## System Operation Contracts

|  |  |
| --- | --- |
| Name | **StartHorde3DApplication(Horde3DApplication)** |
| Task | Configures and starts the given Horde3DApplication. |
| Preconditions | There is currently no other started Horde3DApplication. |
| Postconditions | * The original Horde3D.dll was renamed to Horde3D\_org.dll. * The system’s proxy Horde3D.dll was copied to the location of the original Horde3D.dll. * The settings file was copied to the location of the Horde3DApplication’s main executable. * The Horde3DApplication was started and the proxy Horde3D.dll initialized itself with the settings found in the settings file. * The communication between the debugger process and the Horde3DApplication process was established. |
| Exceptions | The given Horde3DApplication object is configured incorrectly (e.g. the original Horde3D.dll cannot be found, the main executable cannot be found, unable to rename the original Horde3D.dll, unable to copy the proxy Horde3D.dll, unable to copy the settings file, unable to start the application, the application terminates unexpectedly). |
| Type | System |
| Links | UC “Debug Scene”, SSD “Debug Scene“ |
| Result | void |
| Notes | - |

|  |  |
| --- | --- |
| Name | **SuspendHorde3DApplication()** |
| Task | Waits for the next time the Horde3D render() function is called. Then for each Horde3D function call a FunctionCall object is generated until render() is called again. Returns the list of FunctionCall objects in the order they were created. Then suspends the Horde3DApplication by replacing the underlying window procedure and by suspending all threads found in the Horde3DApplication’s process. |
| Preconditions | There is currently a running Horde3DApplication. |
| Postconditions | * The FunctionCall objects were created and were associated to the Horde3DApplication. * The FunctionCall objects were returned sorted by the order of their creation. |
| Exceptions | The Horde3DApplication cannot be suspended. |
| Type | System |
| Links | UC “Debug Scene”, SSD “Debug Scene“ |
| Result | FunctionCall[\*] |
| Notes | In the design model the FunctionCall class is no longer associated to the Horde3DApplication class. This association has been replaced with an association to the Horde3DCall class. |

|  |  |
| --- | --- |
| Name | **AdvanceToNextRenderCall()** |
| Task | Resumes the Horde3DApplication by replacing the underlying window procedure with the original handler and by resuming all threads. Until the next time the Horde3D render() function is called a FunctionCall object is created for each Horde3D function call. Adds those new objects to the already existing list of FunctionCall objects in the order they were created. Then suspends the Horde3DApplication by replacing the underlying window procedure and by suspending all threads found in the Horde3DApplication’s process. |
| Preconditions | There is currently a suspended Horde3DApplication. |
| Postconditions | * The new FunctionCall objects were created and were associated to the Horde3DApplication. * The FunctionCall objects were returned sorted by the order of their creation. |
| Exceptions | The Horde3DApplication cannot be resumed or suspended. |
| Type | System |
| Links | UC “Debug Scene”, SSD “Debug Scene“ |
| Result | FunctionCall[\*] |
| Notes | In the design model the FunctionCall class is no longer associated to the Horde3DApplication class. This association has been replaced with an association to the Horde3DCall class. |

|  |  |
| --- | --- |
| Name | **ResumeHorde3DApplication()** |
| Task | Resumes the Horde3DApplication by replacing the underlying window procedure with the original handler and by resuming all threads. |
| Preconditions | There is currently a suspended Horde3DApplication. |
| Postconditions | * The Horde3DApplication was resumed. * All FunctionCall objects were removed as well as their associations to the Horde3DApplication. |
| Exceptions | The Horde3DApplication cannot be resumed. |
| Type | System |
| Links | UC “Debug Scene”, SSD “Debug Scene“ |
| Result | void |
| Notes | In the design model the FunctionCall class is no longer associated to the Horde3DApplication class. This association has been replaced with an association to the Horde3DCall class. |

|  |  |
| --- | --- |
| Name | **ShutDownHorde3DApplication()** |
| Task | Tells Horde3DApplication to shut down. |
| Preconditions | There is currently a running or suspended Horde3DApplication. |
| Postconditions | The Horde3DApplication was shut down. |
| Exceptions | The Horde3DApplication cannot be shut down. |
| Type | System |
| Links | UC “Debug Scene”, SSD “Debug Scene“ |
| Result | void |
| Notes | - |

|  |  |
| --- | --- |
| Name | **LoadHorde3DApplication(filePath : string)** |
| Task | Loads a Horde3DApplication object from given file on the hard drive. |
| Preconditions | - |
| Postconditions | A Horde3DApplication object was created and all attributes were set to the values stored in the file. |
| Exceptions | The specified file could not be read or does not contain a serialized Horde3DApplication object. |
| Type | System |
| Links | UC “Configure Horde3D Application Settings”, SSDs “Debug Scene“, “Configure Horde3D Application Settings” |
| Result | Horde3DApplication |
| Notes | The Horde3DApplication objects are deserialized using the standard .NET xml to objects deserialization. Since the implementation is trivial, there is no sequence diagram in the design model. |

|  |  |
| --- | --- |
| Name | **SaveHorde3DApplication(Horde3DApplication, filePath : string)** |
| Task | Saves a Horde3DApplication object to given file on the hard drive. Validates the Horde3DApplication’s settings. |
| Preconditions | - |
| Postconditions | The Horde3DApplication object was stored in the specified file and can later be loaded again using LoadHorde3DApplication(filePath). |
| Exceptions | The specified file could not be created or overwritten. The Horde3DApplication’s settings are invalid. |
| Type | System |
| Links | UC “Configure Horde3D Application Settings”, SSD “Configure Horde3D Application Settings “ |
| Result | void |
| Notes | The Horde3DApplication objects are serialized using the standard .NET objects to xml serialization. Since the implementation is trivial, there is no sequence diagram in the design model. |

|  |  |
| --- | --- |
| Name | **GetSceneNodes() | GetSceneNodes(Resource)** |
| Task | If no Resource is specified, gets all SceneNodes currently managed by the Horde3DApplication. Otherwise only the SceneNodes using the given Resource are returned. |
| Preconditions | There currently is a suspended Horde3DApplication. |
| Postconditions | * The SceneNode objects were created, all associations to other SceneNodes were established and the objects’ attribute values were copied from the current Horde3D state. * The associations to LogMessages and Resources were not established. When needed, they can be lazy-loaded using the overloaded GetResources(SceneNode) and GetLogMessages(SceneNode) functions. |
| Exceptions | There was an error getting the scene nodes from the Horde3DApplication. |
| Type | System |
| Links | UC “Browse Scene Graph”, SSD “Browse Scene Nodes“, “Manage Resource” |
| Result | SceneNode[\*] |
| Notes | Resources and LogMessages are lazy-loaded for performance reasons. In most cases, the returned SceneNodes are only needed to show the scene graph, but only very few SceneNodes will actually be viewed in detail by the user. Only in that case the associated Resources and LogMessages must be loaded.  Implementation note: The result of GetSceneNodes() (without the Resource parameter) should be cached by the GUI for performance reasons. The cache must be invalidated whenever the application is resumed or advanced to the next frame.  Implementation note: The user might be viewing a SceneNode. He then might resume the Horde3DApplication or advance to the next frame. Now the scene node cache is invalidated. Calling GetSceneNodes() again would return a different instance of the same SceneNode (or the SceneNode might already have been removed from the application so that the SceneNode is not returned at all). Furthermore, SceneNode objects are copied when they travel from the GUI to the Horde3DApplication and vice versa. The GUI must ensure that the new SceneNode instance is mapped to the old one and the system must implement a meaningful value-based equality for SceneNodes. |

|  |  |
| --- | --- |
| Name | **GetLogMessages() | GetLogMessages(SceneNode) | GetLogMessages(Resource)** |
| Task | If no Resource or SceneNode is specified, gets all LogMessages that were generated by the Horde3DApplication since the last call to GetLogMessages(). Otherwise only the LogMessages referring to the given Resource or SceneNode are returned. |
| Preconditions | There currently is a running or suspended Horde3DApplication. |
| Postconditions | * The LogMessage objects were created. * If the LogMessages refers to a SceneNode or Resource, the association to the referred SceneNode or Resource was established. |
| Exceptions | There was an error getting the log messages from the Horde3DApplication. |
| Type | System |
| Links | UC “Browse Log Messages”, SSD “Browse Log Messages“, “Show Scene Node”, “Manage Resource” |
| Result | LogMessage[\*] |
| Notes | The Resource or SceneNode associated with a LogMessage must be directly returned here, because the LogMessage might be referring to a SceneNode or Resource that has already been deleted (and therefore won’t be returned by the GetResources() or GetSceneNodes() functions) and the GUI needs to know the SceneNode or Resource when displaying the LogMessage.  Implementation note: GetLogMessages() without any parameters will probably be implemented both as a method and as a callback contract to inform the GUI immediately about new LogMessages. |

|  |  |
| --- | --- |
| Name | **GetResources() | GetResources(SceneNode)** |
| Task | If no SceneNode is specified, gets all Resources that the Horde3DApplication currently uses (unused loaded or unloaded Resources are not returned). Otherwise only the Resources used by the given SceneNode are returned. |
| Preconditions | There currently is a suspended Horde3DApplication. |
| Postconditions | * The Resource objects were created, all associations to other Resources were established and the objects’ attribute values were copied from the current Horde3D state. * Some associations and/or attributes might have required parsing the Resource’s xml file on the disk. * The associations to LogMessages and SceneNodes were not established. When needed, they have to be lazy-loaded using the overloaded GetSceneNodes(Resource) and GetLogMessages(Resource) functions. |
| Exceptions | There was an error getting the resources from the Horde3DApplication. |
| Type | System |
| Links | UC “Browse Resources”, SSD “Browse Resources“, SSD “Show Scene Node” |
| Result | Resource[\*] |
| Notes | SceneNodes and LogMessages are lazy-loaded for performance reasons. In most cases, the returned Resources are only needed to show the resource graph, but only very few Resources will actually be viewed in detail by the user. Only in that case the associated SceneNodes and LogMessages must be loaded.  Implementation note: The result of GetResources() (without the SceneNode parameter) should be cached by the GUI for performance reasons. The cache must be invalidated whenever the application is resumed or advanced to the next frame.  Implementation note: The user might be editing a Resource. He then might resume the Horde3DApplication or advance to the next frame. Now the resource cache is invalidated. Calling GetResources() again would return a different instance of the same Resource (or the Resource might already have been removed from the application so that the Resource is not returned at all). Furthermore, Resource objects are copied when they travel from the GUI to the Horde3DApplication and vice versa. The GUI must ensure that no work is lost and/or that the new Resource instance is mapped to the old one and the system must implement a meaningful value-based equality for Resources. |

|  |  |
| --- | --- |
| Name | **UpdateResource(EditableResource)** |
| Task | Updates the Horde3DApplication’s resource data and redraws the current frame. Saves the changes in the resource’s xml definition. |
| Preconditions | There currently is a suspended Horde3DApplication. |
| Postconditions | * The resource’s data was updated both in its xml definition file and within the Horde3DApplication. * The current frame was redrawn with the updated resource. |
| Exceptions | There was an error validating the changes or writing the changes to the xml file on the hard disk or updating the resource within the Horde3DApplication. |
| Type | System |
| Links | UC “Manage Resources”, SSD “Manage Resources“ |
| Result | void |
| Notes | - |