Asset Framework Documentation

This documentation covers all the scripts / textures / materials used in the Demo Scene.
This will help you understand how it all works, so that you can do your own tweaks / modifications or additions to the existing framework if required.

The common framework.
It is important to note that I am trying to create a common standard/framework for all my assets with navigable interiors (and others). This means common scripts, materials, even meshes. **Therefore if you familiarize yourself with how I structure my scenes/assets in one package, things should be familiar for my other assets** (that use the same standard).

Some assets from this documentation will not appear in your purchased pack.
Each pack only includes the required assets for their particular demo implementation. More complex asset packs will include more of the below mentioned assets.

If you are interested to know which of my assets use the shared framework, or if you have any other questions, please send me an e-mail at **VattalusAssets@gmail.com**

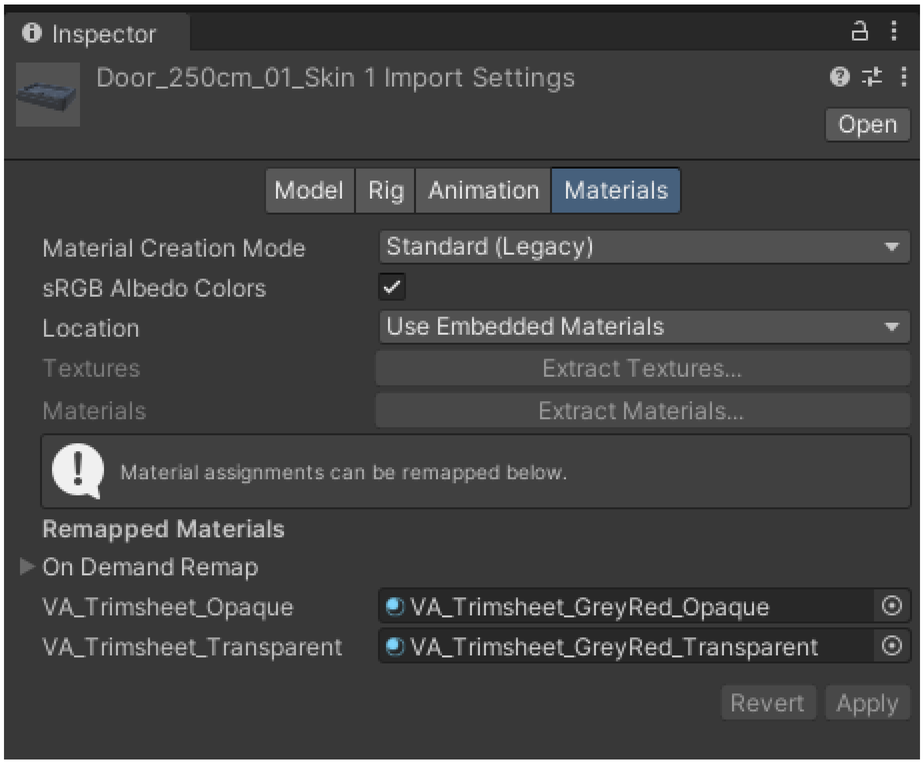
- **Materials and textures**

Main material and texture set
The meshes are textured using the trimsheet method. This means multiple meshes use the same material. Around 90% of the surfaces use the same 2 materials (one opaque, one transparent. Same textures but different shaders). Some other materials are used where the trimsheet is not enough (lights, glass, fabric etc).
The main advantage of using this method is: vastly reduces build size, memory usage and a reduction in drawcalls.

Location: VattalusAssets>Common>Materials>Trimsheet

Here you can find the base textures used for the trimsheet materials (in various formats, depending on your needs. My shader uses the standard unity PBR format, which is MetalSmooth). Multiple color variants are provided, but you can create your own using the included .PSD file. You can customize the 3 different color zones (Primary color, Accent color and Neutral).
The main material also uses a triplanar grunge texture to add some more detail.

Changing trimsheet color variants
If you want to change the color variant or if something goes wrong, make sure to assign the 2 Trimsheet material versions to their respective slots. (Some meshes only use the Opaque material, some use both, some use additional material slots for the aforementioned additional materials like lights)



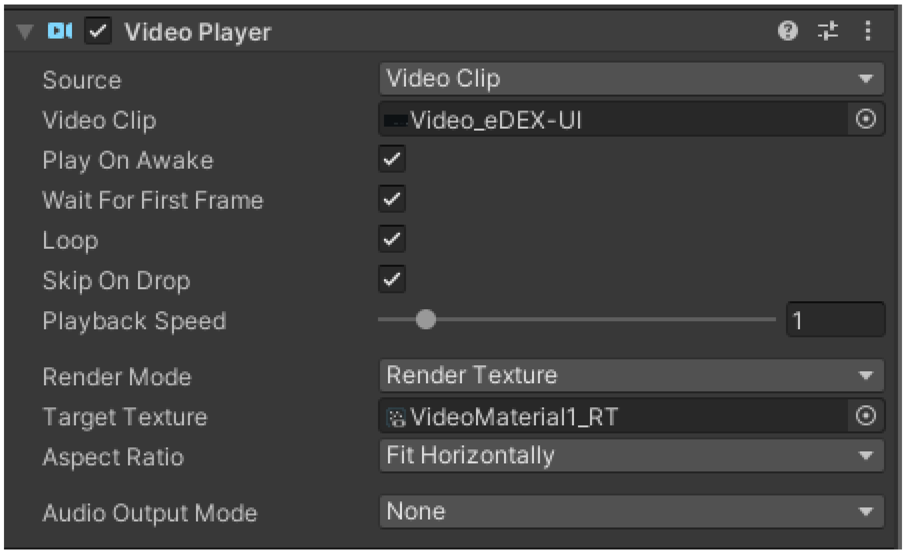
Material Creation Mode: Standard (Legacy)
sRGB Albedo Colors: ✓
Location: Use Embedded Materials
Textures: Extract Textures...
Materials: Extract Materials...
Material assignments can be remapped below.
Remapped Materials
On Demand Remap
VA_Trimsheet_Opaque: VA_Trimsheet_GreyRed_Opaque
VA_Trimsheet_Transparent: VA_Trimsheet_GreyRed_Transparent

You can change the material color variant on individual sub-components of the mesh, like wall panels, inside the scene.

Animated Lights Material
This is another material commonly used across many asset packs. It consists of a single texture map, a shader, and a material. It is used to texture various blinking lights of different colors and blinking patterns, all using the same material.

During texturing the desired light is mapped to a small area on the map, this will dictate its color and blinking pattern.

Video Materials
6 video materials are included. These are used decoratively on screens.
Each material consists of the video file, a render texture, and a material.
In order for it to work, this component needs to be in the scene (1 for each material).



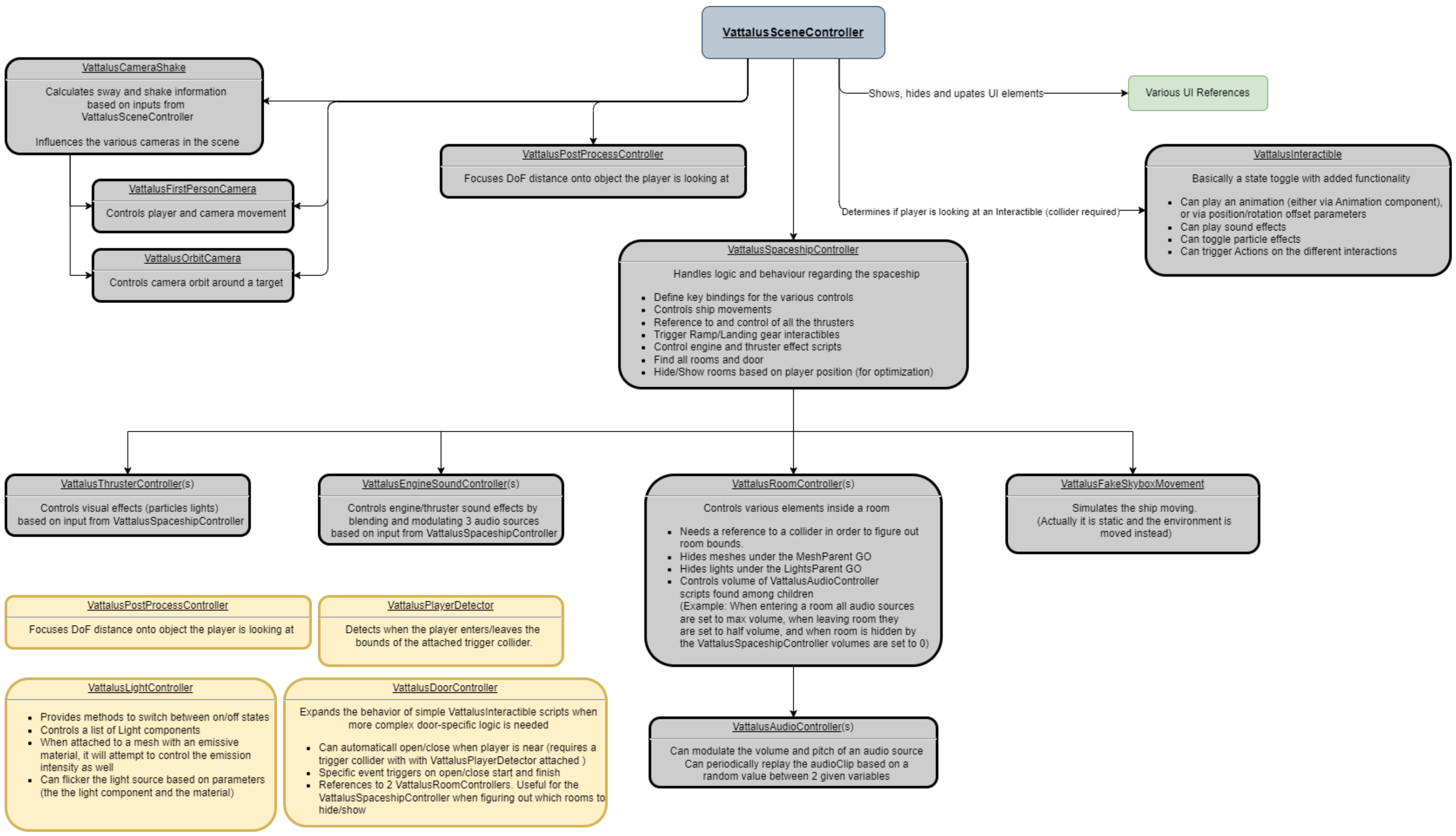
Video Player

Source	Video Clip
Video Clip	Video_eDEX-UI
Play On Awake	✓
Wait For First Frame	✓
Loop	✓
Skip On Drop	✓
Playback Speed	1
Render Mode	Render Texture
Target Texture	VideoMaterial1_RT
Aspect Ratio	Fit Horizontally
Audio Output Mode	None

Various other simple materials
An assortment of other example materials are included (such as emissive lights, fabric, different types of glass etc). These are used where the main trimsheet is not enough.

- **Scripts**

Several scripts are included in order to bring the demo scenes to life, usually to make interactions possible (open/close doors, sit in seats etc), and some are simply aesthetic (flickering lights). Below is a quick breakdown if the important scripts and their applications:



VattalusSceneController
Shows, hides and upates UI elements
Various UI References
Determines if player is looking at an Interactible (collider required)
VattalusCameraShake
Calculates sway and shake information based on inputs from VattalusSceneController
Influences the various cameras in the scene
VattalusFirstPersonCamera
Controls player and camera movement
VattalusOrbitCamera
Controls camera orbit around a target
VattalusPostProcessController
Focuses DoF distance onto object the player is looking at
VattalusInteractible
Basically a state toggle with added functionality
Can play an animation (either via Animation component), or via position/rotation offset parameters
Can play sound effects
Can toggle particle effects
Can trigger Actions on the different interactions
VattalusSpaceshipController
Handles logic and behaviour regarding the spaceship
Define key bindings for the various controls
Controls ship movements
Reference to and control of all the thrusters
Trigger Ramp/Landing gear interactibles
Control engine and thruster effect scripts
Find all rooms and door
Hide/Show rooms based on player position (for optimization)
VattalusThrusterController(s)
Controls visual effects (particles lights) based on input from VattalusSpaceshipController
VattalusEngineSoundController(s)
Controls engine/thruster sound effects by blending and modulating 3 audio sources based on input from VattalusSpaceshipController
VattalusRoomController(s)
Controls various elements inside a room
Needs a reference to a collider in order to figure out room bounds.
Hides meshes under the MeshParent GO
Hides lights under the LightsParent GO
Controls volume of VattalusAudioController scripts found among children (Example: When entering a room all audio sources are set to max volume, when leaving room they are set to half volume, and when room is hidden by the VattalusSpaceshipController volumes are set to 0)
VattalusFakeSkyboxMovement
Simulates the ship moving. (Actually it is static and the environment is moved instead)
VattalusAudioController(s)
Can modulate the volume and pitch of an audio source
Can periodically replay the audioClip based on a random value between 2 given variables
VattalusPostProcessController
Focuses DoF distance onto object the player is looking at
VattalusPlayerDetector
Detects when the player enters/leaves the bounds of the attached trigger collider.
VattalusLightController
Provides methods to switch between on/off states
Controls a list of Light components
When attached to a mesh with an emissive material, it will attempt to control the emission intensity as well
Can flicker the light source based on parameters (the the light component and the material)
VattalusDoorController
Expands the behavior of simple VattalusInteractible scripts when more complex door-specific logic is needed
Can automaticall open/close when player is near (requires a trigger collider with with VattalusPlayerDetector attached)
Specific event triggers on open/close start and finish
References to 2 VattalusRoomControllers. Useful for the VattalusSpaceshipController when figuring out which rooms to hide/show

The **VattalusFirstPersonCamera** handles FPS movement and checks if the player is looking at an intractable object (a collider with an attached VattalusInteractible script component)

The **VattalusSceneController** handles UI and player related logic. It has the following roles:

- References, updates and manages all the UI elements
- References and enables/disables the different camera views

The **VattalusSpaceshipController** controls the ship-oriented behaviors:

- Sets key bindings that control the ship (movement, landing gear etc)
- Moves the Joystick and Throttle control inside the cockpit/bridge based on player input
- Deploys Ramp/Landing gear upon correct keypress
- Controls all the thrusters (VattalusThrusterController) based on ship movement key inputs
- Controls the different types of Engine/Thruster sounds (VattalusEngineSoundController)
- Controls the fake skybox rotation (VattalusFakeSkyboxMovement) when the ship is rotating in order to fake actual movement

VattalusSpaceshipManager mainy optimizes the scene by hiding rooms that are not visible to the player. It does this by checking all rooms (VattalusRoomController) among the children and only showing that one and the ones it is connected to (either via an open door, or via a window. Rooms with permanent view between them have to have references to each other in the VattalusRoomController script).
In case the player is outside the ship (or when in orbit camera mode) only rooms marked with “VisibleFromExterior” are shown.
The optimization check is done whenever a door is opened/closer, when the player enters/exits the ship and when the camera mode is changed.

VattalusInteractable is commonly used on everything that the player interacts with (doors, drawers, switches, seats, beds etc). At the core, it is simply toggles between two states, with added functionality on each switch:

- Play a sound effect on activation/deactivation
- Play an animation (forward and in reverse) upon activation/deactivation
- Or, animate by transitioning between the initial position/rotation and a provided offset
- If “IsSeat” is enabled, upon interaction the FPS camera moves the to target anchor position, and a rotation angle restriction is applied.
- Toggle particle effects
- Trigger 4 distinct Event callbacks on activation/deactivation and when animation starts/ends.
- Can link to other interactables to propagate the interaction further. When “ConformLinked” is checked, propagation only occurs as to ensure the same state on all linked interactables. (Commonly used on doors with multiple moving panels. Each part has its own interactable script, because each script needs its own collider. When interacting with any part of a door, the interaction is propagated to all other parts to ensure all parts play the open animation).

VattalusDoorController can be viewed as an extension of the VattalusInteractable script, but with specific functionality for doors. It still uses VattalusInteractable components, but has the following extra functionality:

- References to 2 VattalusRoomController components that the door connects.
- Specific Event callbacks for then the door open/close starts/ends.
- Can automatically open/close when player enters/exits a bounds trigger collider (Requires a child object with a trigger collider, and a VattalusPlayerDetector script)

VattalusRoomController controls aspects regarding individual rooms, for example show/hide meshes/lights, and control the volume of ambient sound effects (VattalusAudioController).
It also needs a reference to a collider that marks the bounds of the room. With an added VattalusPlayerDetector script added to said collider we can trigger certain events when the player enters/exits the room. (We could easily implement a behavior where lights are toggled when the player enters/leaves the room)

All other scripts are simple and self contained. If in doubt, refer to the annotations inside the code files. I tried to keep everything as well explained as possible.