Class Documentation for «Tales of Rydonia»

During the development, there is going to be a lot of code involved (you don’t say!). Logically we are going to create a lot of new classes during this entire project and documenting them all inline in the code files. But this is not enough and quite risky, to be honest. Should any of the code files be deleted or otherwise compromised, then the documentation of the classes and their functions is gone.

To prevent this scenario, we created this document. In the following table, we are going to document every self-developed class that we are going to use as well as their main methods. This way, we can recreate any method and class that we use in the development, should anything happen to the code files at any time. This is also a good place to look up the methods in a more detailed description and see what exactly this class is for. As an example, we are going to use the **TestClass**. Keep in mind that for every new class, a new table must be generated.

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| TestClass | Class description. | |
|  | **Method** | **Description** |
|  | Public void TestFunction()  string param1 | Method description  Param description |
|  | Public void TestFunction()  string param2 | Method description  Param description |
|  |  |  |

**Color - Description**

Methodnames()

Inherit or Interface, Code

Event Function, Overload, Object-Class, Click\_Event

Not finished

INHALTSVERZEICHNIS

[Class Documentation for «Tales of Rydonia» 1](#_Toc136340728)

[TestClass 1](#_Toc136340729)

[Logger 3](#_Toc136340730)

[GameLogger 3](#_Toc136340731)

[GameManager 4](#_Toc136340732)

[GameLogger 4](#_Toc136340733)

[SaveManager / Load 5](#_Toc136340734)

[SaveManager 5](#_Toc136340735)

[GameData 5](#_Toc136340736)

[Story Dialogue 6](#_Toc136340737)

[StoryAsset 6](#_Toc136340738)

[StoryViewer 6](#_Toc136340739)

[StoryHolder 7](#_Toc136340740)

[StoryNode 7](#_Toc136340741)

[StoryUI 7](#_Toc136340742)

[Item Scritableobject 8](#_Toc136340743)

[Item 8](#_Toc136340744)

[StatItem 8](#_Toc136340745)

[StatItem 8](#_Toc136340746)

[ItemInfo 8](#_Toc136340747)

[ItemInfo 8](#_Toc136340748)

[ItemSlot 9](#_Toc136340749)

[ItemSlot 9](#_Toc136340750)

Logger

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| GameLogger | Logs Messages and Exceptions from other Classes.  Needs to be instantiated. (How-To in the Code Conventions Doc / Class) | |
|  | **Method** | **Description** |
|  | Public GameLogger()  string classname | Constructor Takes the Classname as Parameter to create the Log File |
|  | Private void CreateLogFile()  string classname | Creates a Log File with the name of the Class Name of the Class |
|  | Private void LogEntry()  string type  string message  int lineNumber | Creates a new Log Entry by creating a new Log Event Object  Type of Log (Exception, Information, Log, ...)  Message or Exception message to log  Linenumber where the Log happens |
|  | Private void AddLogEntry()  LogEvent log | Appends a new Log to the Log File Log Event Object, created in LogEntry() |
|  | Private void RemoveLogEntry()  LogEvent log | Removes all Log Entries that are older than 5 days. This Method is automatically called whenever AddLogEntry() is called.  Path of the Log File |

GameManager

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| GameLogger | Logs Messages and Exceptions from other Classes.  Needs to be instantiated. (How-To in the Code Conventions Doc / Class) | |
|  | **Method** | **Description** |
| Event Function | Private void Start() | Start of the GameManager. |
| Click\_Event | NewGame\_Click() | Opens the character select window and sets the Image to false. |
| Click\_Event | StartNewGame\_Click() | Starts a new game and checks if a save slot is empty, else asks to override another slot. |
| Click\_Event | LoadGame\_Click() | Loads a saved Game. |
| Click\_Event | NextChapter\_Click() | When the next chapter Button is clicked. |
| Click\_Event | NextStory\_Click() | When the next story Button is clicked. |
| Click\_Event | BackToMainMenu\_Click() | When the back button is clicked. |

Game Data Controller

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| GameLogger | Logs Messages and Exceptions from other Classes.  Needs to be instantiated. (How-To in the Code Conventions Doc / Class) | |
|  | **Method** | **Description** |
|  | Public static Remove\_Data\_Click() | Deletes the selected Data. |
|  | Private static void SortFiles() | Sorts the Files after one was deleted. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| GameData | This class contains the Variables that are needed to save the status of the game. | |
| Object-Class | **Method** | **Description** |
|  | Public GameData()  SaveData saveData | Constructor to save an Object of type GameData.  Contains all necessary elements to save the status. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| SaveManager | This class returns the next choices nodes or the next story node according to the Story Files. Provides information about the next nodes.  Contains a lot of Getters like IsRootNode, IsEndOfChapter, IsGameOver, … | |
|  | **Method** | **Description** |
|  | Public class SaveData | Provides a temporary object to store and load the data |
| Event Function | Private void Awake() | Sets the language of the program to en-US |
| Event Function | Private void Start() | If there are any save files, then the LoadGame Button in the Menu is activated |
| Click\_Event | Public void NewGame\_Click() | When a new Game is started, it checks for a open save slot, if there aren’t any,  then the User has to choose an old save slot to override the date with the new Game |
| Click\_Event | Public void LoadDataIntoSlots\_Click() | When the LoadGame Button is clicked, then the save files getting loaded into the save slots,  Is there no save for a slot, then the slot stays empty |
|  | private static void LoadGame()  int slotNum | Loads the Clicked Game  Number of the clicked slot |
|  | Public static bool LoadData() | Returns true if Data has been loaded, else when not |
|  | Public static SaveData GetSaveData() | Returns the loaded Data |
|  | Publics static void SaveGame()  SaveData save | Saves the status of Game in a JSON File |

Story Dialogue

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryAsset | This class Is a Scriptable Object.  Creates the Story and Choice Nodes for the Editor.  Contains several Getters to get the Root node, all the nodes, the Child nodes, or the Choice nodes. | |
|  | **Method** | **Description** |
|  | Public StoryAsset ReadNodes()  StoryAsset chapter | Reads the Nodes from the Xml File and puts them in the right order.  Chapter to be read in. |
|  | Private void ReadProperties()  StoryNode node  XmlDocument xmlDoc | Reads the Properties from the Xml to the according node.  Node whose properties must be read.  The currently opened xml document. |
|  | Private static StoryNode CreateNode()  XmlNode node  string id  bool isChoice | Creates a new Node and a unique GUID Parent node to add the new child, Saves the label, id, text, and type of the node.  Name of the node.  Id Attribute in xml File  Declares if Node is a choice or not |
|  | Private void SaveNodesToAssetDatabase() | Adds to or removes from the asset database |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryViewer |  | |
|  | **Method** | **Description** |
|  | Public static bool OnOpenAsset()  Int instanceId | Shows the Editor Window, depending on if a Story is loaded or not. Returns the StoryAsset.  Instance ID of the Editor Window. |
| Event Function | Private void OnEnable() | When the Dialog is Enabled, initialize the node Styles. |
|  | Private void OnSelectionChanged() | Loads the selected chapter |
| Event Function | Private void OnGUI() | Gets called every time a change happens on the Editor. Displays the nodes. |
|  | Privates static DrawSurface() | Draws the Surface of the Editor Window. |
|  | Private bool DrawNode()  StoryNode node | Draws the nodes.  Next node to draw. |
|  | DrawConnections()  StoryNode node | Add Bezier Curve between the nodes to connect parent and child nodes. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryHolder | This class returns the next choices nodes or the next story node according to the Story Files. Provides information about the next nodes.  Contains a lot of Getters like IsRootNode, IsEndOfChapter, IsGameOver, … | |
|  | **Method** | **Description** |
|  | Public void LoadChapterProperties()  StoryAsset chapter | Loads the Save Data or Starts a new Chapter.  Is either null or a new chapter |
|  | Private void Next()  StoryNode node | Get next choice nodes.  Parent that contains the next choices nodes. |
| Overload #1 | Public void Next() | Get the next story node. |
|  | Public bool HasNext() | Returns true if the current nod has children. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryNode | Is a scriptable Object and contains Setter and Getter for the Node properties.  Maybe this class will be extended with more properties. | |
| Object-Class | **Properties** | **Description** |
|  | Public void AddChildNode() | Adds the node name to the childnodes list. |
|  | RemoveChildNode() | Removes node from childnodes. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryUI | Displays the Story  Takes the node information and puts in the Game objects so the Story can be played | |
|  | **Method** | **Description** |
|  | Private void Start() | When the Game starts, gets the story, adds the next Button click Event and updates the UI. |
| Click\_Event | Private void Next\_Clickt() | When the next button is clicked, it loads the next part of the story |
|  | Private void UpdateUI() | Updates the UI, loads the next story or choice nodes and their properties. Saves the node state. |
|  | Private void NextChapter() | Loads the next Chapter when the EndOfChapter node is reached, or the GameOver Screen when the GameOver node is reached |
|  | Private void BuildChoiceList() | Builds the choice list, depending on the count of the nodes. Some choices are only visible to players with the required background |

Item Scritableobject

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| Item | This class gives a scriptableobject that is then used in the inventory as an item. | |
|  | **Method** | **Description** |
|  | Public Class Item | ScriptableObject that provides data for item to be used in the inventory. |
| Click\_Event | Public virtual void Use() | Uses the item. At the moment it only debugs |
| Event Function | Public virtual string GetItemDescription | If there are any save files, then the LoadGame Button in the Menu is activated |
|  |  |  |

StatItem

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StatItem | This class gives a scriptableobject that is then used in the inventory as an item. | |
|  | **Method** | **Description** |
|  | Public Class StatItem | ScriptableObject that provides data for an item to be used in the inventory. This item type can give boons to the player such as health, food etc. Could be used in a later state of the game |
| Click\_Event | Public override void Use() | Uses the statitem to give stats to the player at a later stage. |
| Datatype | Public enum StatitemType | Defines statitem types such as healthitems, food items etc. |
|  |  |  |

ItemInfo

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| ItemInfo | This class returns the information about the item. Namely the item description as well as the item name. | |
|  | **Method** | **Description** |
|  | Public Class ItemInfo | This class returns the information about the item. Namely the item description as well as the item name. |
| Event Function | Public void SetUp()  \_itemName  \_itemDescription | Uses the statitem to give stats to the player at a later stage.  Return the name of the item.  Return the description of the item. |
|  |  |  |

ItemSlot

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| ItemSlot | Adds an item slot to the inventory windows. I this slot, an item is then used to be put into the item slot. | |
|  | **Method** | **Description** |
|  | Public Class ItemSlot | Adds an item slot to the inventory windows. I this slot, an item is then used to be put into the item slot. |
| Event Function | Public void AddItem()  newItem | Add items to the item slots.  The item that should be added to the slot. |
| Click\_Event | Public void UseItem() | Uses the item and gets the function from the item class. |
| Event Function | Public void DestroySlot() | Destroys item slot when the item is used or removed by the user. |
| Click\_Event | Public void OnRemoveButtonClicked() | This function removes an item from the inventory. |
| Hover\_Event | Public void OnCourserEnter() | This function activates the tooltip over an item, when it is hovered over it. |
| Hover\_Event | Public void OnCourserExit() | This function deactivates the tooltip over an item, when the courser is removed from said item. |
|  |  |  |

Controller

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryAsset | This class Is a Scriptable Object.  Creates the Story and Choice Nodes for the Editor.  Contains several Getters to get the Root node, all the nodes, the Child nodes, or the Choice nodes. | |
|  | **Method** | **Description** |
|  | Public StoryAsset ReadNodes()  StoryAsset chapter | Reads the Nodes from the Xml File and puts them in the right order  Chapter to be read in |
|  | Private void ReadProperties()  StoryNode node  XmlDocument xmlDoc | Reads the Properties from the Xml to the according node.  Node whose properties must be read.  The currently opened xml document. |
|  | Private static StoryNode CreateNode()  XmlNode node  string id  bool isChoice | Creates a new Node and a unique GUID Parent node to add the new child, Saves the label, id, text, and type of the node.  Name of the node.  Id Attribute in xml File  Declares if Node is a choice or not |
|  | Private void SaveNodesToAssetDatabase() | Adds to or removes from the asset database |