Class Documentation for «Project TextingSpree»

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 «Project TextingSpree» 1](#_Toc124275997)

[TestClass 1](#_Toc124275998)

[Logger 3](#_Toc124275999)

[GameLogger 3](#_Toc124276000)

[Story 4](#_Toc124276001)

[Story 4](#_Toc124276002)

[StoryEditor 4](#_Toc124276003)

[StoryHolder 5](#_Toc124276004)

[StoryNode 5](#_Toc124276005)

[StoryUI 5](#_Toc124276006)

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 |

Story

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| Story | This class Is a Scriptable Object and implements the ISerializationCallbackReceiver.  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** |
|  | Private void Awake() | Is called every time a Scriptable Object is loaded. |
|  | Private void OnValidate()  string classname | When Exporting the Game, OnValidate() needs to be called from Awake() Method. |
|  | Public void AddNode()  StoryNode parentNode  Bool isChoice | Adds a new node, either Choice or Story node  Parent node to add the new child  Declares if the node to add is a story node or a Choice node |
|  | Private static StoryNode CreateNode()  StoryNode parentNode  Bool isChoice | Creating a new node by setting the name, the position and add it to the parent  Parent node to add the new child  Declares if the node to add is a story node or a Choice node |
|  | Public void DeleteNode()  StoryNode nodeToDelete | Delete selected node  Node to delete |
|  | Public void OnBeforeSerialize() | Method from ISerializationCallbackReceiver  Creates the first node |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryEditor |  | |
|  | **Method** | **Description** |
|  | Public static OnOpenAsset()  Int instanceId | Shows the Editor Window, depending if a Story is loaded or not  Instance ID of the Editor Window. |
|  | Private void OnSelectionChanged() | Loads the selected chapter |
| Event Function | Private void OnGUI() | Gets called every time a change happens on the Editor. Creates and deletes the nodes. |
|  | Privates static DrawSurface() | Draws the Surface of the Editor Window. |
|  | Private bool DrawNode()  StoryNode node | Reads the node information from the corresponding xml file. Either creates story or choice nodes. |
|  | Private void SetProperties()  StoryNode node | Reads the properties of the Story nodes in the corresponding xml file.  Properties like: (Image, End Of Chapter, Game over, …) |
|  | DrawLinkButtons()  StoryNode node | Draws the Buttons to link or unlink the nodes. |
|  | DrawConnections()  StoryNode node | Draws the connections between the 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** |
| Event Function | Private Start()  string classname | If there is loaded data, then it loads this data. Else this function loads the new Chapter of the Scene. |
|  | Private void Next()  StoryNode node | Gets next choice next choices nodes  Parent nodes that contains the next choice nodes |
| Overload #1 | Public void Next() | Gets the next story node |
|  | Public bool HasNext() | This method returns true when the parent node has some child nodes, and returns false when not |

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| StoryNode | Contains Setter and Getter for the Node properties.  Maybe this class will be extended with more properties. | |
| Object-Class | **Properties** | **Description** |
|  | String text | Contains the text for each node |
|  | Bool isChoiceNode | Node is a choice node |
|  | Bool isRootNode | Node is a root node |
|  | Bool IsEndOfChapter | Node is at the end of the chapter |
|  | Bool isGameOver | Node is a game over node |
|  | String image | Only the story node can hold images, saves the name (without the type, of the picture as a string |
|  | List<string>childNodes | Contains the childNodes for every node |
|  | Rect storyRect | Rectangle that contains the button and node information |

|  |  |  |
| --- | --- | --- |
| **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 add calls UpdateUI() |
| 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 Story, loads the next part of story and the choices nodes, after that it saves the actual node and their information |
|  | 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 |

SaveManager / Load

|  |  |  |
| --- | --- | --- |
| **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 |

|  |  |  |
| --- | --- | --- |
| **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 |