Custom Quest

Random Dragon Games
Version 1.0
Wed Jul 26 2017

Table of Contents

Table of contents

Welcome to Custom Quests' documentations and guides

Welcome to Custom **Quest**' documentations and guides. If you want to see a quick start guide for getting started, go to `http://randomdragongames.com/games/custom-quest-3/`

Namespace Index

Packages

Here are the packages with brief descriptions (if available):

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Namespace Documentation

CustomQuest Namespace Reference

The custom quest namespace. Contains a series of enums, used in the different classes: "criteriaType", "rewardType", "editorRewardType", "editorCriteriaType"

Classes

1 class **CustomQuestSettings**

A static class, used for controlling the different settings in the custom quest extension. Also used for this purpose, is the SettingsHolder **Enumerations**

enum **criteriaType** { **Kill**, **Gather**, **Deliver** }*Different types of criterias If you make a new type, add it here*

enum **rewardType** { **Resource**, **Item** }*Different types of rewards If you make a new type, add it here*

enum **editorRewardType** { **Standard**, **Criteria**, **Optional** }*Different type of editor reward types. Used to determin, where in the editor a reward should be visible, and which list it should be added to*

enum **editorCriteriaType** { **Standard**, **Criteria**, **Optional** }*Different type of editor criteria types. Used to determin, where in the editor a criteria should be visible, and which list it should be added to*

Detailed Description

The custom quest namespace. Contains a series of enums, used in the different classes: "criteriaType", "rewardType", "editorRewardType", "editorCriteriaType"

Enumeration Type Documentation

enum CustomQuest.criteriaType `[strong]`

Different types of criterias If you make a new type, add it here

enum CustomQuest.editorCriteriaType `[strong]`

Different type of editor criteria types. Used to determin, where in the editor a criteria should be visible, and which list it should be added to

enum CustomQuest.editorRewardType `[strong]`

Different type of editor reward types. Used to determin, where in the editor a reward should be visible, and which list it should be added to

enum CustomQuest.rewardType `[strong]`

Different types of rewards If you make a new type, add it here

UnityEngine Namespace Reference

Namespaces

UnityEngine.UI Namespace Reference

Namespaces

UnityEngine.UI.Extensions Namespace Reference

Classes

2 class **UIPolygon**

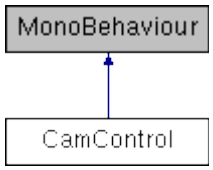
*Credit CiaccoDavide Sourced from - http://ciaccodavi.de/unity/uipolygon Used for Custom **Quest** 28-04-2017*

Class Documentation

CamControl Class Reference

Controls an object to follow a target, with a distance behind and above. Made to make a camera follow a player

Inheritance diagram for CamControl:



Public Attributes

3 Transform **target**

The target the camera should follow

4 float **distanceBehind**

The distance behind the target the camera should be

5 float **distanceTop**

The distance above the target the camera should be

Detailed Description

Controls an object to follow a target, with a distance behind and above. Made to make a camera follow a player

---

Member Data Documentation

float CamControl.distanceBehind

The distance behind the target the camera should be

float CamControl.distanceTop

The distance above the target the camera should be

Transform CamControl.target

The target the camera should follow

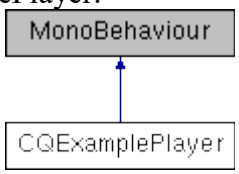
The documentation for this class was generated from the following file:

6 Assets/CustomQuest/Assets/Scripts/Demo Scripts/CamControl.cs

CQExamplePlayer Class Reference

Contains a sample player logic to: Move with arrow keys or "WASD". Opening and using the quest wheel. Clicking on a quest giver And picking up an item

Inheritance diagram for CQExamplePlayer:



Public Member Functions

7 void **Movement** ()

8 void **pickUpItem** (**Item** item)

Logic for picking up and item

Public Attributes

9 float **movementSpeed**

10 float **rotationSpeed**

11 float **resources**

Float for resources, use your own resources here.

12 List< **Item** > **items** = new List<**Item**>()

A list of the items this player has picked up

13 bool **attacking**

Whether this player is currently attacking or not

Properties

14 int **Damage** `[get, set]`

The dmg this player does

Detailed Description

Contains a sample player logic to: Move with arrow keys or "WASD". Opening and using the quest wheel. Clicking on a quest giver And picking up an item

---

Member Function Documentation

void CQExamplePlayer.pickUpItem (Item *item*)

Logic for picking up and item

Parameters:

item	The item picked up by the player

Member Data Documentation

bool CQExamplePlayer.attacking

Whether this player is currently attacking or not

List<Item> CQExamplePlayer.items = new List<Item>()

A list of the items this player has picked up

float CQExamplePlayer.resources

Float for resources, use your own resources here.

Property Documentation

int CQExamplePlayer.Damage `[get], [set]`

The dmg this player does

The documentation for this class was generated from the following file:

15 Assets/CustomQuest/Assets/Scripts/Demo Scripts/CQExamplePlayer.cs

CQPlayerObject Class Reference

A component which should be added to the games player

Inheritance diagram for CQPlayerObject:

Detailed Description

A component which should be added to the games player

The documentation for this class was generated from the following file:

16 Assets/CustomQuest/Assets/Scripts/CQPlayerObject.cs

CQUnitObject Class Reference

Inheritance diagram for CQUnitObject:

The documentation for this class was generated from the following file:

17 Assets/CustomQuest/Assets/Scripts/CQUnitObject.cs

Criteria Class Reference

A criteria for a quest Contains amount, type, name, individual player progressing, spawnzones, rewards and many other options

Inheritance diagram for Criteria:

Public Member Functions

18 virtual void **Start** ()

Use this for initialization

19 virtual void **EditorStart** ()

Used to set values when converting to a custom criteria

20 virtual void **Update** ()

Runs once every frame

21 virtual void **StartCriteria** (**CQPlayerObject** player)

Starts the time and spawn on this criteria

22 virtual void **Progress** (**CQPlayerObject** player, **CQExamplePlayer** unit)

Updates the player progress.

23 virtual void **Remove** (GameObject obj)

Removes an object from this criterias spawnedObjects lists, and destroys it

24 virtual void **Complete** (**CQPlayerObject** player, **CQExamplePlayer** unit)

Is run when this criteria is completed

25 virtual void **Fail** (**CQPlayerObject** player)

Is run when this criteria fails

Public Attributes

26 int **amount**

The amount of objects the player have to kill to complete the quest.

27 string **criteriaName**

*Name of the **Criteria***

28 **criteriaType type**

*Type of the **Criteria***

29 GameObject **criteriaObject**

*The **Criteria** Object.*

30 Dictionary< **CQPlayerObject**, int > **playerProgression** = new Dictionary<**CQPlayerObject**, int>()

Dictionary over the players progression.

31 List< **SpawnZone** > **spawnZones** = new List<**SpawnZone**>()

A list of spawnZones for this criteria

32 List< **Reward** > **rewards** = new List<**Reward**>()

The rewards this criteria contains

33 bool **giveRewardsOnCompletion**

If true, this criteria will give its reward as soon as the criteria is completed. Otherwise, the reward is given when the quest is completed

34 bool **timed**

If this criteria is timed

35 bool **dontDespawnObjectsWhenComplete**

Dont despawns all the spawned objects for this criteria when its completed

36 float **time**

If its timed, how long (in seconds)

Properties

37 **Quest Quest** `[get, set]`

*A reference to the **Quest** script.*

38 int **Level** `[get, set]`

An int controlling when this criteria is avalible for completion. All level '0' will have to be

completed, before level '1' will activate, and so on. It have a hard limit of 100.

Detailed Description

A criteria for a quest Contains amount, type, name, individual player progressing, spawnzones, rewards and many other options

---

Member Function Documentation

virtual void Criteria.Complete (CQPlayerObject *player*, CQExamplePlayer *unit*) `[virtual]`

Is run when this criteria is completed

Parameters:

player	The player who completed the criteria

virtual void Criteria.EditorStart () `[virtual]`

Used to set values when converting to a custom criteria

virtual void Criteria.Fail (CQPlayerObject *player*) `[virtual]`

Is run when this criteria fails

virtual void Criteria.Progress (CQPlayerObject *player*, CQExamplePlayer *unit*) `[virtual]`

Updates the player progress.

Parameters:

player	

virtual void Criteria.Remove (GameObject *obj*) `[virtual]`

Removes an object from this criterias spawnedObjects lists, and destroys it

Parameters:

obj	The object to remove and destroy

virtual void Criteria.Start () [virtual]

Use this for initialization

virtual void Criteria.StartCriteria (CQPlayerObject *player*) [virtual]

Starts the time and spawn on this criteria

virtual void Criteria.Update () [virtual]

Runs once every frame

Member Data Documentation

int Criteria.amount

The amount of objects the player have to kill to complete the quest.

string Criteria.criteriaName

Name of the **Criteria**

GameObject Criteria.criteriaObject

The **Criteria** Object.

bool Criteria.dontDespawnObjectsWhenComplete

Dont despawns all the spawned objects for this criteria when its completed

bool Criteria.giveRewardsOnCompletion

If true, this criteria will give its reward as soon as the criteria is completed. Otherwise, the reward is given when the quest is completed

Dictionary<CQPlayerObject, int> Criteria.playerProgression = new Dictionary<CQPlayerObject, int>()

Dictionary over the players progression.

List<Reward> Criteria.rewards = new List<Reward>()

The rewards this criteria contains

List<SpawnZone> Criteria.spawnZones = new List<SpawnZone>()

A list of spawnZones for this criteria

float Criteria.time

If its timed, how long (in seconds)

bool Criteria.timed

If this criteria is timed

criteriaType Criteria.type

Type of the **Criteria**

Property Documentation

int Criteria.Level `[get], [set]`

An int controlling when this criteria is avalible for completion. All level '0' will have to be completed, before level '1' will activate, and so on. It have a hard limit of 100.

Quest Criteria.Quest `[get], [set]`

A reference to the **Quest** script.

The documentation for this class was generated from the following file:

39 Assets/CustomQuest/Assets/Scripts/Criteria.cs

CriteriaPrefablDeleteWindow Class Reference

An editor window used to display a confirmation pop up, when deleting a criteria.

Inheritance diagram for CriteriaPrefablDeleteWindow:

Public Member Functions

40 void **SetQuestEditor** (**CustomQuestEditor** editor, **Criteria** c)

Sets the editor controlling this window, and the criteria about to be deleted

Properties

41 static **CriteriaPrefablDeleteWindow Instance** `[get]`

Detailed Description

An editor window used to display a confirmation pop up, when deleting a criteria.

Member Function Documentation

void CriteriaPrefablDeleteWindow.SetQuestEditor (CustomQuestEditor *editor*, Criteria *c*)

Sets the editor controlling this window, and the criteria about to be deleted

Parameters:

editor	The editor which spawned this window
c	The criteria about to be deleted

The documentation for this class was generated from the following file:

42 Assets/CustomQuest/Assets/Scripts/Editor/CriteriaPrefablDeleteWindow.cs

CriteriaUILogic Class Reference

The UI logic holder for a criteria. Used by UI logic.

Inheritance diagram for CriteriaUILogic:

Public Member Functions

43 void **Start** ()

Use this for initialization

Public Attributes

44 **Criteria criteria**
45 Text **criteriaName**
46 Text **amountDone**
47 Text **slash**
48 Text **totalAmount**
49 Text **criteriaType**
50 RectTransform **rectTransform**
51 bool **completed**

Detailed Description

The UI logic holder for a criteria. Used by UI logic.

Member Function Documentation

void CriteriaUILogic.Start ()

Use this for initialization

The documentation for this class was generated from the following file:

52 Assets/CustomQuest/Assets/Scripts/CriteriaUILogic.cs

CustomCriteriaPopUp Class Reference

An editor window used to display a confirmation pop up, when converting a criteria.

Inheritance diagram for CustomCriteriaPopUp:

Public Member Functions

53 void **SetQuestEditor** (**CustomQuestEditor** editor, **Criteria** c)

Sets the quest editor and the criteria about to be converted

Properties

54 static **CustomCriteriaPopUp Instance** `[get]`

Detailed Description

An editor window used to display a confirmation pop up, when converting a criteria.

Member Function Documentation

void CustomCriteriaPopUp.SetQuestEditor (CustomQuestEditor *editor*, Criteria *c*)

Sets the quest editor and the criteria about to be converted

Parameters:

editor	The editor who spawned this window
c	The criteria being converted

The documentation for this class was generated from the following file:

55 Assets/CustomQuest/Assets/Scripts/Editor/CustomCriteriaPopUp.cs

CustomQuestEditor Class Reference

The editor showing all information about quest, criterias and rewards in the inspector. Requires the quest component to show any info.

Inheritance diagram for CustomQuestEditor:

Public Member Functions

56 void **OnHierarchyChange** ()

Runs when a change in the hierarchy is made

57 void **CopyAll< T >** (T source, T target)
58 void **ConvertToCustomQuest** (**Quest** q)
59 void **DeleteQuest** (**Quest** q)
60 void **DeleteCriteriaPrefab** (**Criteria** c)
61 void **ConvertToCustomCriteria** (**Criteria** c)
62 void **DeleteRewardPrefab** (**Reward** r)
63 void **ConvertToCustomReward** (**Reward** r, **Quest** q)

Static Public Member Functions

64 static void **OpenQuestSystem** ()

Opens the editor.

65 static void **DrawCurves** (Rect startRect, Rect endRect, Color color)

Public Attributes

66 **Quest selectedQuest**

The currently selected quest

67 **Criteria selectedCriteria**

The currently selected criteria

68 **Reward selectedReward**

The currently selected reward

69 List< **Quest** > **allQuests** = new List<**Quest**>()

A list of all the quests prefabs

70 List< **Criteria** > **allCriterias** = new List<**Criteria**>()

A list of all the criteria prefabs

71 List< **Reward** > **allRewards** = new List<**Reward**>()

A list of all the reward prefabs

72 bool **deletingNode**

Bool used to determine if a node should be deleted

73 **QuestNode nodeToDelete**

The node to delete if the deletingNode bool is true

74 ReorderableList **R_questPrefabList** = null

The reordable list of the quest prefabs

75 ReorderableList **R_questInSceneList** = null

The reordable list of the quest prefabs, used in quest in scene

76 ReorderableList **R_criteriaPrefabList** = null

The reordable list of the criteria prefabs

77 ReorderableList **R_rewardPrefabList** = null

The reordable list of the reward prefabs

78 ReorderableList **R_CriteriaList** = null

A reordable list of the criterias for a quest

79 ReorderableList **R_RewardList** = null

A reordable list of the rewards for a quest

80 ReorderableList **R_OptionalCriteriaList** = null

A reordable list of the optional criterias for a quest

81 ReorderableList **R_OptionalRewardsList** = null

A reordable list of the optional rewards for a quest

82 GUISkin **thisGUISkin**

83 Texture **background**

Protected Attributes

84 GUIStyle **boldStyle**

A guistyle, used for to make tekst bold

85 GUIStyle **headLineStyle**

A guistyle, used for big headlines

Detailed Description

The editor showing all information about quest, criterias and rewards in the inspector. Requires the quest component to show any info.

Member Function Documentation

void CustomQuestEditor.OnHierarchyChange ()

Runs when a change in the hierarchy is made

static void CustomQuestEditor.OpenQuestSystem () [static]

Opens the editor.

Member Data Documentation

List<Criteria> CustomQuestEditor.allCriterias = new List<Criteria>()

A list of all the criteria prefabs

List<Quest> CustomQuestEditor.allQuests = new List<Quest>()

A list of all the quests prefabs

List<Reward> CustomQuestEditor.allRewards = new List<Reward>()

A list of all the reward prefabs

GUIStyle CustomQuestEditor.boldStyle [protected]

A guistyle, used for to make tekst bold

bool CustomQuestEditor.deletingNode

Bool used to determine if a node should be deleted

GUIStyle CustomQuestEditor.headLineStyle `[protected]`

A guistyle, used for big headlines

QuestNode CustomQuestEditor.nodeToDelete

The node to delete if the deletingNode bool is true

ReorderableList CustomQuestEditor.R_CriteriaList = null

A reordable list of the criterias for a quest

ReorderableList CustomQuestEditor.R_criteriaPrefabList = null

The reordable list of the criteria prefabs

ReorderableList CustomQuestEditor.R_OptionalCriteriaList = null

A reordable list of the optional criterias for a quest

ReorderableList CustomQuestEditor.R_OptionalRewardsList = null

A reordable list of the optional rewards for a quest

ReorderableList CustomQuestEditor.R_questInSceneList = null

The reordable list of the quest prefabs, used in quest in scene

ReorderableList CustomQuestEditor.R_questPrefabList = null

The reordable list of the quest prefabs

ReorderableList CustomQuestEditor.R_RewardList = null

A reordable list of the rewards for a quest

ReorderableList CustomQuestEditor.R_rewardPrefabList = null

The reordable list of the reward prefabs

Criteria CustomQuestEditor.selectedCriteria

The currently selected criteria

Quest CustomQuestEditor.selectedQuest

The currently selected quest

Reward CustomQuestEditor.selectedReward

The currently selected reward

The documentation for this class was generated from the following file:

86 Assets/CustomQuest/Assets/Scripts/Editor/CustomQuestEditor.cs

CustomQuestPopUp Class Reference

An editor window used to display a confirmation pop up, when converting a quest.

Inheritance diagram for CustomQuestPopUp:

Public Member Functions

87 void **SetQuestEditor** (**CustomQuestEditor** editor)
Sets the quest editor

Properties

88 static **CustomQuestPopUp Instance** `[get]`

Detailed Description

An editor window used to display a confirmation pop up, when converting a quest.

Member Function Documentation

void CustomQuestPopUp.SetQuestEditor (CustomQuestEditor *editor*)

Sets the quest editor

Parameters:

editor	The editor who spawned this window

The documentation for this class was generated from the following file:

89 Assets/CustomQuest/Assets/Scripts/Editor/CustomQuestPopUp.cs

CustomRewardPopUp Class Reference

An editor window used to display a confirmation pop up, when converting a reward.
Inheritance diagram for CustomRewardPopUp:

Public Member Functions

90 void **SetQuestEditor** (**CustomQuestEditor** editor, **Reward** r)
Sets the quest editor and the reward about to be converted

Properties

91 static **CustomRewardPopUp Instance** `[get]`

Detailed Description

An editor window used to display a confirmation pop up, when converting a reward.

Member Function Documentation

void CustomRewardPopUp.SetQuestEditor (CustomQuestEditor *editor*, Reward *r*)

Sets the quest editor and the reward about to be converted

Parameters:

editor	The editor who spawned this window
r	The reward being converted

The documentation for this class was generated from the following file:

92 Assets/CustomQuest/Assets/Scripts/Editor/CustomRewardPopUp.cs

Dokumentation Class Reference

The documentation for this class was generated from the following file:

93 Assets/CustomQuest/Assets/Scripts/Dokumentation.cs

Enemy Class Reference

An example enemy class. Will take damage when colliding with a player, and when its health is 0, will process the criteria its a part of (If its part of a criteria)

Inheritance diagram for Enemy:

Public Member Functions

94 void **OnTriggerEnter** (Collider other)

Runs when this objects trigger colliders with another

Public Attributes

95 int **health** = 100

The health of this enemy

Detailed Description

An example enemy class. Will take damage when colliding with a player, and when its health is 0, will process the criteria its a part of (If its part of a criteria)

Member Function Documentation

void Enemy.OnTriggerEnter (Collider *other*)

Runs when this objects trigger colliders with another

Parameters:

other	The other object colliding with this one

Member Data Documentation

int Enemy.health = 100

The health of this enemy

The documentation for this class was generated from the following file:

96 Assets/CustomQuest/Assets/Scripts/Demo Scripts/Enemy.cs

EventInfoHolder Class Reference

Public Attributes

97 **CQPlayerObject player**
98 **CQExamplePlayer unit**
99 **Quest quest**
100 **Criteria criteria**
101 **Reward reward**
102 GameObject **gameObject**
103 **SpawnZone spawnZone**
104 **QuestGiver questGiver**
105 float **f1**
106 float **f2**
107 int **i1**
108 int **i2**
109 string **s1**
110 string **s2**

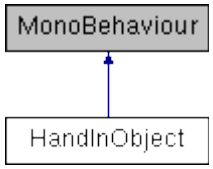
The documentation for this class was generated from the following file:

111 Assets/CustomQuest/Assets/Scripts/EventInfoHolder.cs

HandInObject Class Reference

Script used by handinobject, which is an object a player can hand its quest in at, if needed.

Inheritance diagram for HandInObject:



Public Member Functions

112 void **OnTriggerEnter** (Collider coll)

When a object collides with this object, it runs some checks to check if the its a player, and if it has completed all the needed criterias. If all that is good, it completed the quest.

Public Attributes

113 float **radius** = 3

The radius of the sphereCollider

114 SphereCollider **sphere**

The sphere collider of this handInObject

115 List< **Quest** > **quests** = new List<**Quest**>()

A list of quests, this is the handInObject of.

116 bool **handInByCollision** = true

If true, will run the attached quest's collision method, when gameobject collides with this object. Otherwise, another method will be needed to hand in the quest

Properties

117 float **Radius** `[get, set]`

Gets the radius of this handinobject, or sets it and the attached sqherecolliders radius

---

Detailed Description

Script used by handinobject, which is an object a player can hand its quest in at, if needed.

Member Function Documentation

void HandInObject.OnTriggerEnter (Collider *coll*)

When a object collides with this object, it runs some checks to check if the its a player, and if it has completed all the needed criterias. If all that is good, it completed the quest.

Parameters:

coll	The other collider

Member Data Documentation

bool HandInObject.handInByCollision = true

If true, will run the attached quest's collision method, when gameobject collides with this object. Otherwise, another method will be needed to hand in the quest

List<Quest> HandInObject.quests = new List<Quest>()

A list of quests, this is the handInObject of.

float HandInObject.radius = 3

The radius of the sphereCollider

SphereCollider HandInObject.sphere

The sphere collider of this handInObject

Property Documentation

float HandInObject.Radius `[get], [set]`

Gets the radius of this handinobject, or sets it and the attached sqherecolliders radius

---

The documentation for this class was generated from the following file:

118 Assets/CustomQuest/Assets/Scripts/HandInObject.cs

Item Class Reference

A sample class for an **Item**. When it collides with a player, the players pickUpItem method is run.

Inheritance diagram for Item:

Public Member Functions

119 void **OnTriggerEnter** (Collider other)

Is run when another trigger enters this gameobjects trigger

Detailed Description

A sample class for an **Item**. When it collides with a player, the players pickUpItem method is run.

Member Function Documentation

void Item.OnTriggerEnter (Collider *other*)

Is run when another trigger enters this gameobjects trigger

Parameters:

other	The other trigger colliding

The documentation for this class was generated from the following file:

120 Assets/CustomQuest/Assets/Scripts/Demo Scripts/Item.cs

Manager< T > Class Template Reference

A manager class, used to make a monobehavior into a singleton, so everyone can acces it from anywhere.

Inheritance diagram for Manager< T >:

Public Member Functions

121 virtual void **Awake** ()

Properties

122 static T **Instance** `[get]`

Detailed Description

A manager class, used to make a monobehavior into a singleton, so everyone can acces it from anywhere.

Template Parameters:

T	

Type Constraints

T : ***Manager***<*T*>

The documentation for this class was generated from the following file:

123 Assets/CustomQuest/Assets/Scripts/Manager.cs

NodeDeleteWindow Class Reference

A confirmation window for when deleting a quest in scene (A quest node)

Inheritance diagram for NodeDeleteWindow:

Public Member Functions

124 void **SetQuestEditor** (**CustomQuestEditor** editor, **QuestNode** qn)

Sets the quest editor and the quest node about to be deleted

Properties

125 static **NodeDeleteWindow Instance** `[get]`

Detailed Description

A confirmation window for when deleting a quest in scene (A quest node)

Member Function Documentation

void NodeDeleteWindow.SetQuestEditor (CustomQuestEditor *editor*, QuestNode *qn*)

Sets the quest editor and the quest node about to be deleted

Parameters:

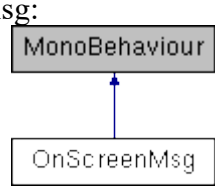
editor	The editor who spawned this window
qn	The quest node being deleted

The documentation for this class was generated from the following file:

126 Assets/CustomQuest/Assets/Scripts/Editor/NodeDeleteWindow.cs

OnScreenMsg Class Reference

Inheritance diagram for OnScreenMsg:



Properties

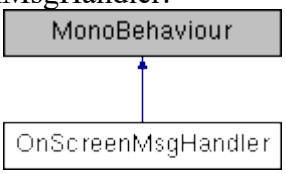
127 float **LifeTime** `[get, set]`
128 string **Msg** `[get, set]`
129 int **Size** `[get, set]`
130 Color **Color** `[get, set]`
131 Vector2 **MsgPosition** `[get, set]`

The documentation for this class was generated from the following file:
132 Assets/CustomQuest/Assets/Scripts/OnScreenMsg.cs

OnScreenMsgHandler Class Reference

A handler used for handling on screen messages

Inheritance diagram for OnScreenMsgHandler:



Public Member Functions

133 void **AddMsg** (float lifeTime, string msg, int size, Color color)
Adds an On Screen Msg

Properties

134 List< **OnScreenMsg** > **Msgs** `[get, set]`

Detailed Description

A handler used for handling on screen messages

Member Function Documentation

void OnScreenMsgHandler.AddMsg (float *lifeTime*, string *msg*, int *size*, Color *color*)

Adds an On Screen Msg

Parameters:

lifeTime	The time the msg should be displayed
msg	The msg to be shown
size	The size of text
color	The color of the text

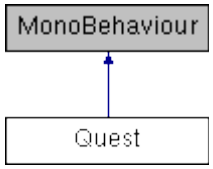
The documentation for this class was generated from the following file:

135 Assets/CustomQuest/Assets/Scripts/OnScreenMsgHandler.cs

Quest Class Reference

The quest script. This is the big one. Contains all information, lists and funktionality about a quest.

Inheritance diagram for Quest:



Public Member Functions

136 virtual void **Start** ()

Use this for initialization

137 virtual void **StartCriterias** ()

Adds the correct criterias to activeCriterias and other start related things

138 virtual void **EditorStart** ()

A start function used in the editor

139 virtual void **Update** ()

Update is called once per frame

140 virtual void **OnCompletion** (**CQPlayerObject** player, **CQExamplePlayer** unit)

*Run if **Quest** is completed*

141 virtual void **OnFail** (**CQPlayerObject** player)

*Run if **Quest** failed*

142 virtual void **CreateCriteria** (**Criteria** critera)

Adds a criteria to the quest

143 virtual void **DeleteCriteria** (**Criteria** criteria)

Deletes a specific criteria

144 virtual void **CreateReward** (**Reward** reward)

Adds a reward to the quest

145 virtual void **DeleteReward** (**Reward** reward)

Deletes a specifik reward script

146 virtual void **SpawnQuestObjects** (GameObject questObject, int objectAmount, **SpawnZone** zone)

*Spawns **Quest** Objects*

147 virtual void **CriteriaCompleted** (**CQExamplePlayer** unit, **CQPlayerObject** player, **Criteria** criteria)

*Completes a **Criteria**.*

148 virtual void **ProcessCriteria** (**CQExamplePlayer** unit, **CQPlayerObject** player, **Criteria** criteria)

Processes a criteria

149 virtual void **ProcessOptionalCriteria** (**CQExamplePlayer** unit, **CQPlayerObject** player, **Criteria** criteria)

Processes an optional criteria

150 virtual void **CriteriaFailed** (**CQPlayerObject** player, **Criteria** criteria)

Run this to fail a criteria

151 virtual void **GiveReward** (**CQExamplePlayer** unit, **CQPlayerObject** player, List< **Reward** > **rewards**)

*Give's the **Reward** when the player completes the quest.*

152 virtual void **UnlockQuest** (**Quest** quest, **CQPlayerObject** player)

Unlocks the quest if no quests are left uncompleted.

153 virtual void **StartSpawning** (**CQPlayerObject** player)

Starts all the criterias spawn

154 virtual void **StopSpawning** ()

Stops all the criterias spawn

155 virtual void **ResetCriterias** (**CQPlayerObject** player)

Resest the criterias for this quest

156 virtual void **AddPlayerToCriterias** ()

*Adds players to the **Criteria** dictionaries, if they are not already there.*

Public Attributes

157 Sprite **questIcon**

The icon of the quest

158 string **questName**

*The name of the **Quest***

159 string **description**

*The description of the **Quest***

160 string **toolTip**

*The tooltip of the **Quest***

161 bool **startAvailability**

Sets whether the quest should be Available from start or not.

162 bool **constantAvailability**

Sets whether the quest always should be available to all players. Works best with repeaterable.

163 bool **autoComplete**

A bool whether the quest should auto complete when all criterias are done, or not

164 bool **questCompleted**

Whether the quest is completed or not

165 bool **repeatable**

If the quest can be picked up and completed by the same player, multiple times

166 float **repeatableTime**

The time before this quest is available again after completion in seconds, if repeatable is true

167 Dictionary< **CQPlayerObject**, float > **remainingRepeatableTime** = new Dictionary<**CQPlayerObject**, float>()

A dictonary of the remainig time on the induvidual players

168 bool **dontDelete**

If the quest should ever be deleted

169 bool **singleComplete**

If the quest can only be completed by one player, before its gone

170 bool **startSpawningOnDiscover**

If the quest should start spawning its criterias when its discovered

171 bool **noSpawnIfNoPlayer**

If the quest should stop spawning its criterias when no player has the quest

172 bool **timed**

If this quest is timed, and then fails if its time is up

173 bool **matchOptionalLevels**

If the quest should match criteria levels with optional criteria levels. So when a criteria level is done, optional criterias levels up aswell

174 bool **pickUpAble**

If the player is able to pick this quest up at a quest giver

175 float **time**

The time before this quest fails

176 List< **Criteria** > **criterias** = new List<**Criteria**>()

*A List of the **Quest**'s Criterias*

177 List< **Reward** > **rewards** = new List<**Reward**>()

*A list of the **Quest**'s Rewards*

178 List< int > **thresholds** = new List<int>()

The different thresholds for the different levels of quests

179 List< int > **optionalThresholds** = new List<int>()

The different thresholds for the different levels of optional quests

180 List< **Criteria** > **optionalCriterias** = new List<**Criteria**>()

A list of this quests optional criterias

181 List< **Reward** > **optionalRewards** = new List<**Reward**>()

A list of this quests optional rewards

182 Dictionary< **CQPlayerObject**, List< **Criteria** > > **activeOptionalCriterias** = new Dictionary<**CQPlayerObject**, List<**Criteria**>>()

Dictonary of each players active optionalCriterias for this quest

183 Dictionary< **CQPlayerObject**, List< **Criteria** > > **completedOptionalCriterias** = new Dictionary<**CQPlayerObject**, List<**Criteria**>>()

Dictonary of each players completed optionalCriterias for this quest

184 Dictionary< **CQPlayerObject**, List< **Criteria** > > **failedOptionalCriterias** = new Dictionary<**CQPlayerObject**, List<**Criteria**>>()

Dictonary of each players failed optionalCriterias for this quest

185 int **completedOptionalThreshold**

The threshold of when to give the optional rewards. (3 = one player must complete 3 optional criterias to get the bonus reward)

186 List< **Quest** > **unCompletedQuests** = new List<**Quest**>()

Quests that has to be completed before this quest activates.

187 List< **Quest** > **questsToUnlock** = new List<**Quest**>()

List to put the Quests you want to be able to use as chain quests.

188 Dictionary< **CQPlayerObject**, List< **Criteria** > > **unCompletedCriterias** = new Dictionary<**CQPlayerObject**, List<**Criteria**>>()

Dictionary of the not yet completed criterias.

189 Dictionary< **CQPlayerObject**, List< **Criteria** > > **completedCriterias** = new Dictionary<**CQPlayerObject**, List<**Criteria**>>()

Dictionary of the completed criterias.

190 Dictionary< **CQPlayerObject**, List< **Criteria** > > **activeCriterias** = new Dictionary<**CQPlayerObject**, List<**Criteria**>>()

A list of the active criterias on this quest

191 List< **HandInObject** > **handInObjects** = new List<**HandInObject**>()

A list of handInObjects this quest can hand in its quest to

192 List< **QuestGiver** > **questGivers** = new List<**QuestGiver**>()

A list of questGivers this quets has

193 Dictionary< **CQPlayerObject**, float > **remainingTime** = new Dictionary<**CQPlayerObject**, float>()

A dictonary of the remainig time on the induvidual players

194 List< **CQPlayerObject** > **playersUnCompleted** = new List<**CQPlayerObject**>()

A list of player currently on the quest

195 List< **CQPlayerObject** > **playersCompleted** = new List<**CQPlayerObject**>()

A list of the players who has already done the quest

Detailed Description

The quest script. This is the big one. Contains all information, lists and funktionality about a quest.

Member Function Documentation

virtual void Quest.AddPlayerToCriterias () `[virtual]`

Adds players to the **Criteria** dictionaries, if they are not already there.

virtual void Quest.CreateCriteria (Criteria *critera*) `[virtual]`

Adds a criteria to the quest

Parameters:

critera	The criteria to add

virtual void Quest.CreateReward (Reward *reward*) `[virtual]`

Adds a reward to the quest

Parameters:

reward	The reward to be added

virtual void Quest.CriteriaCompleted (CQExamplePlayer *unit*, CQPlayerObject *player*, Criteria *criteria*) `[virtual]`

Completes a **Criteria**.

Parameters:

player	The player who completed it
criteria	The criteria to complete
unit	The unit who completed the criteria

virtual void Quest.CriteriaFailed (CQPlayerObject *player*, Criteria *criteria*) `[virtual]`

Run this to fail a criteria

Parameters:

critera	The criteria to fail
player	The player who failed this criteria

virtual void Quest.DeleteCriteria (Criteria *criteria*) `[virtual]`

Deletes a specific criteria

Parameters:

criteria	The criteria to be deleted

virtual void Quest.DeleteReward (Reward *reward*) `[virtual]`

Deletes a specifik reward script

Parameters:

reward	The reward to be deleted

virtual void Quest.EditorStart () `[virtual]`

A start function used in the editor

virtual void Quest.GiveReward (CQExamplePlayer *unit*, CQPlayerObject *player*, List< Reward > *rewards*) `[virtual]`

Give's the **Reward** when the player completes the quest.

Parameters:

unit	The unit who completed the quest
player	The player who completed the quest

virtual void Quest.OnCompletion (CQPlayerObject *player*, CQExamplePlayer *unit*) `[virtual]`

Run if **Quest** is completed

Parameters:

player	The player completing the quest
unit	The unit completing the quest

virtual void Quest.OnFail (CQPlayerObject *player*) `[virtual]`

Run if **Quest** failed

Parameters:

player	The player failing the quest

virtual void Quest.ProcessCriteria (CQExamplePlayer *unit*, CQPlayerObject *player*, Criteria *criteria*) `[virtual]`

Processes a criteria

Parameters:

unit	The unit who completed the criteria
player	The player who completed the criteria
criteria	The criteria which is completed

virtual void Quest.ProcessOptionalCriteria (CQExamplePlayer *unit*, CQPlayerObject *player*, Criteria *criteria*) `[virtual]`

Processes an optional criteria

Parameters:

unit	The unit who completed the criteria
player	The player who completed the criteria
criteria	The criteria which is completed

virtual void Quest.ResetCriterias (CQPlayerObject *player*) `[virtual]`

Resest the criterias for this quest

Parameters:

player	The player to reset the criterias for

virtual void Quest.SpawnQuestObjects (GameObject *questObject*, int *objectAmount*, SpawnZone *zone*) `[virtual]`

Spawns **Quest** Objects

Parameters:

questObject	The object to spawn
objectAmount	The amount to spawn
criteria	The criteria spawning them

virtual void Quest.Start () `[virtual]`

Use this for initialization

virtual void Quest.StartCriterias () `[virtual]`

Adds the correct criterias to activeCriterias and other start related things

virtual void Quest.StartSpawning (CQPlayerObject *player*) `[virtual]`

Starts all the criterias spawn

Parameters:

player	The player starting the spawn

virtual void Quest.StopSpawning () `[virtual]`

Stops all the criterias spawn

virtual void Quest.UnlockQuest (Quest *quest*, CQPlayerObject *player*) `[virtual]`

Unlocks the quest if no quests are left uncompleted.

Parameters:

quest	The quest this quest removes from its unCompletedQuests
player	The player unlocking the quest

virtual void Quest.Update () `[virtual]`

Update is called once per frame

Member Data Documentation

Dictionary<CQPlayerObject, List<Criteria> > Quest.activeCriterias = new Dictionary<CQPlayerObject, List<Criteria>>()

A list of the active criterias on this quest

Dictionary<CQPlayerObject, List<Criteria> > Quest.activeOptionalCriterias = new Dictionary<CQPlayerObject, List<Criteria>>()

Dictonary of each players active optionalCriterias for this quest

bool Quest.autoComplete

A bool whether the quest should auto complete when all criterias are done, or not

Dictionary<CQPlayerObject, List<Criteria> > Quest.completedCriterias = new Dictionary<CQPlayerObject, List<Criteria>>()

Dictionary of the completed criterias.

Dictionary<CQPlayerObject, List<Criteria> > Quest.completedOptionalCriterias = new Dictionary<CQPlayerObject, List<Criteria>>()

Dictonary of each players completed optionalCriterias for this quest

int Quest.completedOptionalThreshold

The threshold of when to give the optional rewards. (3 = one player must complete 3 optional criterias to get the bonus reward)

bool Quest.constantAvailability

Sets whether the quest always should be available to all players. Works best with repeaterable.

List<Criteria> Quest.criterias = new List<Criteria>()

A List of the **Quest**'s Criterias

string Quest.description

The description of the **Quest**

bool Quest.dontDelete

If the quest should ever be deleted

//Is there to make sure a quest always stays around, even if all the players in the scene have completed it. What if a new player joins the room @ runtime?

Dictionary<CQPlayerObject, List<Criteria> > Quest.failedOptionalCriterias = new Dictionary<CQPlayerObject, List<Criteria>>()

Dictonary of each players failed optionalCriterias for this quest

List<HandInObject> Quest.handInObjects = new List<HandInObject>()

A list of handInObjects this quest can hand in its quest to

bool Quest.matchOptionalLevels

If the quest should match criteria levels with optional criteria levels. So when a criteria level is done, optional criterias levels up aswell

bool Quest.noSpawnIfNoPlayer

If the quest should stop spawning its criterias when no player has the quest

List<Criteria> Quest.optionalCriterias = new List<Criteria>()

A list of this quests optional criterias

List<Reward> Quest.optionalRewards = new List<Reward>()

A list of this quests optional rewards

List<int> Quest.optionalThresholds = new List<int>()

The different thresholds for the different levels of optional quests

bool Quest.pickUpAble

If the player is able to pick this quest up at a quest giver

List<CQPlayerObject> Quest.playersCompleted = new List<CQPlayerObject>()

A list of the players who has already done the quest

List<CQPlayerObject> Quest.playersUnCompleted = new List<CQPlayerObject>()

A list of player currently on the quest

bool Quest.questCompleted

Whether the quest is completed or not

List<QuestGiver> Quest.questGivers = new List<QuestGiver>()

A list of questGivers this quets has

Sprite Quest.questIcon

The icon of the quest

string Quest.questName

The name of the **Quest**

List<Quest> Quest.questsToUnlock = new List<Quest>()

List to put the Quests you want to be able to use as chain quests.

Dictionary<CQPlayerObject, float> Quest.remainingRepeatableTime = new Dictionary<CQPlayerObject, float>()

A dictonary of the remainig time on the induvidual players

Dictionary<CQPlayerObject, float> Quest.remainingTime = new Dictionary<CQPlayerObject, float>()

A dictonary of the remainig time on the induvidual players

bool Quest.repeatable

If the quest can be picked up and completed by the same player, multiple times

float Quest.repeatableTime

The time before this quest is available again after completion in seconds, if repeatable is true

List<Reward> Quest.rewards = new List<Reward>()

A list of the **Quest**'s Rewards

bool Quest.singleComplete

If the quest can only be completed by one player, before its gone

bool Quest.startAvailability

Sets whether the quest should be Available from start or not.

bool Quest.startSpawningOnDiscover

If the quest should start spawning its criterias when its discovered

List<int> Quest.thresholds = new List<int>()

The different thresholds for the different levels of quests

float Quest.time

The time before this quest fails

bool Quest.timed

If this quest is timed, and then fails if its time is up

string Quest.toolTip

The tooltip of the **Quest**

Dictionary<CQPlayerObject, List<Criteria> > Quest.unCompletedCriterias = new Dictionary<CQPlayerObject, List<Criteria>>()

Dictionary of the not yet completed criterias.

List<Quest> Quest.unCompletedQuests = new List<Quest>()

Quests that has to be completed before this quest activates.

The documentation for this class was generated from the following file:

196 Assets/CustomQuest/Assets/Scripts/Quest.cs

QuestCompass Class Reference

The quest compass script. Used for controlling the direction of the compass arrows. Gets an origin point, from the center of the camera.

Inheritance diagram for QuestCompass:

Public Member Functions

197 virtual void **Start** ()

Use this for initialization

198 virtual void **Update** ()

Is run every frame

Public Attributes

199 GameObject **origin**

Where we are looking from, or where the player is

200 Vector3 **targetLocation**

Where the compas is pointing from

201 bool **player**

Point from player, or middle of screen

Detailed Description

The quest compass script. Used for controlling the direction of the compass arrows. Gets an origin point, from the center of the camera.

Member Function Documentation

virtual void QuestCompass.Start () [virtual]

Use this for initialization

Reimplemented in **QuestCompassArrow** (*p.44*).

virtual void QuestCompass.Update () [virtual]

Is run every frame

Reimplemented in **QuestCompassArrow** (*p.44*).

Member Data Documentation

GameObject QuestCompass.origin

Where we are looking from, or where the player is

bool QuestCompass.player

Point from player, or middle of screen

Vector3 QuestCompass.targetLocation

Where the compas is pointing from

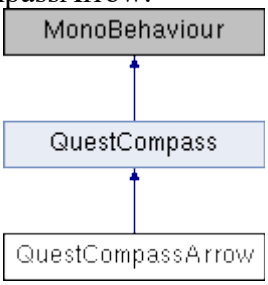
The documentation for this class was generated from the following file:

202 Assets/CustomQuest/Assets/Scripts/QuestCompass.cs

QuestCompassArrow Class Reference

A quest compass arrow, pointing at its target

Inheritance diagram for QuestCompassArrow:



Public Member Functions

203 override void **Start** ()

Use this for initialization

204 override void **Update** ()

Update is called once per frame

Public Attributes

205 Transform **target**

Where is the quest? - Where is the compass pointing? // gameobject or the like

---

Detailed Description

A quest compass arrow, pointing at its target

Member Function Documentation

override void QuestCompassArrow.Start () `[virtual]`

Use this for initialization

Reimplemented from **QuestCompass** (*p.43*).

override void QuestCompassArrow.Update () `[virtual]`

Update is called once per frame

Reimplemented from **QuestCompass** (*p.43*).

Member Data Documentation

Transform QuestCompassArrow.target

Where is the quest? - Where is the compass pointing? // gameobject or the like

---

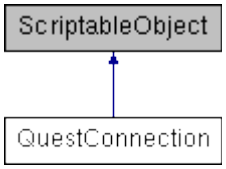
The documentation for this class was generated from the following file:

206 Assets/CustomQuest/Assets/Scripts/QuestCompassArrow.cs

QuestConnection Class Reference

A connection between two edges

Inheritance diagram for QuestConnection:



Public Attributes

207 **QuestEdge leftEdge**
208 **QuestEdge rightEdge**

Detailed Description

A connection between two edges

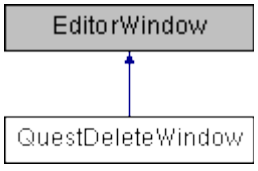
The documentation for this class was generated from the following file:

209 Assets/CustomQuest/Assets/Scripts/QuestConnection.cs

QuestDeleteWindow Class Reference

An editor window used to display a confirmation pop up, when deleting a quest.

Inheritance diagram for QuestDeleteWindow:



Public Member Functions

210 void **SetQuestEditor** (**CustomQuestEditor** editor, **Quest** q)
Sets the quest editor and the quest about to be converted

Properties

211 static **QuestDeleteWindow Instance** `[get]`

Detailed Description

An editor window used to display a confirmation pop up, when deleting a quest.

Member Function Documentation

void QuestDeleteWindow.SetQuestEditor (CustomQuestEditor *editor*, Quest *q*)

Sets the quest editor and the quest about to be converted

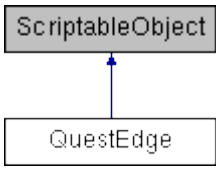
Parameters:

editor	The editor who spawned this window
q	The quest being converted

The documentation for this class was generated from the following file:
212 Assets/CustomQuest/Assets/Scripts/Editor/QuestDeleteWindow.cs

QuestEdge Class Reference

An edge on a quest
Inheritance diagram for QuestEdge:



Public Attributes

213 **QuestNode questNode**
The quest node this edge is on
214 List< **QuestConnection** > **connections** = new List<**QuestConnection**>()
The connections this edge has
215 bool **left**

If its on the left side of the node, or not. (Used for drawing lines correctly)
216 Rect **rect**
The rectangle of this edge

Detailed Description

An edge on a quest

Member Data Documentation

List<QuestConnection> QuestEdge.connections = new List<QuestConnection>()

The connections this edge has

bool QuestEdge.left

If its on the left side of the node, or not. (Used for drawing lines correctly)

QuestNode QuestEdge.questNode

The quest node this edge is on

Rect QuestEdge.rect

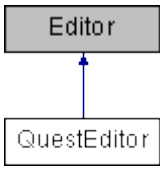
The rectangle of this edge

The documentation for this class was generated from the following file:
217 Assets/CustomQuest/Assets/Scripts/QuestEdge.cs

QuestEditor Class Reference

Contains the editor logic for displaying all the info about a quest. It also contains info about **Criteria** and **Reward** components attached to this quest. If you make new fields in the quest, which you want to acces through the inspector, this is where you would write a way to display it.

Inheritance diagram for QuestEditor:



Public Member Functions

218 override void **OnInspectorGUI** ()

Runs the GUI logic of the quest

219 void **CopyAll< T >** (T source, T target)

Public Attributes

220 ReorderableList **R_CriteriaList** = null
221 ReorderableList **R_RewardList** = null
222 ReorderableList **R_OptionalCriteriaList** = null
223 ReorderableList **R_OptionalRewardsList** = null

Protected Attributes

224 GUIStyle **foldOutStyle**
225 GUIStyle **headLineStyle**

Detailed Description

Contains the editor logic for displaying all the info about a quest. It also contains info about **Criteria** and **Reward** components attached to this quest. If you make new fields in the quest, which you want to acces through the inspector, this is where you would write a way to display it.

Member Function Documentation

override void QuestEditor.OnInspectorGUI ()

Runs the GUI logic of the quest

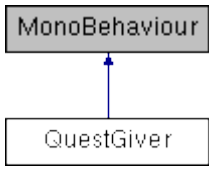
The documentation for this class was generated from the following file:

226 Assets/CustomQuest/Assets/Scripts/Editor/QuestEditor.cs

QuestGiver Class Reference

Quest giver script, contains functionality for a quest giver.

Inheritance diagram for QuestGiver:



Public Member Functions

227 void **OnTriggerEnter** (Collider coll)

*If a player collides with the **QuestGiver**, the questgiver will give the quest to the player by instantiating it.*

228 void **StartQuestPopUp** (**CQPlayerObject** player, **Quest** quest)

Starts the progress of a quest pop up.

Public Attributes

229 List< **Quest** > **quests** = new List<**Quest**>()

*A list of quests this **QuestGiver** can give*

230 float **radius** = 3

Radius the quest giver will give the quest in.

231 float **declineDistance** = 5

Radius player has to move away, before quest is considered declined

232 bool **walkIntoStartQuest** = true

If true, player will be able to pickup quest by walking within radius

233 GameObject **questSymbol**

The quest symbol over the quest giver

Detailed Description

Quest giver script, contains functionality for a quest giver.

Member Function Documentation

void QuestGiver.OnTriggerEnter (Collider *coll*)

If a player collides with the **QuestGiver**, the questgiver will give the quest to the player by instantiating it.

Parameters:

coll	The collider colliding with the questGiver

void QuestGiver.StartQuestPopUp (CQPlayerObject *player*, Quest *quest*)

Starts the progress of a quest pop up.

Parameters:

player	The player recieving the quest pop up

Member Data Documentation

float QuestGiver.declineDistance = 5

Radius player has to move away, before quest is considered declined

List<Quest> QuestGiver.quests = new List<Quest>()

A list of quests this **QuestGiver** can give

GameObject QuestGiver.questSymbol

The quest symbol over the quest giver

float QuestGiver.radius = 3

Radius the quest giver will give the quest in.

bool QuestGiver.walkIntoStartQuest = true

If true, player will be able to pickup quest by walking within radius

---

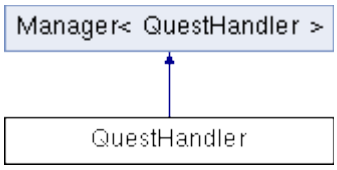
The documentation for this class was generated from the following file:

234 Assets/CustomQuest/Assets/Scripts/QuestGiver.cs

QuestHandler Class Reference

The quest handler, keeps track of players and quests. Can be accessed from anywhere, thanks to the **Manager**

Inheritance diagram for QuestHandler:



Public Member Functions

235 override void **Awake** ()

Awake is run before Start

236 void **QuestsDiscovered** (**Quest** quest, **CQPlayerObject** player)

Tells the quest system that a quest has been discovered by a player

237 void **FindQuests** ()

Finds all the quests in the scene, and adds them to the allQuests list

238 void **FindPlayers** ()

Finds all the players in the scene, and adds them to the players list

239 void **AddPlayer** (**CQPlayerObject** p)

Adds a player to the players list

Static Public Member Functions

240 static System.Type **GetComponentTypeByName** (string name)

Find a componet by name, and returns it

241 static void **StartListening** (string eventName, UnityAction< **EventInfoHolder** > listener)
242 static void **StopListening** (string eventName, UnityAction< **EventInfoHolder** > listener)
243 static void **TriggerEvent** (string eventName, **EventInfoHolder** e)

Public Attributes

244 List< **Quest** > **allQuests** = new List<**Quest**>()

A list over the quests currently in the scene.

245 List< **CQPlayerObject** > **players** = new List<**CQPlayerObject**>()

A list over the players currently in the scene. To add a new player, use "AddPlayer".

246 Dictionary< **CQPlayerObject**, List< **Quest** > > **availableQuests** = new Dictionary<**CQPlayerObject**, List<**Quest**>>()

A dictonary which contains all the players, and the quests they each have

247 **CQPlayerObject selectedPlayer**

The currently selected player. In networking enviroments, this would be the local client.

Properties

248 **CQPlayerObject SelectedPlayer** `[get, set]`

The properties for selectedPlayer - Tries to find a player, if its null

249 Dictionary< string, List< UnityAction< **EventInfoHolder** > > > **EventDictonary** `[get, set]`

The properties for EventDictonary - Creates a new dictionary, if its null

---

Detailed Description

The quest handler, keeps track of players and quests. Can be accessed from anywhere, thanks to the **Manager**

Member Function Documentation

void QuestHandler.AddPlayer (CQPlayerObject *p*)

Adds a player to the players list

Parameters:

p	The player to be added

override void QuestHandler.Awake () `[virtual]`

Awake is run before Start

Reimplemented from **Manager< QuestHandler >** (*p.27*).

void QuestHandler.FindPlayers ()

Finds all the players in the scene, and adds them to the players list

void QuestHandler.FindQuests ()

Finds all the quests in the scene, and adds them to the allQuests list

static System.Type QuestHandler.GetComponentTypeByName (string *name*) `[static]`

Find a componet by name, and returns it

Parameters:

name	The name of the component to find

Returns:

The component which has the name, or null

void QuestHandler.QuestsDiscovered (Quest *quest*, CQPlayerObject *player*)

Tells the quest system that a quest has been discovered by a player

Parameters:

quest	The quest discovered
player	The player who discovered the quest

Member Data Documentation

List<Quest> QuestHandler.allQuests = new List<Quest>()

A list over the quests currently in the scene.

Dictionary<CQPlayerObject, List<Quest> > QuestHandler.availableQuests = new Dictionary<CQPlayerObject, List<Quest>>()

A dictonary which contains all the players, and the quests they each have

List<CQPlayerObject> QuestHandler.players = new List<CQPlayerObject>()

A list over the players currently in the scene. To add a new player, use "AddPlayer".

CQPlayerObject QuestHandler.selectedPlayer

The currently selected player. In networking enviroments, this would be the local client.

Property Documentation

Dictionary<string, List<UnityAction<EventInfoHolder> > > QuestHandler.EventDictonary `[get], [set]`

The properties for EventDictonary - Creates a new dictionary, if its null

CQPlayerObject QuestHandler.SelectedPlayer `[get], [set]`

The properties for selectedPlayer - Tries to find a player, if its null

---

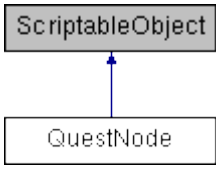
The documentation for this class was generated from the following file:

250 Assets/CustomQuest/Assets/Scripts/QuestHandler.cs

QuestNode Class Reference

The node for displaying a quest in scene. Contains Edges and a rectangle for pos and size.

Inheritance diagram for QuestNode:



Public Member Functions

251 void **Start** ()

Use this for initialization

Public Attributes

252 int **windowID**

The ID of this questNode

253 **Quest quest**

The quest this quetsNode is attached to

254 List< **QuestEdge** > **allEdges** = new List<**QuestEdge**>()

All the edges this quetsnode has

255 **QuestEdge startEdge**

The start edge for this questNode

256 **QuestEdge completeEdge**

The completede edge for this questNode

257 **QuestEdge failEdge**

The fail edge for this questNode

Properties

258 Rect **Rectangle** `[get, set]`

The rectangle of this questNode

Detailed Description

The node for displaying a quest in scene. Contains Edges and a rectangle for pos and size.

Member Function Documentation

void QuestNode.Start ()

Use this for initialization

Member Data Documentation

List<QuestEdge> QuestNode.allEdges = new List<QuestEdge>()

All the edges this quetsnode has

QuestEdge QuestNode.completeEdge

The completede edge for this questNode

QuestEdge QuestNode.failEdge

The fail edge for this questNode

Quest QuestNode.quest

The quest this quetsNode is attached to

QuestEdge QuestNode.startEdge

The start edge for this questNode

int QuestNode.windowID

The ID of this questNode

Property Documentation

Rect QuestNode.Rectangle `[get], [set]`

The rectangle of this questNode

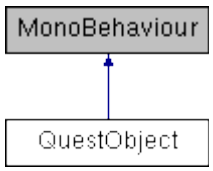
The documentation for this class was generated from the following file:

259 Assets/CustomQuest/Assets/Scripts/QuestNode.cs

QuestObject Class Reference

Component for a questObject. Holds a reference to the criteria its a part of. Used on all quest objects, enemies, gather objects etc.

Inheritance diagram for QuestObject:



Public Attributes

260 **Criteria criteria**

A reference to the criteria.

Detailed Description

Component for a questObject. Holds a reference to the criteria its a part of. Used on all quest objects, enemies, gather objects etc.

Member Data Documentation

Criteria QuestObject.criteria

A reference to the criteria.

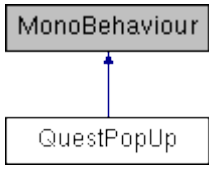
The documentation for this class was generated from the following file:

261 Assets/CustomQuest/Assets/Scripts/QuestObject.cs

QuestPopUp Class Reference

A pop up for when picking up a quest, when the game is running

Inheritance diagram for QuestPopUp:



Public Member Functions

262 void **SetStartValues** (**QuestUI questUI**)

*Sets this **QuestPopUp**'s start values*

263 void **AcceptQuest** ()

The player accepted this questPopUps quest

264 void **DeclineQuest** ()

The player declined this questPopUps quest

265 void **OnDrag** (UnityEngine.EventSystems.BaseEventData eventData)

Drags the window with the mouse

Public Attributes

266 **Quest quest**

The quest this quest pop up is giving

267 **CQPlayerObject player**

The player currently recieving this quest

268 **QuestGiver questGiver**

The giver of this quest

269 Text **title**

The text to contains the title

270 Text **description**

The text to contain the description

271 Text **criterias**

The text to contain the criterias

272 Text **rewards**

The text to contains the rewards

273 Image **icon**

The image to contain the icon

274 **QuestUI questUI**

The questUI this popup is a part of

Detailed Description

A pop up for when picking up a quest, when the game is running

Member Function Documentation

void QuestPopUp.AcceptQuest ()

The player accepted this questPopUps quest

void QuestPopUp.DeclineQuest ()

The player declined this questPopUps quest

void QuestPopUp.OnDrag (UnityEngine.EventSystems.BaseEventData *eventData*)

Drags the window with the mouse

Parameters:

eventData	

void QuestPopUp.SetStartValues (QuestUI *questUI*)

Sets this **QuestPopUp**'s start values

Member Data Documentation

Text QuestPopUp.criterias

The text to contain the criterias

Text QuestPopUp.description

The text to contain the description

Image QuestPopUp.icon

The image to contain the icon

CQPlayerObject QuestPopUp.player

The player currently recieving this quest

Quest QuestPopUp.quest

The quest this quest pop up is giving

QuestGiver QuestPopUp.questGiver

The giver of this quest

QuestUI QuestPopUp.questUI

The questUI this popup is a part of

Text QuestPopUp.rewards

The text to contains the rewards

Text QuestPopUp.title

The text to contains the title

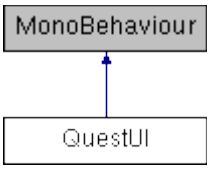
The documentation for this class was generated from the following file:

275 Assets/CustomQuest/Assets/Scripts/QuestPopUp.cs

QuestUI Class Reference

Used for displaying a UI quest list. Is not a list for holding quests.

Inheritance diagram for QuestUI:



Public Member Functions

276 void **ResetQuest** (**Quest** q)

Resets a quest, by deleting it from the UI list. And then update will add it again

277 void **AddOnSreenMsg** (float lifeTime, string msg, int size, Color color)

*Adds an **OnScreenMsg** to be displayed for the player*

278 void **UpdateQuestTracker** ()

Updates the quest tracker, pointing arrows on all the active quests

279 void **StartQuestPopUp** (**QuestGiver** questGiver, **CQPlayerObject** player, **Quest** quest)

Opens a quest pop up

280 void **UpdateSelectionCircle** (int highlightedItem)

Updates the quest wheel selection

Public Attributes

281 List< **Quest** > **canvasQuests** = new List<**Quest**>()

A list of quests in the questList

282 List< **QuestUILogic** > **questUis** = new List<**QuestUILogic**>()

A list of the quest uis currently used for showing the quests

283 GameObject **questHolder**

The questHolder, the UI prefabs are initialized under this gameobject

284 GameObject **questTemplate**

The template for quests UI

285 GameObject **criteriaTemplate**

The template for criterias UI

286 GameObject **rewardTemplate**

The template for reward UI

287 GameObject **optionalTemplate**

The template for the 'optional' ui element

288 GameObject **messagePrefab**

The messages prefab, used for instantiating new on screen messages

289 **OnScreenMsgHandler messageHolder**

The holder for messages, determines where the messages are added

290 Text **resourceAmount**

The ui text element, showing the selected players current resource amount

291 Text **itemList**

The ui text element, showing the selected players current list of items

292 GameObject **compassArrow**

The prefab of a compassArrow

293 List< GameObject > **compassArrows**

A list of current aktive compassArrows

294 GameObject **compass**

*The gameobject holding the **Quest** Compass script*

295 **QuestPopUp questPopUpPrefab**

Prefab for the quest pop up, used by the quest giver

296 Dictionary< **CQPlayerObject**, List< **Quest** > > **activeQuestPopUpQuests** = new Dictionary<**CQPlayerObject**, List<**Quest**>>()

A dictionary of the active questPopUps by player

297 int **questWheelSelection**

The current questWheel part selected

298 bool **questWheelAktive** = false

Used to ditermin if the quest wheel is visible and active, or not

299 GameObject **questWheel**

The quest wheel to toggle and use

300 GameObject **questWheelActions**

A holder which all the UI parts of the quest wheel is under.

301 Text **middleText**

To be assigned to the text in the middle of the quest wheel. Will change depending on which quest is hovered

302 List< **UIPolygon** > **questWheelBackgrounds** = new List<**UIPolygon**>()

Contains all the UIPolygon backgrounds for the quest wheel circle

303 List< Image > **questWheelImages** = new List<Image>()

Contains all the images for the quest wheel circle

304 Text **questNameText**

The text for displaying the quest name

305 Text **descriptionText**

The text for displaying the desciption name

306 Text **criteriaText**

The text for displaying the criterias

307 Text **rewardsText**

The text for displaying the rewards

308 GraphicRaycaster **gr**

A reference to the graphic raycaster of this scene. Used to find the graphic raycaster component.

Detailed Description

Used for displaying a UI quest list. Is not a list for holding quests.

Member Function Documentation

void QuestUI.AddOnSreenMsg (float *lifeTime*, string *msg*, int *size*, Color *color*)

Adds an **OnScreenMsg** to be displayed for the player

Parameters:

lifeTime	The lifetime of the msg
msg	The actualt msg
size	The size of the msg
color	The color of the text

void QuestUI.ResetQuest (Quest *q*)

Resets a quest, by deleting it from the UI list. And then update will add it again

Parameters:

q	The quest to reset

void QuestUI.StartQuestPopUp (QuestGiver *questGiver*, CQPlayerObject *player*, Quest *quest*)

Opens a quest pop up

Parameters:

questGiver	The questgiver giving the quest
player	The player recieving the quest

void QuestUI.UpdateQuestTracker ()

Updates the quest tracker, pointing arrows on all the active quests

void QuestUI.UpdateSelectionCircle (int *highlightedItem*)

Updates the quest wheel selection

Parameters:

highlightedItem	The new highlighted item

Member Data Documentation

Dictionary<CQPlayerObject, List<Quest> > QuestUI.activeQuestPopUpQuests = new Dictionary<CQPlayerObject, List<Quest>>()

A dictionary of the active questPopUps by player

List<Quest> QuestUI.canvasQuests = new List<Quest>()

A list of quests in the questList

GameObject QuestUI.compass

The gameobject holding the **Quest** Compass script

GameObject QuestUI.compassArrow

The prefab of a compassArrow

List<GameObject> QuestUI.compassArrows

A list of current aktive compassArrows

GameObject QuestUI.criteriaTemplate

The template for criterias UI

Text QuestUI.criteriaText

The text for displaying the criterias

Text QuestUI.descriptionText

The text for displaying the desciption name

GraphicRaycaster QuestUI.gr

A reference to the graphic raycaster of this scene. Used to find the graphic raycaster component.

Text QuestUI.itemList

The ui text element, showing the selected players current list of items

OnScreenMsgHandler QuestUI.messageHolder

The holder for messages, determines where the messages are added

GameObject QuestUI.messagePrefab

The messages prefab, used for instantiating new on screen messages

Text QuestUI.middleText

To be assigned to the text in the middle of the quest wheel. Will change depending on which quest is hovered

GameObject QuestUI.optionalTemplate

The template for the 'optional' ui element

GameObject QuestUI.questHolder

The questHolder, the UI prefabs are initialized under this gameobject

Text QuestUI.questNameText

The text for displaying the quest name

QuestPopUp QuestUI.questPopUpPrefab

Prefab for the quest pop up, used by the quest giver

GameObject QuestUI.questTemplate

The template for quests UI

List<QuestUILogic> QuestUI.questUis = new List<QuestUILogic>()

A list of the quest uis currently used for showing the quests

GameObject QuestUI.questWheel

The quest wheel to toggle and use

GameObject QuestUI.questWheelActions

A holder which all the UI parts of the quest wheel is under.

bool QuestUI.questWheelAktive = false

Used to ditermin if the quest wheel is visible and active, or not

List<UIPolygon> QuestUI.questWheelBackgrounds = new List<UIPolygon>()

Contains all the UIPolygon backgrounds for the quest wheel circle

List<Image> QuestUI.questWheelImages = new List<Image>()

Contains all the images for the quest wheel circle

int QuestUI.questWheelSelection

The current questWheel part selected

Text QuestUI.resourceAmount

The ui text element, showing the selected players current resource amount

Text QuestUI.rewardsText

The text for displaying the rewards

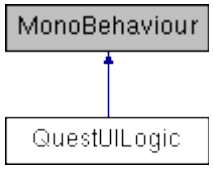
GameObject QuestUI.rewardTemplate

The template for reward UI

The documentation for this class was generated from the following file:
309 Assets/CustomQuest/Assets/Scripts/QuestUI.cs

QuestUILogic Class Reference

A UI logic holder for a quests. Used for displaying a list of quest.
Inheritance diagram for QuestUILogic:



Public Member Functions

310 void **Start** ()
The method used for initialization

Public Attributes

311 **Quest quest**
*The **Quest** Script this **QuestUILogic** is the logic off*
312 Text **questName**
The text element which should show the name of the quest
313 Text **description**
The text element which should show the description of the quest
314 Image **questIcon**
The icon of the quest this element is showing
315 List< **CriteriaUILogic** > **criteriaUis** = new List<**CriteriaUILogic**>()
A list of the CriteriasUIs this questsUIlogic has
316 List< **Criteria** > **criterias** = new List<**Criteria**>()

A list of criterias from the quest, this ui is showing
317 List< **RewardUILogic** > **rewardUis** = new List<**RewardUILogic**>()
A list of rewardUIs this questUIlogic has
318 List< **Reward** > **rewards** = new List<**Reward**>()
A list of rewards from the quest, this ui is showing
319 RectTransform **rectTransform**
The rect transform of this object

Detailed Description

A UI logic holder for a quests. Used for displaying a list of quest.

Member Function Documentation

void QuestUILogic.Start ()

The method used for initialization

Member Data Documentation

List<Criteria> QuestUILogic.criterias = new List<Criteria>()

A list of criterias from the quest, this ui is showing

List<CriteriaUILogic> QuestUILogic.criteriaUis = new List<CriteriaUILogic>()

A list of the CriteriasUIs this questsUIlogic has

Text QuestUILogic.description

The text element which should show the description of the quest

Quest QuestUILogic.quest

The **Quest** Script this **QuestUILogic** is the logic off

Image QuestUILogic.questIcon

The icon of the quest this element is showing

Text QuestUILogic.questName

The text element which should show the name of the quest

RectTransform QuestUILogic.rectTransform

The rect transform of this object

List<Reward> QuestUILogic.rewards = new List<Reward>()

A list of rewards from the quest, this ui is showing

List<RewardUILogic> QuestUILogic.rewardUis = new List<RewardUILogic>()

A list of rewardUIs this questUIlogic has

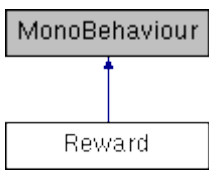
The documentation for this class was generated from the following file:

320 Assets/CustomQuest/Assets/Scripts/QuestUILogic.cs

Reward Class Reference

A reward given by a quest or a criteria.

Inheritance diagram for Reward:



Public Member Functions

321 virtual void **Start** ()
Use this for initialization
322 virtual void **EditorStart** ()
Used when converting a script, so the fields are set correctly (It's supposed to be empthy, see the .txt files for explanation)
323 virtual void **Update** ()
Update is called once per frame

Public Attributes

324 string **rewardName**
*Name of the **Reward**.*
325 **rewardType type**
*Type of the **Reward**.*
326 GameObject **rewardObject**
The reward Object.
327 int **amount**
Amount of rewards.

Detailed Description

A reward given by a quest or a criteria.

Member Function Documentation

virtual void Reward.EditorStart () `[virtual]`

Used when converting a script, so the fields are set correctly (It's supposed to be empthy, see the .txt files for explanation)

virtual void Reward.Start () `[virtual]`

Use this for initialization

virtual void Reward.Update () `[virtual]`

Update is called once per frame

Member Data Documentation

int Reward.amount

Amount of rewards.

string Reward.rewardName

Name of the **Reward**.

GameObject Reward.rewardObject

The reward Object.

rewardType Reward.type

Type of the **Reward**.

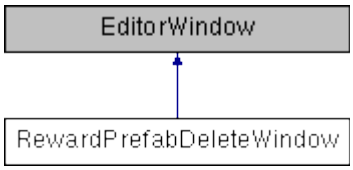
The documentation for this class was generated from the following file:

328 Assets/CustomQuest/Assets/Scripts/Reward.cs

RewardPrefabDeleteWindow Class Reference

An editor window used to display a confirmation pop up, when deleting a criteria.

Inheritance diagram for RewardPrefabDeleteWindow:



Public Member Functions

329 void **SetQuestEditor** (**CustomQuestEditor** editor, **Reward** r)

Sets the editor controlling this window, and the reward about to be deleted

Properties

330 static **RewardPrefabDeleteWindow Instance** `[get]`

Detailed Description

An editor window used to display a confirmation pop up, when deleting a criteria.

Member Function Documentation

void RewardPrefabDeleteWindow.SetQuestEditor (CustomQuestEditor *editor*, Reward *r*)

Sets the editor controlling this window, and the reward about to be deleted

Parameters:

editor	The editor which spawned this window
r	The reward about to be deleted

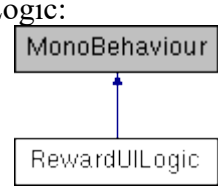
The documentation for this class was generated from the following file:

331 Assets/CustomQuest/Assets/Scripts/Editor/RewardPrefabDeleteWindow.cs

RewardUILogic Class Reference

A UI logic holder for a reward. Used for displaying a list of quest.

Inheritance diagram for RewardUILogic:



Public Member Functions

332 void **Start** ()

Use this for initialization

Public Attributes

333 **Reward reward**

The reward of this rewardUILogic object

334 Text **rewardName**

The text which should be showing the name of the reward

335 Text **rewardType**

The text which should be showing the type of the reward

336 Text **rewardAmount**

The text which should be showing the amount of the reward

337 RectTransform **rectTransform**

The recttransform of this rewardUILogic object

Detailed Description

A UI logic holder for a reward. Used for displaying a list of quest.

Member Function Documentation

void RewardUILogic.Start ()

Use this for initialization

Member Data Documentation

RectTransform RewardUILogic.rectTransform

The recttransform of this rewardUILogic object

Reward RewardUILogic.reward

The reward of this rewardUILogic object

Text RewardUILogic.rewardAmount

The text which should be showing the amount of the reward

Text RewardUILogic.rewardName

The text which should be showing the name of the reward

Text RewardUILogic.rewardType

The text which should be showing the type of the reward

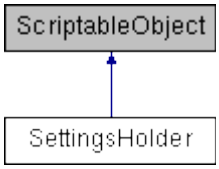
The documentation for this class was generated from the following file:

338 Assets/CustomQuest/Assets/Scripts/RewardUILogic.cs

SettingsHolder Class Reference

A holder for settings for the custom quest system

Inheritance diagram for SettingsHolder:



Public Member Functions

339 void **DestroyObject** (ScriptableObject o)

Destroys an scriptableObject immediately, warning: will destroy assets without warning.

Public Attributes

340 List< **Quest** > **prefabQuests** = new List<**Quest**>()
341 List< **Criteria** > **prefabCriteria** = new List<**Criteria**>()
342 List< **Reward** > **prefabReward** = new List<**Reward**>()
343 List< **QuestNode** > **questNodes** = new List<**QuestNode**>()
344 bool **showQuestName**
345 bool **showDescription**
346 bool **showCriterias**
347 bool **showRewards**
348 GameObject **handInObjectPrefab**
349 GameObject **criteriaSpawnPrefab**
350 GameObject **questGiverPrefab**
351 bool **criteriaSpecificRewards**
352 bool **optionalCriteriaSpecificRewards**
353 bool **optional**
354 GUISkin **randomDragonGUISkin**

Detailed Description

A holder for settings for the custom quest system

Member Function Documentation

void SettingsHolder.DestroyObject (ScriptableObject *o*)

Destroys an scriptableObject immediately, warning: will destroy assets without warning.

Parameters:

o	The object to destroy

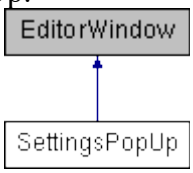
The documentation for this class was generated from the following file:

355 Assets/CustomQuest/Assets/Scripts/SettingsHolder.cs

SettingsPopUp Class Reference

The settings window

Inheritance diagram for SettingsPopUp:



Properties

356 static **SettingsPopUp Instance** `[get]`

Detailed Description

The settings window

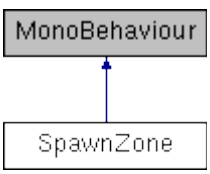
The documentation for this class was generated from the following file:

357 Assets/CustomQuest/Assets/Scripts/Editor/SettingsPopUp.cs

SpawnZone Class Reference

A spawn zone for a quest object. Contains information about spawning

Inheritance diagram for SpawnZone:



Public Member Functions

358 void **DespawnQuestObjects** ()
Despawns the QuestObjects

Public Attributes

359 float **spawnRateTimer**
A spawn timer, goes up with deltaTime, and is compared to the criterias spawnrate
360 GameObject **spawnAreaObject**
*The object the quest will spawn the **Criteria** Objects around.*
361 float **spawnRadius**
The radius around the spawn object, where the objects will spawn inside
362 int **spawnAmount**
The amount of objects you wan't the quest to spawn.
363 float **spawnRate**
The rate of how often the objects should spawn.
364 int **initialSpawnAmount**
How many objects the quest should spawn from the beginning.
365 int **maxSpawnAmount**
The max amount of spawns at once.
366 List< GameObject > **spawnedObjects** = new List<GameObject>()
List of the spawnedObjects.

Properties

367 bool **Spawn** `[get, set]`
368 **Criteria Criteria** `[get, set]`
The criteria this spawnzone is the spawnzone for
369 string **SpawnName** `[get, set]`
The name of this spawnZone

Detailed Description

A spawn zone for a quest object. Contains information about spawning

---

Member Function Documentation

void SpawnZone.DespawnQuestObjects ()

Despawns the QuestObjects

Parameters:

questObjects	The objects to despawn

Member Data Documentation

int SpawnZone.initialSpawnAmount

How many objects the quest should spawn from the beginning.

int SpawnZone.maxSpawnAmount

The max amount of spawns at once.

int SpawnZone.spawnAmount

The amount of objects you wan't the quest to spawn.

GameObject SpawnZone.spawnAreaObject

The object the quest will spawn the **Criteria** Objects around.

List<GameObject> SpawnZone.spawnedObjects = new List<GameObject>()

List of the spawnedObjects.

float SpawnZone.spawnRadius

The radius around the spawn object, where the objects will spawn inside

float SpawnZone.spawnRate

The rate of how often the objects should spawn.

float SpawnZone.spawnRateTimer

A spawn timer, goes up with deltaTime, and is compared to the criterias spawnrate

Property Documentation

Criteria SpawnZone.Criteria `[get], [set]`

The criteria this spawnzone is the spawnzone for

string SpawnZone.SpawnName `[get], [set]`

The name of this spawnZone

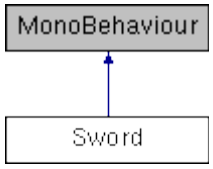
The documentation for this class was generated from the following file:

370 Assets/CustomQuest/Assets/Scripts/SpawnZone.cs

Sword Class Reference

A class for controlling when the sword is lethal, and for dealing dmg

Inheritance diagram for Sword:



Detailed Description

A class for controlling when the sword is lethal, and for dealing dmg

---

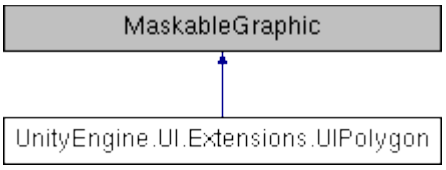
The documentation for this class was generated from the following file:

371 Assets/CustomQuest/Assets/Scripts/Demo Scripts/Sword.cs

UnityEngine.UI.Extensions.UIPolygon Class Reference

Credit CiaccoDavide Sourced from - `http://ciaccodavi.de/unity/uipolygon` Used for Custom **Quest** 28-04-2017

Inheritance diagram for UnityEngine.UI.Extensions.UIPolygon:



Public Member Functions

372 void **DrawPolygon** (int _sides)
373 void **DrawPolygon** (int _sides, float[] _VerticesDistances)
374 void **DrawPolygon** (int _sides, float[] _VerticesDistances, float _rotation)

Public Attributes

375 bool **fill** = true
376 float **thickness** = 5
377 int **sides** = 3
378 float **rotation** = 0
379 float [] **VerticesDistances** = new float[3]

Protected Member Functions

380 UIVertex [] **SetVbo** (Vector2[] vertices, Vector2[] uvs)
381 override void **OnPopulateMesh** (VertexHelper vh)

Properties

382 override Texture **mainTexture** `[get]`
383 Texture **texture** `[get, set]`

Detailed Description

Credit CiaccoDavide Sourced from - `http://ciaccodavi.de/unity/uipolygon` Used for Custom **Quest** 28-04-2017

This is **UI** logic component, which makes you able to do polygons in the existing Unity **UI** logic

---

The documentation for this class was generated from the following file:
384 Assets/CustomQuest/Assets/Scripts/Demo Scripts/UIPolygon.cs

Index

INDEX