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](#_Toc147321640)

[TestClass 1](#_Toc147321641)

[Logger 4](#_Toc147321642)

[GameLogger 4](#_Toc147321643)

[GameManager 5](#_Toc147321644)

[GameLogger 5](#_Toc147321645)

[Game Data Controller 6](#_Toc147321646)

[DataRemover 6](#_Toc147321647)

[GameData 6](#_Toc147321648)

[InventoryData 6](#_Toc147321649)

[TimeAndProgress 6](#_Toc147321650)

[GameDataController 7](#_Toc147321651)

[Story Dialogue 8](#_Toc147321652)

[StoryAsset 8](#_Toc147321653)

[StoryViewer 8](#_Toc147321654)

[StoryHolder 9](#_Toc147321655)

[StoryNode 9](#_Toc147321656)

[StoryUI 9](#_Toc147321657)

[Inventory 10](#_Toc147321658)

[Item 10](#_Toc147321659)

[StatItem 10](#_Toc147321660)

[ItemInfo 10](#_Toc147321661)

[ItemSlot 10](#_Toc147321662)

[Controller 11](#_Toc147321663)

[FileController 11](#_Toc147321664)

[XmlController 11](#_Toc147321665)

[CharacterSelect 11](#_Toc147321666)

[ObjectRotation 11](#_Toc147321667)

[SaveSlotImage 12](#_Toc147321668)

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** | |
| GameManager | Is in Control of the Game, Story and handles the Scenes | |
|  | **Method** | **Description** |
| Event Function | Private void Start() | Start of the GameManager. More information in the class itself |
| Click\_Event | NewGame\_Click() | Opens the character select window and disables the select Images |
| Click\_Event | StartNewGame\_Click() | Checks if a character was selected and a Name was given.  Starts a new game and checks if a save placeholder is empty, else asks to override another placeholder. |
| 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** | |
| DataRemover | This function is only available in the Load Game Screen.  The Player can select a save slot and remove its data. | |
|  | **Method** | **Description** |
| Click\_Event | 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** | |
| InventoryData | This class saves and loads the Items from the Inventory. | |
|  | **Method** | **Description** |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| TimeAndProgress | Calculates the elapsed Time and Progress. | |
|  | **Method** | **Description** |
|  | Public static void StartTime() | Starts the Time. |
|  | Public static void StopTime() | Stops the Time. |
|  | GetElapsedTime() | Returns the elapsed Time. |
|  | CalculateProgress()  string chapter | Calculates the Progress.  The chapter is used to calculate the progress. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| GameDataController | This class loads and saves the status of the Game. | |
|  | **Method** | **Description** |
|  | Public class SaveData() | Provides a temporary object to store and load the data |
| Event Function | Private void Start() | If there are any save files, then the LoadSaveGame Button in the Menu is activated. |
|  | Public bool NewGame() | When a new Game is started, it checks for a open save slot, are there none, then the User has to choose an old save slot to override the date with the new Game.  Returns false, when no open slot is left and true if there is at least one open. |
|  | Public void LoadGame() | Sets the SaveScreen and loads the data. |
| Click\_Event | Public void LoadGame\_Click() | Loads the selected Game. |
| Click\_Event | Public void Continue\_Click() | When continue is clicked, the User can select a save slot to override the old data. |
| Click\_Event | Public void Cancel\_Click() | The MessageBox closes. |
| Click\_Event | Public void BackToMenu\_Click() | To get back to the main menu. |
|  | Private void LoadDataIntoSlots() | 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 void UpdateSlotView()  int slotNum | Updates the Slot view with loaded data.  Slot number where the save data has to be placed. |
|  | Public void UpdateEmptySlot() | Updates the Slot view with empty data, if there is no save. |
|  | Private static void LoadSelectedGame() | Loads the Selected Game. |
|  | Private void SaveNewGame() | Saves the name and the Character Properties as first Save for a New Game. |
|  | Public void SaveGame()  SaveData save | Saves the status of the Game each time the Player selects a choice or clicks the next Button. This is the Autosave function.  The SaveData sent from StoryUI. |

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 |

Inventory

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| Item | This class gives a Scriptable object that is then used in the inventory as an item. | |
|  | **Method** | **Description** |
| 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 |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StatItem | This item type can give boons to the player such as health, food etc. Could be used in a later state of the game | |
|  | **Method** | **Description** |
| 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 health items, food items etc. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| ItemInfo | This class returns the information about the item. Namely the item description as well as the item name. | |
|  | **Method** | **Description** |
| Event Function | Public void SetUp()  Text itemName  Text \_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. |

|  |  |  |
| --- | --- | --- |
| **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** |
| Event Function | Public void AddItem()  Item 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** | |
| FileController | Controls the CRUD of Files and Folders | |
|  | **Method** | **Description** |
|  | Public static void CreateFolders() | Creates Folders  SaveData: stores the Save Data from the Game  StoryAssets: stores the node Information as Json files |
|  | Public static void SortSaveFilest() | Sorts and renames the files after one was deleted |
|  | Public static void DeleteFile()  string path | Deletes a specific file.  path to the file, that needs to be deleted |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| XmlController | This class contains Methods to access Xml Files.  Returns XmlDocument, XmlReader, XmlWriter, XmlNode, Node InnerText or Content. | |
|  | **Method** | **Description** |
|  | Public static string GetMessageBoxText()  int index | Reads the MessageBox Nodes from the OutsourcedStrings/strings.xml  Determines the node to return. |
|  | Public static string GetInformationText()  int index | Reads the Information Nodes from the OutsourcedStrings/strings.xml  Determines the node to return. |
|  | Public static string GetChapterTitle()  StoryAsset currentChapter | Reads the title node from the current StoryAssets/StoryChapter.xml  Current Chapter that is loaded |
|  | Public static XmlDocument GetXmlDocOfStoryFile()  string chapter | Reads the Node information of the chapter  The current chapter that need's to be created |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| CharacterSelect-Controller | When a Character is selected. | |
|  | **Method** | **Description** |
| Click\_Event | Public void Character\_Click() | Loads the Character name in the invisible label, so the GameDataController knows which character to save. |
|  | Private void SetImage() | Sets the select Image, when a character is selected. |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| ObjectRotation-Controller | Rotates an Element around the Z Axis. | |
|  | **Method** | **Description** |
| Event Function | Private void Update() | Every frame the Game Object gets rotated |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| SaveSlotImage-Controller | When a save slot is clicked. | |
|  | **Method** | **Description** |
| Click\_Event | GameObject\_Click() | Sets the select Image, when a save slot is selected. |