TaskComplete

- Defined in file_rootex_os_thread.h

### Struct Documentation

**struct TaskComplete**
Contains the information of completed jobs.

#### Public Members

__int32 **m_Jobs**

*Vector*<__int32> **m_IDs**

### Struct TaskQueue

- Defined in file_rootex_os_thread.h

### Struct Documentation

**struct TaskQueue**
A queue of jobs.

#### Public Members

__int32 **m_Jobs**

**unsigned __int32 TaskQueue::m_Write**

**unsigned __int32 TaskQueue::m_Read**

*Vector*<*Ref*<*Task*>> **m_QueueJobs**

### Struct TaskReady

- Defined in file_rootex_os_thread.h

### Struct Documentation

**struct TaskReady**
Contains information of jobs ready for execution.

#### Public Members

__int32 **m_Jobs**

**unsigned __int32 TaskReady::m_Write**

**unsigned __int32 TaskReady::m_Read**

*Vector*<__int32> **m_IDs**

### Struct ToolbarDock::ToolbarDockSettings

- Defined in file_editor_gui_toolbar_dock.h

### Nested Relationships

This struct is a nested type of *Class ToolbarDock*.

### Struct Documentation

**struct ToolbarDockSettings**

#### Public Members

bool **m_IsActive** = true

bool **m_InEditorPlaying** = false

### Struct TransformAnimationComponent::Keyframe

- Defined in file_rootex_framework_components_space_transform_animation_component.h

### Nested Relationships

This struct is a nested type of *Class TransformAnimationComponent*.

### Struct Documentation

**struct Keyframe**

#### Public Members

float **timePosition**

*Matrix* **transform**

### Struct TransformComponent::TransformBuffer

- Defined in file_rootex_framework_components_space_transform_component.h

### Nested Relationships

This struct is a nested type of *Class TransformComponent*.

### Struct Documentation

**struct TransformBuffer**

#### Public Members

*Vector3* **position**

*Quaternion* **rotation**

*Vector3* **scale**

*BoundingBox* **boundingBox**

*Matrix* **transform**

### Struct TranslationKeyframe

- Defined in file_rootex_core_animation_animation.h

### Struct Documentation

**struct TranslationKeyframe**

#### Public Members

float **m_Time**

*Vector3* **m_Translation**

### Struct UIVertexData

- Defined in file_rootex_core_renderer_vertex_data.h

### Struct Documentation

**struct UIVertexData**

#### Public Members

*Vector2* **position**

char **color**[4]

*Vector2* **textureCoord**

### Struct VertexBufferElement

- Defined in file_rootex_core_renderer_buffer_format.h

### Struct Documentation

**struct VertexBufferElement**

Struct encapsulating the vertex buffer formats renderer currently supports.

#### Public Types

**enum Type**

Abstracts the DXGI input format types.

*Values:*

**FloatFloatFloatFloat** = DXGI_FORMAT_R32G32B32A32_FLOAT

**FloatFloatFloat** = DXGI_FORMAT_R32G32B32_FLOAT

**FloatFloat** = DXGI_FORMAT_R32G32_FLOAT

**ByteByteByteByte** = DXGI_FORMAT_R8G8B8A8_UNORM

**UInt** = DXGI_FORMAT_R32_UINT

**IntIntIntInt** = DXGI_FORMAT_R32G32B32A32_SINT

#### Public Members

*Type* **m_Type**

What type of objects are present in buffer.

LPCSTR **m_Name**

Used as the semantic of the Vertex Buffer element in shaders.

D3D11_INPUT_CLASSIFICATION **m_Class**

Per-what kind of object? (Changes for instance buffers mainly)

int **m_Slot**

Buffer slot to add this element to.

bool **m_ResetOffset**

Reset the offset counter. Used to starting a new buffer.

UINT **m_RendersPerInstance**

Number of instances to be rendered per instance.

#### Public Static Functions

**static** unsigned int **GetSize** (*Type type*)
Total size of the Vertex Buffer.

### Struct VertexData

- Defined in file_rootex_core_renderer_vertex_data.h

### Inheritance Relationships

### Derived Type

- `public AnimatedVertexData` (*Struct AnimatedVertexData*)

### Struct Documentation

**struct VertexData**
Data to be sent in a vertex.

Subclassed by *AnimatedVertexData*

#### Public Members

*Vector3* **position**

*Vector3* **normal**

*Vector2* **textureCoord**

*Vector3* **tangent**

### Struct ViewportDock::ViewportDockSettings

- Defined in file_editor_gui_viewport_dock.h

### Nested Relationships

This struct is a nested type of *Class ViewportDock*.

### Struct Documentation

**struct ViewportDockSettings**

#### Public Members

bool **m_IsActive** = true

bool **m_IsClosed**

float **m_AspectRatio**

### Struct WorkerParameters

- Defined in file_rootex_os_thread.h

### Struct Documentation

**struct WorkerParameters**
Worker thread parameters.

#### Public Members

__int32 **m_Thread**

*ThreadPool* ***m_ThreadPool**

### Class AnimatedBasicMaterialResourceFile

- Defined in file_rootex_core_resource_files_animated_basic_material_resource_file.h

### Inheritance Relationships

### Base Type

- `public BasicMaterialResourceFile` (*Class BasicMaterialResourceFile*)

### Class Documentation

**class AnimatedBasicMaterialResourceFile** : **public** *BasicMaterialResourceFile*
Representation of a skeletal animation basic material.

#### Public Functions

**AnimatedBasicMaterialResourceFile**(**const** *FilePath* &*path*)

**~AnimatedBasicMaterialResourceFile**()

void **uploadAnimationBuffer**(**const** *PerModelAnimationVSCBData* &*animationBuffer*)

**const** *Shader* ***getShader**() **const**

void **bindShader**()

void **bindVSCB**()

void **reimport**()
Reload the file buffer from disk.

#### Public Static Functions

**static** void **Load**()

**static** void **Destroy**()

### Class AnimatedModelComponent

- Defined in file_rootex_framework_components_visual_model_animated_model_component.h

### Inheritance Relationships

### Base Type

- `public RenderableComponent` (*Class RenderableComponent*)

### Class Documentation

**class AnimatedModelComponent** : **public** *RenderableComponent*
*Component* for skeletal animation models.

#### Public Types

**enum AnimationMode**
*Values:*

**None** = 0

**Looping** = 1

**Alternating** = 2

#### Public Functions

**AnimatedModelComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~AnimatedModelComponent**()

bool **preRender**(float *deltaMilliseconds*)

void **render**(float *viewDistance*)

*String* **getCurrentAnimationName**() **const**

float **getCurrentTime**() **const**

void **checkCurrentAnimationExists**()

void **update** (float *deltaMilliseconds*)

void **setPlaying** (bool *enabled*)

void **play** ()

void **stop** ()

void **setAnimation** (**const** *String* &*name*)

void **swapAnimation** (**const** *String* &*name*)

void **transition** (**const** *String* &*name*, float *transitionTime*)

void **swapTransition** (**const** *String* &*name*, float *transitionTime*)

void **setSpeedMultiplier** (float *speedMul*)

float **getStartTime** () **const**

float **getEndTime** () **const**

bool **isPlaying** () **const**

bool **hasEnded** () **const**

void **assignBoundingBox** ()

void **assignOverrides** (*Ref*<*AnimatedModelResourceFile*> *file*, **const** *HashMap*<*String*, *String*> &*materialOverrides*)

void **setAnimatedResourceFile** (*Ref*<*AnimatedModelResourceFile*> *file*, **const** *HashMap*<*String*, *String*> &*materialOverrides*)

*Ref*<*AnimatedModelResourceFile*> **getAnimatedResourceFile** () **const**

bool **setupData** ()
Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON** () **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw** ()
Expose the component data with ImGui.

### Protected Attributes

float **m_TimeDirection** = 1.0f

float **m_TransitionTime** = 1.0f

float **m_RemainingTransitionTime** = 0.0f

*Ref*<*AnimatedModelResourceFile*> **m_AnimatedModelResourceFile**

*String* **m_CurrentAnimationName**

float **m_CurrentTimePosition**

float **m_SpeedMultiplier**

*RootExclusion* **m_RootExclusion**

bool **m_IsPlaying**

bool **m_IsPlayOnStart**

*AnimationMode* **m_AnimationMode**

*Vector*<*Matrix*> **m_FinalTransforms**

### Class AnimatedModelResourceFile

- Defined in file_rootex_core_resource_files_animated_model_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class AnimatedModelResourceFile** : **public** *ResourceFile*

Representation of an animated 3D model file. Supports .dae files.

#### Public Functions

**AnimatedModelResourceFile**(*AnimatedModelResourceFile*&)

**AnimatedModelResourceFile**(*AnimatedModelResourceFile*&&)

**~AnimatedModelResourceFile**()

void **reimport**()

Reload the file buffer from disk.

*Vector*<*Pair*<*Ref*<*AnimatedBasicMaterialResourceFile*>, *Vector*<*Mesh*>>> &**getMeshes**()

*HashMap*<*String*, *SkeletalAnimation*> &**getAnimations**()

size_t **getBoneCount**() **const**

void **setNodeHierarchy**(aiNode **currentAiNode*, *Ptr*<*SkeletonNode*> &*currentNode*)

void **setAnimationTransforms**(*Ptr*<*SkeletonNode*> &*node*, float *currentTime*, **const** *String* &*animationName*, **const** *Matrix* &*parentModelTransform*, float *transitionTightness*, *RootExclusion* *rootExclusion*, bool *isRootFound*)

*Vector*<*String*> **getAnimationNames**()

float **getAnimationStartTime**(**const** *String* &*animationName*) **const**

float **getAnimationEndTime**(**const** *String* &*animationName*) **const**

void **getFinalTransforms**(*Vector*<*Matrix*> &*transforms*, **const** *String* &*animationName*, float *currentTime*, float *transitionTightness*, *RootExclusion* *rootExclusion*)

#### Public Static Functions

**static** *Matrix* **AiMatrixToMatrix**(**const** aiMatrix4x4 &*aiMatrix*)

### Class AnimationSystem

- Defined in file_rootex_framework_systems_animation_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class AnimationSystem** : **public** *System*

*System* that handles all skeletal animations.

#### Public Functions

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *AnimationSystem* ***GetSingleton**()

### Class Application

- Defined in file_rootex_app_application.h

### Inheritance Relationships

### Derived Types

- `public EditorApplication` (*Class EditorApplication*)
- `public GameApplication` (*Class GameApplication*)

## Class Documentation

**class Application**

Interface for a Rootex application. Every application that uses Rootex should derive this class.

Subclassed by *EditorApplication*, *GameApplication*

### Public Functions

**Application**(**const** *String* &*appTitle*, **const** *String* &*settingsFile*)

**Application**(*Application*&)

**virtual ~Application**()

void **run**()

**virtual** void **process**(float *deltaMilliseconds*)

void **end**()

void **createSaveSlot**(int *slot*)

bool **loadSave**(int *slot*)

JSON::json &**getSaveData**()

bool **saveSlot**()

**const** *String* &**getAppTitle**() **const**

**const** *Timer* &**getAppTimer**() **const**

*ThreadPool* &**getThreadPool**()

**const** *FrameTimer* &**getAppFrameTimer**() **const**

*Window* ***getWindow**()

*ApplicationSettings* ***getSettings**()

*Vector*<*FilePath*> **getLibrariesPaths**()

Returns paths of all third-party libraries provided by rootex/vendor/.

void **destroySplashWindow**()

float ***getDeltaMultiplierPtr**()

float **getDeltaMultiplier**() **const**

void **setDeltaMultiplier**(float *gain*)

void **resetDeltaMultiplier**()

### Public Static Functions

**static** *Application* ***GetSingleton**()

#### Protected Functions

*String* **getSaveSlotPath** (int *slot*)

#### Protected Attributes

*Timer* **m_ApplicationTimer**

*FrameTimer* **m_FrameTimer**

*ThreadPool* **m_ThreadPool**

float **m_DeltaMultiplier** = 1.0f

*String* **m_ApplicationTitle**

int **m_CurrentSaveSlot**

JSON::json **m_CurrentSaveData**

*Ptr*<*SplashWindow*> **m_SplashWindow**

*Ptr*<*Window*> **m_Window**

*Ptr*<*ApplicationSettings*> **m_ApplicationSettings**

### Class ApplicationSettings

- Defined in file_rootex_app_application_settings.h

### Class Documentation

**class ApplicationSettings**

#### Public Functions

**ApplicationSettings** (*Ref*<*TextResourceFile*> *settingsFile*)

**ApplicationSettings** (*ApplicationSettings*&)

**~ApplicationSettings** ()

JSON::json::iterator **find** (**const** *String* &*setting*)

JSON::json::iterator **end** ()

*TextResourceFile* ***getTextFile** () **const**

JSON::json &**getJSON** ()

#### Public Static Functions

**static** *ApplicationSettings* ***GetSingleton** ()

### Class AudioBuffer

- Defined in file_rootex_core_audio_audio_buffer.h

### Inheritance Relationships

### Derived Types

- `public StaticAudioBuffer` (*Class StaticAudioBuffer*)
- `public StreamingAudioBuffer` (*Class StreamingAudioBuffer*)

### Class Documentation

**class AudioBuffer**

Interface for an audio buffer that is used inside an *AudioSource*.

Subclassed by *StaticAudioBuffer*, *StreamingAudioBuffer*

#### Public Functions

**AudioBuffer**(*AudioBuffer*&)

**virtual ~AudioBuffer**()

*AudioResourceFile* ***getAudioFile**()

#### Protected Functions

**AudioBuffer**(*Ref*<*AudioResourceFile*> *audioFile*)

**virtual** void **initializeBuffers**() = 0

**virtual** void **destroyBuffers**() = 0

#### Protected Attributes

*Ref*<*AudioResourceFile*> **m_AudioFile**

### Class AudioComponent

- Defined in file_rootex_framework_components_audio_audio_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Derived Types

- `public MusicComponent` (*Class MusicComponent*)
- `public ShortMusicComponent` (*Class ShortMusicComponent*)

### Class Documentation

**class AudioComponent** : **public** *Component*

*Component* that plays audio according to the listener's position relative to the component.

Subclassed by *MusicComponent*, *ShortMusicComponent*

#### Public Functions

**AudioComponent** (*Entity* &*owner*, bool *playOnStart*, float *volume*, bool *isLooping*, bool *attenuation*, *AudioSource*::*AttenuationModel* *model*, *ALfloat* *rolloffFactor*, *ALfloat* *referenceDistance*, *ALfloat* *maxDistance*)

**virtual ~AudioComponent** ()

void **update** ()

bool **isPlayOnStart** () **const**

bool **isAttenuated** ()

void **setPlaying** (bool *enabled*)

void **play** ()

void **stop** ()

void **setLooping** (bool *enabled*)

bool **isLooping** ()

*AudioSource* ***getAudioSource** ()

*RigidBodyComponent* ***getCollider** ()

bool **setupData** ()
Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON** () **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw** ()
Expose the component data with ImGui.

#### Protected Functions

void **setAudioSource** (*AudioSource* **audioSource*)

#### Protected Attributes

bool **m_IsPlayOnStart**

bool **m_IsLooping**

*String* **m_AttenuationModelName** = "Linear"

### Class AudioListenerComponent

- Defined in file_rootex_framework_components_audio_audio_listener_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class AudioListenerComponent** : **public** *Component*

Listening component which is used to put sound output to. Useful for marking 3D sound attenuation in moving listeners

#### Public Functions

**AudioListenerComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~AudioListenerComponent**()

void **update**()

*Vector3* **getPosition**()

*Vector3* **getUp**()

*Vector3* **getAt**()

*RigidBodyComponent* ***getCollider**()

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **onRemove**()

Perform operations prior to detachment from owning entity and destruction.

void **draw**()

Expose the component data with ImGui.

### Class AudioPlayer

- Defined in file_editor_gui_audio_player.h

### Class Documentation

**class AudioPlayer**

#### Public Functions

**AudioPlayer**()

**AudioPlayer**(*AudioPlayer*&)

**~AudioPlayer**()

*Ref*<*ResourceFile*> **load**(**const** *FilePath* &*filePath*)

void **unload**()

void **draw**(float *deltaMilliseconds*)

### Class AudioResourceFile

- Defined in file_rootex_core_resource_files_audio_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class AudioResourceFile** : **public** *ResourceFile*

Representation of an audio file. Only .wav files are supported.

#### Public Functions

**AudioResourceFile**(*AudioResourceFile*&)

**AudioResourceFile**(*AudioResourceFile*&&)

**~AudioResourceFile**()

void **reimport**()

Reload the file buffer from disk.

**const** char ***getAudioData**() **const**

ALsizei **getAudioDataSize**() **const**

Get size of decompressed audio data.

ALenum **getFormat**() **const**

Returns the same enum value that OpenAL uses.

float **getFrequency**() **const**

int **getBitDepth**() **const**

int **getChannels**() **const**

float **getDuration**() **const**
Get the duration of the audio in seconds.

### Class AudioSource

- Defined in file_rootex_core_audio_audio_source.h

### Inheritance Relationships

### Derived Types

- `public StaticAudioSource` (*Class StaticAudioSource*)
- `public StreamingAudioSource` (*Class StreamingAudioSource*)

### Class Documentation

**class AudioSource**
An interface for an audio source in the game world.

Subclassed by *StaticAudioSource*, *StreamingAudioSource*

#### Public Types

**enum AttenuationModel**
Defines all attenuation models provided by OpenAL.

*Values:*

**Linear** = AL_LINEAR_DISTANCE

**Inverse** = AL_INVERSE_DISTANCE

**Exponential** = AL_EXPONENT_DISTANCE

**LinearClamped** = AL_LINEAR_DISTANCE_CLAMPED

**InverseClamped** = AL_INVERSE_DISTANCE_CLAMPED

**ExponentialClamped** = AL_EXPONENT_DISTANCE_CLAMPED

#### Public Functions

**virtual** void **setLooping**(bool *enabled*)

**virtual** void **queueNewBuffers**()
Queue new buffers to the audio card if possible.

void **play**()

void **pause**()

void **stop**()

bool **isPlaying**() **const**

bool **isPaused**() **const**

bool **isStopped**() **const**

**virtual** bool **isLooping**() **const**

*ALuint* **getSourceID**() **const**

**virtual** float **getDuration**() **const** = 0
Get audio duration in seconds.

void **setVelocity**(**const** *Vector3* &*velocity*)

void **setVolume**(float *volume*)

void **setPosition**(*Vector3* &*position*)

void **setModel**(*AttenuationModel distanceModel*)

void **setRollOffFactor**(*ALfloat rolloffFactor*)
Roll Off Factor: The rate of change of attenuation.

void **setReferenceDistance**(*ALfloat referenceDistance*)
Reference Distance: Distance until which clamping occurs.

void **setMaxDistance**(*ALfloat maxDistance*)

#### Protected Functions

**AudioSource**(bool *isStreaming*)

**virtual** **~AudioSource**()

#### Protected Attributes

*ALuint* **m_SourceID**

bool **m_IsStreaming**
RTTI for storing if the audio buffer is being streamed.

### Class AudioSystem

- Defined in file_rootex_framework_systems_audio_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

## Class Documentation

**class AudioSystem** : **public** *System*

Audio *System* responsible for streaming and static audio.

### Public Functions

*AudioListenerComponent* ***getListener**() **const**

void **setListener**(*AudioListenerComponent* **listenerComponent*)

void **restoreListener**()

bool **initialize**(**const** JSON::json &*systemData*)

void **setConfig**(**const** *SceneSettings* &*sceneSettings*)

void **shutDown**()

void **update**(float *deltaMilliseconds*)

void **begin**()

void **end**()

### Public Static Functions

**static** *AudioSystem* ***GetSingleton**()

**static** *String* **GetALErrorString**(int *errID*)

Returns error string corresponding to AL error codes.

**static** *String* **GetALCErrorString**(int *errID*)

Returns error string corresponding to ALC error codes.

**static** void **CheckALError**(**const** char **msg*, **const** char **fname*, int *line*)

Wrapper over alGetError function.

**static** void **CheckALCError**(**const** char **msg*, **const** char **fname*, int *line*)

Wrapper over alcGetError function.

**static** void **CheckALUTError**(**const** char **msg*, **const** char **fname*, int *line*)

Wrapper over alutGetError function.

## Class BaseComponentSet

- Defined in file_rootex_framework_ecs_factory.h

## Inheritance Relationships

## Derived Type

- `public ComponentSet< T >` (*Template Class ComponentSet*)

### Class Documentation

**class BaseComponentSet**
Subclassed by *ComponentSet< T >*

#### Public Functions

**virtual** bool **addComponent**(*Entity* &*owner*, **const** JSON::json &*componentData*, bool *checks* = true) = 0

**virtual** bool **addDefaultComponent**(*Entity* &*owner*, bool *checks*) = 0

**virtual** bool **removeComponent**(*Entity* &*entity*) = 0

**virtual const** *String* &**getName**() **const** = 0

**virtual const** *String* &**getCategory**() **const** = 0

**virtual const** ComponentID &**getID**() **const** = 0

### Class BasicMaterialResourceFile

- Defined in file_rootex_core_resource_files_basic_material_resource_file.h

### Inheritance Relationships

### Base Type

- `public MaterialResourceFile` (*Class MaterialResourceFile*)

### Derived Types

- `public AnimatedBasicMaterialResourceFile` (*Class AnimatedBasicMaterialResourceFile*)
- `public InstancingBasicMaterialResourceFile` (*Class InstancingBasicMaterialResourceFile*)

### Class Documentation

**class BasicMaterialResourceFile** : **public** *MaterialResourceFile*
Representation of a Basic material.

Subclassed by *AnimatedBasicMaterialResourceFile*, *InstancingBasicMaterialResourceFile*

#### Public Functions

**BasicMaterialResourceFile**(**const** *FilePath* &*path*)

**virtual ~BasicMaterialResourceFile**()

void **setColor**(**const** *Color* &*color*)

void **setDiffuse**(*Ref*<*ImageResourceFile*> *diffuse*)

void **setNormal**(*Ref*<*ImageResourceFile*> *normal*)

void **setSpecular**(*Ref*<*ImageResourceFile*> *spec*)

void **setLightmap**(*Ref*<*ImageResourceFile*> *lightmap*)

void **setAffectedBySky**(bool *status*)

void **setAffectedByLight**(bool *status*)

*Color* **getColor**()

*Ref*<*ImageResourceFile*> **getDiffuse**()

*Ref*<*ImageResourceFile*> **getNormal**()

*Ref*<*ImageResourceFile*> **getSpecular**()

*Ref*<*ImageResourceFile*> **getLightmap**()

**const** *Shader* ***getShader**() **const**

*Vector*<*Ref*<*GPUTexture*>> **getTextures**() **const**

void **bindShader**()

void **bindTextures**()

void **bindSamplers**()

void **bindVSCB**()

void **bindPSCB**()

JSON::json **getJSON**() **const**

ID3D11ShaderResourceView ***getPreview**() **const**

void **reimport**()
Reload the file buffer from disk.

bool **save**()

void **draw**()

### Public Static Functions

**static** void **Load**()

**static** void **Destroy**()

### Protected Functions

**BasicMaterialResourceFile**(**const** Type *type*, **const** *FilePath* &*path*)

#### Protected Attributes

Microsoft::WRL::ComPtr<ID3D11Buffer> **m_PSCB**

Microsoft::WRL::ComPtr<ID3D11Buffer> **m_VSCB**

### Class BoneAnimation

- Defined in file_rootex_core_animation_animation.h

### Class Documentation

**class BoneAnimation**
Stores the animation keyframes for a bone.

#### Public Functions

**BoneAnimation**()

**BoneAnimation**(**const** *BoneAnimation*&)

**~BoneAnimation**()

void **addTranslationKeyframe**(*TranslationKeyframe* &*keyframe*)

void **addRotationKeyframe**(*RotationKeyframe* &*keyframe*)

void **addScalingKeyframe**(*ScalingKeyframe* &*keyframe*)

*Matrix* **interpolate**(float *time*)
Lineary interpolates the two nearest keyframes to the current time.

### Class BoxColliderComponent

- Defined in file_rootex_framework_components_physics_box_collider_component.h

### Inheritance Relationships

### Base Type

- `public RigidBodyComponent` (*Class RigidBodyComponent*)

### Class Documentation

**class BoxColliderComponent** : **public** *RigidBodyComponent*
Collider component in the shape of a box.

#### Public Functions

**BoxColliderComponent** (*Entity* &*owner*, **const** JSON::json &*data*)

**~BoxColliderComponent** ()

void **setDimensions** (**const** *Vector3* &*dimensions*)

*Vector3* **getDimensions** () **const**

JSON::json **getJSON** () **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw** ()
Expose the component data with ImGui.

### Class BufferFormat

- Defined in file_rootex_core_renderer_buffer_format.h

### Class Documentation

**class BufferFormat**
Stores vector of Vertex Buffer Elements to be used as the input for Vertex Shaders.

#### Public Functions

**BufferFormat** ()

void **push** (*VertexBufferElement*::*Type* *type*, LPCSTR *name*, D3D11_INPUT_CLASSIFICATION *elementClass*, int *slot*, bool *resetOffset*, UINT *rendersPerInstance*)

**const** *Vector*<*VertexBufferElement*> &**getElements** () **const**

### Class CameraComponent

- Defined in file_rootex_framework_components_visual_camera_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class CameraComponent** : **public** *Component*

#### Public Functions

**CameraComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~CameraComponent**()

*Matrix* &**getViewMatrix**()

*Matrix* &**getProjectionMatrix**()

*Vector3* **getAbsolutePosition**()

*PostProcessingDetails* **getPostProcessingDetails**() **const**

bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

void **onRemove**()
Perform operations prior to detachment from owning entity and destruction.

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **addCustomPostProcessingDetails**(**const** *String* &*path*)

void **draw**()
Expose the component data with ImGui.

### Class CapsuleColliderComponent

- Defined in file_rootex_framework_components_physics_capsule_collider_component.h

#### Inheritance Relationships

#### Base Type

- `public RigidBodyComponent` (*Class RigidBodyComponent*)

#### Class Documentation

**class CapsuleColliderComponent** : **public** *RigidBodyComponent*
Collider component in the shape of a capsule.

#### Public Functions

**CapsuleColliderComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~CapsuleColliderComponent**()

float **getSideHeight**() **const**

void **setSideHeight**(float *s*)

float **getRadius**() **const**

void **setRadius**(float *r*)

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class CollisionComponent

- Defined in file_rootex_framework_components_physics_collision_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Derived Types

- `public RigidBodyComponent` (*Class RigidBodyComponent*)
- `public TriggerComponent` (*Class TriggerComponent*)

### Class Documentation

**class CollisionComponent** : **public** *Component*
Base class for collider components.

Subclassed by *RigidBodyComponent*, *TriggerComponent*

#### Public Functions

**CollisionComponent**(*Entity* &*owner*, int *collisionGroup*, int *collisionMask*)

**virtual ~CollisionComponent**()

**virtual** void **handleHit**(*Hit* **h*)

void **onRemove**()
Perform operations prior to detachment from owning entity and destruction.

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

void **displayCollisionLayers**(unsigned int &*collision*)

#### Protected Functions

void **detachCollisionObject**()

void **attachCollisionObject**()

#### Protected Attributes

*Ref*<btCollisionObject> **m_CollisionObject**

unsigned int **m_CollisionGroup**

unsigned int **m_CollisionMask**

### Class CollisionModelResourceFile

- Defined in file_rootex_core_resource_files_collision_model_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class CollisionModelResourceFile** : **public** *ResourceFile*
Representation of a 3D model file used to initialise a mesh collider.

#### Public Functions

**CollisionModelResourceFile**(**const** *CollisionModelResourceFile*&)

**CollisionModelResourceFile**(**const** *CollisionModelResourceFile*&&)

**~CollisionModelResourceFile**()

void **reimport**()
Reload the file buffer from disk.

btTriangleIndexVertexArray ***getCollisionMesh**()

### Class Component

- Defined in file_rootex_framework_component.h

### Nested Relationships

### Nested Types

- *Struct Component::Category*

### Inheritance Relationships

### Derived Types

- `public AudioComponent` (*Class AudioComponent*)
- `public AudioListenerComponent` (*Class AudioListenerComponent*)
- `public CameraComponent` (*Class CameraComponent*)
- `public CollisionComponent` (*Class CollisionComponent*)
- `public DirectionalLightComponent` (*Class DirectionalLightComponent*)
- `public FogComponent` (*Class FogComponent*)
- `public ParticleEffectComponent` (*Class ParticleEffectComponent*)
- `public PlayerController` (*Class PlayerController*)
- `public PointLightComponent` (*Class PointLightComponent*)
- `public RenderableComponent` (*Class RenderableComponent*)
- `public RenderUIComponent` (*Class RenderUIComponent*)
- `public SkyComponent` (*Class SkyComponent*)
- `public SpotLightComponent` (*Class SpotLightComponent*)
- `public TransformAnimationComponent` (*Class TransformAnimationComponent*)
- `public TransformComponent` (*Class TransformComponent*)
- `public UIComponent` (*Class UIComponent*)

### Class Documentation

**`class Component`**

An ECS style interface of a collection of data that helps implement a behaviour. Also allows operations on that data.

Subclassed by *AudioComponent*, *AudioListenerComponent*, *CameraComponent*, *CollisionComponent*, *DirectionalLightComponent*, *FogComponent*, *ParticleEffectComponent*, *PlayerController*, *PointLightComponent*, *RenderableComponent*, *RenderUIComponent*, *SkyComponent*, *SpotLightComponent*, *TransformAnimationComponent*, *TransformComponent*, *UIComponent*

#### Public Functions

**`Component`**(*Entity* &*owner*)

**`virtual ~Component`**()

void **registerDependency**(*Dependable* *dependable)
Only use to register dependency through a *Dependency* object.

**const** *Vector*<*Dependable* *> &**getDependencies**() **const**

bool **resolveDependencies**()
Establish inter-component links after all components have been added on the owner entity. Return true if successful.

**virtual** bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

**virtual** bool **setupEntities**()
Perform setting up operations which are possible only after all entities have been set up.

**virtual** void **onRemove**()
Perform operations prior to detachment from owning entity and destruction.

*Entity* &**getOwner**()

**virtual** ComponentID **getComponentID**() **const** = 0

**virtual const** char ***getName**() **const** = 0

**virtual** JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

**virtual** void **draw**()
Expose the component data with ImGui.

### Protected Attributes

*Entity* ***m_Owner**

**struct Category**

#### Public Static Attributes

**const** *String* **General** = "General"

**const** *String* **Audio** = "Audio"

**const** *String* **Game** = "Game"

**const** *String* **Physics** = "Physics"

**const** *String* **Model** = "Model"

**const** *String* **Effect** = "Effect"

**const** *String* **Light** = "Light"

**const** *String* **UI** = "UI"

## Template Class ComponentArray

- Defined in file_rootex_utility_component_array.h

### Class Documentation

template<typename **Component**, class **A** = std::allocator<*Component*>>
**class ComponentArray**

#### Public Functions

**ComponentArray**()

ComponentArrayIterator<*Component*> **begin**()

ComponentArrayIterator<*Component*> **end**()

void **push_back**(**const** *Component* &*item*)

void **emplace_back**(*Entity* &*owner*, **const** JSON::json &*componentData*)

bool **erase**(*Entity* &*entity*)

*Component* &**operator[]**(int *index*)

size_t **size**() **const**

bool **empty**() **const**

*Component* **front**()

*Component* **back**()

### Template Class ComponentArrayIterator

- Defined in file_rootex_utility_component_array_iterator.h

### Class Documentation

template<typename **DataType**>
**class ComponentArrayIterator**

#### Public Functions

**ComponentArrayIterator**(*Vector*<bool> &*isValid*, **typename** *Vector*<DataType>::iterator *itr*)

**ComponentArrayIterator**(**const** *ComponentArrayIterator*<DataType> &*rawIterator*)

**~ComponentArrayIterator**()

ComponentArrayIterator<DataType> &**operator=**(**const** ComponentArrayIterator<DataType> &*rawIterator*)

bool **operator==**(**const** ComponentArrayIterator<DataType> &*rawIterator*) **const**

bool **operator!=**(**const** ComponentArrayIterator<DataType> &*rawIterator*) **const**

ComponentArrayIterator<DataType> **operator++**()

DataType &**operator***()

**const** DataType &**operator***() **const**

#### Protected Attributes

*Vector*<DataType>::iterator **m_Itr**

int **m_Index**

*Vector*<bool> ***m_IsValid**

### Template Class ComponentSet

- Defined in file_rootex_framework_ecs_factory.h

### Inheritance Relationships

### Base Type

- `public BaseComponentSet` (*Class BaseComponentSet*)

### Class Documentation

template<class **T**>
**class ComponentSet** : **public** *BaseComponentSet*

#### Public Functions

**ComponentSet**()

**ComponentSet**(**const** *ComponentSet*&)

ComponentSet &**operator=**(**const** ComponentSet&)

ComponentArray<T> &**getAll**()

bool **addComponent**(*Entity* &*owner*, **const** JSON::json &*componentData*, bool *checks*)

bool **addDefaultComponent**(*Entity* &*owner*, bool *checks*)

bool **removeComponent**(*Entity* &*entity*)

**const** *String* &**getName**() **const**

**const** *String* &**getCategory**() **const**

**const** ComponentID &**getID**() **const**

### Class ContentBrowser

- Defined in file_editor_gui_content_browser.h

### Nested Relationships

### Nested Types

- *Struct ContentBrowser::ContentBrowserSettings*

### Class Documentation

**class ContentBrowser**

#### Public Functions

**ContentBrowser**()

**ContentBrowser**(**const** *ContentBrowser*&)

**~ContentBrowser**()

void **draw**(float *deltaMilliseconds*)

*ContentBrowserSettings* &**getSettings**()

void **setActive**(bool *enabled*)

#### Public Static Functions

**static** *ContentBrowser* ***GetSingleton**()

**struct ContentBrowserSettings**

##### Public Members

bool **m_IsActive** = true

## Class CPUParticlesComponent

- Defined in file_rootex_framework_components_visual_effect_cpu_particles_component.h

### Nested Relationships

### Nested Types

- *Struct CPUParticlesComponent::Particle*

### Inheritance Relationships

### Base Type

- `public ModelComponent` (*Class ModelComponent*)

### Class Documentation

**class CPUParticlesComponent** : **public** *ModelComponent*
Creates particle systems on the CPU. Not preferred over *ParticleEffectComponent*.

#### Public Functions

**CPUParticlesComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~CPUParticlesComponent**()

void **setMaterial**(*Ref*<*InstancingBasicMaterialResourceFile*> *particlesMaterial*)

void **emit**(**const** *ParticleTemplate* &*particleTemplate*)

void **expandPool**(**const** size_t &*poolSize*)

bool **preRender**(float *deltaMilliseconds*)

void **render**(float *viewDistance*)

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class CPUTexture

- Defined in file_rootex_core_renderer_texture.h

### Class Documentation

**class CPUTexture**
Texture accessible on CPU.

#### Public Functions

**CPUTexture**(unsigned char **pixelData*, int *width*, int *height*)
pixelData should contain values in RGBA format

**~CPUTexture**()

*Color* **getPixel**(int *x*, int *y*)

void **setPixel**(int *x*, int *y*, *Color color*)

unsigned int **getWidth**() **const**

unsigned int **getHeight**() **const**

**const** unsigned char ***getBuffer**() **const**

### Class CustomMaterialResourceFile

- Defined in file_rootex_core_resource_files_custom_material_resource_file.h

### Inheritance Relationships

### Base Type

- `public MaterialResourceFile` (*Class MaterialResourceFile*)

### Class Documentation

**class CustomMaterialResourceFile** : **public** *MaterialResourceFile*

Representation of a custom material.

#### Public Functions

**CustomMaterialResourceFile**(**const** *FilePath* &*path*)

**~CustomMaterialResourceFile**()

void **setShaders**(**const** *String* &*vertexShader*, **const** *String* &*pixelShader*)

void **setVS**(**const** *String* &*vertexShader*)

void **setPS**(**const** *String* &*pixelShader*)

void **recompileShaders**()

**const** *Shader* ***getShader**() **const**

*Vector*<*Ref*<*GPUTexture*>> **getTextures**() **const**

void **bindShader**()

void **bindTextures**()

void **bindSamplers**()

void **bindVSCB**()

void **bindPSCB**()

JSON::json **getJSON**() **const**

ID3D11ShaderResourceView ***getPreview**() **const**

void **reimport**()
Reload the file buffer from disk.

bool **save**()

void **draw**()

void **drawTextureSlots**(**const** char **label*, *Vector*<*Ref*<*ImageResourceFile*>> &*textures*)

float **getFloat**(int *index*)

*Vector3* **getFloat3**(int *index*)

*Color* **getColor**(int *index*)

bool **setFloat**(int *index*, float *value*)

bool **setFloat3**(int *index*, *Vector3* *value*)

bool **setColor**(int *index*, *Color* *value*)

#### Public Static Functions

**static** void **Load**()

**static** void **Destroy**()

#### Public Static Attributes

**const** *String* **s_DefaultCustomVSPath** = "rootex/core/renderer/shaders/custom_vertex_shader.hlsl"

**const** *String* **s_DefaultCustomPSPath** = "rootex/core/renderer/shaders/custom_pixel_shader.hlsl"

### Class CustomPostProcess

- Defined in file_rootex_core_renderer_post_processor.h

### Inheritance Relationships

### Base Type

- `public PostProcess` (*Class PostProcess*)

### Class Documentation

**class CustomPostProcess** : **public** *PostProcess*

#### Public Functions

**CustomPostProcess**(**const** *String* &*path*)

void **draw**(*CameraComponent* **camera*, ID3D11ShaderResourceView *&*nextSource*)
Performs the post processing step. The source for next post process step is altered when a post process step takes place.

#### Public Members

*String* **m_PostProcessPath**

### Class CustomRenderInterface

- Defined in file_rootex_core_ui_custom_render_interface.h

### Nested Relationships

### Nested Types

- *Struct CustomRenderInterface::GeometryData*

### Inheritance Relationships

### Base Type

- `public RenderInterface`

### Class Documentation

**class CustomRenderInterface** : **public** RenderInterface
Provides a render interface for RmlUi.

#### Public Functions

**CustomRenderInterface**(int *width*, int *height*)

**CustomRenderInterface**(**const** *CustomRenderInterface*&)

**virtual ~CustomRenderInterface**()

**virtual** void **RenderGeometry**(Rml::Vertex **vertices*, int *numVertices*, int **indices*, int *numIndices*, Rml::TextureHandle *texture*, **const** Rml::Vector2f &*translation*)

**virtual** Rml::CompiledGeometryHandle **CompileGeometry**(Rml::Vertex **vertices*, int *numVertices*, int **indices*, int *numIndices*, Rml::TextureHandle *texture*)

**virtual** void **RenderCompiledGeometry**(Rml::CompiledGeometryHandle *geometry*, **const** Rml::Vector2f &*translation*)

**virtual** void **ReleaseCompiledGeometry**(Rml::CompiledGeometryHandle *geometry*)

**virtual** bool **LoadTexture**(Rml::TextureHandle &*textureHandle*, Rml::Vector2i &*textureDimensions*, **const** *String* &*source*)

**virtual** bool **GenerateTexture**(Rml::TextureHandle &*textureHandle*, **const** byte **source*, **const** Rml::Vector2i &*sourceDimensions*)

**virtual** void **ReleaseTexture**(Rml::TextureHandle *texture*)

**virtual** void **EnableScissorRegion**(bool *enable*)

**virtual** void **SetScissorRegion**(int *x*, int *y*, int *width*, int *height*)

**virtual** void **SetTransform**(**const** Rml::Matrix4f **transform*)

### Class CustomSystemInterface

- Defined in file_rootex_framework_systems_ui_system.h

### Inheritance Relationships

### Base Type

- `public SystemInterface`

### Class Documentation

**class CustomSystemInterface** : **public** SystemInterface

### Class DebugDrawer

- Defined in file_rootex_core_physics_debug_drawer.h

### Inheritance Relationships

### Base Type

- `public btIDebugDraw`

### Class Documentation

**class DebugDrawer** : **public** btIDebugDraw
Provides an API for Bullet3D to draw debug information.

#### Public Functions

**DebugDrawer**()

**DebugDrawer**(*DebugDrawer*&)

**~DebugDrawer**()

**virtual** void **drawLine**(**const** btVector3 &*from*, **const** btVector3 &*to*, **const** btVector3 &*color*)

**virtual** void **drawContactPoint**(**const** btVector3 &*pointOnB*, **const** btVector3 &*normalOnB*, btScalar *distance*, int *lifeTime*, **const** btVector3 &*color*)

**virtual** void **reportErrorWarning**(**const** char **warningString*)

**virtual** void **draw3dText**(**const** btVector3 &*location*, **const** char **textString*)

**virtual** void **setDebugMode**(int *debugMode*)

**virtual** int **getDebugMode**() **const**

### Class DebugSystem

- Defined in file_game_systems_debug_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class DebugSystem** : **public** *System*

#### Public Functions

bool **initialize**(**const** JSON::json &*systemData*)

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *DebugSystem* ***GetSingleton**()

### Class DecalComponent

- Defined in file_rootex_framework_components_visual_effect_decal_component.h

### Inheritance Relationships

### Base Type

- `public ModelComponent` (*Class ModelComponent*)

### Class Documentation

**class DecalComponent** : **public** *ModelComponent*

Used to add decals (textures rendered on a surface like a sticker) to a object.

#### Public Functions

**DecalComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~DecalComponent**()

void **render**(float *viewDistance*)

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class DecalMaterialResourceFile

- Defined in file_rootex_core_resource_files_decal_material_resource_file.h

### Inheritance Relationships

### Base Type

- `public MaterialResourceFile` (*Class MaterialResourceFile*)

### Class Documentation

**class DecalMaterialResourceFile** : **public** *MaterialResourceFile*

Representation of a material used for drawing decals. Uses *ImageResourceFile* to store the Decal's texture.

#### Public Functions

**DecalMaterialResourceFile**(**const** *FilePath* &*path*)

**virtual ~DecalMaterialResourceFile**()

void **setColor**(**const** *Color* &*color*)

void **setDecal**(*Ref*<*ImageResourceFile*> *decal*)

**const** *Shader* ***getShader**() **const**

*Vector*<*Ref*<*GPUTexture*>> **getTextures**() **const**

void **bindShader**()

void **bindTextures**()

void **bindSamplers**()

void **bindVSCB**()

void **bindPSCB**()

JSON::json **getJSON**() **const**

ID3D11ShaderResourceView ***getPreview**() **const**

void **reimport**()
Reload the file buffer from disk.

bool **save**()

void **draw**()

### Public Static Functions

**static** void **Load**()

**static** void **Destroy**()

### Protected Functions

**DecalMaterialResourceFile**(**const** Type *type*, **const** *FilePath* &*path*)

### Protected Attributes

Microsoft::WRL::ComPtr<ID3D11Buffer> **m_PSCB**

Microsoft::WRL::ComPtr<ID3D11Buffer> **m_VSCB**

## Class Dependable

- Defined in file_rootex_framework_component.h

## Inheritance Relationships

## Derived Type

- `public Dependency< ComponentType, isSoft >` (*Template Class Dependency*)

### Class Documentation

**class Dependable**
Subclassed by *Dependency< ComponentType, isSoft >*

#### Public Functions

**virtual** ComponentID **getID**() **const** = 0

**virtual** bool **isValid**() **const** = 0

**virtual** void **setComponent**(*Component* *component) = 0

### Template Class Dependency

- Defined in file_rootex_framework_component.h

### Inheritance Relationships

### Base Type

- `public Dependable` (*Class Dependable*)

### Class Documentation

template<class **ComponentType**, bool **isSoft**>
**class Dependency** : **public** *Dependable*
Depend on a component either softly (recoverable from dependency breakage) or harshly (cannot recover from breakage).

#### Public Functions

**Dependency**(*Component* *dependedBy)

**~Dependency**()

ComponentType ***getComponent**()

**const** ComponentType ***getComponent**() **const**

void **setComponent**(*Component* *component)

ComponentID **getID**() **const**
Get the ID of the component which is depended upon. This should not use the component object because it is nullptr when this is called.

bool **isValid**() **const**
Return if the dependency has been evaluated properly.

### Class DirectionalLightComponent

- Defined in file_rootex_framework_components_visual_light_directional_light_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class DirectionalLightComponent** : **public** *Component*

*Component* to apply a dynamic directional light to the scene, only the first created instance is used in case of multiple such components.

#### Public Functions

**DirectionalLightComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~DirectionalLightComponent**()

*Vector3* **getDirection**()

**const** *DirectionalLight* &**getDirectionalLight**() **const**

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class DxgiDebugInterface

- Defined in file_rootex_core_renderer_dxgi_debug_interface.h

### Class Documentation

**class DxgiDebugInterface**

#### Public Functions

void **set**()

void **getMessages**(*String file*, *String func*) **const**

#### Public Static Functions

**static** *DxgiDebugInterface* ***GetSingleton**()

### Class EditorApplication

- Defined in file_editor_editor_application.h

### Inheritance Relationships

### Base Type

- `public Application` (*Class Application*)

### Class Documentation

**class EditorApplication** : **public** *Application*

#### Public Functions

**EditorApplication**()

**EditorApplication**(*EditorApplication*&)

**~EditorApplication**()

void **setGameMode**(bool *enabled*)

**virtual** void **process**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *EditorApplication* ***GetSingleton**()

### Class EditorSystem

- Defined in file_editor_editor_system.h

### Nested Relationships

### Nested Types

- *Struct EditorSystem::Icons*

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class EditorSystem** : **public** *System*

#### Public Functions

bool **initialize**(**const** JSON::json &*systemData*)

void **update**(float *deltaMilliseconds*)

void **pushRegularFont**()

void **pushBoldFont**()

void **pushItalicFont**()

void **pushMonospaceFont**()

void **popFont**()

void **openScene**(*String sceneName*)

ImColor **getWarningColor**() **const**

ImColor **getFatalColor**() **const**

ImColor **getSuccessColor**() **const**

ImColor **getNormalColor**() **const**

ImColor **getLinkColor**() **const**

#### Public Static Functions

**static** *EditorSystem* ***GetSingleton**()

### Class Entity

- Defined in file_rootex_framework_entity.h

### Class Documentation

**class Entity**

A collection of ECS style components that define an ECS style entity.

#### Public Functions

**Entity**(*Scene* **scene*)

**Entity**(*Entity*&)

**Entity**(*Entity*&&)

*Entity* &**operator=**(*Entity*&&)

**~Entity**()

bool **onAllComponentsAdded**()

bool **onAllEntitiesAdded**()

void **registerComponent**(*Component* **component*)

bool **addDefaultComponent**(**const** *String* &*componentName*)

bool **addComponent**(**const** *String* &*componentName*, **const** JSON::json &*componentData*)

bool **removeComponent**(ComponentID *toRemoveComponentID*, bool *hardRemove* = false)

bool **hasComponent**(ComponentID *componentID*)

void **clear**()
Remove all components.

void **destroy**()
Destruct all components.

*Scene* ***getScene**() **const**

template<class **ComponentType** = *Component*>
*ComponentType* ***getComponent**()

template<class **ComponentType** = *Component*>
*ComponentType* ***getComponentFromID**(ComponentID *ID*)

JSON::json **getJSON**() **const**

**const** *String* &**getName**() **const**

**const** *SceneID* **getID**() **const**

**const** *String* &**getFullName**() **const**

**const** *HashMap*<ComponentID, *Component* *> &**getAllComponents**() **const**

void **bind**(**const** *Event*::*Type* &*event*, **const** sol::function &*function*)

bool **call**(**const** *String* &*function*, **const** *Vector*<*Variant*> &*args*)

void **evaluateScriptOverrides**()

bool **setScript**(**const** *String* &*path*)

bool **setScriptJSON**(**const** JSON::json &*script*)

*Script* ***getScript**() **const**

void **draw**()

#### Protected Attributes

*Scene* ***m_Scene**

*HashMap*<ComponentID, *Component* *> **m_Components**

*Ref*<*Script*> **m_Script**

### Class Event

- Defined in file_rootex_core_event.h

### Class Documentation

**class Event**

An *Event* that is sent out by *EventManager*.

#### Public Types

**typedef** *String* **Type**

String defining the type of the event.

#### Public Functions

**Event**(**const** *Type* &*type*, **const** *Variant* &*data*)

**Event**(*Event*&)

**~Event**()

**const** *Type* &**getType**() **const**

**const** *Variant* &**getData**() **const**

Returns the payload data sent with an event. Extract typed data after getting the data.

### Template Class EventBinder

- Defined in file_rootex_core_event_manager.h

### Inheritance Relationships

### Base Type

- `public EventBinderBase` (*Class EventBinderBase*)

### Class Documentation

template<class **T**>
**class EventBinder** : **public** *EventBinderBase*

### Public Functions

**EventBinder**()

**~EventBinder**()

**void EventBinder::bind(const Event::Type & event, T * self, Variant (T::*)(const Ev**
Duplicate bindings will override the previous ones.

void **bind**(**const** *Event*::*Type* &*event*, EventFunction *function*)

void **unbind**(**const** *Event*::*Type* &*event*)

void **unbindAll**()

bool **hasBinding**(**const** *Event*::*Type* &*binding*) **const**

*Variant* **handle**(**const** *Event* &*event*)
Call only if binding exists.

## Class EventBinderBase

- Defined in file_rootex_core_event_manager.h

## Inheritance Relationships

## Derived Types

- public EventBinder< T > (*Template Class EventBinder*)
- public EventBinder< ContentBrowser > (*Template Class EventBinder*)
- public EventBinder< CustomPostProcess > (*Template Class EventBinder*)
- public EventBinder< CustomRenderInterface > (*Template Class EventBinder*)
- public EventBinder< EditorSystem > (*Template Class EventBinder*)
- public EventBinder< Entity > (*Template Class EventBinder*)
- public EventBinder< FileEditor > (*Template Class EventBinder*)
- public EventBinder< FileViewer > (*Template Class EventBinder*)
- public EventBinder< InputInterface > (*Template Class EventBinder*)
- public EventBinder< InputSystem > (*Template Class EventBinder*)
- public EventBinder< InspectorDock > (*Template Class EventBinder*)
- public EventBinder< LuaInterpreter > (*Template Class EventBinder*)
- public EventBinder< OutputDock > (*Template Class EventBinder*)
- public EventBinder< RenderingDevice > (*Template Class EventBinder*)
- public EventBinder< RenderSystem > (*Template Class EventBinder*)
- public EventBinder< SceneDock > (*Template Class EventBinder*)
- public EventBinder< SceneLoader > (*Template Class EventBinder*)

- `public EventBinder< ToolbarDock >` (*Template Class EventBinder*)
- `public EventBinder< UISystem >` (*Template Class EventBinder*)
- `public EventBinder< Window >` (*Template Class EventBinder*)

## Class Documentation

**class EventBinderBase**

Subclassed by *EventBinder< T >*, *EventBinder< ContentBrowser >*, *EventBinder< CustomPostProcess >*, *EventBinder< CustomRenderInterface >*, *EventBinder< EditorSystem >*, *EventBinder< Entity >*, *EventBinder< FileEditor >*, *EventBinder< FileViewer >*, *EventBinder< InputInterface >*, *EventBinder< InputSystem >*, *EventBinder< InspectorDock >*, *EventBinder< LuaInterpreter >*, *EventBinder< OutputDock >*, *EventBinder< RenderingDevice >*, *EventBinder< RenderSystem >*, *EventBinder< SceneDock >*, *EventBinder< SceneLoader >*, *EventBinder< ToolbarDock >*, *EventBinder< UISystem >*, *EventBinder< Window >*

### Public Functions

**virtual** bool **hasBinding**(**const** *Event*::*Type* &*binding*) **const** = 0

**virtual** *Variant* **handle**(**const** *Event* &*event*) = 0

Call only if binding exists.

## Class EventManager

- Defined in file_rootex_core_event_manager.h

## Class Documentation

**class EventManager**

An *Event* dispatcher and registrar that also allows looking up registered events.

### Public Functions

void **defer**(*Function*<void)

> *function*Defer a singular function till the end of the frame.

void **addBinder**(*EventBinderBase* **binder*)

Add an event binder which binds to several events per object. Does not need to be called externally.

void **removeBinder**(*EventBinderBase* **binder*)

*Variant* **returnCall**(**const** *Event* &*event*) **const**

Publish an event. Returns the result of the first event handled.

*Variant* **returnCall**(**const** *Event*::*Type* &*eventType*, **const** *Variant* &*data* = 0) **const**

void **call**(**const** *Event* &*event*) **const**

void **call**(**const** *Event*::*Type* &*eventType*, **const** *Variant* &*data* = 0) **const**

void **deferredCall**(*Ref*<*Event*> *event*)
Publish an event that gets evaluated the end of the current frame.

void **deferredCall**(**const** *Event*::*Type* &*eventType*, **const** *Variant* &*data* = 0)

void **dispatchDeferred**()
Dispatch deferred events collected so far.

**const** *HashMap*<*EventBinderBase* *, bool> &**getBinders**() **const**

#### Public Static Functions

**static** *EventManager* ***GetSingleton**()

### Class FileEditor

- Defined in file_editor_gui_file_editor.h

### Class Documentation

**class FileEditor**

#### Public Functions

**FileEditor**()

void **draw**(float *deltaMilliseconds*)

### Class FileViewer

- Defined in file_editor_gui_file_viewer.h

### Class Documentation

**class FileViewer**

#### Public Functions

**FileViewer**()

**FileViewer**(*FileViewer*&)

**~FileViewer**()

void **draw**(float *deltaMilliseconds*)

### Class FlipbookDecorator

- Defined in file_rootex_core_ui_flipbook_decorator.h

### Nested Relationships

### Nested Types

- *Struct FlipbookDecorator::FlipbookElementData*

### Inheritance Relationships

### Base Type

- `public RootexDecorator` (*Class RootexDecorator*)

### Class Documentation

**class FlipbookDecorator** : **public** *RootexDecorator*

UI element to show a simple image based animations using a sequence of sprites.

#### Public Functions

**FlipbookDecorator**()

**~FlipbookDecorator**()

bool **addFrame**(**const** Rml::Sprite **sprite*)

void **setFPS**(float *fps*)

void **update**(float *deltaSeconds*)

Rml::DecoratorDataHandle **GenerateElementData**(Rml::Element **element*) **const**

void **ReleaseElementData**(Rml::DecoratorDataHandle *elementData*) **const**

void **RenderElement**(Rml::Element **element*, Rml::DecoratorDataHandle *elementData*) **const**

### Class FlipbookDecoratorInstancer

- Defined in file_rootex_core_ui_flipbook_decorator.h

### Inheritance Relationships

### Base Type

- `public DecoratorInstancer`

### Class Documentation

**class FlipbookDecoratorInstancer** : **public** DecoratorInstancer

#### Public Functions

**FlipbookDecoratorInstancer**()

Rml::SharedPtr<Rml::Decorator> **InstanceDecorator**(**const** Rml::String &*name*, **const** Rml::PropertyDictionary &*properties*, **const** Rml::DecoratorInstancerInterface &*instancerInterface*)

### Class FogComponent

- Defined in file_rootex_framework_components_visual_effect_fog_component.h

#### Inheritance Relationships

#### Base Type

- public Component (*Class Component*)

#### Class Documentation

**class FogComponent** : **public** *Component*

Adds fog and obscures objects w.r.t the near and far values.

#### Public Functions

**FogComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**FogComponent**()

*Color* **getColor**() **const**

float **getNearDistance**() **const**

float **getFarDistance**() **const**

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class FontResourceFile

- Defined in file_rootex_core_resource_files_font_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class FontResourceFile** : **public** *ResourceFile*

Representation of a font file. Supports .spritefont files.

#### Public Functions

**FontResourceFile**(**const** *FontResourceFile*&)

**FontResourceFile**(**const** *FontResourceFile*&&)

**~FontResourceFile**()

void **reimport**()

Reload the file buffer from disk.

*Ref*<DirectX::SpriteFont> **getFont**() **const**

### Class FrameTimer

- Defined in file_rootex_os_timer.h

### Inheritance Relationships

### Base Type

- `public LoggingScopeTimer` (*Class LoggingScopeTimer*)

### Class Documentation

**class FrameTimer** : **public** *LoggingScopeTimer*

*Timer* that helps keep track of frame time.

#### Public Functions

**FrameTimer**()

**~FrameTimer**()

void **reset**()

Reset frame time to 0. Call at the beginning of the frame for accurate results.

void **showTime**()

void **showFPS**()

float **getFrameTime**() **const**
Call at the end of the frame for accurate results.

float **getLastFrameTime**() **const**

float **getLastFPS**() **const**

### Class GameApplication

- Defined in file_game_game_application.h

### Inheritance Relationships

### Base Type

- `public Application` (*Class Application*)

### Class Documentation

**class GameApplication** : **public** *Application*
*Application* that runs when game is run without the editor.

#### Public Functions

**GameApplication**()

**GameApplication**(*GameApplication*&)

**~GameApplication**()

### Class GameRenderSystem

- Defined in file_game_systems_game_render_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class GameRenderSystem** : **public** *System*

#### Public Functions

**virtual** bool **initialize**(**const** JSON::json &*systemData*)

**virtual** void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *GameRenderSystem* ***GetSingleton**()

### Class GPUTexture

- Defined in file_rootex_core_renderer_texture.h

### Class Documentation

**class GPUTexture**

Encapsulates all *GPUTexture* related functionalities, uses DirectXTK behind the scenes.

#### Public Functions

**GPUTexture**(**const** char **pixelData*, int *width*, int *height*)

**GPUTexture**(**const** char **imageFileData*, size_t *size*)

**GPUTexture**(**const** *GPUTexture*&)

*GPUTexture* &**operator=**(**const** *GPUTexture*&)

**~GPUTexture**()

ID3D11ShaderResourceView ***getTextureResourceView**() **const**

ID3D11Texture2D ***getD3D11Texture2D**() **const**

unsigned int **getWidth**() **const**

unsigned int **getHeight**() **const**

unsigned int **getMipLevels**() **const**

unsigned char ***download**()

### Class GridModelComponent

- Defined in file_rootex_framework_components_visual_model_grid_model_component.h

### Inheritance Relationships

### Base Type

- `public ModelComponent` (*Class ModelComponent*)

### Class Documentation

**class GridModelComponent** : **public** *ModelComponent*

Renders a grid uptil the view distance.

#### Public Functions

**GridModelComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~GridModelComponent**()

void **render**(float *viewDistance*)

bool **setupData**()

Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class ImageCubeResourceFile

- Defined in file_rootex_core_resource_files_image_cube_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class ImageCubeResourceFile** : **public** *ResourceFile*

Representation of an image file which is loaded into a cube texture. Supports only certain DDS formats. Use texassemble.exe (in DirectXTex) to assemble cube sides in a suitable format.

#### Public Functions

**ImageCubeResourceFile**(**const** *ImageCubeResourceFile*&)

**ImageCubeResourceFile**(**const** *ImageCubeResourceFile*&&)

**~ImageCubeResourceFile**()

void **reimport**()
Reload the file buffer from disk.

**const** *TextureCube* ***getTexture**() **const**

### Class ImageResourceFile

- Defined in file_rootex_core_resource_files_image_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class ImageResourceFile** : **public** *ResourceFile*
Representation of an image file in Direct3D supported texture format. Cannot be used for processing on CPU Supports BMP, JPEG, PNG, TIFF, GIF, HD Photo, or other WIC supported file containers

#### Public Functions

**ImageResourceFile**(**const** *ImageResourceFile*&)

**ImageResourceFile**(**const** *ImageResourceFile*&&)

**~ImageResourceFile**()

void **reimport**()
Reload the file buffer from disk.

**const** *Ref*<*GPUTexture*> **getGPUTexture**()

**const** *Ref*<*CPUTexture*> **getCPUTexture**()

unsigned int **getWidth**() **const**

unsigned int **getHeight**() **const**

void **setCPUAccess**(bool *cpuAccess*)

bool **isCPUAccess**()

void **uploadCPUTexturetoGPU**()

### Class ImageViewer

- Defined in file_editor_gui_image_viewer.h

### Class Documentation

**class ImageViewer**

#### Public Functions

*Ref*<*ResourceFile*> **load**(**const** *FilePath* &*filePath*)

void **unload**()

void **draw**(float *deltaMilliseconds*)

### Class IndexBuffer

- Defined in file_rootex_core_renderer_index_buffer.h

### Class Documentation

**class IndexBuffer**

Encapsulates Index Buffer data, to be supplied to the Input Assembler.

#### Public Functions

**IndexBuffer**(**const** *Vector*<unsigned short> &*indices*, bool *dynamicWrite* = false)

**IndexBuffer**(**const** *Vector*<unsigned int> &*indices*)

**IndexBuffer**(**const** int **indices*, size_t *size*)

**~IndexBuffer**()

void **bind**() **const**

unsigned int **getCount**() **const**

ID3D11Buffer ***getBuffer**()

#### Protected Attributes

Microsoft::WRL::ComPtr<ID3D11Buffer> **m_IndexBuffer**

unsigned int **m_Count**

DXGI_FORMAT **m_Format**

### Class InputInterface

- Defined in file_rootex_core_ui_input_interface.h

### Class Documentation

**class InputInterface**
Provides an interface to pass user input into RmlUi.

#### Public Functions

bool **initialise**()

void **processWindowsEvent**(UINT *message*, WPARAM *wParam*, LPARAM *lParam*)
Process the Windows message.

void **setContext**(Rml::Context **context*)

Rml::Character **getCharacterCode**(Rml::Input::KeyIdentifier *keyIdentifier*, int *keyModifier_state*)

*Variant* **windowResized**(**const** *Event* **event*)

#### Public Members

bool **m_IsMouseOver** = false

bool **m_IsEnabled** = true

float **m_ScaleX** = 1.0f

float **m_ScaleY** = 1.0f

int **m_Left** = 0

int **m_Right** = 0

int **m_Top** = 0

int **m_Bottom** = 0

#### Public Static Functions

**static** *InputInterface* ***GetSingleton**()

#### Protected Attributes

Rml::Context ***m_Context** = nullptr

### Class InputListener

- Defined in file_rootex_core_input_input_listener.h

### Inheritance Relationships

### Base Type

- `public MappedInputListener`

### Class Documentation

**class InputListener** : **public** MappedInputListener
Helper class to receive inputs from Gainput.

#### Public Functions

**InputListener**(*InputBoolListenerFunction* *boolListener*, *InputFloatListenerFunction* *floatListener*)

**InputListener**(*InputListener*&)

**~InputListener**()

**virtual** bool **OnUserButtonBool**(gainput::UserButtonId *userButton*, bool *oldValue*, bool *newValue*)

**virtual** bool **OnUserButtonFloat**(gainput::UserButtonId *userButton*, float *oldValue*, float *newValue*)

int **getID**() **const**

void **setID**(int *id*)

### Class InputManager

- Defined in file_rootex_core_input_input_manager.h

### Class Documentation

**class InputManager**
Allows interfacing to game controlling hardware, including mouse, keyboard and XInput controllers. Allows detecting inputs through *Event* dispatch. *Event* data for boolean buttons consists of a Vector2 where Vector2.x and Vector2.y carry the old and new values for the input event respectively. Float buttons should be queried directly by invoking *InputManager*.

#### Public Functions

void **initialize**(unsigned int *width*, unsigned int *height*)

void **setEnabled**(bool *enabled*)

void **loadSchemes**(**const** *HashMap*<*String*, *InputScheme*> &*inputSchemes*)

void **addScheme**(**const** *String* &*name*, **const** *InputScheme* &*inputScheme*)

void **pushScheme**(**const** *String* &*schemeName*)

void **popScheme**()

void **flushSchemes**()

void **mapBool**(**const** *Event*::*Type* &*action*, *Device* *device*, *DeviceButtonID* *button*)
Bind an event to a button on a device.

void **mapFloat**(**const** *Event*::*Type* &*action*, *Device* *device*, *DeviceButtonID* *button*)
Bind an event to a float on a device.

void **unmap**(**const** *Event*::*Type* &*action*)

bool **isPressed**(**const** *Event*::*Type* &*action*)

bool **hasPressed**(**const** *Event*::*Type* &*action*)

bool **wasPressed**(**const** *Event*::*Type* &*action*)

float **getFloat**(**const** *Event*::*Type* &*action*)

float **getFloatDelta**(**const** *Event*::*Type* &*action*)

*Vector2* **getMousePosition**()

void **update**()

void **setDisplaySize**(**const** *Vector2* &*newSize*)

*Array*<*String*, 23> **getMouseButtonNames**()

*Array*<*String*, 166> **getKeyboardButtonNames**()

*Array*<*String*, 20> **getPadButtonNames**()

**const** gainput::InputMap &**getMap**() **const**

gainput::InputDeviceMouse ***getMouse**()

gainput::InputDeviceKeyboard ***getKeyboard**()

gainput::InputDevicePad ***getPad1**()

gainput::InputDevicePad ***getPad2**()

### Public Static Functions

**static** *InputManager* ***GetSingleton**()

**static** void **SetEnabled**(bool *enabled*)

**static** void **MapBool**(**const** *Event*::*Type* &*action*, *Device* *device*, *DeviceButtonID* *button*)

**static** void **MapFloat**(**const** *Event*::*Type* &*action*, *Device* *device*, *DeviceButtonID* *button*)

**static** bool **IsPressed**(**const** *Event*::*Type* &*action*)

**static** bool **HasPressed**(**const** *Event*::*Type* &*action*)

**static** bool **WasPressed**(**const** *Event*::*Type* &*action*)

**static** float **GetFloat**(**const** *Event*::*Type* &*action*)

**static** float **GetFloatDelta**(**const** *Event*::*Type* &*action*)

**static** void **Unmap**(**const** *Event*::*Type* &*action*)

**static** *Vector2* **GetMousePosition**()

**static** *Array*<*String*, 23> **GetMouseButtonNames**()

**static** *Array*<*String*, 166> **GetKeyboardButtonNames**()

**static** *Array*<*String*, 20> **GetPadButtonNames**()

### Class InputSystem

- Defined in file_rootex_framework_systems_input_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class InputSystem** : **public** *System*

Input system responsible for registering and loading inputs.

#### Public Functions

void **loadSchemes**(**const** *HashMap*<*String*, *InputScheme*> &*schemes*)

void **addScheme**(**const** *String* &*name*, **const** *InputScheme* &*scheme*)

void **pushScheme**(**const** *String* &*name*)

void **popScheme**()

void **flushSchemes**()

bool **initialize**(**const** JSON::json &*systemData*)

void **setConfig**(**const** *SceneSettings* &*sceneSettings*)

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *InputSystem* ***GetSingleton**()

### Class InspectorDock

- Defined in file_editor_gui_inspector_dock.h

### Nested Relationships

### Nested Types

- *Struct InspectorDock::InspectorSettings*

### Class Documentation

**class InspectorDock**

#### Public Functions

*Variant* **closeScene**(**const** *Event* **event*)

**InspectorDock**()

**InspectorDock**(*InspectorDock*&)

**~InspectorDock**()

void **draw**(float *deltaMilliseconds*)

void **drawSceneActions**(*Scene* **scene*)

*Scene* ***getOpenedScene**()

*InspectorSettings* &**getSettings**()

void **setActive**(bool *enabled*)

#### Public Static Functions

**static** *InspectorDock* ***GetSingleton**()

**struct InspectorSettings**

##### Public Members

bool **m_IsActive** = true

### Class InstancingBasicMaterialResourceFile

- Defined in file_rootex_core_resource_files_instancing_basic_material_resource_file.h

### Inheritance Relationships

### Base Type

- `public BasicMaterialResourceFile` (*Class BasicMaterialResourceFile*)

### Class Documentation

**class InstancingBasicMaterialResourceFile** : **public** *BasicMaterialResourceFile*
Representation of a hardware instancing material.

#### Public Functions

**InstancingBasicMaterialResourceFile**(**const** *FilePath* &*path*)

**~InstancingBasicMaterialResourceFile**()

**const** *Shader* ***getShader**() **const**

void **bindShader**()

#### Public Static Functions

**static** void **Load**()

**static** void **Destroy**()

### Class LightSystem

- Defined in file_rootex_framework_systems_light_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class LightSystem** : **public** *System*
Interface for setting up point, directional and spot lights.

#### Public Functions

*StaticPointLightsInfo* **getStaticPointLights**()

*LightsInfo* **getDynamicLights**()

#### Public Static Functions

**static** *LightSystem* ***GetSingleton**()

### Class Locale

- Defined in file_rootex_core_language_locale.h

### Class Documentation

**class Locale**

Loads game strings from a .json file. The expected format for the .json file consisting of game strings is {“key1”:”game string 1” , “key2”:”game string 2”}

#### Public Functions

*String* **getString**(**const** *String key*)

Used to get game string of the current language by passing the key defined for the string in the .json file.

void **loadLanguage**(**const** *String location*)

Used to load game strings from the .json file present in the location passed.

#### Public Static Functions

**static** *Locale* ***GetSingleton**()

### Class LoggingScopeTimer

- Defined in file_rootex_os_timer.h

### Inheritance Relationships

### Base Type

- `public Timer` (*Class Timer*)

### Derived Type

- `public FrameTimer` (*Class FrameTimer*)

### Class Documentation

**class LoggingScopeTimer** : **public** *Timer*

Display a message with the time taken when the scope where this is instantiated ends.

Subclassed by *FrameTimer*

#### Public Functions

**LoggingScopeTimer**(**const** *String* &*msg*)

**LoggingScopeTimer**(*LoggingScopeTimer*&)

**virtual** **~LoggingScopeTimer**()

### Class LuaInterpreter

- Defined in file_rootex_script_interpreter.h

### Class Documentation

**class** **LuaInterpreter**

Lua interpreter that runs all Lua scripts inside the same Lua state. This means that all Lua code-snippets that are run can cross-reference each other.

#### Public Functions

sol::state &**getLuaState**()

#### Public Static Functions

**static** *LuaInterpreter* ***GetSingleton**()

### Class LuaTextResourceFile

- Defined in file_rootex_core_resource_files_lua_text_resource_file.h

### Inheritance Relationships

### Base Type

- `public TextResourceFile` (*Class TextResourceFile*)

### Class Documentation

**class** **LuaTextResourceFile** : **public** *TextResourceFile*

Representation of a text file that has Lua code.

#### Public Functions

**LuaTextResourceFile**(*TextResourceFile*&)

**LuaTextResourceFile**(*TextResourceFile*&&)

**~LuaTextResourceFile**()

### Class MaterialResourceFile

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Derived Types

- `public BasicMaterialResourceFile` (*Class BasicMaterialResourceFile*)
- `public CustomMaterialResourceFile` (*Class CustomMaterialResourceFile*)
- `public DecalMaterialResourceFile` (*Class DecalMaterialResourceFile*)
- `public SkyMaterialResourceFile` (*Class SkyMaterialResourceFile*)

### Class Documentation

**class MaterialResourceFile** : **public** *ResourceFile*

Representation of a material file. Every material type is inherited from this class.

Subclassed by *BasicMaterialResourceFile*, *CustomMaterialResourceFile*, *DecalMaterialResourceFile*, *SkyMaterialResourceFile*

#### Public Functions

bool **saveMaterialData**(**const** JSON::json &*j*)

**virtual const** *Shader* ***getShader**() **const** = 0

**virtual** *Vector*<*Ref*<*GPUTexture*>> **getTextures**() **const** = 0

**virtual** void **bindShader**() = 0

**virtual** void **bindTextures**() = 0

**virtual** void **bindSamplers**() = 0

**virtual** void **bindVSCB**() = 0

**virtual** void **bindPSCB**() = 0

**virtual** JSON::json **getJSON**() **const**

**virtual** ID3D11ShaderResourceView ***getPreview**() **const** = 0

void **readJSON**(**const** JSON::json &*j*)

void **setAlpha**(bool *enabled*)

bool **isAlpha**() **const**

void **draw**()

template<class **T**>
*T* ***as**()

#### Protected Functions

**MaterialResourceFile**(**const** Type &*type*, **const** *FilePath* &*path*)

### Class MaterialViewer

- Defined in file_editor_gui_material_viewer.h

### Class Documentation

**class MaterialViewer**

#### Public Functions

*Ref*<*ResourceFile*> **load**(**const** *FilePath* &*filePath*)

void **unload**()

void **draw**(float *deltaMilliseconds*)

### Class ModelComponent

- Defined in file_rootex_framework_components_visual_model_model_component.h

### Inheritance Relationships

### Base Type

- `public RenderableComponent` (*Class RenderableComponent*)

### Derived Types

- `public CPUParticlesComponent` (*Class CPUParticlesComponent*)
- `public DecalComponent` (*Class DecalComponent*)
- `public GridModelComponent` (*Class GridModelComponent*)
- `public SpriteComponent` (*Class SpriteComponent*)

## Class Documentation

**class ModelComponent** : **public** *RenderableComponent*

Used to render static 3D models.

Subclassed by *CPUParticlesComponent*, *DecalComponent*, *GridModelComponent*, *SpriteComponent*

### Public Functions

**ModelComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**virtual ~ModelComponent**()

bool **preRender**(float *deltaMilliseconds*)

void **render**(float *viewDistance*)

void **setModelResourceFile**(*Ref*<*ModelResourceFile*> *newModel*, **const** *HashMap*<*String*, *String*> &*materialOverrides*)

*ModelResourceFile* ***getModelResourceFile**() **const**

**const** *Vector*<*Pair*<*Ref*<*BasicMaterialResourceFile*>, *Vector*<*Mesh*>>> &**getMeshes**() **const**

bool **setupData**()

Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Protected Functions

void **assignBoundingBox**()

void **assignOverrides**(*Ref*<*ModelResourceFile*> *newModel*, **const** *HashMap*<*String*, *String*> &*materialOverrides*)

### Protected Attributes

*Ref*<*ModelResourceFile*> **m_ModelResourceFile**

## Class ModelResourceFile

- Defined in file_rootex_core_resource_files_model_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class ModelResourceFile** : **public** *ResourceFile*
Representation of a 3D model file.

#### Public Functions

**ModelResourceFile**(**const** *ModelResourceFile*&)

**ModelResourceFile**(**const** *ModelResourceFile*&&)

**~ModelResourceFile**()

void **reimport**()
Reload the file buffer from disk.

*Vector*<*Pair*<*Ref*<*BasicMaterialResourceFile*>, *Vector*<*Mesh*>>> &**getMeshes**()

int **getMaterialCount**()

*Ref*<*BasicMaterialResourceFile*> **getMaterialAt**(int *i*)

*Vector*<*Mesh*> &**getMeshesOfMaterialAt**(int *i*)

### Class MusicComponent

- Defined in file_rootex_framework_components_audio_music_component.h

### Inheritance Relationships

### Base Type

- `public AudioComponent` (*Class AudioComponent*)

### Class Documentation

**class MusicComponent** : **public** *AudioComponent*

### Public Functions

**MusicComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~MusicComponent**()

*AudioResourceFile* ***getAudioFile**() **const**

void **setAudioFile**(*Ref*<*AudioResourceFile*> *audioFile*)

bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

## Class OS

- Defined in file_rootex_os_os.h

## Class Documentation

**class OS**
Provides features that are provided directly by the *OS*.

### Public Functions

**~OS**()

### Public Static Functions

**static** bool **Initialize**()

**static** void **Execute**(**const** *String* &*string*)
Execute a command.

**static** void **RunApplication**(**const** *String* &*commandLine*)

**static** bool **ElevateThreadPriority**()

**static** int **GetCurrentThreadPriority**()

**static** *String* **GetBuildDate**()

**static** *String* **GetBuildTime**()

**static** *String* **GetBuildType**()

**static** *String* **GetGameExecutablePath**()

**static** *String* **GetOrganizationName** ()

**static** *String* **GetAppDataFolder** ()

**static** *String* **GetAbsoluteSaveGameFolder** (**const** *String* &*appName*)

**static** int **GetDisplayWidth** ()

**static** int **GetDisplayHeight** ()

**static** *Optional*<*String*> **SelectFile** (**const** char **filter*, **const** char **dir* = nullptr)

Open a dialog box which the user can select a file from. Selected filepath is returned if successful. Filter needs 2 '\0' characters at the end.

**static** *Optional*<*String*> **SaveSelectFile** (**const** char **filter*, **const** char **dir* = nullptr)

**static** void **OpenFileInSystemEditor** (**const** *String* &*filePath*)

**static** void **OpenFileInExplorer** (**const** *String* &*filePath*)

**static** void **EditFileInSystemEditor** (**const** *String* &*filePath*)

**static** *FileTimePoint* **GetFileLastChangedTime** (**const** *String* &*filePath*)

**static** bool **IsExistsAbsolute** (*String absPath*)

**static** bool **IsExists** (*String relativePath*)

**static** *FileBuffer* **LoadFileContents** (*String stringPath*)

**static** JSON::json **LoadFileContentsToJSONObject** (*String stringPath*)

**static** *FileBuffer* **LoadFileContentsAbsolute** (*String absPath*)

**static** *FilePath* **GetAbsolutePath** (*String stringPath*)

**static** *FilePath* **GetRootRelativePath** (*String stringPath*)

**static** *FilePath* **GetRelativePath** (*String stringPath*, *String base*)

**static** *String* **GetFileStem** (*String stringPath*)

**static** *FilePath* **GetParentPath** (*String stringPath*)

**static** *Vector*<*FilePath*> **GetAllFilesInDirectory** (**const** *String* &*directory*)

**static** *Vector*<*FilePath*> **GetAllInDirectory** (**const** *String* &*directory*)

**static** *Vector*<*FilePath*> **GetAllInDirectoryRoot** (**const** *String* &*directory*)

**static** *Vector*<*FilePath*> **GetDirectoriesInDirectory** (**const** *String* &*directory*)

**static** bool **DeleteDirectory** (**const** *String* &*dirPath*)

**static** bool **Rename** (**const** *String* &*sourcePath*, **const** *String* &*destinationPath*)

**static** *Vector*<*FilePath*> **GetFilesInDirectory** (**const** *String* &*directory*)

**static** bool **RelativeCopyFile** (**const** *String* &*src*, **const** *String* &*dest*)

**static** void **RelativeCopyDirectory** (**const** *String* &*src*, **const** *String* &*dest*)

**static** bool **IsDirectory**(**const** *String* &*path*)

**static** bool **IsFile**(**const** *String* &*path*)

**static** void **RegisterFileChangesWatcher**(**const** *String* &*path*, void (**callback*)) PVOID, BOOLEAN
, PVOID *param*

**static** void **RegisterDirectoryChangesWatcher**(**const** *String* &*path*, void (**callback*)) PVOID, BOOLEAN
, PVOID *param*

**static** bool **CreateDirectoryName**(**const** *String* &*dirPath*)

**static** bool **CreateDirectoryAbsoluteName**(**const** *String* &*dirPath*)

**static** *InputOutputFileStream* **CreateFileName**(**const** *String* &*filePath*)

**static** *InputOutputFileStream* **CreateFileNameAbsolute**(**const** *String* &*absFilePath*)

**static** bool **SaveFile**(**const** *FilePath* &*filePath*, **const** char **fileBuffer*, size_t *fileSize*)

**static** bool **SaveFileAbsolute**(**const** *FilePath* &*absFilePath*, **const** char **fileBuffer*, size_t *fileSize*)

**static** void **Print**(**const** *String* &*msg*, **const** *String* &*type* = "Print")

**static** void **PrintInline**(**const** *String* &*msg*, **const** *String* &*type* = "Print")

**static** void **Print**(**const** float &*real*)

**static** void **Print**(**const** int &*number*)

**static** void **Print**(**const** unsigned int &*number*)

**static** void **PrintLine**(**const** *String* &*msg*)

**static** void **PrintWarning**(**const** *String* &*warning*)

**static** void **PrintWarningInline**(**const** *String* &*warning*)

**static** void **PrintError**(**const** *String* &*error*)

**static** void **PrintErrorInline**(**const** *String* &*error*)

**static** void **PrintIf**(**const** bool &*expr*, **const** *String* &*error*)

**static** void **PrintSilent**(**const** *String* &*msg*)

**static** void **PrintInlineSilent**(**const** *String* &*msg*)

**static** void **PrintSilent**(**const** float &*real*)

**static** void **PrintSilent**(**const** int &*number*)

**static** void **PrintSilent**(**const** unsigned int &*number*)

**static** void **PrintLineSilent**(**const** *String* &*msg*)

**static** void **PrintWarningSilent**(**const** *String* &*warning*)

**static** void **PrintWarningInlineSilent**(**const** *String* &*warning*)

**static** void **PrintErrorSilent**(**const** *String* &*error*)

**static** void **PrintErrorInlineSilent**(**const** *String* &*error*)

**static** void **PrintIfSilent**(**const** bool &*expr*, **const** *String* &*error*)

**static** void **PostError**(*String* *message*, LPSTR *caption*)

#### Public Static Attributes

std::filesystem::file_time_type::clock **s_FileSystemClock**

**const** std::chrono::time_point<std::chrono::system_clock> **s_ApplicationStartTime**

*FilePath* **s_RootDirectory**

*FilePath* **s_GameDirectory**

*FilePath* **s_EngineDirectory**

### Class OutputDock

- Defined in file_editor_gui_output_dock.h

### Nested Relationships

### Nested Types

- *Struct OutputDock::OutputDockSettings*

### Class Documentation

**class OutputDock**

#### Public Functions

**OutputDock**()

**OutputDock**(*OutputDock*&)

**~OutputDock**()

void **draw**(float *deltaMilliseconds*)

*OutputDockSettings* &**getSettings**()

void **setActive**(bool *enabled*)

**struct OutputDockSettings**

##### Public Members

bool **m_IsActive** = true

### Class ParticleEffectComponent

- Defined in file_rootex_framework_components_visual_effect_particle_effect_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class ParticleEffectComponent** : **public** *Component*

Creates particle systems using the Effekseer library.

#### Public Functions

**ParticleEffectComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~ParticleEffectComponent**()

bool **isPlayOnStart**() **const**

bool **isMoving**() **const**

void **setMoving**(bool *enabled*)

bool **isPaused**() **const**

void **setPlaying**(bool *enabled*)

void **play**()

void **stop**()

*ParticleEffectResourceFile* ***getEffectResource**()

void **setEffect**(*Ref*<*ParticleEffectResourceFile*> *effect*)

Effekseer::Handle **getHandle**() **const**

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class ParticleEffectResourceFile

- Defined in file_rootex_core_resource_files_particle_effect_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Class Documentation

**class ParticleEffectResourceFile** : **public** *ResourceFile*
An Effekseer effect file used in particle effects.

#### Public Functions

**ParticleEffectResourceFile**(**const** *ParticleEffectResourceFile*&)

**ParticleEffectResourceFile**(**const** *ParticleEffectResourceFile*&&)

**~ParticleEffectResourceFile**()

Effekseer::Effect ***getEffect**() **const**

void **reimport**()
Reload the file buffer from disk.

### Class ParticleSystem

- Defined in file_rootex_framework_systems_particle_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class ParticleSystem** : **public** *System*
*System* for handling particle effects made using the Effekseer libary.

#### Public Functions

**~ParticleSystem**()

Effekseer::Handle **play**(Effekseer::Effect **effect*, **const** *Vector3* &*position*, int *startFrame*)

void **stop**(Effekseer::Handle *handle*)

void **setMatrix**(Effekseer::Handle *handle*, **const** *Matrix* &*mat*)

void **setSpeed**(Effekseer::Handle *handle*, **const** float &*speed*)

void **setTargetLocation**(Effekseer::Handle *handle*, **const** *Vector3* &*target*)

bool **getPaused**(Effekseer::Handle *handle*)

Effekseer::Effect ***loadEffect**(**const** *String* &*path*)

void **release**(Effekseer::Effect **effect*)

bool **initialize**(**const** JSON::json &*systemData*)

void **begin**()

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *ParticleSystem* ***GetSingleton**()

### Class PauseSystem

- Defined in file_rootex_framework_systems_pause_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class PauseSystem** : **public** *System*

*System* which handles the change in behavior of everything on pausing and playing.

#### Public Functions

bool &**getIsPausingEnabled**()

void **setIsPausingEnabled**(bool *pausing*)

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *PauseSystem* ***GetSingleton**()

### Class PhysicsSystem

- Defined in file_rootex_framework_systems_physics_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class PhysicsSystem** : **public** *System*

#### Public Functions

**virtual ~PhysicsSystem**()

bool **initialize**(**const** JSON::json &*systemData*)
Initialization and Maintenance of the Physics World.

void **addRigidBody**(btRigidBody **body*, int *group*, int *mask*)

void **removeRigidBody**(btRigidBody **rigidBody*)

void **addCollisionObject**(btCollisionObject **body*, int *group*, int *mask*)

void **removeCollisionObject**(btCollisionObject **collisionObject*)

**const** *PhysicsMaterialData* &**getMaterialData**(*PhysicsMaterial material*)

**const** char ***getMaterialNames**()

btCollisionWorld::AllHitsRayResultCallback **reportAllRayHits**(**const** btVector3 &*m_From*, **const** btVector3 &*m_To*)

btCollisionWorld::ClosestRayResultCallback **reportClosestRayHits**(**const** btVector3 &*m_From*, **const** btVector3 &*m_To*)

void **debugDrawComponent**(**const** btTransform &*worldTransform*, **const** btCollisionShape **shape*, **const** btVector3 &*color*)

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *PhysicsSystem* ***GetSingleton**()

**static** void **InternalTickCallback**(btDynamicsWorld ***const** *world*, btScalar **const** *timeStep*)
Callback from bullet for each physics time step.

### Class PlayerController

- Defined in file_rootex_framework_components_game_player_controller.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class PlayerController** : **public** *Component*

Defines a *PlayerController* that can be directly used to create a controllable player element.

#### Public Functions

**PlayerController**(*Entity* &*owner*, **const** JSON::json &*data*)

**~PlayerController**()

void **update**(float *deltaMilliseconds*)

bool **setupData**()

Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

void **drawAnimation**(**const** char **animation*, *String* &*editing*)

#### Public Members

*String* **m_WalkAnimation**

*String* **m_RunAnimation**

*String* **m_IdleAnimation**

*String* **m_TurnLeftAnimation**

*String* **m_TurnRightAnimation**

float **m_MaxWalkSpeed**

float **m_MaxRunSpeed**

float **m_StoppingPower**

float **m_IdleThreshold**

*Ref*<*StateManager*> **m_StateManager**

float **m_Acceleration**

*Vector3* **m_Velocity**

### Class PlayerSystem

- Defined in file_rootex_framework_systems_player_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class PlayerSystem** : **public** *System*
Player system to update the player controller.

#### Public Functions

bool **initialize**(**const** JSON::json &*systemData*)

void **setConfig**(**const** *SceneSettings* &*sceneSettings*)

void **begin**()

void **update**(float *deltaMilliseconds*)

void **end**()

#### Public Static Functions

**static** *PlayerSystem* ***GetSingleton**()

### Class PointLightComponent

- Defined in file_rootex_framework_components_visual_light_point_light_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Derived Type

- `public StaticPointLightComponent` (*Class StaticPointLightComponent*)

### Class Documentation

**class PointLightComponent** : **public** *Component*

*Component* to apply dynamic point lights to the scene.

Subclassed by *StaticPointLightComponent*

#### Public Functions

**PointLightComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~PointLightComponent**()

*Matrix* **getAbsoluteTransform**()

**const** *PointLight* &**getPointLight**() **const**

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class PostProcess

- Defined in file_rootex_core_renderer_post_processor.h

### Inheritance Relationships

### Derived Type

- `public CustomPostProcess` (*Class CustomPostProcess*)

### Class Documentation

**class PostProcess**

Subclassed by *CustomPostProcess*

#### Public Functions

**virtual ~PostProcess**()

**virtual** void **draw**(*CameraComponent* **camera*, ID3D11ShaderResourceView **&nextSource*) = 0

Performs the post processing step. The source for next post process step is altered when a post process step takes place.

### Class PostProcessor

- Defined in file_rootex_core_renderer_post_processor.h

### Class Documentation

**class PostProcessor**

#### Public Functions

**PostProcessor**()

**PostProcessor**(*PostProcessor*&)

**~PostProcessor**()

void **draw**(*CameraComponent* *camera)

#### Friends

**friend PostProcessor::PostProcessSystem**

### Class PostProcessSystem

- Defined in file_rootex_framework_systems_post_process_system.h

### Inheritance Relationships

### Base Type

- public System (*Class System*)

### Class Documentation

**class PostProcessSystem** : **public** *System*
Handles all the custom post processes.

#### Public Functions

void **update**(float *deltaMilliseconds*)

void **addCustomPostProcessing**(**const** *String* &*path*)

#### Public Static Functions

**static** *PostProcessSystem* ***GetSingleton**()

### Class Random

- Defined in file_rootex_core_random.h

### Class Documentation

**class Random**
A random number generator.

#### Public Static Functions

**static** float **Float**()
Returns a random float between 0.0f and 1.0f.

### Class RenderableComponent

- Defined in file_rootex_framework_components_visual_model_renderable_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Derived Types

- `public AnimatedModelComponent` (*Class AnimatedModelComponent*)
- `public ModelComponent` (*Class ModelComponent*)

### Class Documentation

**class RenderableComponent** : **public** *Component*
Base class for all components related to rendering.

Subclassed by *AnimatedModelComponent*, *ModelComponent*

#### Public Functions

**virtual ~RenderableComponent**()

void **setVisible**(bool *enabled*)

bool **isVisible**() **const**

**virtual** bool **preRender**(float *deltaMilliseconds*)

**virtual** void **render**(float *viewDistance*)

**virtual** void **postRender**()

**virtual** bool **addAffectingStaticLight**(*SceneID id*)

**virtual** void **removeAffectingStaticLight**(*SceneID id*)

**virtual** void **setMaterialOverride**(*MaterialResourceFile* **oldMaterial*, *Ref*<*MaterialResourceFile*> *newMaterial*)

*Ref*<*MaterialResourceFile*> **getMaterialOverride**(*MaterialResourceFile* **material*)

unsigned int **getRenderPass**() **const**

bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

bool **setupEntities**()
Perform setting up operations which are possible only after all entities have been set up.

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Protected Functions

**RenderableComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

float **getLODFactor**(float *viewDistance*)

### Protected Attributes

bool **m_IsVisible**

unsigned int **m_RenderPass**

bool **m_LODEnable**

float **m_LODBias**

float **m_LODDistance**

*HashMap*<*MaterialResourceFile* *, *Ref*<*MaterialResourceFile*>> **m_MaterialOverrides**

*Vector*<*SceneID*> **m_AffectingStaticLightIDs**

*Vector*<int> **m_AffectingStaticLights**

Microsoft::WRL::ComPtr<ID3D11Buffer> **m_PerModelCB**

## Class Renderer

- Defined in file_rootex_core_renderer_renderer.h

## Class Documentation

**class Renderer**
Makes the rendering draw call and set viewport, instrumental in seperating Game and HUD rendering.

### Public Functions

**Renderer**()

**Renderer**(**const** *Renderer*&)

*Renderer* &**operator=**(**const** *Renderer*&)

**virtual ~Renderer**()

void **setViewport**(*Viewport* &*viewport*)

void **resetCurrentShader**()

void **bind**(*MaterialResourceFile* **newMaterial*, *MaterialResourceFile* **oldMaterial*)

void **bind**(*MaterialResourceFile* **Material*)

void **draw**(**const** *VertexBuffer* **vertexBuffer*, **const** *IndexBuffer* **indexBuffer*) **const**

void **drawInstanced**(**const** *VertexBuffer* **vertexBuffer*, **const** *IndexBuffer* **indexBuffer*, **const** *VertexBuffer* **instanceBuffer*, unsigned int *instances*) **const**

## Class RenderingDevice

- Defined in file_rootex_core_renderer_rendering_device.h

## Class Documentation

**class RenderingDevice**

The boss of all rendering, all DirectX API calls requiring the Device or Context go through this.

### Public Types

**enum RasterizerState**

*Values:*

**Default**

**UI**

**UIScissor**

**Wireframe**

**Sky**

**enum SamplerState**

*Values:*

**Default**

**Anisotropic**

### Public Functions

void **initialize**(HWND *hWnd*, int *width*, int *height*)

void **createOffScreenViews**(int *width*, int *height*)

void **createSwapChainAndRTVs**(int *width*, int *height*, **const** HWND &*hWnd*)
Create resources which depend on window height and width.

void **setScreenState**(bool *fullscreen*)

ID3D11Device *__getDevice__()

ID3D11DeviceContext *__getContext__()

void **enableSkyDSS**()

void **disableSkyDSS**()

void **disableDSS**()

void **enableDSS**()

void **createRTVAndSRV**(Microsoft::WRL::ComPtr<ID3D11RenderTargetView> &*rtv*, Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> &*srv*)

Microsoft::WRL::ComPtr<ID3D11Buffer> **createBuffer**(**const** char **data*, size_t *size*, D3D11_BIND_FLAG *bindFlags*, D3D11_USAGE *usage*, int *cpuAccess*)

void **editBuffer**(**const** char **data*, size_t *byteSize*, ID3D11Buffer **bufferPointer*)

template<class **T**>
Microsoft::WRL::ComPtr<ID3D11Buffer> **createBuffer**(**const** *T* &*data*, D3D11_BIND_FLAG *bindFlags*, D3D11_USAGE *usage*, D3D11_CPU_ACCESS_FLAG *cpuAccess*)

template<typename **T**>
void **editBuffer**(**const** *T* &*data*, ID3D11Buffer **bufferPointer*)

Microsoft::WRL::ComPtr<ID3DBlob> **compileShader**(**const** *String* &*shaderPath*, **const** char **entryPoint*, **const** char **profile*)
To hold shader blobs loaded from the compiled shader files.

Microsoft::WRL::ComPtr<ID3D11PixelShader> **createPS**(ID3DBlob **blob*)

Microsoft::WRL::ComPtr<ID3D11VertexShader> **createVS**(ID3DBlob **blob*)

Microsoft::WRL::ComPtr<ID3D11InputLayout> **createVL**(ID3DBlob **vertexShaderBlob*, **const** D3D11_INPUT_ELEMENT_DESC **ied*, UINT *size*)

*Ref*<DirectX::SpriteFont> **createFont**(**const** *String* &*fontFilePath*)

Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> **createDDSTexture**(**const** char **imageDDSFileData*, size_t *size*)

Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> **createTexture**(**const** char **imageFileData*, size_t *size*)

Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> **createTextureFromPixels**(**const** char **imageRawData*, unsigned int *width*, unsigned int *height*)

unsigned char ***downloadTexture**(ID3D11Texture2D **texture*, unsigned int *width*, unsigned int *height*)

Microsoft::WRL::ComPtr<ID3D11SamplerState> **createSS**(*SamplerState type*)

void **setVSSRV**(unsigned int *slot*, unsigned int *count*, ID3D11ShaderResourceView ***texture*)

void **setPSSRV**(unsigned int *slot*, unsigned int *count*, ID3D11ShaderResourceView ***texture*)

void **setVSSS**(unsigned int *slot*, unsigned int *count*, ID3D11SamplerState ***samplerState*)

void **setPSSS**(unsigned int *slot*, unsigned int *count*, ID3D11SamplerState ***samplerState*)

void **setVSCB**(unsigned int *slot*, unsigned int *count*, ID3D11Buffer ***constantBuffer*)

void **setPSCB**(unsigned int *slot*, unsigned int *count*, ID3D11Buffer ***constantBuffer*)

void **bind**(ID3D11Buffer ***const** **vertexBuffer*, int *count*, **const** unsigned int **stride*, **const** unsigned int **offset*)

void **bind**(ID3D11Buffer **indexBuffer*, DXGI_FORMAT *format*)

void **bind**(ID3D11VertexShader **vertexShader*)

void **bind**(ID3D11PixelShader **pixelShader*)

void **bind**(ID3D11InputLayout **inputLayout*)

void **mapBuffer**(ID3D11Buffer **buffer*, D3D11_MAPPED_SUBRESOURCE &*subresource*)

void **unmapBuffer**(ID3D11Buffer **buffer*)

void **setDefaultBS**()

void **setAlphaBS**()

void **setCurrentRS**()

*RasterizerState* **getRSType**()

void **setRSType**(*RasterizerState rs*)

void **setTemporaryUIRS**()

void **setTemporaryUIScissoredRS**()

void **setDSS**()

void **setScissorRectangle**(int *x*, int *y*, int *width*, int *height*)

void **setResolutionAndRefreshRate**(int *width*, int *height*, int *refreshRateNum*, int *refreshRateDeno*)

void **setOffScreenRTVDSV**()

void **setOffScreenRTVOnly**()

void **setMainRT**()

void **setRTV**(Microsoft::WRL::ComPtr<ID3D11RenderTargetView> *rtv*)

void **setRTV**(ID3D11RenderTargetView **rtv*)

void **setInputLayout**(ID3D11InputLayout **inputLayout*)

void **unbindSRVs**()

void **unbindRTVs**()

void **unbindDepthSRV**()

Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> **getMainSRV**()

Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> **getDepthSSRV**()

Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> **getStencilSRV**()

Microsoft::WRL::ComPtr<ID3D11ShaderResourceView> **getOffScreenSRV**()

*Ref*<DirectX::SpriteBatch> **getUIBatch**()

void **setPrimitiveTopology**(D3D11_PRIMITIVE_TOPOLOGY *pt*)

void **setViewport**(**const** D3D11_VIEWPORT **vp*)

void **drawIndexed**(UINT *indices*)
The last boss, draws Triangles.

void **drawIndexedInstanced**(UINT *indices*, UINT *instances*, UINT *startInstance*)

void **beginDrawUI**()

void **endDrawUI**()

void **draw**(UINT *vertexCount*, UINT *startVertexLocation*)

void **clearRTV**(Microsoft::WRL::ComPtr<ID3D11RenderTargetView> *rtv*, float *r*, float *g*, float *b*, float *a*)

void **clearMainRT**(float *r*, float *g*, float *b*, float *a*)

void **clearOffScreenRT**(float *r*, float *g*, float *b*, float *a*)

void **clearDSV**()

### Public Static Functions

**static** *RenderingDevice* ***GetSingleton**()

### Class RenderSystem

- Defined in file_rootex_framework_systems_render_system.h

### Nested Relationships

### Nested Types

- *Struct RenderSystem::LineRequests*

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class RenderSystem** : **public** *System*

Responsible for handling all the rendering in the editor.

#### Public Functions

void **setConfig**(**const** *SceneSettings* &*sceneSettings*)

void **update**(float *deltaMilliseconds*)

void **renderLines**()

void **submitLine**(**const** *Vector3* &*from*, **const** *Vector3* &*to*)

void **submitBox**(**const** *Vector3* &*min*, **const** *Vector3* &*max*)

void **submitSphere**(**const** *Vector3* &*center*, **const** float &*radius*)

void **submitCone**(**const** *Matrix* &*transform*, **const** float &*height*, **const** float &*radius*)

void **recoverLostDevice**()

void **setCamera**(*CameraComponent* **camera*)

void **restoreCamera**()

void **enableWireframeRasterizer**()

void **resetDefaultRasterizer**()

void **setPerCameraVSCBs**()

void **setPerFrameVSCBs**(float *fogStart*, float *fogEnd*)

void **setPerCameraChangePSCBs**()

void **setPerFramePSCBs**(**const** *Color* &*fogColor*)

void **setPerScenePSCBs**()

void **updateStaticLights**()

void **updatePerSceneBinds**()

void **setIsEditorRenderPass**(bool *enabled*)

void **enableLineRenderMode**()

void **resetRenderMode**()

*CameraComponent* ***getCamera**() **const**

*Renderer* ***getRenderer**() **const**

void **draw**()

#### Public Static Functions

**static** *RenderSystem* ***GetSingleton**()

### Class RenderUIComponent

- Defined in file_rootex_framework_components_visual_ui_render_ui_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Derived Type

- `public TextUIComponent` (*Class TextUIComponent*)

### Class Documentation

**class RenderUIComponent** : **public** *Component*
Our UI base class used to render UI with *RenderingDevice*.

Subclassed by *TextUIComponent*

#### Public Functions

**virtual ~RenderUIComponent**()

bool **preRender**()

**virtual** void **render**() = 0

void **postRender**()

void **setIsVisible**(bool *enabled*)

bool **isVisible**() **const**

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

#### Protected Functions

**RenderUIComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

#### Protected Attributes

bool **m_IsVisible**

### Class RenderUISystem

- Defined in file_rootex_framework_systems_render_ui_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class RenderUISystem** : **public** *System*
Render system for the UI.

#### Public Functions

void **update**(float *deltaMilliseconds*)

void **pushUIMatrix**(**const** *Matrix* &*transform*)

void **popUIMatrix**()

*Matrix* **getTopUIMatrix**()

#### Public Static Functions

**static** *RenderUISystem* ***GetSingleton**()

### Class ResourceFile

- Defined in file_rootex_core_resource_file.h

### Inheritance Relationships

### Derived Types

- `public AnimatedModelResourceFile` (*Class AnimatedModelResourceFile*)
- `public AudioResourceFile` (*Class AudioResourceFile*)
- `public CollisionModelResourceFile` (*Class CollisionModelResourceFile*)
- `public FontResourceFile` (*Class FontResourceFile*)
- `public ImageCubeResourceFile` (*Class ImageCubeResourceFile*)
- `public ImageResourceFile` (*Class ImageResourceFile*)
- `public MaterialResourceFile` (*Class MaterialResourceFile*)
- `public ModelResourceFile` (*Class ModelResourceFile*)
- `public ParticleEffectResourceFile` (*Class ParticleEffectResourceFile*)
- `public TextResourceFile` (*Class TextResourceFile*)

### Class Documentation

**class ResourceFile**

Interface of a file loaded from disk. Use *ResourceLoader* to load, create or save files.

Subclassed by *AnimatedModelResourceFile*, *AudioResourceFile*, *CollisionModelResourceFile*, *FontResourceFile*, *ImageCubeResourceFile*, *ImageResourceFile*, *MaterialResourceFile*, *ModelResourceFile*, *ParticleEffectResourceFile*, *TextResourceFile*

#### Public Types

**enum Type**

RTTI storage for the type of file being represented.

*Values:*

**None** = 0

Signifies an error in loading. Every valid *ResourceFile* will have a non-None type.

**Lua** = 1

**Audio** = 2

**Text** = 3

**Model** = 4

**AnimatedModel** = 5

**CollisionModel** = 6

**Image** = 7

**ImageCube** = 8

**Font** = 9

**ParticleEffect** = 10

**BasicMaterial** = 11

**InstancingBasicMaterial** = 12

**AnimatedBasicMaterial** = 13

**SkyMaterial** = 14

**CustomMaterial** = 15

**DecalMaterial** = 16

### Public Functions

**ResourceFile**(**const** *ResourceFile*&)

**ResourceFile**(**const** *ResourceFile*&&)

**virtual ~ResourceFile**()

**virtual** void **reimport**()
Reload the file buffer from disk.

**virtual** bool **save**()

**virtual** void **draw**()

bool **isDirty**()
If the file has been changed on disk.

*FilePath* **getPath**() **const**

*Type* **getType**() **const**

**const** *FileTimePoint* &**getLastReadTime**() **const**

**const** *FileTimePoint* &**getLastChangedTime**()

### Public Members

*Type* **m_Type**

*FilePath* **m_Path**

*FileTimePoint* **m_LastReadTime**

*FileTimePoint* **m_LastChangedTime**

#### Public Static Attributes

**const Map<Type, const String> ResourceFile::s_TypeNames= { { Type::None, "None" }, { Ty**

#### Protected Functions

**ResourceFile**(**const** *Type* &*type*, **const** *FilePath* &*path*)

#### Friends

**friend ResourceFile::ResourceLoader**

### Class ResourceLoader

- Defined in file_rootex_core_resource_loader.h

### Class Documentation

**class ResourceLoader**

Factory for *ResourceFile* objects. Implements creating, loading and saving files. Maintains an internal cache that doesn't let the same file to be loaded twice. Cache misses force file loading.

This just means you can load the same file multiple times without worrying about unnecessary copies.

All path arguments should be relative to Rootex root.

The resource creation API is internally synchronised (threadsafe).

#### Public Static Functions

**static** void **Initialize**()

**static** void **Destroy**()

**static const** *HashMap*<*ResourceFile*::*Type*, *Vector*<*Weak*<*ResourceFile*>>> &**GetResources**()

**static const** char *__**GetCreatableExtension**(*ResourceFile*::*Type* *type*)

**static** void **SaveResources**(*ResourceFile*::*Type* *type*)

**static** void **ClearDeadResources**()

**static** *Ref*<*TextResourceFile*> **CreateTextResourceFile**(**const** *String* &*path*)

**static** *Ref*<*LuaTextResourceFile*> **CreateLuaTextResourceFile**(**const** *String* &*path*)

**static** *Ref*<*AudioResourceFile*> **CreateAudioResourceFile**(**const** *String* &*path*)

**static** *Ref*<*ModelResourceFile*> **CreateModelResourceFile**(**const** *String* &*path*)

**static** *Ref*<*CollisionModelResourceFile*> **CreateCollisionModelResourceFile**(**const** *String* &*path*)

**static** *Ref*<*AnimatedModelResourceFile*> **CreateAnimatedModelResourceFile**(**const** *String* &*path*)

**static** *Ref*<*ImageResourceFile*> **CreateImageResourceFile**(**const** *String* &*path*)

**static** *Ref*<*ImageCubeResourceFile*> **CreateImageCubeResourceFile**(**const** *String* &*path*)

**static** *Ref*<*FontResourceFile*> **CreateFontResourceFile**(**const** *String* &*path*)

**static** *Ref*<*ParticleEffectResourceFile*> **CreateParticleEffectResourceFile**(**const** *String* &*path*)

**static** *Ref*<*MaterialResourceFile*> **CreateMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*BasicMaterialResourceFile*> **CreateBasicMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*InstancingBasicMaterialResourceFile*> **CreateInstancingBasicMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*AnimatedBasicMaterialResourceFile*> **CreateAnimatedBasicMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*SkyMaterialResourceFile*> **CreateSkyMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*CustomMaterialResourceFile*> **CreateCustomMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*DecalMaterialResourceFile*> **CreateDecalMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*ResourceFile*> **CreateResourceFile**(**const** *ResourceFile*::*Type* &*type*, **const** *String* &*path*)

**static** *Ref*<*TextResourceFile*> **CreateNewTextResourceFile**(**const** *String* &*path*)

**static** *Ref*<*BasicMaterialResourceFile*> **CreateNewBasicMaterialResourceFile**(**const** *String* &*path*)

**static** *Ref*<*AnimatedBasicMaterialResourceFile*> **CreateNewAnimatedBasicMaterialResourceFile**(**const** *String* &*path*)

**static** int **Preload**(*ResourceCollection paths*, *Atomic*<int> &*progress*)
Load all the files passed in parallely. Return total tasks generated.

**static** void **Persist**(*Ref*<*ResourceFile*> *res*)
Add a resource to be kept alive till explicitly ordered to clear them. Internally synchronised.

**static** void **ClearPersistentResources**()

### Class RigidBodyComponent

- Defined in file_rootex_framework_components_physics_rigid_body_component.h

### Inheritance Relationships

### Base Types

- `public CollisionComponent` (*Class CollisionComponent*)
- `public btMotionState`

### Derived Types

- `public BoxColliderComponent` (*Class BoxColliderComponent*)
- `public CapsuleColliderComponent` (*Class CapsuleColliderComponent*)
- `public SphereColliderComponent` (*Class SphereColliderComponent*)
- `public StaticMeshColliderComponent` (*Class StaticMeshColliderComponent*)

### Class Documentation

**class RigidBodyComponent** : **public** *CollisionComponent*, **public** btMotionState

*Component* that reacts to and moves according to physics.

Subclassed by *BoxColliderComponent*, *CapsuleColliderComponent*, *SphereColliderComponent*, *StaticMeshColliderComponent*

#### Public Functions

**virtual ~RigidBodyComponent**()

void **applyForce**(**const** *Vector3* &*force*)

void **applyTorque**(**const** *Vector3* &*torque*)

*Vector3* **getAngularFactor**() **const**

void **setAngularFactor**(**const** *Vector3* &*factors*)

void **setAxisLock**(bool *enabled*)

*Vector3* **getOffset**() **const**

void **setOffset**(**const** *Vector3* &*offset*)

*Vector3* **getGravity**() **const**

void **setGravity**(**const** *Vector3* &*gravity*)

*PhysicsMaterial* **getMaterial**() **const**

*Vector3* **getVelocity**()

void **setVelocity**(**const** *Vector3* &*velocity*)

*Vector3* **getAngularVelocity**()

void **setAngularVelocity**(**const** *Vector3* &*angularVel*)

void **translate**(**const** *Vector3* &*vec*)

void **setTransform**(**const** *Matrix* &*mat*)

*Matrix* **getTransform**()

bool **isMoveable**()

void **setMoveable**(bool *enabled*)

bool **canSleep**()

void **setSleepable**(bool *enabled*)

bool **isCCD**()

void **setCCD**(bool *enabled*)

bool **isGeneratesHitEvents**()

void **setGeneratedHitEvents**(bool *enabled*)

bool **isKinematic**()

void **setKinematic**(bool *enabled*)

void **setupRigidBody**()

bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

void **highlight**()

### Protected Functions

**RigidBodyComponent**(*Entity* &*owner*, **const** *PhysicsMaterial* &*material*, float *volume*, **const** *Vector3* &*offset*, **const** *Vector3* &*gravity*, **const** *Vector3* &*angularFactor*, int *collisionGroup*, int *collisionMask*, bool *isMoveable*, bool *isKinematic*, bool *generatesHitEvents*, bool *canSleep*, bool *isCCD*, **const** *Ref*<btCollisionShape> &*collisionShape*)

void **getWorldTransform**(btTransform &*worldTrans*) **const**

void **setWorldTransform**(**const** btTransform &*worldTrans*)

void **updateTransform**()

void **handleHit**(*Hit* **hit*)

#### Protected Attributes

*Ref*<btCollisionShape> **m_CollisionShape**

btRigidBody ***m_Body** = nullptr

bool **m_IsGeneratesHitEvents**

btScalar **m_Mass**

*Vector3* **m_Gravity**

*Vector3* **m_AngularFactor**

*Vector3* **m_Offset**

float **m_Volume**

bool **m_IsMoveable**

bool **m_IsKinematic**

bool **m_IsSleepable**

bool **m_IsCCD**

*PhysicsMaterial* **m_Material**

btVector3 **m_LocalInertia**

### Class RootexDecorator

- Defined in file_rootex_core_ui_rootex_decorator.h

### Inheritance Relationships

### Base Type

- `public Decorator`

### Derived Type

- `public FlipbookDecorator` (*Class FlipbookDecorator*)

### Class Documentation

**class RootexDecorator** : **public** Decorator

Base class for other decorator elements like Flipbook Decorator.

Subclassed by *FlipbookDecorator*

### Public Functions

**RootexDecorator**()

**virtual ~RootexDecorator**()

**virtual** void **update**(float *deltaSeconds*) = 0

### Public Static Functions

**static** void **UpdateAll**(float *deltaSeconds*)

## Class Scene

- Defined in file_rootex_framework_scene.h

## Class Documentation

**class Scene**

### Public Types

**enum ImportStyle**

*Values:*

**Local**
If scene is not imported but created raw inside this scene.

**External**
If scene is linked to another scene file.

### Public Functions

**Scene**(*SceneID id*, **const** *String &name*, **const** *SceneSettings &settings*, *ImportStyle importStyle*, **const** *String &sceneFile*)

**~Scene**()

*Scene* ***findScene**(*SceneID scene*)

void **reimport**()

void **onLoad**()

bool **snatchChild**(*Scene *child*)

bool **addChild**(*Ptr*<*Scene*> *&child*)

bool **removeChild**(*Scene *toRemove*)

void **setName**(**const** *String &name*)

JSON::json **getJSON**() **const**

bool &**getIsScenePaused**()

void **setIsScenePaused**(bool *pause*)

*Vector*<*Ptr*<*Scene*>> &**getChildren**()

*SceneID* **getID**() **const**

*ImportStyle* **getImportStyle**() **const**

*String* **getScenePath**() **const**

*Scene* ***getParent**() **const**

*Entity* &**getEntity**()

**const** *String* &**getName**() **const**

**const** *String* &**getFullName**() **const**

void **setFullName**(*String* &*name*)

*SceneSettings* &**getSettings**()

### Public Static Functions

**static** void **ResetNextID**()

**static** *Ptr*<*Scene*> **Create**(**const** JSON::json &*sceneData*, **const** bool *assignNewIDs*)

**static** *Ptr*<*Scene*> **CreateFromFile**(**const** *String* &*sceneFile*)

**static** *Ptr*<*Scene*> **CreateEmpty**()

**static** *Ptr*<*Scene*> **CreateEmptyAtPath**(**const** *String* &*sceneFile*)

**static** *Ptr*<*Scene*> **CreateRootScene**()

**static** bool **isReservedName**(**const** *String* &*sceneName*)

**static** *Vector*<*Scene* *> **FindScenesByName**(**const** *String* &*name*)

**static** *Scene* ***FindSceneByID**(**const** *SceneID* &*id*)

**static** **const** *Vector*<*Scene* *> &**FindAllScenes**()

## Class SceneDock

- Defined in file_editor_gui_scene_dock.h

## Nested Relationships

## Nested Types

- *Struct SceneDock::SceneDockSettings*

### Class Documentation

**class SceneDock**

#### Public Functions

**SceneDock**()

**SceneDock**(**const** *SceneDock*&)

**~SceneDock**()

void **draw**(float *deltaMilliseconds*)

void **showEntities**(*Scene* **scene*)

*SceneDockSettings* &**getSettings**()

void **setActive**(bool *enabled*)

**struct SceneDockSettings**

##### Public Members

bool **m_IsActive** = true

### Class SceneLoader

- Defined in file_rootex_framework_scene_loader.h

### Class Documentation

**class SceneLoader**

Loads, saves and destroys scenes.

#### Public Functions

int **preloadScene**(**const** *String* &*sceneFile*, *Atomic*<int> &*progress*)

void **loadPreloadedScene**(**const** *String* &*sceneFile*, **const** *Vector*<*String*> &*arguments*)

void **loadScene**(**const** *String* &*sceneFile*, **const** *Vector*<*String*> &*arguments*)

bool **saveScene**(*Scene* **scene*)

bool **saveSceneAtFile**(*Scene* **scene*, **const** *String* &*filePath*)

void **destroyAllScenes**()

*Scene* ***getCurrentScene**() **const**

*Scene* ***getRootScene**() **const**

*Ptr*<*Scene*> &**getRootSceneEx**()

*Vector*<*String*> **getArguments**()

#### Public Static Functions

**static** *SceneLoader* ***GetSingleton**()

### Class Script

- Defined in file_rootex_script_script.h

### Class Documentation

**class Script**

#### Public Functions

**Script**(**const** JSON::json &*script*)

**Script**(**const** *Script*&)

**~Script**()

bool **setup**(*Entity* **entity*)

void **evaluateOverrides**()

bool **call**(**const** *String* &*function*, **const** *Vector*<*Variant*> &*args*)

JSON::json **getJSON**() **const**

**const** *String* &**getFilePath**()

sol::table &**getScriptInstance**()

void **draw**()

### Class ScriptSystem

- Defined in file_rootex_framework_systems_script_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class ScriptSystem** : **public** *System*

Interface for initialisation, maintenance and deletion of script components.

#### Public Functions

void **addInitScriptEntity**(*Entity* *e)

void **removeInitScriptEntity**(*Entity* *e)

void **addEnterScriptEntity**(*Entity* *e)

void **removeEnterScriptEntity**(*Entity* *e)

void **update**(float *deltaMilliseconds*)

Calls *update()* function of script components.

void **end**()

Calls *end()* function of script components.

#### Public Static Functions

**static** *ScriptSystem* ***GetSingleton**()

### Class Shader

- Defined in file_rootex_core_renderer_shader.h

### Class Documentation

**class Shader**

#### Public Functions

**Shader**()

**Shader**(**const** *String* &*vertexPath*, **const** *String* &*pixelPath*, **const** *BufferFormat* &*vertexBufferFormat*)

**~Shader**()

void **bind**() **const**

bool **isValid**() **const**

#### Public Members

Microsoft::WRL::ComPtr<ID3D11VertexShader> **m_VertexShader**

Microsoft::WRL::ComPtr<ID3D11PixelShader> **m_PixelShader**

Microsoft::WRL::ComPtr<ID3D11InputLayout> **m_InputLayout**

bool **m_IsValid** = true

### Class ShortMusicComponent

- Defined in file_rootex_framework_components_audio_short_music_component.h

### Inheritance Relationships

### Base Type

- `public AudioComponent` (*Class AudioComponent*)

### Class Documentation

**class ShortMusicComponent** : **public** *AudioComponent*

#### Public Functions

**ShortMusicComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~ShortMusicComponent**()

*AudioResourceFile* ***getAudioFile**() **const**

void **setAudioFile**(*Ref*<*AudioResourceFile*> *audioFile*)

bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class SkeletalAnimation

- Defined in file_rootex_core_animation_animation.h

### Class Documentation

**class SkeletalAnimation**
Represents a single animation of a model with its bone animation data.

#### Public Functions

**SkeletalAnimation**()

**SkeletalAnimation**(**const** *SkeletalAnimation*&)

**~SkeletalAnimation**()

*Matrix* **interpolate**(**const** *String* &*nodeName*, float *currentTime*)
Calls the *interpolate()* function of each *BoneAnimation*.

float **getStartTime**() **const**

float **getEndTime**() **const**

void **setDuration**(float *time*)

void **addBoneAnimation**(*String boneName*, *BoneAnimation* &*boneAnimation*)

### Class SkyComponent

- Defined in file_rootex_framework_components_visual_effect_sky_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class SkyComponent** : **public** *Component*
Used to add a skybox to the scene.

#### Public Functions

**SkyComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~SkyComponent**()

*ModelResourceFile* ***getSkySphere**() **const**

*SkyMaterialResourceFile* ***getSkyMaterial**() **const**

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class SkyMaterialResourceFile

- Defined in file_rootex_core_resource_files_sky_material_resource_file.h

### Inheritance Relationships

### Base Type

- `public MaterialResourceFile` (*Class MaterialResourceFile*)

### Class Documentation

**class SkyMaterialResourceFile** : **public** *MaterialResourceFile*
Representation of a sky material.

#### Public Functions

**SkyMaterialResourceFile**(**const** *FilePath* &*path*)

**~SkyMaterialResourceFile**()

void **setSky**(*Ref*<*ImageCubeResourceFile*> *sky*)

**const** *Shader* ***getShader**() **const**

*Vector*<*Ref*<*GPUTexture*>> **getTextures**() **const**

void **bindShader**()

void **bindTextures**()

void **bindSamplers**()

void **bindVSCB**()

void **bindPSCB**()

JSON::json **getJSON**() **const**

ID3D11ShaderResourceView ***getPreview**() **const**

void **reimport**()
Reload the file buffer from disk.

bool **save**()

void **draw**()

#### Public Static Functions

**static** void **Load**()

**static** void **Destroy**()

### Class SphereColliderComponent

- Defined in file_rootex_framework_components_physics_sphere_collider_component.h

### Inheritance Relationships

### Base Type

- `public RigidBodyComponent` (*Class RigidBodyComponent*)

### Class Documentation

**class SphereColliderComponent** : **public** *RigidBodyComponent*
Collider component in the shape of a sphere.

#### Public Functions

**SphereColliderComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~SphereColliderComponent**()

float **getRadius**() **const**

void **setRadius**(float *r*)

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class SplashWindow

- Defined in file_rootex_main_splash_window.h

### Class Documentation

**class SplashWindow**

#### Public Functions

**SplashWindow**(**const** *String* &*title*, **const** *String* &*icon*, **const** *String* &*image*, int *width*, int *height*)

**SplashWindow**(**const** *SplashWindow*&)

**~SplashWindow**()

### Class SpotLightComponent

- Defined in file_rootex_framework_components_visual_light_spot_light_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class SpotLightComponent** : **public** *Component*

*Component* to apply dynamic spot lights to the scene.

#### Public Functions

**SpotLightComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~SpotLightComponent**()

*Matrix* **getAbsoluteTransform**()

**const** *SpotLight* &**getSpotLight**() **const**

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class SpriteComponent

- Defined in file_rootex_framework_components_visual_model_sprite_component.h

### Inheritance Relationships

### Base Type

- `public ModelComponent` (*Class ModelComponent*)

### Class Documentation

**class SpriteComponent** : **public** *ModelComponent*
*Component* for rendering sprites. Supports billboarding.

#### Public Functions

**SpriteComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~SpriteComponent**()

*Ref*<*MaterialResourceFile*> **getOverridingMaterialResourceFile**()

bool **preRender**(float *deltaMilliseconds*)

void **postRender**()

bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class State

- Defined in file_rootex_core_state_manager.h

### Class Documentation

**class State**

#### Public Functions

**virtual ~State**()

**virtual** void **update**(*StateManager* &*manager*, float *deltaMilliseconds*)

**virtual** void **enter**(*StateManager* &*manager*)

**virtual** *Ptr*<*State*> **exit**(*StateManager* &*manager*, float *deltaMilliseconds*) = 0

### Class StateManager

- Defined in file_rootex_core_state_manager.h

### Class Documentation

**class StateManager**

A state-machine-like class used to mantain and update states in classes like the *PlayerController*.

#### Public Functions

**virtual** void **update**(float *deltaMilliseconds*)

void **transition**(*Ptr*<*State*> *newState*)

#### Protected Functions

**StateManager**(*Ptr*<*State*> &&*currentState*)

**virtual ~StateManager**()

#### Protected Attributes

*Ptr*<*State*> **m_CurrentState**

## Class StaticAudioBuffer

- Defined in file_rootex_core_audio_static_audio_buffer.h

### Inheritance Relationships

### Base Type

- `public AudioBuffer` (*Class AudioBuffer*)

### Class Documentation

**class StaticAudioBuffer** : **public** *AudioBuffer*

An audio buffer that is sent entirely at once to the audio card. Preferable for smaller audio files. Larger files may be slow to start.

#### Public Functions

**StaticAudioBuffer**(*Ref*<*AudioResourceFile*> *audioFile*)

**~StaticAudioBuffer**()

*ALuint* &**getBuffer**()

### Class StaticAudioSource

- Defined in file_rootex_core_audio_audio_source.h

### Inheritance Relationships

### Base Type

- `public AudioSource` (*Class AudioSource*)

### Class Documentation

**class StaticAudioSource** : **public** *AudioSource*
An audio source that uses *StaticAudioBuffer*.

#### Public Functions

**StaticAudioSource**(*Ref*<*StaticAudioBuffer*> *audio*)

**~StaticAudioSource**()

void **unqueueBuffers**()

float **getDuration**() **const**
Get audio duration in seconds.

float **getElapsedTimeS**()

### Class StaticMeshColliderComponent

- Defined in file_rootex_framework_components_physics_static_mesh_collider_component.h

### Inheritance Relationships

### Base Type

- `public RigidBodyComponent` (*Class RigidBodyComponent*)

### Class Documentation

**class StaticMeshColliderComponent** : **public** *RigidBodyComponent*
Collision component with a 3D model as its shape.

#### Public Functions

**StaticMeshColliderComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~StaticMeshColliderComponent**()

bool **setupData**()
Perform setting up internal data needed from other components after they have been added to the owning entity.

void **setCollisionModel**(*Ref*<*CollisionModelResourceFile*> *file*)

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class StaticPointLightComponent

- Defined in file_rootex_framework_components_visual_light_static_point_light_component.h

### Inheritance Relationships

### Base Type

- `public PointLightComponent` (*Class PointLightComponent*)

### Class Documentation

**class StaticPointLightComponent** : **public** *PointLightComponent*
*Component* to apply static point lights to the scene. These type of lights are rendered in larger numbers with the limitation that they are static i.e. movement of these lights will not show up inside the final render.

#### Public Functions

**StaticPointLightComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~StaticPointLightComponent**()

void **draw**()
Expose the component data with ImGui.

### Class StopTimer

- Defined in file_rootex_os_timer.h

### Inheritance Relationships

### Base Type

- `public Timer` (*Class Timer*)

### Class Documentation

**class StopTimer** : **public** *Timer*
A timer that works like a stopwatch.

#### Public Functions

**StopTimer**()

**StopTimer**(*StopTimer*&)

**virtual ~StopTimer**()

void **reset**()
Reset start time to now.

### Class StreamingAudioBuffer

- Defined in file_rootex_core_audio_streaming_audio_buffer.h

### Inheritance Relationships

### Base Type

- `public AudioBuffer` (*Class AudioBuffer*)

### Class Documentation

**class StreamingAudioBuffer** : **public** *AudioBuffer*
An audio buffer that is streamed to the audio card instead of sent entirely at once. Allows faster startup for large audio files.

#### Public Functions

**StreamingAudioBuffer**(*Ref*<*AudioResourceFile*> *audioFile*)

**~StreamingAudioBuffer**()

void **loadNewBuffers**(int *count*, bool *isLooping*)

*ALuint* ***getBuffers**()

int **getBufferQueueLength**()

### Class StreamingAudioSource

- Defined in file_rootex_core_audio_audio_source.h

### Inheritance Relationships

### Base Type

- `public AudioSource` (*Class AudioSource*)

### Class Documentation

**class StreamingAudioSource** : **public** *AudioSource*
An audio source that uses *StreamingAudioBuffer*.

#### Public Functions

**StreamingAudioSource**(*Ref*<*StreamingAudioBuffer*> *audio*)

**~StreamingAudioSource**()

bool **isLooping**() **const**

void **setLooping**(bool *enabled*)

void **queueNewBuffers**()
Queue new buffers to the audio card if possible.

void **unqueueBuffers**()

**virtual** float **getDuration**() **const**
Get audio duration in seconds.

### Class System

- Defined in file_rootex_framework_system.h

### Inheritance Relationships

### Derived Types

- `public AnimationSystem` (*Class AnimationSystem*)
- `public AudioSystem` (*Class AudioSystem*)
- `public DebugSystem` (*Class DebugSystem*)
- `public EditorSystem` (*Class EditorSystem*)
- `public GameRenderSystem` (*Class GameRenderSystem*)
- `public InputSystem` (*Class InputSystem*)

- `public LightSystem` (*Class LightSystem*)
- `public ParticleSystem` (*Class ParticleSystem*)
- `public PauseSystem` (*Class PauseSystem*)
- `public PhysicsSystem` (*Class PhysicsSystem*)
- `public PlayerSystem` (*Class PlayerSystem*)
- `public PostProcessSystem` (*Class PostProcessSystem*)
- `public RenderSystem` (*Class RenderSystem*)
- `public RenderUISystem` (*Class RenderUISystem*)
- `public ScriptSystem` (*Class ScriptSystem*)
- `public TransformAnimationSystem` (*Class TransformAnimationSystem*)
- `public TransformSystem` (*Class TransformSystem*)
- `public TriggerSystem` (*Class TriggerSystem*)
- `public UISystem` (*Class UISystem*)

## Class Documentation

**class System**

ECS style *System* interface that allows iterating over components directly.

Subclassed by *AnimationSystem*, *AudioSystem*, *DebugSystem*, *EditorSystem*, *GameRenderSystem*, *InputSystem*, *LightSystem*, *ParticleSystem*, *PauseSystem*, *PhysicsSystem*, *PlayerSystem*, *PostProcessSystem*, *RenderSystem*, *RenderUISystem*, *ScriptSystem*, *TransformAnimationSystem*, *TransformSystem*, *TriggerSystem*, *UISystem*

### Public Types

**enum UpdateOrder**

*Values:*

**Input**

**Update**

**PostUpdate**

**Render**

**PostRender**

**RenderUI**

**UI**

**GameRender**

**Editor**

**Async**

**End**

### Public Functions

**System**(**const** *String* &*name*, **const** *UpdateOrder* &*order*, bool *isGameplay*)

**System**(*System*&)

**virtual ~System**()

**virtual** bool **initialize**(**const** JSON::json &*systemData*)

**virtual** void **setConfig**(**const** *SceneSettings* &*sceneSettings*)

**virtual** void **begin**()

**virtual** void **update**(float *deltaMilliseconds*)

**virtual** void **end**()

*String* **getName**() **const**

**const** *UpdateOrder* &**getUpdateOrder**() **const**

bool **isActive**() **const**

void **setActive**(bool *enabled*)

**virtual** void **draw**()

### Public Static Functions

**static const** *Vector*<*Vector*<*System* *>> &**GetSystems**()

**static** void **pause**()

**static** void **unPause**()

### Protected Attributes

*String* **m_SystemName**

*UpdateOrder* **m_UpdateOrder**

bool **m_IsActive**

bool **m_IsSystemPaused**

### Protected Static Attributes

*Vector*<*Vector*<*System* *>> **s_Systems**

### Friends

**friend System::Entity**

### Class Task

- Defined in file_rootex_os_thread.h

### Class Documentation

**class Task**
Defines jobs to be run on threads.

#### Public Functions

**Task**(**const** *Function*<void) > &*executionTask*

**Task**(**const** *Task*&)

**~Task**()

void **execute**()

#### Public Members

__int32 **m_ID**

__int32 **m_Dependencies**

*Vector*<__int32> **m_Permissions**

*Function*<void()> **m_ExecutionTask**

### Class TextResourceFile

- Defined in file_rootex_core_resource_files_text_resource_file.h

### Inheritance Relationships

### Base Type

- `public ResourceFile` (*Class ResourceFile*)

### Derived Type

- `public LuaTextResourceFile` (*Class LuaTextResourceFile*)

### Class Documentation

**class TextResourceFile** : **public** *ResourceFile*
Representation of a text file.

Subclassed by *LuaTextResourceFile*

### Public Functions

**TextResourceFile**(*TextResourceFile*&)

**TextResourceFile**(*TextResourceFile*&&)

**virtual ~TextResourceFile**()

void **putString**(**const** *String* &*newData*)
Replace old data string with new data string.

void **popBack**()
Remove 1 character from the end of the data buffer.

void **append**(**const** *String* &*add*)

*String* **getString**() **const**
Get the resource data buffer as a readable String.

size_t **getSize**() **const**

void **reimport**()
Reload the file buffer from disk.

bool **save**()

### Protected Functions

**TextResourceFile**(**const** *FilePath* &*path*)

### Protected Attributes

*String* **m_FileString**

### Friends

**friend TextResourceFile::ResourceLoader**

## Class TextUIComponent

- Defined in file_rootex_framework_components_visual_ui_text_ui_component.h

## Inheritance Relationships

### Base Type

- `public RenderUIComponent` (*Class RenderUIComponent*)

### Class Documentation

**class TextUIComponent** : **public** *RenderUIComponent*
*Component* to render 2D UI Text.

#### Public Types

**enum Mode**
DirectXTK flipping modes for sprites.

*Values:*

**None** = DirectX::SpriteEffects_None

**FlipX** = DirectX::SpriteEffects_FlipHorizontally

**FlipY** = DirectX::SpriteEffects_FlipVertically

**FlipXY** = DirectX::SpriteEffects_FlipBoth

#### Public Functions

**TextUIComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~TextUIComponent**()

void **setFont**(*Ref*<*FontResourceFile*> *fontFile*)

void **setText**(**const** *String* &*text*)

void **render**()

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class TextureCube

- Defined in file_rootex_core_renderer_texture.h

### Class Documentation

**class TextureCube**
Cube *GPUTexture* in 3D.

#### Public Functions

**TextureCube**(**const** char ***imageDDSFileData*, size_t *size*)

**TextureCube**(**const** *TextureCube*&)

*TextureCube* &**operator=**(**const** *TextureCube*&)

**~TextureCube**()

ID3D11ShaderResourceView ***getTextureResourceView**() **const**

### Class TextViewer

- Defined in file_editor_gui_text_viewer.h

### Class Documentation

**class TextViewer**

#### Public Functions

*Ref*<*ResourceFile*> **load**(**const** *FilePath* &*filePath*)

void **unload**()

void **draw**(float *deltaMilliseconds*)

### Class ThreadPool

- Defined in file_rootex_os_thread.h

### Class Documentation

**class ThreadPool**

#### Public Functions

**ThreadPool**()

**ThreadPool**(*ThreadPool*&)

**~ThreadPool**()

void **submit**(*Vector*<*Ref*<*Task*>> &*tasks*)
To submit a job to the jobs queue.

bool **isCompleted**() **const**
Returns true if all tasks have been completed.

void **join**() **const**
Returns when all the tasks have been completed.

### Class Timer

- Defined in file_rootex_os_timer.h

### Inheritance Relationships

### Derived Types

- `public LoggingScopeTimer` (*Class LoggingScopeTimer*)
- `public StopTimer` (*Class StopTimer*)

### Class Documentation

**class Timer**

Helper to keep track of time.

Subclassed by *LoggingScopeTimer*, *StopTimer*

#### Public Functions

**Timer**()

**Timer**(*Timer*&)

**virtual ~Timer**()

float **getTimeMs**() **const**

float **getTimeNs**() **const**

#### Public Static Functions

**static** *TimePoint* **Now**()

#### Protected Attributes

*TimePoint* **m_StartTime**

*TimePoint* **m_EndTime**

#### Protected Static Attributes

**const** std::chrono::high_resolution_clock **s_Clock**

### Class ToolbarDock

- Defined in file_editor_gui_toolbar_dock.h

### Nested Relationships

### Nested Types

- *Struct ToolbarDock::ToolbarDockSettings*

### Class Documentation

**class ToolbarDock**

#### Public Functions

**ToolbarDock**()

**ToolbarDock**(*ToolbarDock*&)

**~ToolbarDock**()

void **draw**(float *deltaMilliseconds*)

*ToolbarDockSettings* &**getSettings**()

void **setActive**(bool *enabled*)

**struct ToolbarDockSettings**

##### Public Members

bool **m_IsActive** = true

bool **m_InEditorPlaying** = false

## Class TransformAnimationComponent

- Defined in file_rootex_framework_components_space_transform_animation_component.h

### Nested Relationships

### Nested Types

- *Struct TransformAnimationComponent::Keyframe*

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class TransformAnimationComponent** : **public** *Component*
Manages animations based on provided keyframes.

### Public Types

**enum TransitionType**
First word is for incoming, second word for outgoing.

*Values:*

**SmashSmash** = 0

**EaseEase** = 1

**SmashEase** = 2

**EaseSmash** = 3

**enum AnimationMode**
*Values:*

**None** = 0

**Looping** = 1

**Alternating** = 2

### Public Functions

**TransformAnimationComponent** (*Entity* &*owner*, **const** JSON::json &*data*)

**~TransformAnimationComponent** ()

void **pushKeyframe** (**const** *Keyframe* &*keyFrame*)

void **popKeyframe** (int *count*)

bool **isPlaying** () **const**

bool **isPlayOnStart** () **const**

bool **hasEnded** () **const**

float **getStartTime** () **const**

float **getEndTime** () **const**

void **reset** ()

void **interpolate** (float *t*)

void **setPlaying** (bool *enabled*)

bool **setupData** ()
Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON** () **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw** ()
Expose the component data with ImGui.

**struct Keyframe**

#### Public Members

float **timePosition**

*Matrix* **transform**

### Class TransformAnimationSystem

- Defined in file_rootex_framework_systems_transform_animation_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class TransformAnimationSystem** : **public** *System*
*System* which handles the initialisation and updation of transform animations.

#### Public Functions

**TransformAnimationSystem**()

void **begin**()

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *TransformAnimationSystem* ***GetSingleton**()

### Class TransformComponent

- Defined in file_rootex_framework_components_space_transform_component.h

### Nested Relationships

### Nested Types

- *Struct TransformComponent::TransformBuffer*

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class TransformComponent** : **public** *Component*

Stores the transform (position, rotation, scale) information of any entity.

#### Public Functions

**TransformComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~TransformComponent**()

void **setPosition**(**const** *Vector3* &*position*)

void **setRotation**(**const** float &*yaw*, **const** float &*pitch*, **const** float &*roll*)

void **setRotationQuaternion**(**const** *Quaternion* &*rotation*)

void **setScale**(**const** *Vector3* &*scale*)

void **setLocalTransform**(**const** *Matrix* &*transform*)

void **setAbsoluteTransform**(**const** *Matrix* &*transform*)

void **setBounds**(**const** *BoundingBox* &*bounds*)

void **setRotationPosition**(**const** *Matrix* &*transform*)

void **setAbsoluteRotationPosition**(**const** *Matrix* &*transform*)

void **setParentAbsoluteTransform**(**const** *Matrix* &*parentTransform*)

void **addLocalTransform**(**const** *Matrix* &*applyTransform*)

void **addQuaternion**(**const** *Quaternion* &*applyQuaternion*)

void **addRotation**(float *yaw*, float *pitch*, float *roll*)

*Vector3* **getPosition**() **const**

*Quaternion* **getRotation**() **const**

**const** *Vector3* &**getScale**() **const**

**const** *Matrix* &**getLocalTransform**() **const**

*Matrix* **getRotationPosition**() **const**

*Matrix* **getParentAbsoluteTransform**() **const**

int **getPassDowns**() **const**

*BoundingBox* **getWorldSpaceBounds**()

*Matrix* **getAbsoluteTransform**()

*Vector3* **getAbsolutePosition**()

void **setAbsolutePosition**(**const** *Vector3* &*position*)

*Quaternion* **getAbsoluteRotation**()

*Vector3* **getAbsoluteScale**()

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

void **highlight**()

### Class TransformSystem

- Defined in file_rootex_framework_systems_transform_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class TransformSystem** : **public** *System*

#### Public Functions

void **calculateTransforms**(*Scene* **scene*)

void **pushMatrix**(**const** *Matrix* &*transform*)

void **pushMatrixOverride**(**const** *Matrix* &*transform*)

void **popMatrix**()

**const** *Matrix* &**getCurrentMatrix**() **const**

#### Public Static Functions

**static** *TransformSystem* ***GetSingleton**()

### Class TriggerComponent

- Defined in file_rootex_framework_components_physics_trigger_component.h

### Inheritance Relationships

### Base Type

- `public CollisionComponent` (*Class CollisionComponent*)

### Class Documentation

**class TriggerComponent** : **public** *CollisionComponent*

*Component* that notifies all targets when a scene enters its bounds.

#### Public Functions

**TriggerComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~TriggerComponent**()

bool **isEntryRepeat**()

bool **isExitRepeat**()

void **setDimensions**(**const** *Vector3* &*dimensions*)

void **addEntryTarget**(*SceneID toAdd*)

void **addExitTarget**(*SceneID toAdd*)

void **removeEntryTarget**(*SceneID toRemove*)

void **removeExitTarget**(*SceneID toRemove*)

btGhostObject ***getGhostObject**()

bool **setupData**()

Perform setting up internal data needed from other components after they have been added to the owning entity.

JSON::json **getJSON**() **const**

Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()

Expose the component data with ImGui.

### Class TriggerSystem

- Defined in file_rootex_framework_systems_trigger_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class TriggerSystem** : **public** *System*
Responsible for managing the notifications of trigger components.

#### Public Functions

void **update**(float *deltaMilliseconds*)

#### Public Static Functions

**static** *TriggerSystem* ***GetSingleton**()

### Class UIComponent

- Defined in file_rootex_framework_components_visual_ui_ui_component.h

### Inheritance Relationships

### Base Type

- `public Component` (*Class Component*)

### Class Documentation

**class UIComponent** : **public** *Component*
UI component for RmlUi.

#### Public Functions

**UIComponent**(*Entity* &*owner*, **const** JSON::json &*data*)

**~UIComponent**()

Rml::ElementDocument ***getDocument**()

void **setDocument**(**const** *String* &*path*)

JSON::json **getJSON**() **const**
Get JSON representation of the component data needed to re-construct component from memory.

void **draw**()
Expose the component data with ImGui.

### Class UISystem

- Defined in file_rootex_framework_systems_ui_system.h

### Inheritance Relationships

### Base Type

- `public System` (*Class System*)

### Class Documentation

**class UISystem** : **public** *System*
Manages the UI rendering using RmlUi.

#### Public Functions

void **loadFont**(**const** *String* &*path*)

Rml::ElementDocument ***loadDocument**(**const** *String* &*path*)

void **unloadDocument**(Rml::ElementDocument *&*document*)

bool **initialize**(**const** JSON::json &*systemData*)

void **update**(float *deltaMilliseconds*)

void **shutDown**()

void **setDebugger**(bool *enabled*)

Rml::Context ***getContext**()

void **draw**()

#### Public Static Functions

**static** *UISystem* ***GetSingleton**()

### Class VertexBuffer

- Defined in file_rootex_core_renderer_vertex_buffer.h

### Class Documentation

**class VertexBuffer**
Encapsulates a vector of vertices to be used as Vertex Buffer.

#### Public Functions

**VertexBuffer**(**const** char ***buffer*, unsigned int *elementCount*, unsigned int *stride*, D3D11_USAGE *usage*, int *cpuAccess*)

**~VertexBuffer**()

void **bind**() **const**

unsigned int **getCount**() **const**

unsigned int **getStride**() **const**

ID3D11Buffer ***getBuffer**() **const**

### Class Viewport

- Defined in file_rootex_core_renderer_viewport.h

### Class Documentation

**class Viewport**

Encapsulation of the viewport being rendered to.

#### Public Functions

**Viewport**(float *width*, float *height*, float *minDepth*, float *maxDepth*, float *topLeftX*, float *topLeftY*)

**Viewport**(*Viewport*&)

**~Viewport**()

**const** D3D11_VIEWPORT ***getViewport**() **const**

### Class ViewportDock

- Defined in file_editor_gui_viewport_dock.h

### Nested Relationships

### Nested Types

- *Struct ViewportDock::ViewportDockSettings*

### Class Documentation

**class ViewportDock**

### Public Functions

**ViewportDock**(**const** JSON::json &*viewportJSON*)

**ViewportDock**(*ViewportDock*&)

**~ViewportDock**()

void **draw**(float *deltaMilliseconds*)

*ViewportDockSettings* &**getSettings**()

void **setActive**(bool *enabled*)

**struct ViewportDockSettings**

#### Public Members

bool **m_IsActive** = true

bool **m_IsClosed**

float **m_AspectRatio**

## Class Window

- Defined in file_rootex_main_window.h

## Class Documentation

**class Window**

Handles window creation.

### Public Functions

**Window**(int *xOffset*, int *yOffset*, int *width*, int *height*, **const** *String* &*title*, bool *isEditor*, bool *fullScreen*, **const** *String* &*icon*)

**Window**(**const** *Window*&)

*Window* &**operator=**(**const** *Window*&)

**~Window**()

void **show**()

std::optional<int> **processMessages**()

void **swapBuffers**()

void **applyDefaultViewport**()

void **clipCursor**(RECT *clip*)

Clips or blocks the cursor beyond the specified rectangle.

void **resetClipCursor**()
Reset cursor clips and allow free cursor movement.

void **showCursor**(bool *enabled*)

void **clearMain**(**const** *Color* &*color*)

void **clearOffScreen**(**const** *Color* &*color*)

*Variant* **toggleFullscreen**(**const** *Event* **event*)

*Variant* **getScreenState**(**const** *Event* **event*)

int **getWidth**() **const**

int **getHeight**() **const**

int **getTitleBarHeight**() **const**

HWND **getWindowHandle**()

void **setWindowTitle**(*String title*)

void **setWindowSize**(**const** *Vector2* &*newSize*)

### Protected Functions

*Variant* **quitWindow**(**const** *Event* **event*)

*Variant* **quitEditorWindow**(**const** *Event* **event*)

*Variant* **windowResized**(**const** *Event* **event*)

### Protected Attributes

int **m_Width**

int **m_Height**

bool **m_IsEditorWindow**

bool **m_IsFullscreen**

WNDCLASSEX **m_WindowClass** = {0}

LPCSTR **m_ClassName**

HINSTANCE **m_AppInstance**

HWND **m_WindowHandle**

### Protected Static Functions

**static LRESULT CALLBACK Window::WindowsProc(HWND windowHandler, UINT msg, WPARAM wPara**
Wraps DefWindowProc function.

## Enums

### Enum CollisionMask

- Defined in file_rootex_framework_components_physics_collision_component.h

### Enum Documentation

**enum CollisionMask**

*Values:*

**None** = 0

**Player** = 1 << 0

**Enemy** = 1 << 1

**Architecture** = 1 << 2

**TriggerVolume** = 1 << 3

**Other** = 1 << 4

**All** = *Player* | *Enemy* | *Architecture* | *TriggerVolume* | *Other*

### Enum ComponentIDs

- Defined in file_rootex_framework_components_component_ids.h

### Enum Documentation

**enum ComponentIDs**

*Values:*

**ModelComponent**

**GridModelComponent**

**SpriteComponent**

**CPUParticlesComponent**

**DecalComponent**

**TextUIComponent**

**UIComponent**

**CameraComponent**

**TransformComponent**

**PointLightComponent**

**DirectionalLightComponent**

**SpotLightComponent**

**PhysicsColliderComponent**

**SphereColliderComponent**

**BoxColliderComponent**

**CapsuleColliderComponent**

**StaticMeshColliderComponent**

**TriggerComponent**

**ScriptComponent**

**MusicComponent**

**ShortMusicComponent**

**AudioListenerComponent**

**TransformAnimationComponent**

**SkyComponent**

**FogComponent**

**StaticPointLightComponent**

**AnimatedModelComponent**

**RenderableComponent**

**ParticleEffectComponent**

**PlayerController**

### Enum Device

- Defined in file_rootex_core_input_input_manager.h

### Enum Documentation

**enum Device**

Enumeration of the devices supported. Any new device can be added here.

*Values:*

**Mouse**

**Keyboard**

**Pad1**

Player 1 pad. Usually the first pad connected.

**Pad2**

Player 2 pad. Usually the second pad connected.

### Enum PhysicsMaterial

- Defined in file_rootex_framework_systems_physics_system.h

### Enum Documentation

**enum PhysicsMaterial**
*Values:*

**Air** = 0

**Water** = 1

**Wood** = 2

**End**

### Enum RenderPass

- Defined in file_rootex_core_renderer_render_pass.h

### Enum Documentation

**enum RenderPass**
*Values:*

**Basic** = 1 << 0

**Editor** = 1 << 1

**Alpha** = 1 << 2

### Enum RootExclusion

- Defined in file_rootex_core_resource_files_animated_model_resource_file.h

### Enum Documentation

**enum RootExclusion**
*Values:*

**None** = 0

**Translation** = 1

**All** = 2

### Enum TransformPassDown

- Defined in file_rootex_framework_components_space_transform_component.h

### Enum Documentation

**enum TransformPassDown**
*Values:*

**Position** = 1 << 0

**Rotation** = 1 << 1

**Scale** = 1 << 2

**All** = *Position* | *Rotation* | *Scale*

### Enum TYPES_OF_BUFFERS

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Enum Documentation

**enum TYPES_OF_BUFFERS**

*Values:*

**FLOATCB**

**FLOAT3CB**

**COLORCB**

### Functions

### Function BtTransformToMat

- Defined in file_rootex_core_physics_bullet_conversions.h

### Function Documentation

*Matrix* **BtTransformToMat**(btTransform **const** &*trans*)

### Function BtVector3ToVec

- Defined in file_rootex_core_physics_bullet_conversions.h

### Function Documentation

*Vector3* **BtVector3ToVec**(btVector3 **const** &*btvec*)

### Function ColorToImColor

- Defined in file_editor_editor_system.h

### Function Documentation

ImColor **ColorToImColor**(*Color* &*c*)

### Function CompareMaterials

- Defined in file_rootex_framework_components_visual_model_model_component.h

### Function Documentation

bool **CompareMaterials**(**const** *Pair*<*Ref*<*MaterialResourceFile*>, *Vector*<*Mesh*>> &*a*, **const** *Pair*<*Ref*<*MaterialResourceFile*>, *Vector*<*Mesh*>> &*b*)

### Function CreateRootexApplication

- Defined in file_rootex_app_application.h

### Function Documentation

*Ref*<*Application*> **CreateRootexApplication**()

Externally defined function that returns a Ref object of a derived class of *Application*. This should be defined only once in a project. This is used by the main function to construct the Rootex application.

### Function DECLARE_COMPONENT(AnimatedModelComponent)

- Defined in file_rootex_framework_components_visual_model_animated_model_component.h

### Function Documentation

**DECLARE_COMPONENT**(*AnimatedModelComponent*)

### Function DECLARE_COMPONENT(AudioListenerComponent)

- Defined in file_rootex_framework_components_audio_audio_listener_component.h

### Function Documentation

**DECLARE_COMPONENT**(*AudioListenerComponent*)

### Function DECLARE_COMPONENT(BoxColliderComponent)

- Defined in file_rootex_framework_components_physics_box_collider_component.h

### Function Documentation

**DECLARE_COMPONENT**(*BoxColliderComponent*)

### Function DECLARE_COMPONENT(CameraComponent)

- Defined in file_rootex_framework_components_visual_camera_component.h

### Function Documentation

**DECLARE_COMPONENT**(*CameraComponent*)

### Function DECLARE_COMPONENT(CapsuleColliderComponent)

- Defined in file_rootex_framework_components_physics_capsule_collider_component.h

### Function Documentation

**DECLARE_COMPONENT**(*CapsuleColliderComponent*)

### Function DECLARE_COMPONENT(CPUParticlesComponent)

- Defined in file_rootex_framework_components_visual_effect_cpu_particles_component.h

### Function Documentation

**DECLARE_COMPONENT**(*CPUParticlesComponent*)

### Function DECLARE_COMPONENT(DecalComponent)

- Defined in file_rootex_framework_components_visual_effect_decal_component.h

### Function Documentation

**DECLARE_COMPONENT**(*DecalComponent*)

### Function DECLARE_COMPONENT(DirectionalLightComponent)

- Defined in file_rootex_framework_components_visual_light_directional_light_component.h

### Function Documentation

**DECLARE_COMPONENT**(*DirectionalLightComponent*)

### Function DECLARE_COMPONENT(FogComponent)

- Defined in file_rootex_framework_components_visual_effect_fog_component.h

### Function Documentation

**DECLARE_COMPONENT** (*FogComponent*)

### Function DECLARE_COMPONENT(GridModelComponent)

- Defined in file_rootex_framework_components_visual_model_grid_model_component.h

### Function Documentation

**DECLARE_COMPONENT** (*GridModelComponent*)

### Function DECLARE_COMPONENT(ModelComponent)

- Defined in file_rootex_framework_components_visual_model_model_component.h

### Function Documentation

**DECLARE_COMPONENT** (*ModelComponent*)

### Function DECLARE_COMPONENT(MusicComponent)

- Defined in file_rootex_framework_components_audio_music_component.h

### Function Documentation

**DECLARE_COMPONENT** (*MusicComponent*)

### Function DECLARE_COMPONENT(ParticleEffectComponent)

- Defined in file_rootex_framework_components_visual_effect_particle_effect_component.h

### Function Documentation

**DECLARE_COMPONENT** (*ParticleEffectComponent*)

### Function DECLARE_COMPONENT(PlayerController)

- Defined in file_rootex_framework_components_game_player_controller.h

### Function Documentation

**DECLARE_COMPONENT** (*PlayerController*)

### Function DECLARE_COMPONENT(PointLightComponent)

- Defined in file_rootex_framework_components_visual_light_point_light_component.h

### Function Documentation

**DECLARE_COMPONENT** (*PointLightComponent*)

### Function DECLARE_COMPONENT(ShortMusicComponent)

- Defined in file_rootex_framework_components_audio_short_music_component.h

### Function Documentation

**DECLARE_COMPONENT** (*ShortMusicComponent*)

### Function DECLARE_COMPONENT(SkyComponent)

- Defined in file_rootex_framework_components_visual_effect_sky_component.h

### Function Documentation

**DECLARE_COMPONENT** (*SkyComponent*)

### Function DECLARE_COMPONENT(SphereColliderComponent)

- Defined in file_rootex_framework_components_physics_sphere_collider_component.h

### Function Documentation

**DECLARE_COMPONENT** (*SphereColliderComponent*)

### Function DECLARE_COMPONENT(SpotLightComponent)

- Defined in file_rootex_framework_components_visual_light_spot_light_component.h

### Function Documentation

**DECLARE_COMPONENT** (*SpotLightComponent*)

### Function DECLARE_COMPONENT(SpriteComponent)

- Defined in file_rootex_framework_components_visual_model_sprite_component.h

### Function Documentation

**DECLARE_COMPONENT** (*SpriteComponent*)

### Function DECLARE_COMPONENT(StaticMeshColliderComponent)

- Defined in file_rootex_framework_components_physics_static_mesh_collider_component.h

### Function Documentation

**DECLARE_COMPONENT** (*StaticMeshColliderComponent*)

### Function DECLARE_COMPONENT(StaticPointLightComponent)

- Defined in file_rootex_framework_components_visual_light_static_point_light_component.h

### Function Documentation

**DECLARE_COMPONENT** (*StaticPointLightComponent*)

### Function DECLARE_COMPONENT(TextUIComponent)

- Defined in file_rootex_framework_components_visual_ui_text_ui_component.h

### Function Documentation

**DECLARE_COMPONENT** (*TextUIComponent*)

### Function DECLARE_COMPONENT(TransformAnimationComponent)

- Defined in file_rootex_framework_components_space_transform_animation_component.h

### Function Documentation

**DECLARE_COMPONENT** (*TransformAnimationComponent*)

### Function DECLARE_COMPONENT(TransformComponent)

- Defined in file_rootex_framework_components_space_transform_component.h

### Function Documentation

**DECLARE_COMPONENT** (*TransformComponent*)

### Function DECLARE_COMPONENT(TriggerComponent)

- Defined in file_rootex_framework_components_physics_trigger_component.h

### Function Documentation

**DECLARE_COMPONENT**(*TriggerComponent*)

### Function DECLARE_COMPONENT(UIComponent)

- Defined in file_rootex_framework_components_visual_ui_ui_component.h

### Function Documentation

**DECLARE_COMPONENT**(*UIComponent*)

### Function ECSFactory::AddComponent

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

bool ECSFactory::**AddComponent**(*Entity* &*entity*, ComponentID *componentID*, **const** JSON::json &*componentData*, bool *checks* = true)

### Function ECSFactory::AddDefaultComponent

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

bool ECSFactory::**AddDefaultComponent**(*Entity* &*entity*, ComponentID *componentID*, bool *checks* = true)

### Function ECSFactory::CopyEntity

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

void ECSFactory::**CopyEntity**(*Entity* &*entity*, *Entity* &*copyTarget*)

### Function ECSFactory::FillEntity

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

void `ECSFactory::FillEntity`(*Entity* &*entity*, **const** JSON::json &*entityJSON*)

### Function ECSFactory::FillEntityFromFile

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

void `ECSFactory::FillEntityFromFile`(*Entity* &*entity*, *TextResourceFile* **textResourceFile*)

### Function ECSFactory::FillRootEntity

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

void `ECSFactory::FillRootEntity`(*Entity* &*root*)

### Function ECSFactory::GetComponentIDByName

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

ComponentID `ECSFactory::GetComponentIDByName`(**const** *String* &*componentName*)

### Function ECSFactory::GetComponentNameByID

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

*String* `ECSFactory::GetComponentNameByID`(ComponentID *componentID*)

### Function ECSFactory::Initialize

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

void ECSFactory::**Initialize**()

### Function ECSFactory::RemoveComponent

- Defined in file_rootex_framework_ecs_factory.h

### Function Documentation

bool ECSFactory::**RemoveComponent**(*Entity* &*entity*, ComponentID *componentID*)

### Template Function Extract

- Defined in file_rootex_common_types.h

### Function Documentation

template<typename **P**, typename **Q**>
*P* **Extract**(**const** *Q* &*v*)
Extract the value of type TypeName from a Variant.

### Function from_json(const JSON::json&, ParticleTemplate&)

- Defined in file_rootex_framework_components_visual_effect_cpu_particles_component.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *ParticleTemplate* &*p*)

### Function from_json(const JSON::json&, InputDescription&)

- Defined in file_rootex_core_input_input_manager.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *InputDescription* &*s*)

### Function from_json(const JSON::json&, InputScheme&)

- Defined in file_rootex_core_input_input_manager.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *InputScheme* &*s*)

### Function from_json(const JSON::json&, BasicMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *BasicMaterialData* &*b*)

### Function from_json(const JSON::json&, SkyMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *SkyMaterialData* &*s*)

### Function from_json(const JSON::json&, CustomMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *CustomMaterialData* &*s*)

### Function from_json(const JSON::json&, DecalMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *DecalMaterialData* &*d*)

### Function from_json(const JSON::json&, ResourceFile::Type&)

- Defined in file_rootex_core_resource_file.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *ResourceFile*::*Type* &*s*)

### Function from_json(const JSON::json&, SceneSettings&)

- Defined in file_rootex_framework_scene.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *SceneSettings* &*s*)

### Function from_json(const JSON::json&, TransformPassDown&)

- Defined in file_rootex_framework_components_space_transform_component.h

### Function Documentation

void **from_json**(**const** JSON::json &*j*, *TransformPassDown* &*t*)

### Function GetPayloadTypes

- Defined in file_rootex_core_resource_loader.h

### Function Documentation

*Vector*<**const** char *> **GetPayloadTypes**(**const** *String* &*extension*)

### Function Interpolate

- Defined in file_rootex_utility_maths.h

### Function Documentation

*Matrix* **Interpolate**(*Matrix* &*left*, *Matrix* &*right*, float *lerpFactor*)

### Function IsFileSupported

- Defined in file_rootex_core_resource_loader.h

### Function Documentation

bool **IsFileSupported**(**const** *String* &*extension*, *ResourceFile*::*Type* *supportedFileType*)

### Function MatTobtTransform

- Defined in file_rootex_core_physics_bullet_conversions.h

### Function Documentation

btTransform **MatTobtTransform**(*Matrix* **const** &*mat*)
Helpers for conversion to and from Bullet's data types.

### Function RootexFPSGraph

- Defined in file_rootex_utility_imgui_helpers.h

### Function Documentation

void **RootexFPSGraph**(**const** char **name*, *Vector*<float> &*fpsRecords*, float *lastFPS*)

### Function RootexSelectableImage

- Defined in file_rootex_utility_imgui_helpers.h

### Function Documentation

void **RootexSelectableImage**(**const** char **name*, *Ref*<*ImageResourceFile*> *image*, *Function*<void)**const** *String*&
> *onSelected*

### Function RootexSelectableImageCube

- Defined in file_rootex_utility_imgui_helpers.h

### Function Documentation

void **RootexSelectableImageCube**(**const** char **name*, *Ref*<*ImageCubeResourceFile*> *image*, *Function*<void)**const** *String*&
> *onSelected*

### Function Split

- Defined in file_rootex_common_types.h

### Function Documentation

*Vector*<*String*> **Split**(**const** *String* &*s*, char *delim*)

### Function StringToWideString

- Defined in file_rootex_os_os.h

### Function Documentation

std::wstring **StringToWideString**(**const** *String* &*str*)

### Function to_json(JSON::json&, const ParticleTemplate)

- Defined in file_rootex_framework_components_visual_effect_cpu_particles_component.h

### Function Documentation

void **to_json**(JSON::json &*j*, **const** *ParticleTemplate* *p*)

### Function to_json(JSON::json&, const InputDescription&)

- Defined in file_rootex_core_input_input_manager.h

### Function Documentation

void **to_json**(JSON::json &*j*, **const** *InputDescription* &*s*)

### Function to_json(JSON::json&, const InputScheme&)

- Defined in file_rootex_core_input_input_manager.h

### Function Documentation

void **to_json**(JSON::json &*j*, **const** *InputScheme* &*s*)

### Function to_json(JSON::json&, const BasicMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **to_json**(JSON::json &*j*, **const** *BasicMaterialData* &*b*)

### Function to_json(JSON::json&, const SkyMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **to_json**(JSON::json &*j*, **const** *SkyMaterialData* &*s*)

### Function to_json(JSON::json&, const CustomMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **to_json** (JSON::json &*j*, **const** *CustomMaterialData* &*s*)

### Function to_json(JSON::json&, const DecalMaterialData&)

- Defined in file_rootex_core_resource_files_material_resource_file.h

### Function Documentation

void **to_json** (JSON::json &*j*, **const** *DecalMaterialData* &*d*)

### Function to_json(JSON::json&, const ResourceFile::Type&)

- Defined in file_rootex_core_resource_file.h

### Function Documentation

void **to_json** (JSON::json &*j*, **const** *ResourceFile*::*Type* &*t*)

### Function to_json(JSON::json&, const SceneSettings&)

- Defined in file_rootex_framework_scene.h

### Function Documentation

void **to_json** (JSON::json &*j*, **const** *SceneSettings* &*s*)

### Function to_json(JSON::json&, const TransformPassDown&)

- Defined in file_rootex_framework_components_space_transform_component.h

### Function Documentation

void **to_json** (JSON::json &*j*, **const** *TransformPassDown* &*t*)

### Function VecTobtVector3

- Defined in file_rootex_core_physics_bullet_conversions.h

### Function Documentation

btVector3 **VecTobtVector3** (*Vector3* **const** &*vec3*)

### Function WideStringToString

- Defined in file_rootex_os_os.h

### Function Documentation

*String* **WideStringToString** (**const** std::wstring &*wstr*)

### Variables

### Variable CreatableFiles

- Defined in file_rootex_core_resource_loader.h

### Variable Documentation

**const HashMap<ResourceFile::Type, const char*> CreatableFiles= { { ResourceFile::Type::Lua**

### Variable ECSFactory::s_ComponentSets

- Defined in file_rootex_framework_ecs_factory.h

### Variable Documentation

*HashMap*<*String*, *Ptr*<*BaseComponentSet*>> ECSFactory::**s_ComponentSets**

### Variable FONT_ICON_BUFFER_NAME_ROOTEX

- Defined in file_rootex_utility_imgui_helpers.h

### Variable Documentation

**const** char **FONT_ICON_BUFFER_NAME_ROOTEX**[5415 + 1]

### Variable m_PayloadTypes

- Defined in file_rootex_core_resource_loader.h

### Variable Documentation

```
const HashMap<String, Vector<const char*> > m_PayloadTypes= { { ".png", { "IMAGE_PAYLOAD" }
```

### Variable SupportedFiles

- Defined in file_rootex_core_resource_loader.h

### Variable Documentation

```
const HashMap<ResourceFile::Type, const char*> SupportedFiles= { { ResourceFile::Type::Imag
```

### Defines

### Define _WIN32_WINNT

- Defined in file_rootex_common_types.h

### Define Documentation

```
_WIN32_WINNT
```

### Define AL_CHECK

- Defined in file_rootex_framework_systems_audio_system.h

### Define Documentation

**AL_CHECK**(alFunction)

### Define ALUT_CHECK

- Defined in file_rootex_framework_systems_audio_system.h

### Define Documentation

**ALUT_CHECK**(alutFunction)

### Define BONES_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**BONES_VS_CPP**

### Define BONES_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**BONES_VS_HLSL**

### Define BUFFER_COUNT

- Defined in file_rootex_core_audio_streaming_audio_buffer.h

### Define Documentation

**BUFFER_COUNT**
Number of equal slices that an audio buffer is cut down into.

### Define COMPONENT

- Defined in file_rootex_framework_component.h

### Define Documentation

**COMPONENT**(ComponentType, category)

### Define CONCAT

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CONCAT**(a, b)

### Define CONCAT

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CONCAT**(a, b)

### Define CUSTOM_PER_FRAME_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_PER_FRAME_PS_CPP**

### Define CUSTOM_PER_FRAME_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_PER_FRAME_PS_HLSL**

### Define CUSTOM_PER_OBJECT_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_PER_OBJECT_PS_CPP**

### Define CUSTOM_PER_OBJECT_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_PER_OBJECT_PS_HLSL**

### Define CUSTOM_TEXTURE_0_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_0_PS_CPP**

### Define CUSTOM_TEXTURE_0_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_0_PS_HLSL**

### Define CUSTOM_TEXTURE_0_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_0_VS_CPP**

### Define CUSTOM_TEXTURE_0_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_0_VS_HLSL**

### Define CUSTOM_TEXTURE_1_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_1_PS_CPP**

### Define CUSTOM_TEXTURE_1_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_1_PS_HLSL**

### Define CUSTOM_TEXTURE_1_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_1_VS_CPP**

### Define CUSTOM_TEXTURE_1_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_1_VS_HLSL**

### Define CUSTOM_TEXTURE_2_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_2_PS_CPP**

### Define CUSTOM_TEXTURE_2_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_2_PS_HLSL**

### Define CUSTOM_TEXTURE_2_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_2_VS_CPP**

### Define CUSTOM_TEXTURE_2_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_2_VS_HLSL**

### Define CUSTOM_TEXTURE_3_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_3_PS_CPP**

### Define CUSTOM_TEXTURE_3_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_3_PS_HLSL**

### Define CUSTOM_TEXTURE_3_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_3_VS_CPP**

### Define CUSTOM_TEXTURE_3_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_3_VS_HLSL**

### Define CUSTOM_TEXTURE_4_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_4_PS_CPP**

### Define CUSTOM_TEXTURE_4_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**CUSTOM_TEXTURE_4_PS_HLSL**

### Define CUSTOM_TEXTURE_4_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_4_VS_CPP**

### Define CUSTOM_TEXTURE_4_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**CUSTOM_TEXTURE_4_VS_HLSL**

### Define DEBUG_PANIC

- Defined in file_rootex_common_common.h

### Define Documentation

**DEBUG_PANIC**(m_IfTtrue, m_Msg)
Panic, but only in debug mode.

### Define DECLARE_COMPONENT

- Defined in file_rootex_framework_component.h

### Define Documentation

**DECLARE_COMPONENT**(Type)

### Define DEFINE_COMPONENT

- Defined in file_rootex_framework_component.h

### Define Documentation

**DEFINE_COMPONENT**(Type)

### Define DEFINE_EVENT

- Defined in file_rootex_core_event.h

### Define Documentation

**DEFINE_EVENT** (eventName, ...)

### Define DEPENDENCY

- Defined in file_rootex_framework_component.h

### Define Documentation

**DEPENDENCY** (ComponentType, mode)

### Define DEPENDS_ON

- Defined in file_rootex_framework_component.h

### Define Documentation

**DEPENDS_ON** (ComponentType)

### Define DEPTH_TEXTURE_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**DEPTH_TEXTURE_PS_CPP**

### Define DEPTH_TEXTURE_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**DEPTH_TEXTURE_PS_HLSL**

### Define DIFFUSE_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**DIFFUSE_PS_CPP**

### Define DIFFUSE_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**DIFFUSE_PS_HLSL**

### Define ENGINE_DIRECTORY

- Defined in file_rootex_os_os.h

### Define Documentation

**ENGINE_DIRECTORY**

### Define ERR

- Defined in file_rootex_common_common.h

### Define Documentation

**ERR**(m_Msg)
Logs file, line, function, message in red color.

### Define ERR_CUSTOM

- Defined in file_rootex_common_common.h

### Define Documentation

**ERR_CUSTOM**(m_file, m_func, m_Msg)
Logs file, function, message in red color.

### Define ERR_CUSTOM_SILENT

- Defined in file_rootex_common_common.h

### Define Documentation

**ERR_CUSTOM_SILENT**(m_file, m_func, m_Msg)
Logs file, function, message in red color, doesn't log in editor console.

### Define ERR_SILENT

- Defined in file_rootex_common_common.h

### Define Documentation

**ERR_SILENT**(m_Msg)
Logs file, line, function, message in red color, doesn't log in editor console.

### Define FONT_ICON_BUFFER_NAME_ROOTEX

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**FONT_ICON_BUFFER_NAME_ROOTEX**

### Define FONT_ICON_BUFFER_SIZE_ROOTEX

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**FONT_ICON_BUFFER_SIZE_ROOTEX**

### Define GAME_DIRECTORY

- Defined in file_rootex_os_os.h

### Define Documentation

**GAME_DIRECTORY**

### Define GFX_ERR_CHECK

- Defined in file_rootex_core_renderer_dxgi_debug_interface.h

### Define Documentation

**GFX_ERR_CHECK**(hr)
Used to get DIrectX errors/warnings from its debug interface.

### Define GOD_RAYS_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**GOD_RAYS_PS_CPP**

### Define GOD_RAYS_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**GOD_RAYS_PS_HLSL**

### Define ICON_MAX_ROOTEX

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_MAX_ROOTEX**

### Define ICON_MIN_ROOTEX

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_MIN_ROOTEX**

### Define ICON_ROOTEX_BOOKMARK

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_BOOKMARK**

### Define ICON_ROOTEX_CHECK

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_CHECK**

### Define ICON_ROOTEX_CLOCK_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_CLOCK_O**

### Define ICON_ROOTEX_CLOUD

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_CLOUD**

### Define ICON_ROOTEX_DATABASE

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_DATABASE**

### Define ICON_ROOTEX_EXTERNAL_LINK

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_EXTERNAL_LINK**

### Define ICON_ROOTEX_FILE

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FILE**

### Define ICON_ROOTEX_FILE_AUDIO_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FILE_AUDIO_O**

### Define ICON_ROOTEX_FILE_CODE_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FILE_CODE_O**

### Define ICON_ROOTEX_FILE_IMAGE_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FILE_IMAGE_O**

### Define ICON_ROOTEX_FILE_TEXT

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FILE_TEXT**

### Define ICON_ROOTEX_FILES_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FILES_O**

### Define ICON_ROOTEX_FLOPPY_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FLOPPY_O**

### Define ICON_ROOTEX_FOLDER

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FOLDER**

### Define ICON_ROOTEX_FOLDER_OPEN

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FOLDER_OPEN**

### Define ICON_ROOTEX_FONT

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FONT**

### Define ICON_ROOTEX_FORT_AWESOME

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_FORT_AWESOME**

### Define ICON_ROOTEX_MINUS

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_MINUS**

### Define ICON_ROOTEX_MINUS_CIRCLE

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_MINUS_CIRCLE**

### Define ICON_ROOTEX_PENCIL_SQUARE_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_PENCIL_SQUARE_O**

### Define ICON_ROOTEX_PICTURE_O

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_PICTURE_O**

### Define ICON_ROOTEX_PLUS

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_PLUS**

### Define ICON_ROOTEX_REFRESH

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_REFRESH**

### Define ICON_ROOTEX_REPEAT

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_REPEAT**

### Define ICON_ROOTEX_SEARCH

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_SEARCH**

### Define ICON_ROOTEX_WINDOW_CLOSE

- Defined in file_rootex_utility_imgui_helpers.h

### Define Documentation

**ICON_ROOTEX_WINDOW_CLOSE**

### Define interface

- Defined in file_rootex_core_ui_custom_render_interface.h

### Define Documentation

**interface**

### Define interface

- Defined in file_rootex_core_ui_input_interface.h

### Define Documentation

**interface**

### Define interface

- Defined in file_rootex_framework_components_visual_ui_ui_component.h

### Define Documentation

**interface**

### Define interface

- Defined in file_rootex_framework_systems_ui_system.h

### Define Documentation

**interface**

### Define LIGHTMAP_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**LIGHTMAP_PS_CPP**

### Define LIGHTMAP_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**LIGHTMAP_PS_HLSL**

### Define MAX_BONES

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**MAX_BONES**

### Define MAX_BUFFER_QUEUE_LENGTH

- Defined in file_rootex_core_audio_streaming_audio_buffer.h

### Define Documentation

**MAX_BUFFER_QUEUE_LENGTH**
Maximum number of buffers queued at once.

### Define MAX_COMPONENT_ARRAY_SIZE

- Defined in file_rootex_utility_component_array.h

### Define Documentation

**MAX_COMPONENT_ARRAY_SIZE**

### Define MAX_DYNAMIC_POINT_LIGHTS

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**MAX_DYNAMIC_POINT_LIGHTS**

### Define MAX_DYNAMIC_POINT_LIGHTS

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**MAX_DYNAMIC_POINT_LIGHTS**

### Define MAX_DYNAMIC_SPOT_LIGHTS

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**MAX_DYNAMIC_SPOT_LIGHTS**

### Define MAX_DYNAMIC_SPOT_LIGHTS

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**MAX_DYNAMIC_SPOT_LIGHTS**

### Define MAX_LOD_COUNT

- Defined in file_rootex_core_renderer_mesh.h

### Define Documentation

**MAX_LOD_COUNT**

### Define MAX_PARTICLES

- Defined in file_rootex_framework_components_visual_effect_cpu_particles_component.h

### Define Documentation

**MAX_PARTICLES**

### Define MAX_STATIC_POINT_LIGHTS

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**MAX_STATIC_POINT_LIGHTS**

### Define MAX_STATIC_POINT_LIGHTS_AFFECTING_1_OBJECT

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**MAX_STATIC_POINT_LIGHTS_AFFECTING_1_OBJECT**

### Define MIN_TO_S

- Defined in file_rootex_core_audio_audio_source.h

### Define Documentation

**MIN_TO_S**
Convert minutes to seconds.

### Define MS_TO_NS

- Defined in file_rootex_common_types.h

### Define Documentation

**MS_TO_NS**
Convert milliseconds to nanoseconds.

### Define MS_TO_S

- Defined in file_rootex_common_types.h

### Define Documentation

**MS_TO_S**
Convert milliseconds to seconds.

### Define NOMINMAX

- Defined in file_rootex_common_types.h

### Define Documentation

**NOMINMAX**

### Define NORMAL_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**NORMAL_PS_CPP**

### Define NORMAL_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**NORMAL_PS_HLSL**

### Define NS_TO_MS

- Defined in file_rootex_common_types.h

### Define Documentation

**NS_TO_MS**
Convert nanoseconds to milliseconds.

### Define PANIC

- Defined in file_rootex_common_common.h

### Define Documentation

**PANIC**(m_IfTtrue, m_Msg)
Logs file, line, function, message in yellow color in condition is true.

### Define PANIC_SILENT

- Defined in file_rootex_common_common.h

### Define Documentation

**PANIC_SILENT**(m_IfTtrue, m_Msg)
Logs file, line, function, message in yellow color in condition is true, doesn't log in editor console.

### Define PER_CAMERA_CHANGE_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_CAMERA_CHANGE_PS_CPP**

### Define PER_CAMERA_CHANGE_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_CAMERA_CHANGE_PS_HLSL**

### Define PER_CAMERA_CHANGE_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**PER_CAMERA_CHANGE_VS_CPP**

### Define PER_CAMERA_CHANGE_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**PER_CAMERA_CHANGE_VS_HLSL**

### Define PER_DECAL_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_DECAL_PS_CPP**

### Define PER_DECAL_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_DECAL_PS_HLSL**

### Define PER_FRAME_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_FRAME_PS_CPP**

### Define PER_FRAME_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_FRAME_PS_HLSL**

### Define PER_FRAME_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**PER_FRAME_VS_CPP**

### Define PER_FRAME_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**PER_FRAME_VS_HLSL**

### Define PER_MODEL_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_MODEL_PS_CPP**

### Define PER_MODEL_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_MODEL_PS_HLSL**

### Define PER_OBJECT_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_OBJECT_PS_CPP**

### Define PER_OBJECT_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_OBJECT_PS_HLSL**

### Define PER_OBJECT_VS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**PER_OBJECT_VS_CPP**

### Define PER_OBJECT_VS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_vertex_shader.h

### Define Documentation

**PER_OBJECT_VS_HLSL**

### Define PER_SCENE_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_SCENE_PS_CPP**

### Define PER_SCENE_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**PER_SCENE_PS_HLSL**

### Define PRINT

- Defined in file_rootex_common_common.h

### Define Documentation

**PRINT**(m_Msg)
Logs function, message in white color.

### Define PRINT_SILENT

- Defined in file_rootex_common_common.h

### Define Documentation

**PRINT_SILENT**(m_Msg)
Logs function, message in white color, doesn't log in editor console.

### Define ROOT_MARKER_FILENAME

- Defined in file_rootex_os_os.h

### Define Documentation

**ROOT_MARKER_FILENAME**
Filename that marks the starting directory for Rootex.

### Define ROOT_SCENE_ID

- Defined in file_rootex_framework_scene.h

### Define Documentation

**ROOT_SCENE_ID**

### Define S_TO_MS

- Defined in file_rootex_common_types.h

### Define Documentation

**S_TO_MS**
Convert seconds to milliseconds.

### Define SAMPLER_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**SAMPLER_PS_CPP**

### Define SAMPLER_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**SAMPLER_PS_HLSL**

### Define SKY_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**SKY_PS_CPP**

### Define SKY_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**SKY_PS_HLSL**

### Define SOFT_DEPENDS_ON

- Defined in file_rootex_framework_component.h

### Define Documentation

**SOFT_DEPENDS_ON**(ComponentType)

### Define SOL_ALL_SAFETIES_ON

- Defined in file_rootex_script_interpreter.h

### Define Documentation

**SOL_ALL_SAFETIES_ON**

### Define SOL_PRINT_ERRORS

- Defined in file_rootex_script_interpreter.h

### Define Documentation

**SOL_PRINT_ERRORS**

### Define SOL_STD_VARIANT

- Defined in file_rootex_script_interpreter.h

### Define Documentation

**SOL_STD_VARIANT**

### Define SOL_USING_CXX_LUA

- Defined in file_rootex_script_interpreter.h

### Define Documentation

**SOL_USING_CXX_LUA**

### Define SPECULAR_PS_CPP

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**SPECULAR_PS_CPP**

### Define SPECULAR_PS_HLSL

- Defined in file_rootex_core_renderer_shaders_register_locations_pixel_shader.h

### Define Documentation

**SPECULAR_PS_HLSL**

### Define STRICT

- Defined in file_rootex_common_types.h

### Define Documentation

**STRICT**

### Define WARN

- Defined in file_rootex_common_common.h

### Define Documentation

**WARN**(m_Msg)
Logs file, line, function, message in yellow color.

### Define WARN_SILENT

- Defined in file_rootex_common_common.h

### Define Documentation

**WARN_SILENT**(m_Msg)
Logs file, line, function, message in yellow color, doesn't log in editor console.

### Define WINVER

- Defined in file_rootex_common_types.h

### Define Documentation

**WINVER**

## Typedefs

### Typedef ALfloat

- Defined in file_rootex_core_audio_audio_source.h

### Typedef Documentation

**typedef** float **ALfloat**

### Typedef ALuint

- Defined in file_rootex_core_audio_audio_source.h

### Typedef Documentation

**typedef** unsigned int **ALuint**

### Typedef Array

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Array** = std::array<T, N>
std::array

### Typedef Atomic

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Atomic** = std::atomic<T>
Atomic data type.

### Typedef BoundingBox

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::BoundingBox **BoundingBox**
DirectX::SimpleMath::BoundingBox.

### Typedef Color

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::SimpleMath::Color **Color**
DirectX::SimpleMath::Color.

### Typedef ComponentID

- Defined in file_rootex_framework_ecs_factory.h

### Typedef Documentation

**typedef** unsigned int **ComponentID**

### Typedef ComponentID

- Defined in file_rootex_framework_entity.h

### Typedef Documentation

**typedef** unsigned int **ComponentID**

### Typedef DeviceButtonID

- Defined in file_rootex_core_input_input_manager.h

### Typedef Documentation

**typedef** gainput::DeviceButtonId **DeviceButtonID**
ID of any key, button or analog on the device hardware.

### Typedef FileBuffer

- Defined in file_rootex_os_os.h

### Typedef Documentation

**typedef** *Vector*<char> **FileBuffer**

### Typedef FilePath

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using FilePath** = std::filesystem::path
std::filesystem::path

### Typedef FileTimePoint

- Defined in file_rootex_os_os.h

### Typedef Documentation

**typedef** std::chrono::time_point<std::filesystem::file_time_type::clock> **FileTimePoint**

### Typedef Function

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Function** = std::function<T>
std::function

### Typedef Future

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Future** = std::future<T>
Future data type for reading future variables.

### Typedef HashMap

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using HashMap** = std::unordered_map<P, Q>
std::unordered_map

### Typedef InputBoolListenerFunction

- Defined in file_rootex_core_input_input_listener.h

### Typedef Documentation

**typedef** std::function<bool (int userButton, bool oldValue, bool newValue) > **InputBoolListenerFunction**

### Typedef InputFileStream

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** std::ifstream **InputFileStream**
std::ifstream

### Typedef InputFloatListenerFunction

- Defined in file_rootex_core_input_input_listener.h

### Typedef Documentation

**typedef** std::function<bool (int userButton, float oldValue, float newValue) > **InputFloatListenerFunction**

### Typedef InputOutputFileStream

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** std::fstream **InputOutputFileStream**
std::fstream

### Typedef KeyboardButton

- Defined in file_rootex_core_input_input_manager.h

### Typedef Documentation

**typedef** gainput::Key **KeyboardButton**
Alias for a keyboard device. Allows various keyboard specific operations.

### Typedef Map

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Map** = std::map<P, Q>
std::map

### Typedef Matrix

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::SimpleMath::Matrix **Matrix**
DirectX::SimpleMath::Matrix.

### Typedef MouseButton

- Defined in file_rootex_core_input_input_manager.h

### Typedef Documentation

**typedef** gainput::MouseButton **MouseButton**
Alias for a mouse device. Allows various mouse specific operations.

### Typedef Mutex

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** std::mutex **Mutex**
Mutex for mutual exclusion.

### Typedef Optional

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Optional** = std::optional<T>
std::optional

### Typedef OutputFileStream

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** std::ofstream **OutputFileStream**
std::ofstream

### Typedef PadButton

- Defined in file_rootex_core_input_input_manager.h

### Typedef Documentation

**typedef** gainput::PadButton **PadButton**
Alias for a game controller/pad device. Allows various pad specific operations. Gainput (our input library) only supports pads that allow XInput. So non-XInput controllers are not supported.

### Typedef Pair

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Pair** = std::pair<P, Q>
std::pair

### Typedef Promise

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Promise** = std::promise<T>
Promise data types for sharing futures.

### Typedef Ptr

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Ptr** = std::unique_ptr<T>
std::unique_ptr

### Typedef Quaternion

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::SimpleMath::Quaternion **Quaternion**
DirectX::SimpleMath::Quaternion.

### Typedef Ray

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::SimpleMath::Ray **Ray**
DirectX::SimpleMath::Ray.

### Typedef RecursiveMutex

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** std::recursive_mutex **RecursiveMutex**

### Typedef Ref

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Ref** = std::shared_ptr<T>
std::shared_ptr

### Typedef ResourceCollection

- Defined in file_rootex_core_resource_file.h

### Typedef Documentation

**typedef** *Vector*<*Pair*<*ResourceFile*::*Type*, *String*>> **ResourceCollection**

### Typedef SceneID

- Defined in file_rootex_framework_entity.h

### Typedef Documentation

**typedef** unsigned int **SceneID**

### Typedef Stack

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Stack** = std::stack<T>
std::stack

### Typedef String

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** std::string **String**
std::string

### Typedef StringStream

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** std::stringstream **StringStream**
std::stringstream

### Typedef TimePoint

- Defined in file_rootex_os_timer.h

### Typedef Documentation

**typedef** std::chrono::time_point<std::chrono::high_resolution_clock> **TimePoint**
A point in time of the high resolution clock.

### Typedef Tuple

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Tuple** = std::tuple<P...>
std::tuple

### Typedef Variant

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Variant** = std::variant<bool, int, char, float, *String*, *Vector*<*String*>, *Vector2*, *Vector3*, *Vector4*, *Matrix*, *VariantVector*, *Scene*
A variant able to hold multiple kinds of data, one at a time.

### Typedef VariantVector

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** *Vector*<std::variant<bool, int, char, float, *String*, *Vector2*, *Vector3*, *Vector4*, *Matrix*>> **VariantVector**
DirectX::Colors.

Vector of std::variant of bool, int, char, float, String, Vector2, Vector3, Vector4, Matrix

### Typedef Vector

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Vector** = std::vector<T>
std::vector

### Typedef Vector2

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::SimpleMath::Vector2 **Vector2**
DirectX::SimpleMath::Vector2.

### Typedef Vector3

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::SimpleMath::Vector3 **Vector3**
DirectX::SimpleMath::Vector3.

### Typedef Vector4

- Defined in file_rootex_common_types.h

### Typedef Documentation

**typedef** DirectX::SimpleMath::Vector4 **Vector4**
DirectX::SimpleMath::Vector4.

### Typedef Weak

- Defined in file_rootex_common_types.h

### Typedef Documentation

**using Weak** = std::weak_ptr<T>
std::weak_ptr

# INDEX

## Symbols



## A

## B

## C

## D

## E

## F



## G

## H



## I

## K



## L

## M

## N



## O

## P

## Q



## R

## S

## T

## U



## V

## W