**STOP THE WHEEL**

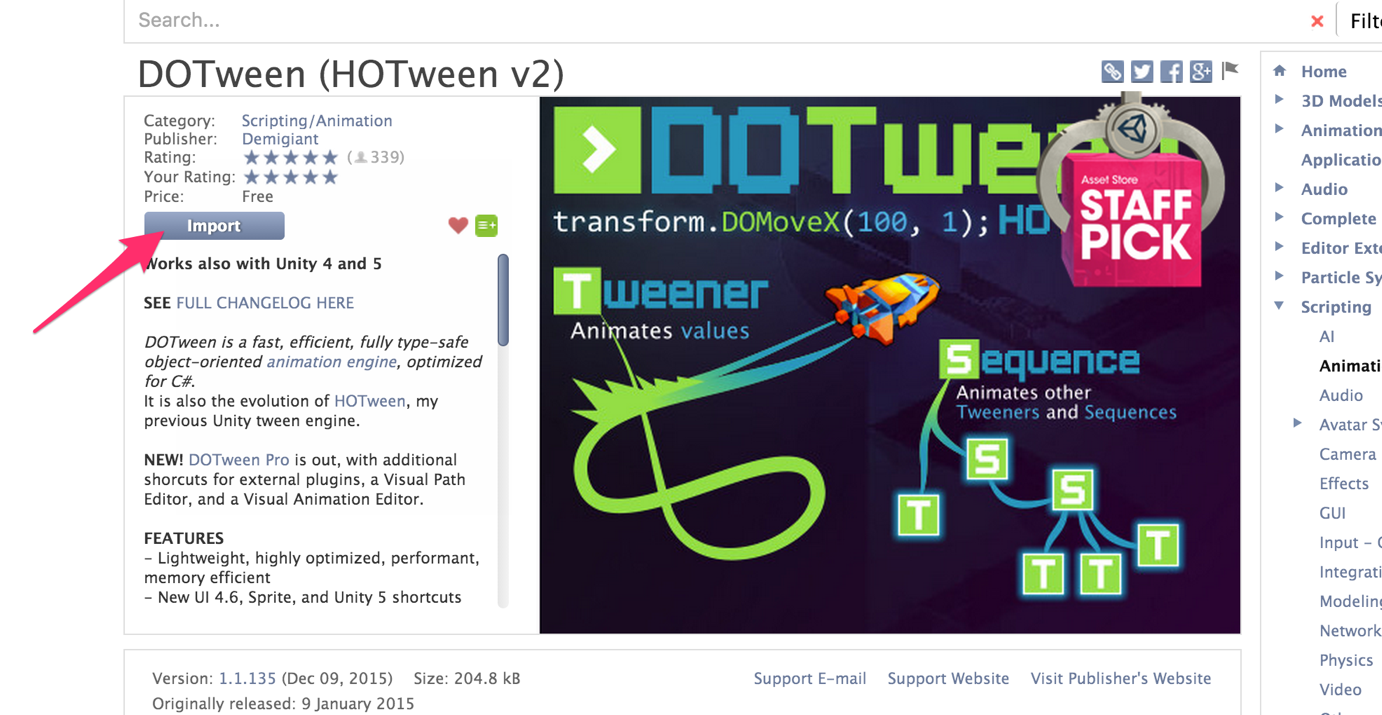
Thanks for your purchase.

First of all, you have to get [DOTWEEN](http://u3d.as/aZ1) from the Asset Store :

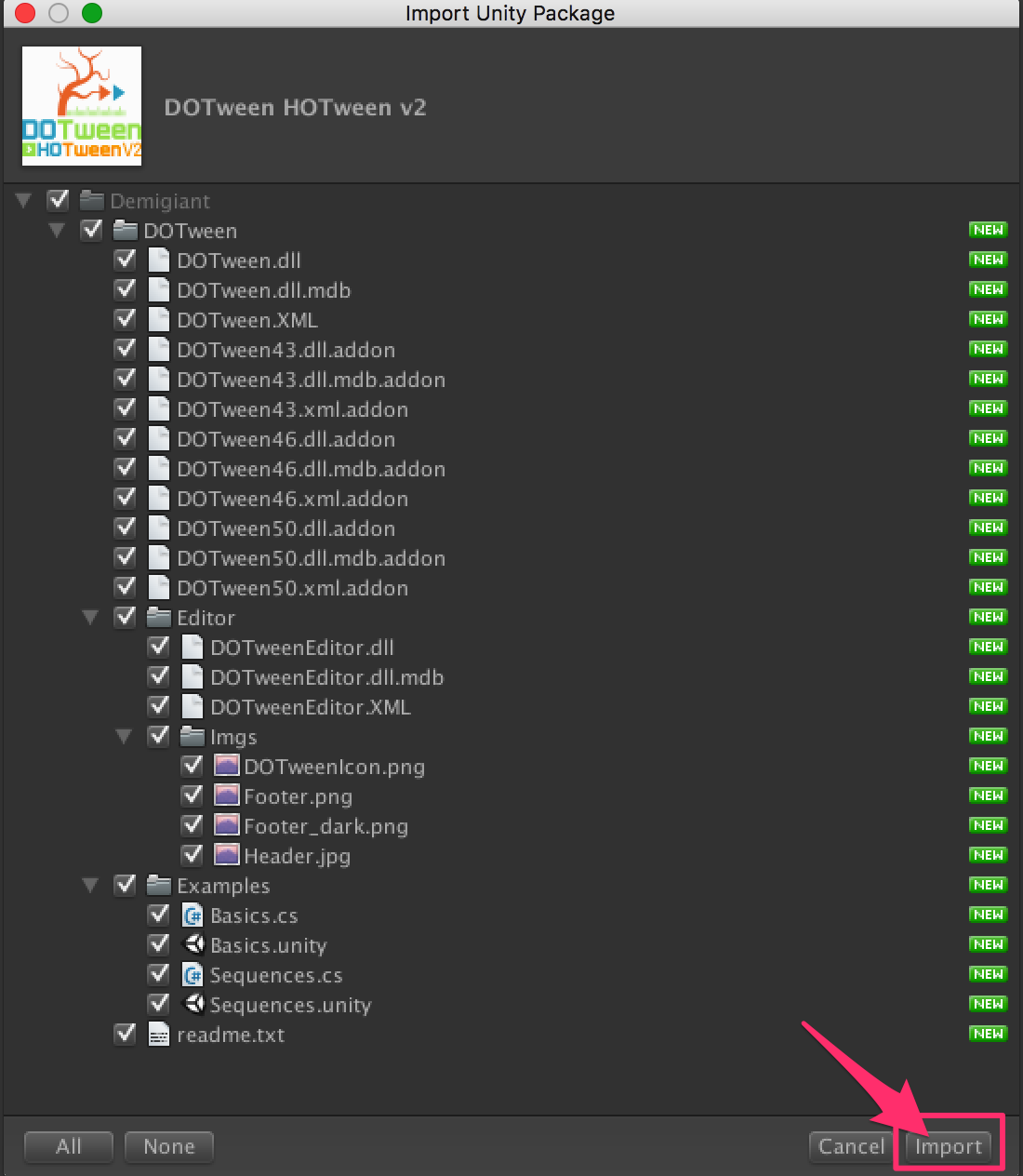
<http://u3d.as/aZ1>

(it’s free)

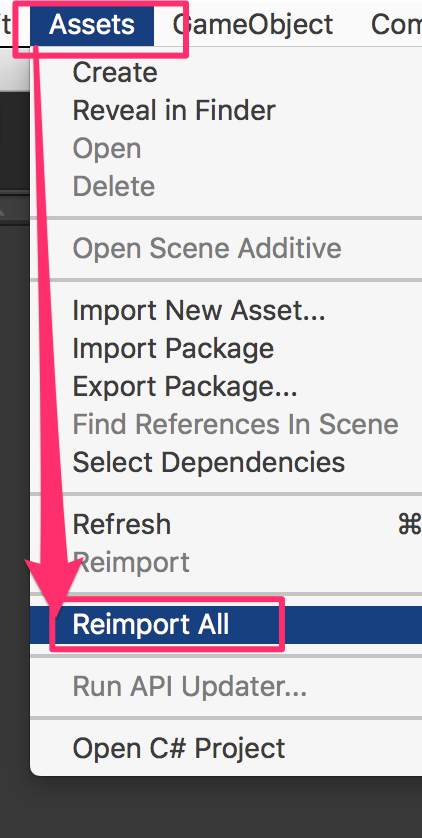
1/ Import Dotween from the asset store : <http://u3d.as/aZ1>



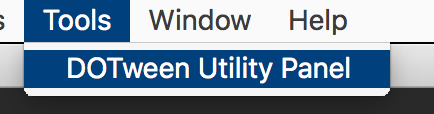
2/ Import the package into Unity



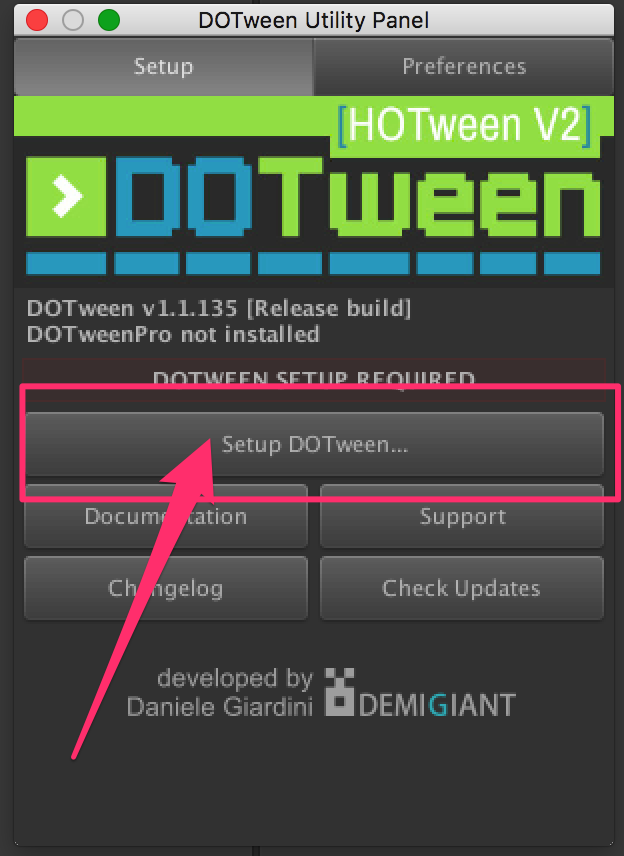
3/If you don’t see the « Tools » in the top of the Unity Screen, please do this :



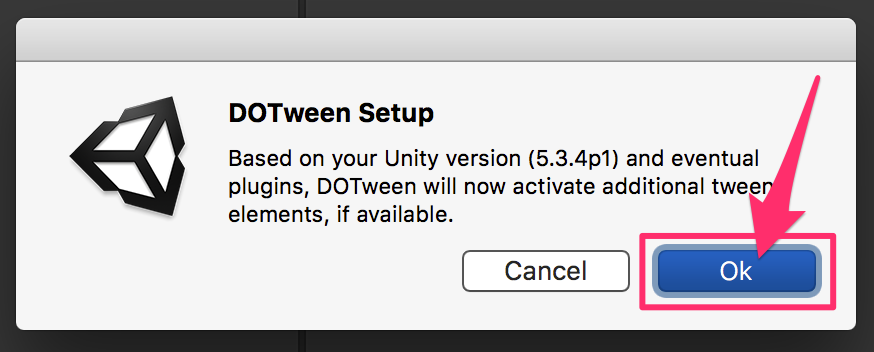
4/Now you have the « Tools ». Open it and click on « DOTween Utility Panel ».



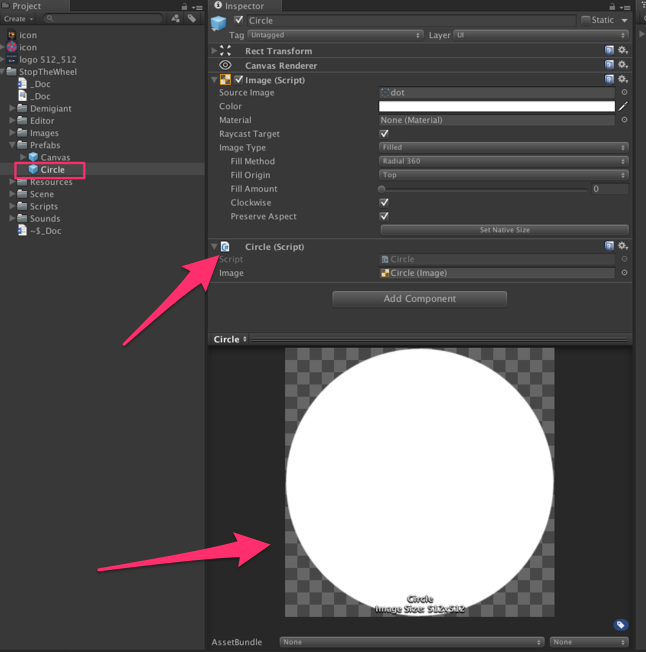
5/Click on « Setup DOTween ».



6/And to finish, click on the « OK » button.

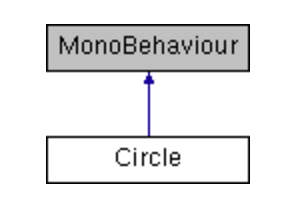


**Circle Class Reference**



Each part of the wheel is a circle. We use the fillAmount component of UI image to get "parts". All the circles are child of the Game Object PartParent (= [**WheelRotator**](class_wheel_rotator.html)). The [**Circle**](class_circle.html) prefab is in the Prefabs folder. Each Circles are instantiate in the [**WheelLogic**](class_wheel_logic.html) at the start of each level [More...](class_circle.html#details)

Inheritance diagram for Circle:



|  |  |
| --- | --- |
| Public Member Functions | |
| [**Circle**](class_circle.html) | [**Init**](class_circle.html#aa3e44e2c36f06da02a69aaa03531eaee) (float fillAmout, float angle, Color color) |
|  | Init the circle = the part of the wheel. Each part is defined with a fillAmount = 1 / number of part in the wheel, an angle and a color [More...](#aa3e44e2c36f06da02a69aaa03531eaee) |
|  | |
| float | [**GetMiddleAngle**](class_circle.html#a3a4a0d6c48e00af4d8a68f6561926603) () |
|  | Get the angle of the middle of the part of wheel [More...](#a3a4a0d6c48e00af4d8a68f6561926603) |
|  | |

|  |  |
| --- | --- |
| Public Attributes | |
| Image | [**image**](class_circle.html#a2bdef1d38104a7cc0a5a478fa164bcda) |
|  | The image = a simple circle [More...](#a2bdef1d38104a7cc0a5a478fa164bcda) |
|  | |

Detailed Description

Each part of the wheel is a circle. We use the fillAmount component of UI image to get "parts". All the circles are child of the Game Object PartParent (= [**WheelRotator**](class_wheel_rotator.html)). The [**Circle**](class_circle.html) prefab is in the Prefabs folder. Each Circles are instantiate in the [**WheelLogic**](class_wheel_logic.html) at the start of each level

Member Function Documentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **float Circle.GetMiddleAngle** | **(** |  | **)** |  | | inline |

Get the angle of the middle of the part of wheel

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | [**Circle**](class_circle.html)**Circle.Init** | **(** | **float** | **fillAmout,** | |  |  | **float** | **angle,** | |  |  | **Color** | **color** | |  | **)** |  |  | | inline |

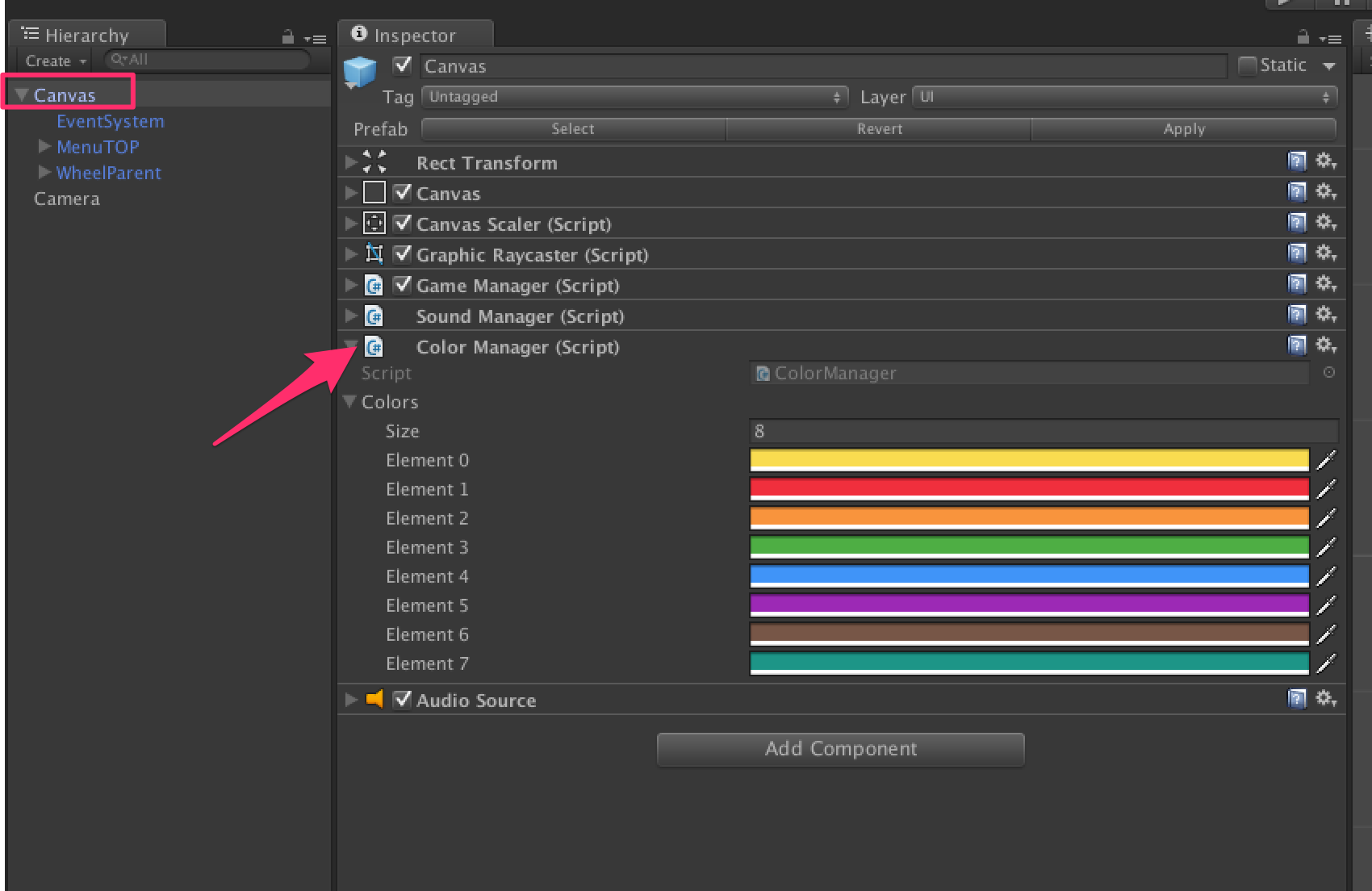
Init the circle = the part of the wheel. Each part is defined with a fillAmount = 1 / number of part in the wheel, an angle and a color

Member Data Documentation

|  |
| --- |
| **Image Circle.image** |

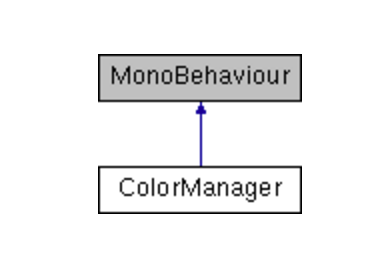
The image = a simple circle

**ColorManager Class Reference**



Class with an array of color. Change the array to customize the colors. Attached to the Canvas game object [More...](class_color_manager.html#details)

Inheritance diagram for ColorManager:

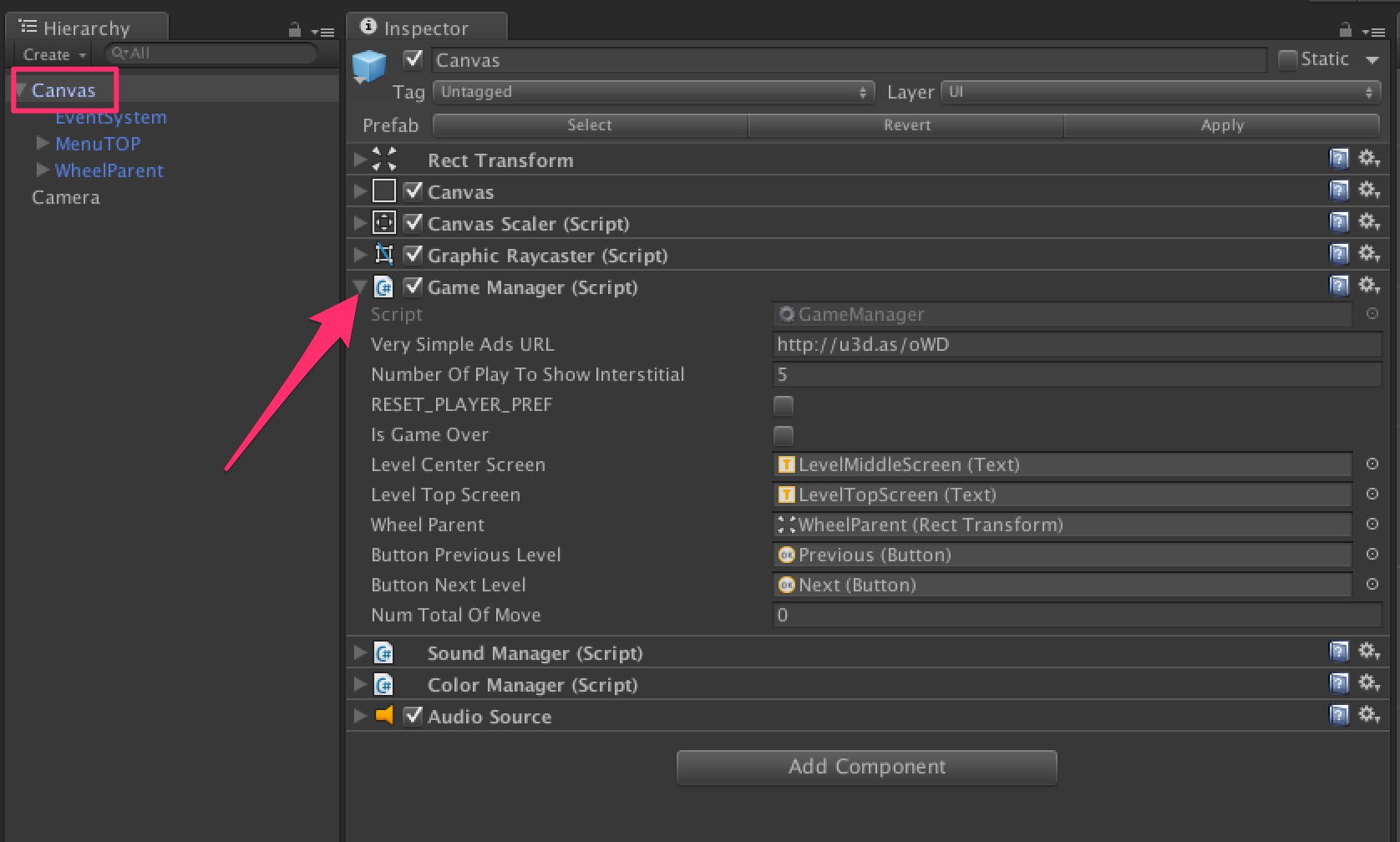


|  |  |
| --- | --- |
| Public Attributes | |
| Color[] | **colors** |
|  | |

Detailed Description

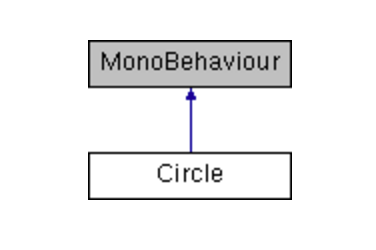
Class with an array of color. Change the array to customize the colors. Attached to the Canvas game object

**GameManager Class Reference**



In charge of the game logic: Game Start, Game Over, Score, Ads etc... Attached to the Canvas game object [More...](class_game_manager.html#details)

Inheritance diagram for GameManager:

****

|  |  |
| --- | --- |
| Public Member Functions | |
| void | [**MoveDone**](class_game_manager.html#abf85bce64b1dc6a833935c982cb92739) () |
|  | When a move is done, ie. player tap at the good moment, we decrease the numTotalOfMove ( -1 ) and we check if success (numTotalOfMove = 0). If success, we call the function LevelClear. If not, play a sound [More...](#abf85bce64b1dc6a833935c982cb92739) |
|  | |
| void | [**GameOver**](class_game_manager.html#a8d69157cb6b97eabeff2374d8e9adeaf) () |
|  | When a move is done, ie. player tap on the screen and the color of the triangle is not equal of the color of the part of the wheel below => Game Over. We restart the game and show interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD> [More...](#a8d69157cb6b97eabeff2374d8e9adeaf) |
|  | |
| void | [**LevelCleared**](class_game_manager.html#ad85fd1b6e237bbc55048f5a117a215b9) () |
|  | If the level is cleared (numTotalOfMove = 0), this function is called. We will animate out the wheel, increase the current level ( +1 ) and go to the next level. We we call to an interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD> [More...](#ad85fd1b6e237bbc55048f5a117a215b9) |
|  | |
| void | [**ShowAds**](class_game_manager.html#abd163e2010eff15a9acb643587c6d573) () |
|  | Show Ads - Interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD> [More...](#abd163e2010eff15a9acb643587c6d573) |
|  | |

|  |  |
| --- | --- |
| Public Attributes | |
| string | [**VerySimpleAdsURL**](class_game_manager.html#a29e66d2485d77e9fbb6415e354c47f8e) = "http://u3d.as/oWD" |
|  | If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD> [More...](#a29e66d2485d77e9fbb6415e354c47f8e) |
|  | |
| int | [**numberOfPlayToShowInterstitial**](class_game_manager.html#ac0e2e2d0daca9ea3ed5c2cc36dad5b53) = 5 |
|  | Number of "play" to show an interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD> [More...](#ac0e2e2d0daca9ea3ed5c2cc36dad5b53) |
|  | |
| bool | [**RESET\_PLAYER\_PREF**](class_game_manager.html#a692985b5bad8435d2b2b52c1771e26db) = false |
|  | to reset the player pref. Use if for debug only!! [More...](#a692985b5bad8435d2b2b52c1771e26db) |
|  | |
| bool | [**isGameOver**](class_game_manager.html#a5e07e0e014f0306ee5c51fcf90ab1479) = false |
|  | True if game over [More...](#a5e07e0e014f0306ee5c51fcf90ab1479) |
|  | |
| Text | [**levelCenterScreen**](class_game_manager.html#aa4de0c4030ea97c1ec0198ee7ab20a07) |
|  | Text in the center of the screen = number of colors to find to clear the level [More...](#aa4de0c4030ea97c1ec0198ee7ab20a07) |
|  | |
| Text | [**levelTopScreen**](class_game_manager.html#a7c1fae0b7a1d162f0a22b32dd873f1a4) |
|  | Text in the center of the screen = number of colors to find to clear the level [More...](#a7c1fae0b7a1d162f0a22b32dd873f1a4) |
|  | |
| RectTransform | [**wheelParent**](class_game_manager.html#a1d7972cdc7ca5842e17a9bc4cfc7c037) |
|  | Reference to wheel parent, to do the animation in and out for transition between level [More...](#a1d7972cdc7ca5842e17a9bc4cfc7c037) |
|  | |
| int | [**numTotalOfMove**](class_game_manager.html#a9f4881222c4fe3782c6a95e33f52c038) = 0 |
|  | The number of move we have to do to clear this level = the level number [More...](#a9f4881222c4fe3782c6a95e33f52c038) |
|  | |

|  |  |
| --- | --- |
| Private Member Functions | |
| void | [**Awake**](class_game_manager.html#a2959177ee51ac31badbf8b1bc8f7f637) () |
|  | Clean the memory and place the wheelparent at the good place [More...](#a2959177ee51ac31badbf8b1bc8f7f637) |
|  | |
| void | [**Start**](class_game_manager.html#a5ccfacd027ad08eeb4ff1f25a7f59c98) () |
|  | Clean the memory and place the wheelparent at the good place [More...](#a5ccfacd027ad08eeb4ff1f25a7f59c98) |
|  | |
| void | [**OnClickedPreviousLevel**](class_game_manager.html#a55a908f2de193affbfc279b3815d9abe) () |
|  | Called when player tap the previous button [More...](#a55a908f2de193affbfc279b3815d9abe) |
|  | |
| void | [**OnClickedNextLevel**](class_game_manager.html#ae040cd6c03e6f3c5e714361ee84e622f) () |
|  | Called when player tap the next button [More...](#ae040cd6c03e6f3c5e714361ee84e622f) |
|  | |
| void | [**OnClick**](class_game_manager.html#af60b97ebd0cd9c09a0fd7200b0ca8a30) (bool isNext) |
|  | Called when player tap the next or previous button [More...](#af60b97ebd0cd9c09a0fd7200b0ca8a30) |
|  | |
| void | [**SetNewGame**](class_game_manager.html#a5a3d50c9743d823fa0f2c5573b984850) () |
|  | Create a new game: Set the texts, the numTotalOfMove and if the last game was not a game over : do the animation in [More...](#a5a3d50c9743d823fa0f2c5573b984850) |
|  | |
| void | [**UpdateButton**](class_game_manager.html#a278ce5ed452abe4f096bdfc1d604525d) () |
|  | Update the button previous and next [More...](#a278ce5ed452abe4f096bdfc1d604525d) |
|  | |
| void | [**DOMoveLevelOut**](class_game_manager.html#a5890b85a3c7a3005d8477e96f4ac00d9) (Action callback) |
|  | Animation out of the wheel (from center to left) [More...](#a5890b85a3c7a3005d8477e96f4ac00d9) |
|  | |
| void | [**DOMoveLevelIn**](class_game_manager.html#abc10f579a36221525962422425ddfa30) (Action callback) |
|  | Animation in of the wheel (from right to center) [More...](#abc10f579a36221525962422425ddfa30) |
|  | |

|  |  |
| --- | --- |
| Private Attributes | |
| Button | [**buttonPreviousLevel**](class_game_manager.html#af5d63233123bc33d798978f1355d3d8d) |
|  | Reference to the button to go to the previous level [More...](#af5d63233123bc33d798978f1355d3d8d) |
|  | |
| Button | [**buttonNextLevel**](class_game_manager.html#ad20fe065208995ab11b58d1eeb3f9afa) |
|  | Reference to the button to go to the next level, if the next level is already unlocked [More...](#ad20fe065208995ab11b58d1eeb3f9afa) |
|  | |

|  |
| --- |
| Additional Inherited Members |
| **Properties inherited from**[**MonoBehaviourHelper**](class_mono_behaviour_helper.html) |

## Detailed Description

In charge of the game logic: Game Start, Game Over, Score, Ads etc... Attached to the Canvas game object

## Member Function Documentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.Awake** | **(** |  | **)** |  | | inlineprivate |

Clean the memory and place the wheelparent at the good place

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **void GameManager.DOMoveLevelIn** | **(** | **Action** | **callback** | **)** |  | | inlineprivate |

Animation in of the wheel (from right to center)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **void GameManager.DOMoveLevelOut** | **(** | **Action** | **callback** | **)** |  | | inlineprivate |

Animation out of the wheel (from center to left)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.GameOver** | **(** |  | **)** |  | | inline |

When a move is done, ie. player tap on the screen and the color of the triangle is not equal of the color of the part of the wheel below => Game Over. We restart the game and show interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.LevelCleared** | **(** |  | **)** |  | | inline |

If the level is cleared (numTotalOfMove = 0), this function is called. We will animate out the wheel, increase the current level ( +1 ) and go to the next level. We we call to an interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.MoveDone** | **(** |  | **)** |  | | inline |

When a move is done, ie. player tap at the good moment, we decrease the numTotalOfMove ( -1 ) and we check if success (numTotalOfMove = 0). If success, we call the function LevelClear. If not, play a sound

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **void GameManager.OnClick** | **(** | **bool** | **isNext** | **)** |  | | inlineprivate |

Called when player tap the next or previous button

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.OnClickedNextLevel** | **(** |  | **)** |  | | inlineprivate |

Called when player tap the next button

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.OnClickedPreviousLevel** | **(** |  | **)** |  | | inlineprivate |

Called when player tap the previous button

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.SetNewGame** | **(** |  | **)** |  | | inlineprivate |

Create a new game: Set the texts, the numTotalOfMove and if the last game was not a game over : do the animation in

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.ShowAds** | **(** |  | **)** |  | | inline |

Show Ads - Interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.Start** | **(** |  | **)** |  | | inlineprivate |

Clean the memory and place the wheelparent at the good place

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameManager.UpdateButton** | **(** |  | **)** |  | | inlineprivate |

Update the button previous and next

## Member Data Documentation

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Button GameManager.buttonNextLevel** | | private |

Reference to the button to go to the next level, if the next level is already unlocked

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Button GameManager.buttonPreviousLevel** | | private |

Reference to the button to go to the previous level

|  |
| --- |
| **bool GameManager.isGameOver = false** |

True if game over

|  |
| --- |
| **Text GameManager.levelCenterScreen** |

Text in the center of the screen = number of colors to find to clear the level

|  |
| --- |
| **Text GameManager.levelTopScreen** |

Text in the center of the screen = number of colors to find to clear the level

|  |
| --- |
| **int GameManager.numberOfPlayToShowInterstitial = 5** |

Number of "play" to show an interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>

|  |
| --- |
| **int GameManager.numTotalOfMove = 0** |

The number of move we have to do to clear this level = the level number

|  |
| --- |
| **bool GameManager.RESET\_PLAYER\_PREF = false** |

to reset the player pref. Use if for debug only!!

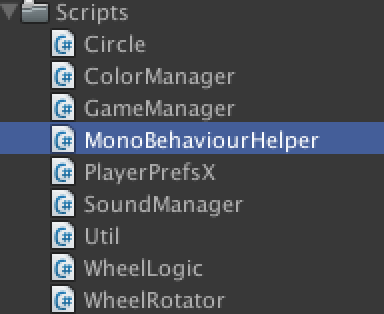
|  |
| --- |
| **string GameManager.VerySimpleAdsURL = "http://u3d.as/oWD"** |

If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>

|  |
| --- |
| **RectTransform GameManager.wheelParent** |

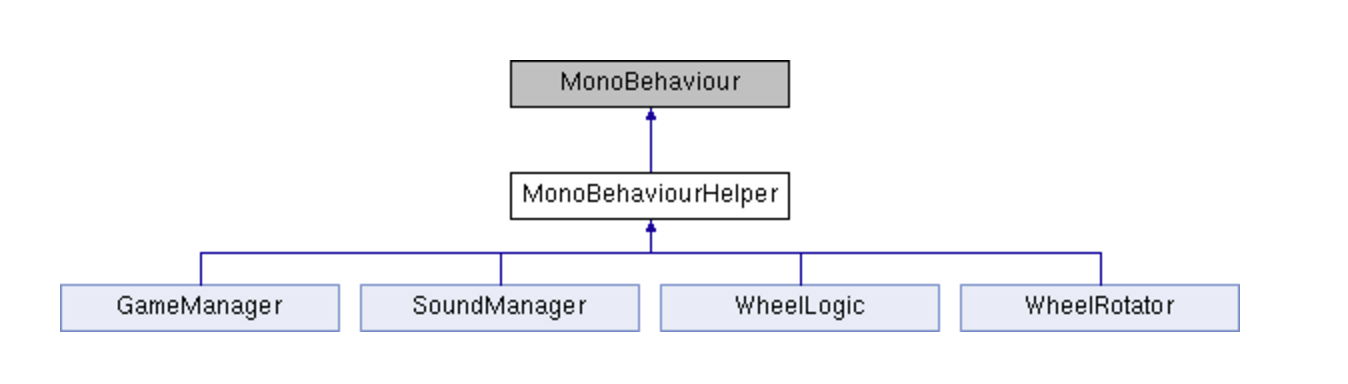
Reference to wheel parent, to do the animation in and out for transition between level

**MonoBehaviourHelper Class Reference**



Script to avoid duplicate code. [More...](class_mono_behaviour_helper.html#details)

Inheritance diagram for MonoBehaviourHelper:



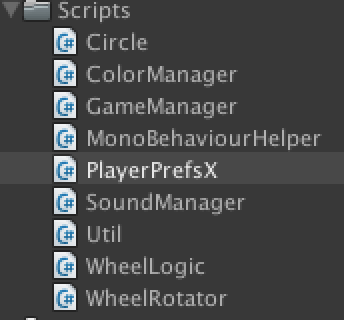
|  |  |
| --- | --- |
| Properties | |
| [**WheelLogic**](class_wheel_logic.html) | **wheelLogic** [get] |
|  | |
| [**WheelRotator**](class_wheel_rotator.html) | **wheelRotator** [get] |
|  | |
| [**GameManager**](class_game_manager.html) | **gameManager** [get] |
|  | |
| [**SoundManager**](class_sound_manager.html) | **soundManager** [get] |
|  | |
| [**ColorManager**](class_color_manager.html) | **colorManager** [get] |
|  | |

|  |  |
| --- | --- |
| Private Attributes | |
| [**WheelLogic**](class_wheel_logic.html) | **\_wheelLogic** |
|  | |
| [**WheelRotator**](class_wheel_rotator.html) | **\_wheelRotator** |
|  | |
| [**GameManager**](class_game_manager.html) | **\_gameManager** |
|  | |
| [**SoundManager**](class_sound_manager.html) | **\_soundManager** |
|  | |
| [**ColorManager**](class_color_manager.html) | **\_colorManager** |
|  | |

## Detailed Description

Script to avoid duplicate code.

**PlayerPrefsX Class Reference**



A player pref extension [More...](class_player_prefs_x.html#details)

|  |  |
| --- | --- |
| Static Public Member Functions | |
| static bool | **SetBool** (String name, bool value) |
|  | |
| static bool | **GetBool** (String name) |
|  | |
| static bool | **GetBool** (String name, bool defaultValue) |
|  | |
| static long | **GetLong** (string key, long defaultValue) |
|  | |
| static long | **GetLong** (string key) |
|  | |
| static void | **SetLong** (string key, long value) |
|  | |
| static bool | **SetVector2** (String key, Vector2 vector) |
|  | |
| static Vector2 | **GetVector2** (String key, Vector2 defaultValue) |
|  | |
| static bool | **SetVector3** (String key, Vector3 vector) |
|  | |
| static Vector3 | **GetVector3** (String key) |
|  | |
| static Vector3 | **GetVector3** (String key, Vector3 defaultValue) |
|  | |
| static bool | **SetQuaternion** (String key, Quaternion vector) |
|  | |
| static Quaternion | **GetQuaternion** (String key) |
|  | |
| static Quaternion | **GetQuaternion** (String key, Quaternion defaultValue) |
|  | |
| static bool | **SetColor** (String key, Color color) |
|  | |
| static Color | **GetColor** (String key) |
|  | |
| static Color | **GetColor** (String key, Color defaultValue) |
|  | |
| static bool | **SetBoolArray** (String key, bool[] boolArray) |
|  | |
| static bool[] | **GetBoolArray** (String key) |
|  | |
| static bool[] | **GetBoolArray** (String key, bool defaultValue, int defaultSize) |
|  | |
| static bool | **SetStringArray** (String key, String[] stringArray) |
|  | |
| static String[] | **GetStringArray** (String key) |
|  | |
| static String[] | **GetStringArray** (String key, String defaultValue, int defaultSize) |
|  | |
| static bool | **SetIntArray** (String key, int[] intArray) |
|  | |
| static bool | **SetFloatArray** (String key, float[] floatArray) |
|  | |
| static bool | **SetVector2Array** (String key, Vector2[] vector2Array) |
|  | |
| static bool | **SetVector3Array** (String key, Vector3[] vector3Array) |
|  | |
| static bool | **SetQuaternionArray** (String key, Quaternion[] quaternionArray) |
|  | |
| static bool | **SetColorArray** (String key, Color[] colorArray) |
|  | |
| static int[] | **GetIntArray** (String key) |
|  | |
| static int[] | **GetIntArray** (String key, int defaultValue, int defaultSize) |
|  | |
| static float[] | **GetFloatArray** (String key) |
|  | |
| static float[] | **GetFloatArray** (String key, float defaultValue, int defaultSize) |
|  | |
| static Vector2[] | **GetVector2Array** (String key) |
|  | |
| static Vector2[] | **GetVector2Array** (String key, Vector2 defaultValue, int defaultSize) |
|  | |
| static Vector3[] | **GetVector3Array** (String key) |
|  | |
| static Vector3[] | **GetVector3Array** (String key, Vector3 defaultValue, int defaultSize) |
|  | |
| static Quaternion[] | **GetQuaternionArray** (String key) |
|  | |
| static Quaternion[] | **GetQuaternionArray** (String key, Quaternion defaultValue, int defaultSize) |
|  | |
| static Color[] | **GetColorArray** (String key) |
|  | |
| static Color[] | **GetColorArray** (String key, Color defaultValue, int defaultSize) |
|  | |
| static void | **ShowArrayType** (String key) |
|  | |

|  |  |
| --- | --- |
| Private Types | |
| enum | **ArrayType** {    **Float**, **Int32**, **Bool**, **String**,    **Vector2**, **Vector3**, **Quaternion**, **Color**  } |
|  | |

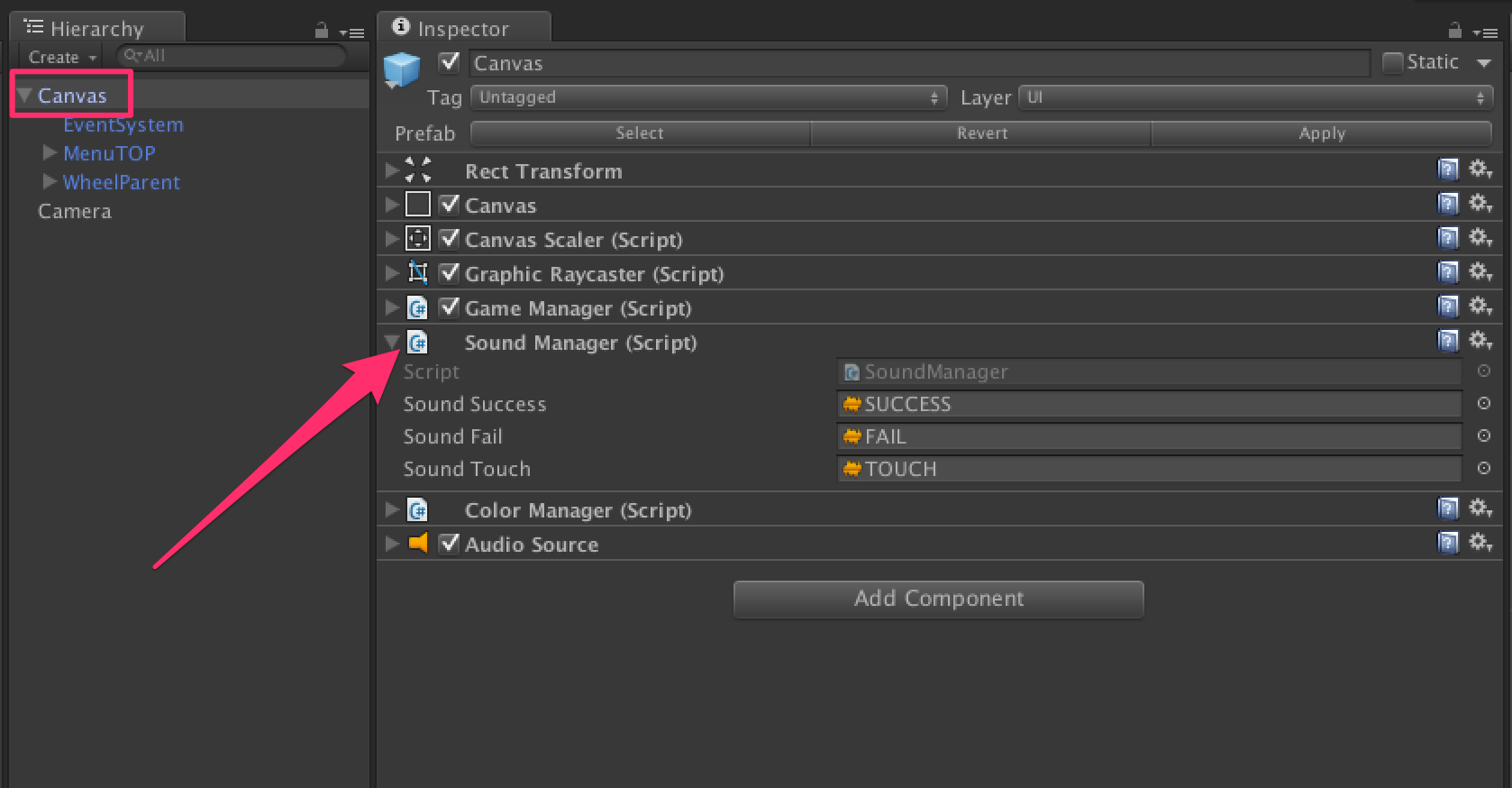
|  |  |
| --- | --- |
| Static Private Member Functions | |
| static void | **SplitLong** (long input, out int lowBits, out int highBits) |
|  | |
| static Vector2 | **GetVector2** (String key) |
|  | |
| static bool | **SetValue< T >** (String key, T array, ArrayType arrayType, int vectorNumber, Action< T, byte[], int > convert) |
|  | |
| static void | **ConvertFromInt** (int[] array, byte[] bytes, int i) |
|  | |
| static void | **ConvertFromFloat** (float[] array, byte[] bytes, int i) |
|  | |
| static void | **ConvertFromVector2** (Vector2[] array, byte[] bytes, int i) |
|  | |
| static void | **ConvertFromVector3** (Vector3[] array, byte[] bytes, int i) |
|  | |
| static void | **ConvertFromQuaternion** (Quaternion[] array, byte[] bytes, int i) |
|  | |
| static void | **ConvertFromColor** (Color[] array, byte[] bytes, int i) |
|  | |
| static void | **GetValue< T >** (String key, T list, ArrayType arrayType, int vectorNumber, Action< T, byte[]> convert) |
|  | |
| static void | **ConvertToInt** (List< int > list, byte[] bytes) |
|  | |
| static void | **ConvertToFloat** (List< float > list, byte[] bytes) |
|  | |
| static void | **ConvertToVector2** (List< Vector2 > list, byte[] bytes) |
|  | |
| static void | **ConvertToVector3** (List< Vector3 > list, byte[] bytes) |
|  | |
| static void | **ConvertToQuaternion** (List< Quaternion > list, byte[] bytes) |
|  | |
| static void | **ConvertToColor** (List< Color > list, byte[] bytes) |
|  | |
| static void | **Initialize** () |
|  | |
| static bool | **SaveBytes** (String key, byte[] bytes) |
|  | |
| static void | **ConvertFloatToBytes** (float f, byte[] bytes) |
|  | |
| static float | **ConvertBytesToFloat** (byte[] bytes) |
|  | |
| static void | **ConvertInt32ToBytes** (int i, byte[] bytes) |
|  | |
| static int | **ConvertBytesToInt32** (byte[] bytes) |
|  | |
| static void | **ConvertTo4Bytes** (byte[] bytes) |
|  | |
| static void | **ConvertFrom4Bytes** (byte[] bytes) |
|  | |

|  |  |
| --- | --- |
| Static Private Attributes | |
| static int | **endianDiff1** |
|  | |
| static int | **endianDiff2** |
|  | |
| static int | **idx** |
|  | |
| static byte[] | **byteBlock** |
|  | |

## Detailed Description

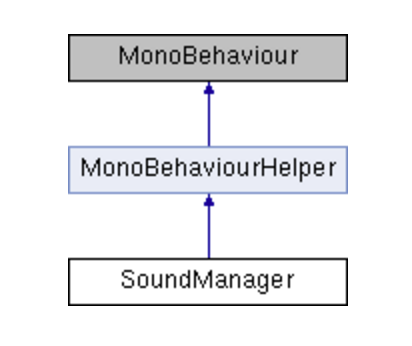
A player pref extension

**SoundManager Class Reference**



Class in charge to play FX in the game. Attached to the Canvas game object. Change the audioSource to customize the sounds. [More...](class_sound_manager.html#details)

Inheritance diagram for SoundManager:



|  |  |
| --- | --- |
| Public Member Functions | |
| void | [**PlaySuccess**](class_sound_manager.html#accc871d4d9436769ed5d1d006d9f2362) () |
|  | Method called when the level is clear = success [More...](#accc871d4d9436769ed5d1d006d9f2362) |
|  | |
| void | [**PlayFail**](class_sound_manager.html#aa06c2d58f950997e0c7ef669feb848ac) () |
|  | Method called when game over [More...](#aa06c2d58f950997e0c7ef669feb848ac) |
|  | |
| void | [**PlayTouch**](class_sound_manager.html#a98e759bc6bb563eaa7fb869984489db0) () |
|  | Method called when the player tap at the good moment on the screen [More...](#a98e759bc6bb563eaa7fb869984489db0) |
|  | |

|  |  |
| --- | --- |
| Private Member Functions | |
| void | [**Awake**](class_sound_manager.html#a06a8d04a0f9ce5d140b5308a4adb9073) () |
|  | Find the audiosource attached to the same game object [More...](#a06a8d04a0f9ce5d140b5308a4adb9073) |
|  | |

|  |  |
| --- | --- |
| Private Attributes | |
| AudioSource | [**audioSource**](class_sound_manager.html#a8feda9b9a6c3a286b38bc19e00e69adc) |
|  | Reference to the audiosouce use to play fx, attached to the same game object [More...](#a8feda9b9a6c3a286b38bc19e00e69adc) |
|  | |
| AudioClip | [**soundSuccess**](class_sound_manager.html#a96f3a0ecd01e5321b8fa31d93b3991d6) |
|  | Sound played when the level is clear = success [More...](#a96f3a0ecd01e5321b8fa31d93b3991d6) |
|  | |
| AudioClip | [**soundFail**](class_sound_manager.html#aa04c95545cc95c26dbbf94044e950378) |
|  | Sound played when game over [More...](#aa04c95545cc95c26dbbf94044e950378) |
|  | |
| AudioClip | [**soundTouch**](class_sound_manager.html#ab9c644abbea15676695996f58cf9a289) |
|  | Sound played when the player tap at the good moment on the screen [More...](#ab9c644abbea15676695996f58cf9a289) |
|  | |

|  |
| --- |
| Additional Inherited Members |
| **Properties inherited from**[**MonoBehaviourHelper**](class_mono_behaviour_helper.html) |

## Detailed Description

Class in charge to play FX in the game. Attached to the Canvas game object. Change the audioSource to customize the sounds.

## Member Function Documentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void SoundManager.Awake** | **(** |  | **)** |  | | inlineprivate |

Find the audiosource attached to the same game object

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void SoundManager.PlayFail** | **(** |  | **)** |  | | inline |

Method called when game over

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void SoundManager.PlaySuccess** | **(** |  | **)** |  | | inline |

Method called when the level is clear = success

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void SoundManager.PlayTouch** | **(** |  | **)** |  | | inline |

Method called when the player tap at the good moment on the screen

## Member Data Documentation

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **AudioSource SoundManager.audioSource** | | private |

Reference to the audiosouce use to play fx, attached to the same game object

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **AudioClip SoundManager.soundFail** | | private |

Sound played when game over

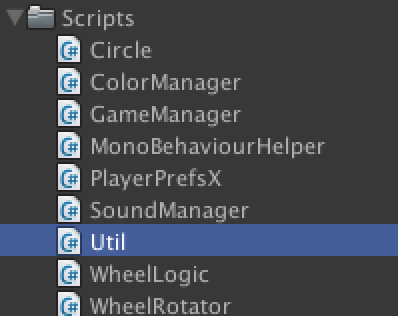
|  |  |  |
| --- | --- | --- |
| |  | | --- | | **AudioClip SoundManager.soundSuccess** | | private |

Sound played when the level is clear = success

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **AudioClip SoundManager.soundTouch** | | private |

Sound played when the player tap at the good moment on the screen

**Util Class Reference**



Utility class. This class is static, so you can use it in all your projects! [More...](class_util.html#details)

|  |  |
| --- | --- |
| Static Public Member Functions | |
| static bool | [**IsEqual**](class_util.html#acc8dde1bc09aa851903c44e41be76a4a) (this Color c, Color o) |
|  | Compare two colors [More...](#acc8dde1bc09aa851903c44e41be76a4a) |
|  | |
| static void | [**Shuffle< T >**](class_util.html#ab86c80a4a0f40bea569bd629447d8fe7) (this IList< T > list) |
|  | Real shuffle of List [More...](#ab86c80a4a0f40bea569bd629447d8fe7) |
|  | |
| static int | [**GetCurrentLevel**](class_util.html#a7b8c6b7f829d060007d9fb01f794a247) () |
|  | Get the current level [More...](#a7b8c6b7f829d060007d9fb01f794a247) |
|  | |
| static bool | [**HavePreviousLevel**](class_util.html#ab6800b62ee6b87d99b76654f04bd8ed2) () |
|  | Check if there is a previous level [More...](#ab6800b62ee6b87d99b76654f04bd8ed2) |
|  | |
| static bool | [**HaveNextLevel**](class_util.html#a9b956a34ab24bdf4d4a4721ff452d0e0) () |
|  | Check if there is a next level, ie. if the next level is unlocked by the player [More...](#a9b956a34ab24bdf4d4a4721ff452d0e0) |
|  | |
| static int | [**GetMaxLevel**](class_util.html#a28043ddde202d5fe6b2b2071bc205700) () |
|  | Get the max level unlocked by the player [More...](#a28043ddde202d5fe6b2b2071bc205700) |
|  | |
| static void | [**SetMaxLevel**](class_util.html#ad172b7c8923b5a747d340a0169e3b6cb) (int level) |
|  | Set the max level unlocked by the player [More...](#ad172b7c8923b5a747d340a0169e3b6cb) |
|  | |
| static void | [**SetCurrentLevel**](class_util.html#a3d9a093769255ea46f7ed6e6af8b7151) (int level) |
|  | Set the current played level [More...](#a3d9a093769255ea46f7ed6e6af8b7151) |
|  | |
| static void | [**ReloadLevel**](class_util.html#a72c46e4d2c3f9d875b03dc1ce321bd2d) () |
|  | Clean the memory and reload the scene [More...](#a72c46e4d2c3f9d875b03dc1ce321bd2d) |
|  | |
| static void | [**CleanMemory**](class_util.html#a9b40d034af9869cb4bf7d5c0f9f7e906) () |
|  | Clean the memory [More...](#a9b40d034af9869cb4bf7d5c0f9f7e906) |
|  | |
| static bool | [**RestartFromGameOver**](class_util.html#abc5dc671111c4c32e6346e64a8ed305e) () |
|  | Resturn true if last time we play we lose (= Game Over) [More...](#abc5dc671111c4c32e6346e64a8ed305e) |
|  | |
| static void | [**SetRestartFromGameOver**](class_util.html#a793da1a5f332515c785f56231adc961e) () |
|  | Set restart from game over = true [More...](#a793da1a5f332515c785f56231adc961e) |
|  | |
| static void | [**SetNotRestartFromGameOver**](class_util.html#a748475d4276eeb73b3a27ea328dc38d2) () |
|  | Set restart from game over = false [More...](#a748475d4276eeb73b3a27ea328dc38d2) |
|  | |

|  |  |
| --- | --- |
| Static Private Attributes | |
| static System.Random | **rng** = new System.Random() |
|  | |

## Detailed Description

Utility class. This class is static, so you can use it in all your projects!

## Member Function Documentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static void Util.CleanMemory** | **(** |  | **)** |  | | inlinestatic |

Clean the memory

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static int Util.GetCurrentLevel** | **(** |  | **)** |  | | inlinestatic |

Get the current level

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static int Util.GetMaxLevel** | **(** |  | **)** |  | | inlinestatic |

Get the max level unlocked by the player

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static bool Util.HaveNextLevel** | **(** |  | **)** |  | | inlinestatic |

Check if there is a next level, ie. if the next level is unlocked by the player

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static bool Util.HavePreviousLevel** | **(** |  | **)** |  | | inlinestatic |

Check if there is a previous level

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **static bool Util.IsEqual** | **(** | **this Color** | **c,** | |  |  | **Color** | **o** | |  | **)** |  |  | | inlinestatic |

Compare two colors

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static void Util.ReloadLevel** | **(** |  | **)** |  | | inlinestatic |

Clean the memory and reload the scene

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static bool Util.RestartFromGameOver** | **(** |  | **)** |  | | inlinestatic |

Resturn true if last time we play we lose (= Game Over)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **static void Util.SetCurrentLevel** | **(** | **int** | **level** | **)** |  | | inlinestatic |

Set the current played level

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **static void Util.SetMaxLevel** | **(** | **int** | **level** | **)** |  | | inlinestatic |

Set the max level unlocked by the player

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static void Util.SetNotRestartFromGameOver** | **(** |  | **)** |  | | inlinestatic |

Set restart from game over = false

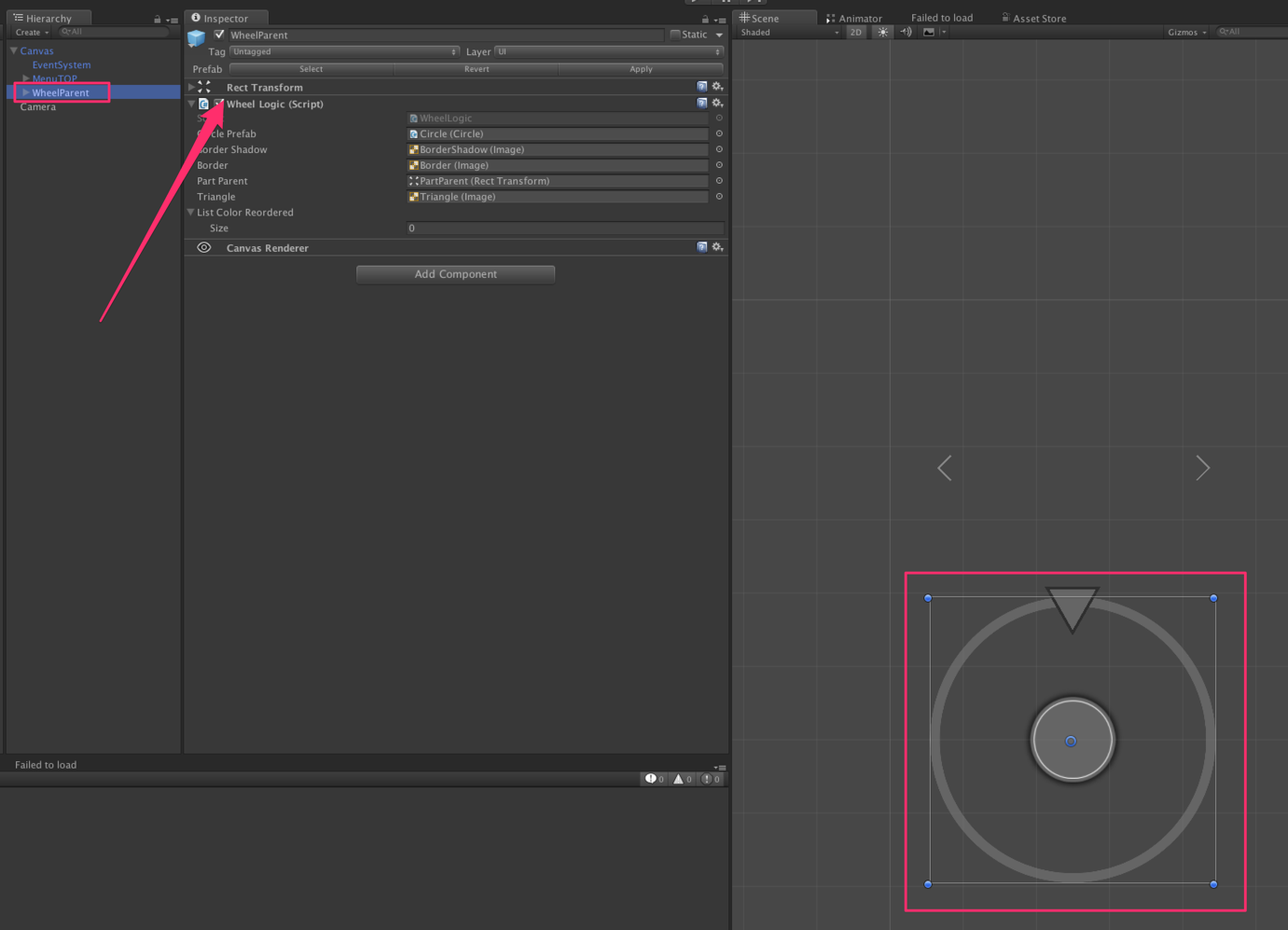
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static void Util.SetRestartFromGameOver** | **(** |  | **)** |  | | inlinestatic |

Set restart from game over = true

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **static void Util.Shuffle< T >** | **(** | **this IList< T >** | **list** | **)** |  | | inlinestatic |

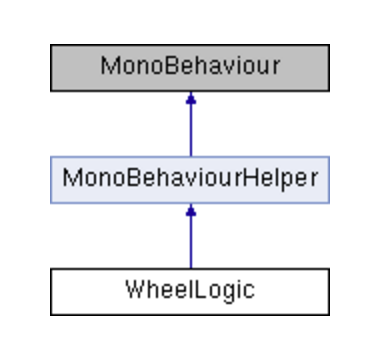
Real shuffle of List

**WheelLogic Class Reference**



In charge of all the wheel logic. Attached to the game object: "WheelParent". Create the colors, Spawn each element of the wheel. Check the color when the player tap the screen etc... [More...](class_wheel_logic.html#details)

Inheritance diagram for WheelLogic:



|  |  |
| --- | --- |
| Public Member Functions | |
| void | [**DOColorTriangle**](class_wheel_logic.html#a6d9ed5db0971bb8be721281db10c4820) () |
|  | Change the color of the triangle = color to find [More...](#a6d9ed5db0971bb8be721281db10c4820) |
|  | |
| bool | [**CheckIfTriangleEqualWheelColor**](class_wheel_logic.html#af659559e7183ffc24f01c4a5291f2cea) () |
|  | Check if the player tap at the good moment on the screen, ie. check if the color of the triangle = the color of the part of the wheel below the triangle [More...](#af659559e7183ffc24f01c4a5291f2cea) |
|  | |
| bool | [**DOCheck**](class_wheel_logic.html#a07789174331962e348744510848b43c7) () |
|  | Call the method CheckIfTriangleEqualWheelColor. If true, move done = minus 1 moove in the total move the player have to do to clear the level. If false => game over [More...](#a07789174331962e348744510848b43c7) |
|  | |

|  |  |
| --- | --- |
| Public Attributes | |
| [**Circle**](class_circle.html) | [**circlePrefab**](class_wheel_logic.html#ab523ade1cd49a2b0297f300d21666b3a) |
|  | Prefab of [**Circle**](class_circle.html). Use to create the wheel. Each part is a UI Image with a certain fillAmount [More...](#ab523ade1cd49a2b0297f300d21666b3a) |
|  | |
| float | [**speedWheel**](class_wheel_logic.html#a874989106dc2c5b6ae9734f3323d2c52) = 3f |
|  | Sped of the wheel, in seconds (total time in seconds to make 360 degree rotation), for the current level [More...](#a874989106dc2c5b6ae9734f3323d2c52) |
|  | |
| Image | [**borderShadow**](class_wheel_logic.html#a69b3421da6055145db2f0c7d3de7df54) |
|  | Image of the shadow of border of the wheek [More...](#a69b3421da6055145db2f0c7d3de7df54) |
|  | |
| Image | [**border**](class_wheel_logic.html#a7d3eb653406c820856640e3097d1a6d4) |
|  | Image of the border of the wheek [More...](#a7d3eb653406c820856640e3097d1a6d4) |
|  | |
| RectTransform | [**partParent**](class_wheel_logic.html#ab9ddc2a2f45ff04b2d1860eceb3b1688) |
|  | Reference to the GameObject who contains all the part of the wheel we will spawn [More...](#ab9ddc2a2f45ff04b2d1860eceb3b1688) |
|  | |
| Image | [**triangle**](class_wheel_logic.html#a738df018347a2592b7597dbb691a6d49) |
|  | Reference to the Triangle Image on the top of the wheel [More...](#a738df018347a2592b7597dbb691a6d49) |
|  | |
| List< Color > | [**listColorReordered**](class_wheel_logic.html#a9c24ba9762bd412e833e99cc9aa806a0) = new List<Color>() |
|  | Reference to a list of color built for a level [More...](#a9c24ba9762bd412e833e99cc9aa806a0) |
|  | |

|  |  |
| --- | --- |
| Private Member Functions | |
| void | [**Awake**](class_wheel_logic.html#a95db6758986c93f5d5f0ca056321f5c7) () |
|  | Create a new list of corlors for this level, randomly : listColorReordered and save it in [**PlayerPrefsX**](class_player_prefs_x.html) to use the same list of colors in case of game over [More...](#a95db6758986c93f5d5f0ca056321f5c7) |
|  | |
| void | [**Start**](class_wheel_logic.html#a8289cb7797e094bb75e1571d636d0e96) () |
|  | Place the border and the border shadow at the good place [More...](#a8289cb7797e094bb75e1571d636d0e96) |
|  | |
| void | [**DefineLevel**](class_wheel_logic.html#aeb20c5a177d1feac435cb370827ca5d1) () |
|  | IMPORTANT ==> It's here we define the levels. Change the formulas if you want. [More...](#aeb20c5a177d1feac435cb370827ca5d1) |
|  | |
| void | [**BuildWheel**](class_wheel_logic.html#af2fcb368f0e97014b1b54f3f9b6856f4) () |
|  | Method to build the wheel. Each part of the wheel is an UI Image, type = fill image. We use the fill amout property to cretae the parts of the wheel [More...](#af2fcb368f0e97014b1b54f3f9b6856f4) |
|  | |
| [**Circle**](class_circle.html) | [**InstantiateCircle**](class_wheel_logic.html#ad63971956205d4af21df59038b111a5a) () |
|  | Method to create a new circle = new part of the wheel [More...](#ad63971956205d4af21df59038b111a5a) |
|  | |
| [**Circle**](class_circle.html) | [**InstantiateCircle**](class_wheel_logic.html#a674e649d84c7fe63fd01eed693168713) (float fillAmout, float angle, Color c) |
|  | Method to create a new circle = new part of the wheel [More...](#a674e649d84c7fe63fd01eed693168713) |
|  | |

|  |  |
| --- | --- |
| Private Attributes | |
| int | [**numOfPart**](class_wheel_logic.html#a9d6a97fd78c1aacd9fb0f690e7ab2bb8) = 12 |
|  | Number of parts in the wheel, for the current level [More...](#a9d6a97fd78c1aacd9fb0f690e7ab2bb8) |
|  | |
| int | [**numOfColor**](class_wheel_logic.html#a637ba06b684196cc9eecd9cf767f8ef9) = 3 |
|  | Number of colors in the wheel, for the current level [More...](#a637ba06b684196cc9eecd9cf767f8ef9) |
|  | |
| List< [**Circle**](class_circle.html) > | [**allCircles**](class_wheel_logic.html#a0e305dd8d560bc07df3d3ce00d64977e) = new List<[**Circle**](class_circle.html)>() |
|  | Reference to all the parts contained in the wheel, for the current level [More...](#a0e305dd8d560bc07df3d3ce00d64977e) |
|  | |
| Color | [**lastColor**](class_wheel_logic.html#a2819fe132399a92197af014e2de6a0ab) |
|  | Reference to the last color to find, to avoid duplicate check [More...](#a2819fe132399a92197af014e2de6a0ab) |
|  | |
| bool | [**firstChangeColor**](class_wheel_logic.html#a6d19ae44dfc38a49e40cf5aaaf1ffb79) = true |
|  | Is it the first time we ask for a color in the game, for this level? If yes, don't get the color behind the triangle [More...](#a6d19ae44dfc38a49e40cf5aaaf1ffb79) |
|  | |

|  |
| --- |
| Additional Inherited Members |
| **Properties inherited from**[**MonoBehaviourHelper**](class_mono_behaviour_helper.html) |

## Detailed Description

In charge of all the wheel logic. Attached to the game object: "WheelParent". Create the colors, Spawn each element of the wheel. Check the color when the player tap the screen etc...

## Member Function Documentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelLogic.Awake** | **(** |  | **)** |  | | inlineprivate |

Create a new list of corlors for this level, randomly : listColorReordered and save it in [**PlayerPrefsX**](class_player_prefs_x.html) to use the same list of colors in case of game over

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelLogic.BuildWheel** | **(** |  | **)** |  | | inlineprivate |

Method to build the wheel. Each part of the wheel is an UI Image, type = fill image. We use the fill amout property to cretae the parts of the wheel

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **bool WheelLogic.CheckIfTriangleEqualWheelColor** | **(** |  | **)** |  | | inline |

Check if the player tap at the good moment on the screen, ie. check if the color of the triangle = the color of the part of the wheel below the triangle

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelLogic.DefineLevel** | **(** |  | **)** |  | | inlineprivate |

IMPORTANT ==> It's here we define the levels. Change the formulas if you want.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **bool WheelLogic.DOCheck** | **(** |  | **)** |  | | inline |

Call the method CheckIfTriangleEqualWheelColor. If true, move done = minus 1 moove in the total move the player have to do to clear the level. If false => game over

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelLogic.DOColorTriangle** | **(** |  | **)** |  | | inline |

Change the color of the triangle = color to find

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | [**Circle**](class_circle.html)**WheelLogic.InstantiateCircle** | **(** |  | **)** |  | | inlineprivate |

Method to create a new circle = new part of the wheel

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | [**Circle**](class_circle.html)**WheelLogic.InstantiateCircle** | **(** | **float** | **fillAmout,** | |  |  | **float** | **angle,** | |  |  | **Color** | **c** | |  | **)** |  |  | | inlineprivate |

Method to create a new circle = new part of the wheel

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelLogic.Start** | **(** |  | **)** |  | | inlineprivate |

Place the border and the border shadow at the good place

## Member Data Documentation

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **List<**[**Circle**](class_circle.html)**> WheelLogic.allCircles = new List<**[**Circle**](class_circle.html)**>()** | | private |

Reference to all the parts contained in the wheel, for the current level

|  |
| --- |
| **Image WheelLogic.border** |

Image of the border of the wheek

|  |
| --- |
| **Image WheelLogic.borderShadow** |

Image of the shadow of border of the wheek

|  |
| --- |
| [**Circle**](class_circle.html)**WheelLogic.circlePrefab** |

Prefab of [**Circle**](class_circle.html). Use to create the wheel. Each part is a UI Image with a certain fillAmount

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **bool WheelLogic.firstChangeColor = true** | | private |

Is it the first time we ask for a color in the game, for this level? If yes, don't get the color behind the triangle

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Color WheelLogic.lastColor** | | private |

Reference to the last color to find, to avoid duplicate check

|  |
| --- |
| **List<Color> WheelLogic.listColorReordered = new List<Color>()** |

Reference to a list of color built for a level

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **int WheelLogic.numOfColor = 3** | | private |

Number of colors in the wheel, for the current level

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **int WheelLogic.numOfPart = 12** | | private |

Number of parts in the wheel, for the current level

|  |
| --- |
| **RectTransform WheelLogic.partParent** |

Reference to the GameObject who contains all the part of the wheel we will spawn

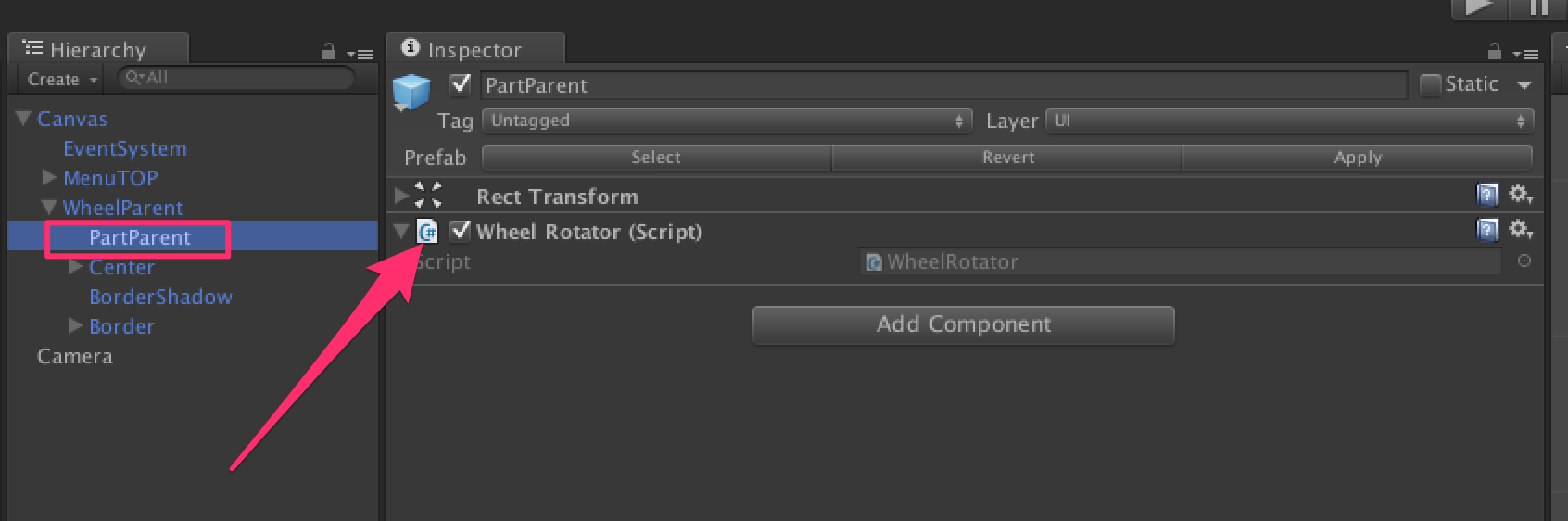
|  |
| --- |
| **float WheelLogic.speedWheel = 3f** |

Sped of the wheel, in seconds (total time in seconds to make 360 degree rotation), for the current level

|  |
| --- |
| **Image WheelLogic.triangle** |

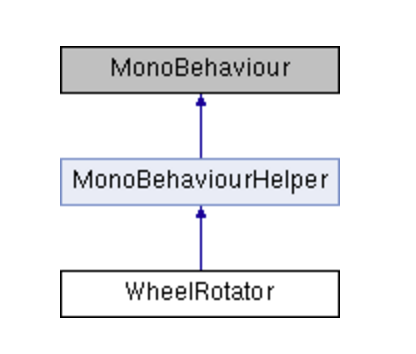
Reference to the Triangle Image on the top of the wheel

**WheelRotator Class Reference**



In charge of the rotation of the wheel and of the input in the game (who will stop the rotation, check the color, and start the rotation in the other direction). Attached to the game object: "PartParent". [More...](class_wheel_rotator.html#details)

Inheritance diagram for WheelRotator:



|  |  |
| --- | --- |
| Public Attributes | |
| int | [**direction**](class_wheel_rotator.html#a27e011ef597238d858a93666febd865e) = 1 |
|  | Two directions: left and right (1 and -1) [More...](#a27e011ef597238d858a93666febd865e) |
|  | |

|  |  |
| --- | --- |
| Properties | |
| bool | [**isSuccess**](class_wheel_rotator.html#a49b14dfb8a22e1b9b0af3c2aa8b348a6) [get] |
|  | Check if success = number of moove to do = 0 [More...](#a49b14dfb8a22e1b9b0af3c2aa8b348a6) |
|  | |
| **Properties inherited from**[**MonoBehaviourHelper**](class_mono_behaviour_helper.html) | |

|  |  |
| --- | --- |
| Private Member Functions | |
| void | [**Awake**](class_wheel_rotator.html#ae12ed0fb0b0960de1c8309ea906e1898) () |
|  | Choose the start direction randmly and set firstStart to true [More...](#ae12ed0fb0b0960de1c8309ea906e1898) |
|  | |
| void | [**Update**](class_wheel_rotator.html#a71a388428b320f4139f5dae144f6d40f) () |
|  | Listen if the player tap or click, and if the game is not game over after the click (so triangle color = part of the wheel color) launch again the rotation but in the oposite direction [More...](#a71a388428b320f4139f5dae144f6d40f) |
|  | |
| void | [**DORotateWheel**](class_wheel_rotator.html#a0aa380a78215e5d846a3d14f939a734c) () |
|  | Start the rotation of the wheel. Check in each updates if the triangle enter a part of the wheel with the same color of him. If we are inside a same color and we go out, that means the player doesn't tap before the triangle go out of the part with the same color, so it's game over. [More...](#a0aa380a78215e5d846a3d14f939a734c) |
|  | |

|  |  |
| --- | --- |
| Private Attributes | |
| bool | [**firstStart**](class_wheel_rotator.html#ab9058c5cec7d2e317ca5878144333edf) = true |
|  | Is it the first time we start the rotation for the level? [More...](#ab9058c5cec7d2e317ca5878144333edf) |
|  | |
| Tweener | [**rotateTweener**](class_wheel_rotator.html#a46ff703b0ae7c83d5b7c52c1c3b37d47) |
|  | Reference to the tweener who rotate the circle [More...](#a46ff703b0ae7c83d5b7c52c1c3b37d47) |
|  | |

## Detailed Description

In charge of the rotation of the wheel and of the input in the game (who will stop the rotation, check the color, and start the rotation in the other direction). Attached to the game object: "PartParent".

## Member Function Documentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelRotator.Awake** | **(** |  | **)** |  | | inlineprivate |

Choose the start direction randmly and set firstStart to true

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelRotator.DORotateWheel** | **(** |  | **)** |  | | inlineprivate |

Start the rotation of the wheel. Check in each updates if the triangle enter a part of the wheel with the same color of him. If we are inside a same color and we go out, that means the player doesn't tap before the triangle go out of the part with the same color, so it's game over.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void WheelRotator.Update** | **(** |  | **)** |  | | inlineprivate |

Listen if the player tap or click, and if the game is not game over after the click (so triangle color = part of the wheel color) launch again the rotation but in the oposite direction

## Member Data Documentation

|  |
| --- |
| **int WheelRotator.direction = 1** |

Two directions: left and right (1 and -1)

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **bool WheelRotator.firstStart = true** | | private |

Is it the first time we start the rotation for the level?

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Tweener WheelRotator.rotateTweener** | | private |

Reference to the tweener who rotate the circle

## Property Documentation

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **bool WheelRotator.isSuccess** | | getprivate |

Check if success = number of moove to do = 0

**ADS :**

Everything is done for you : « Very Simple Ads » is already implemented.

Get it here : http://u3d.as/oWD

Thanks !

Our other assets : <http://u3d.as/9cs>

Inquiries : https://appadvisory.zendesk.com