**SPIN THE CIRCLE**

****

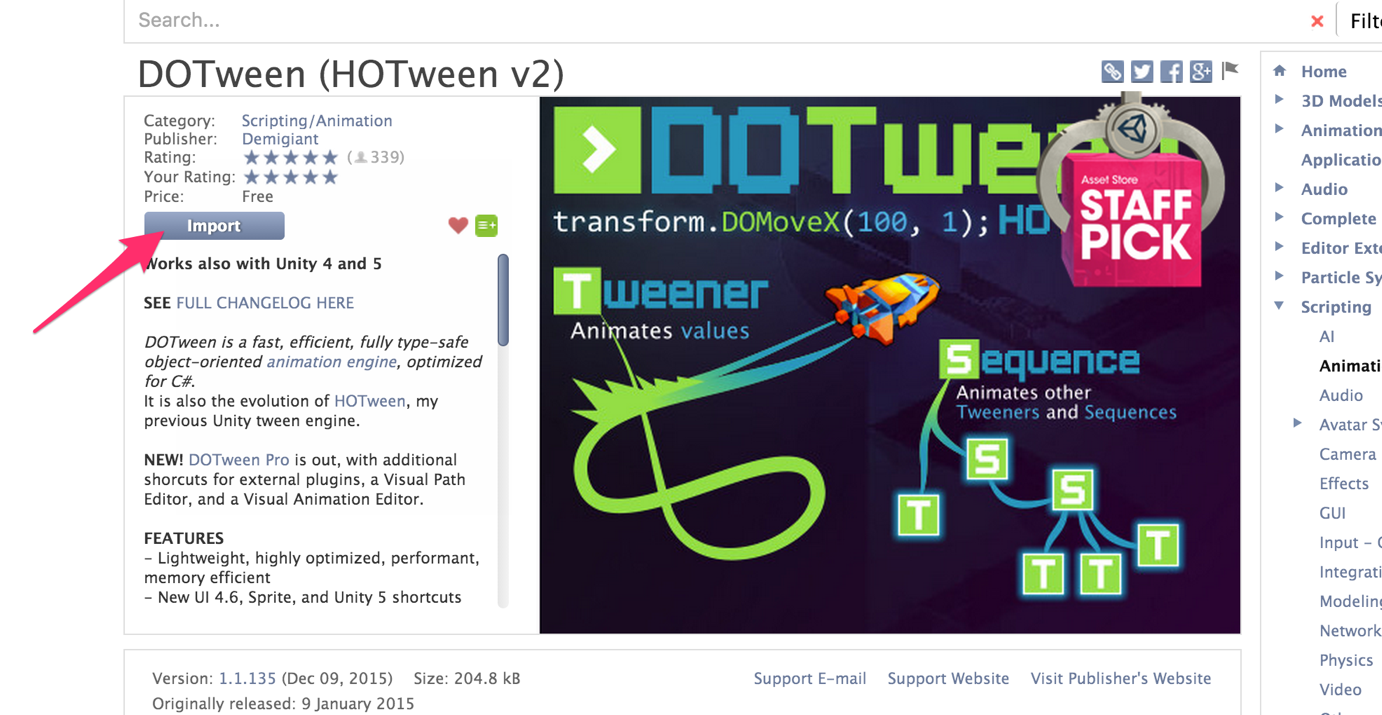
Thanks for your purchase.

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

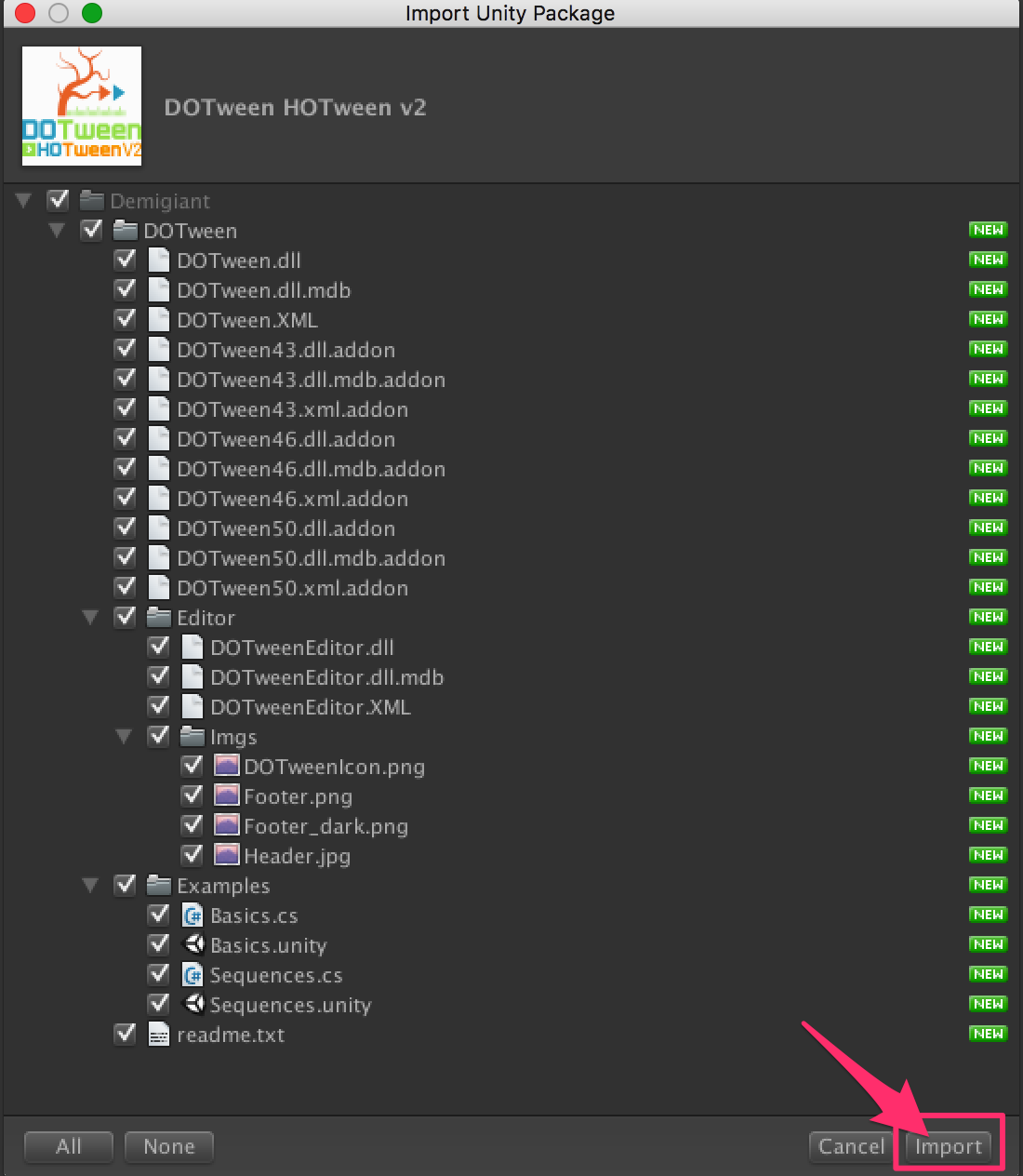
<http://u3d.as/aZ1>

(it’s free)

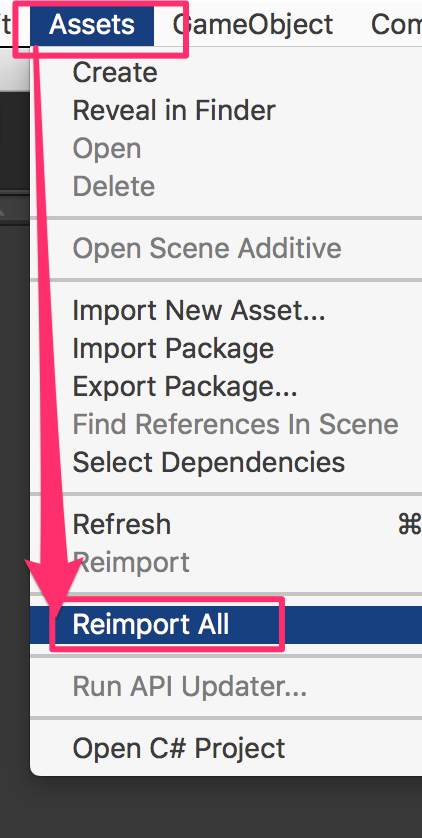
1/ Import Dotween from the asset store : <http://u3d.as/aZ1>



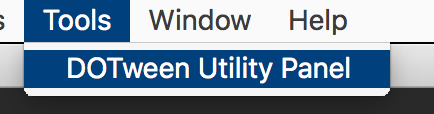
2/ Import the package into Unity



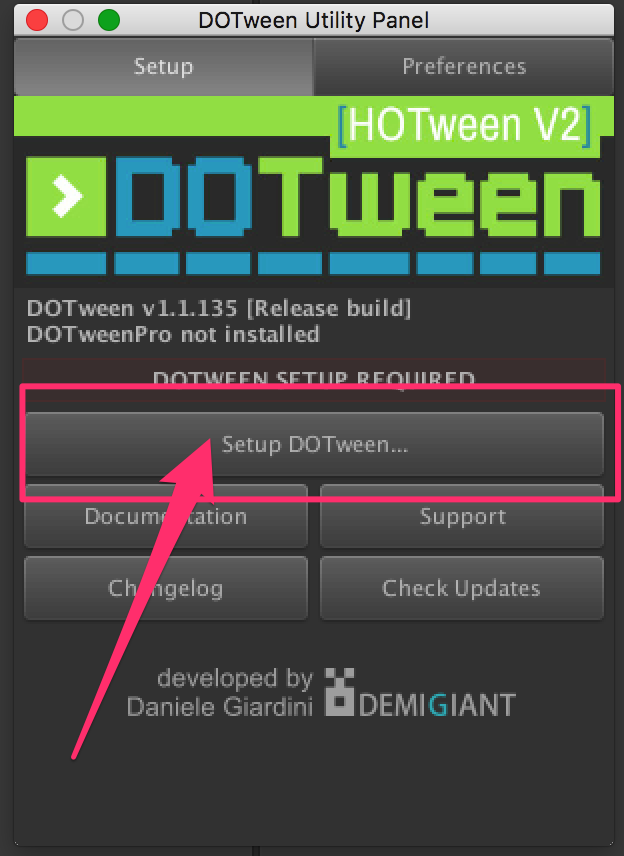
3/If you don’t see the « Tools » in the top of the Unity Screen, please do this :



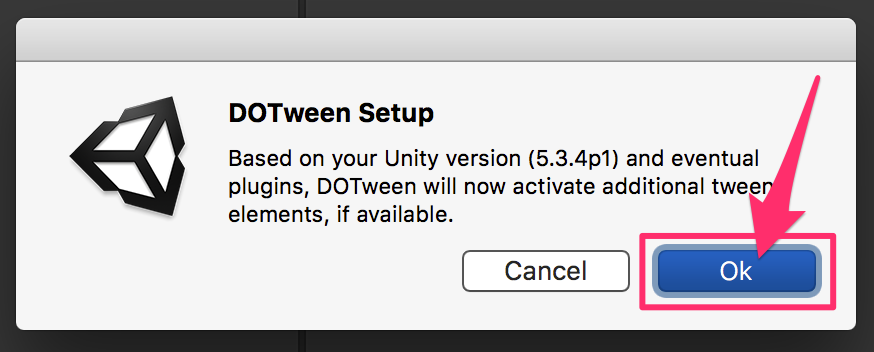
4/Now you have the « Tools ». Open it and click on « DOTween Utility Panel ».



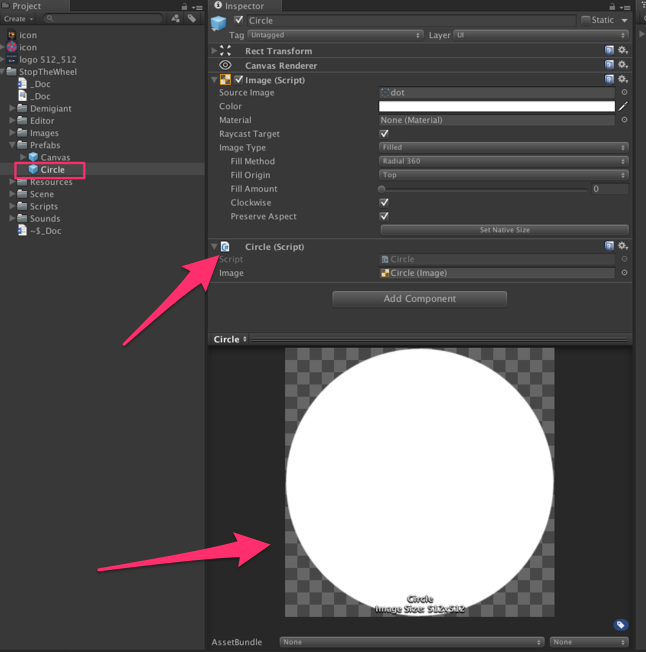
5/Click on « Setup DOTween ».



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

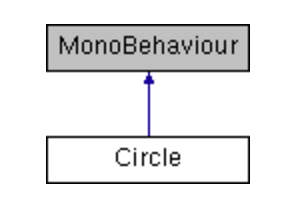


**CirclePart Class Reference**



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

Inheritance diagram for CirclePart:



|  |  |
| --- | --- |
| Public Member Functions | |
| [**CirclePart**](class_circle_part.html) | [**Init**](class_circle_part.html#a2cfc5a21914ae93685a3e32ae1e250b8) (float fillAmout, float angle, Color color) |
|  | Init the circle = the part of the circle. Each part is defined with a fillAmount = 1 / number of part in the circle, an angle and a color [More...](#a2cfc5a21914ae93685a3e32ae1e250b8) |
|  | |
| float | [**GetMiddleAngle**](class_circle_part.html#aec4ff75b02b9b7524aff50da0129ec23) () |
|  | Get the angle of the middle of the part of circle [More...](#aec4ff75b02b9b7524aff50da0129ec23) |
|  | |

|  |  |
| --- | --- |
| Public Attributes | |
| Image | [**image**](class_circle_part.html#af3b2915df0a906d4f5e2c42e8706881e) |
|  | The image = a simple circle [More...](#af3b2915df0a906d4f5e2c42e8706881e) |
|  | |

## Detailed Description

Each part of the circle is a circle. We use the fillAmount component of UI image to get "parts". All the circles are child of the Game Object PartParent (= CircleRotator). The Circle prefab is in the Prefabs folder. Each Circles are instantiate in the CircleLogic at the start of each level

## Member Function Documentation

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

Get the angle of the middle of the part of circle

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

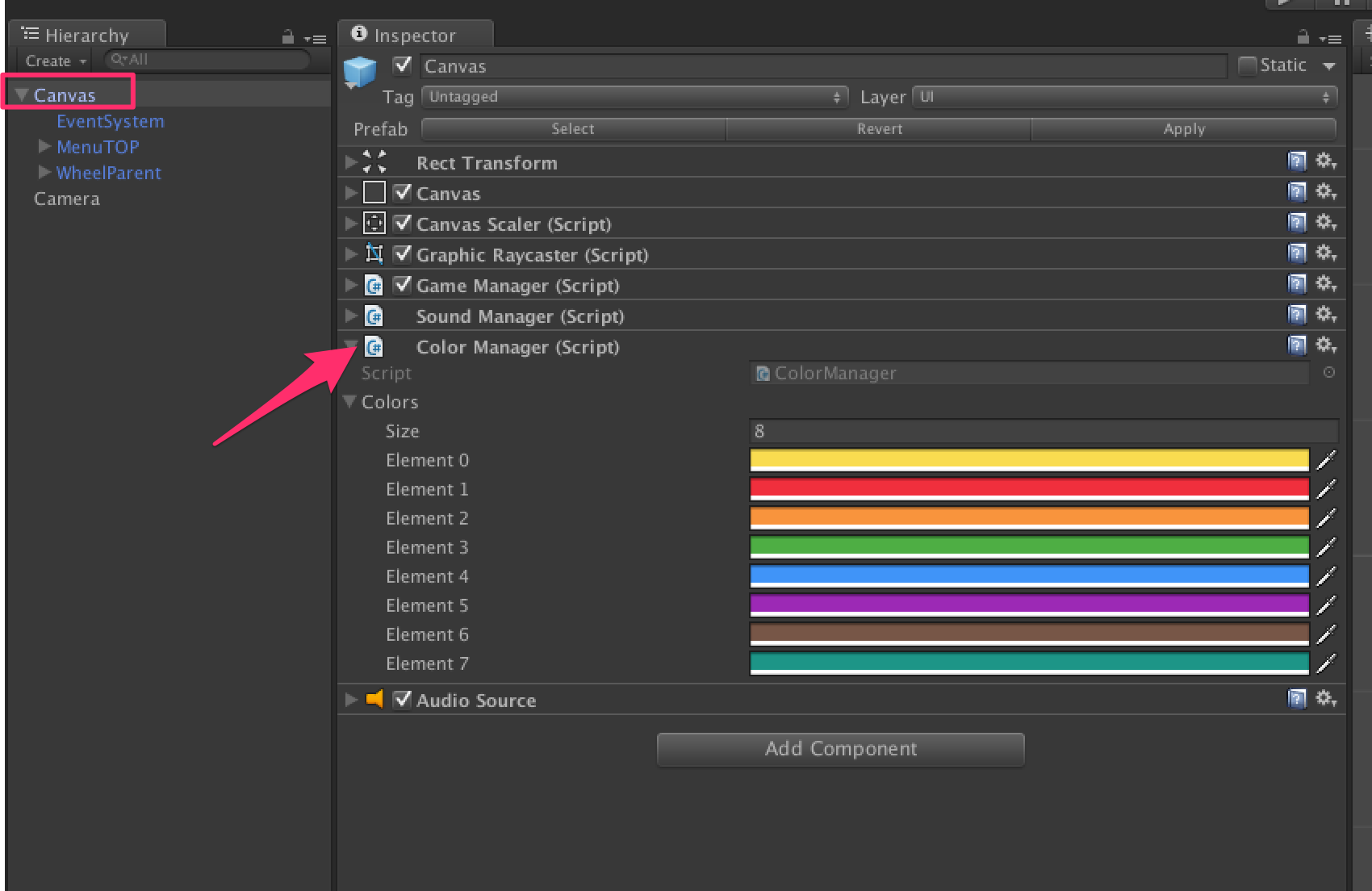
Init the circle = the part of the circle. Each part is defined with a fillAmount = 1 / number of part in the circle, an angle and a color

## Member Data Documentation

|  |
| --- |
| **Image CirclePart.image** |

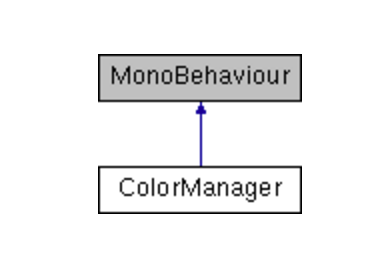
The image = a simple circle

**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:

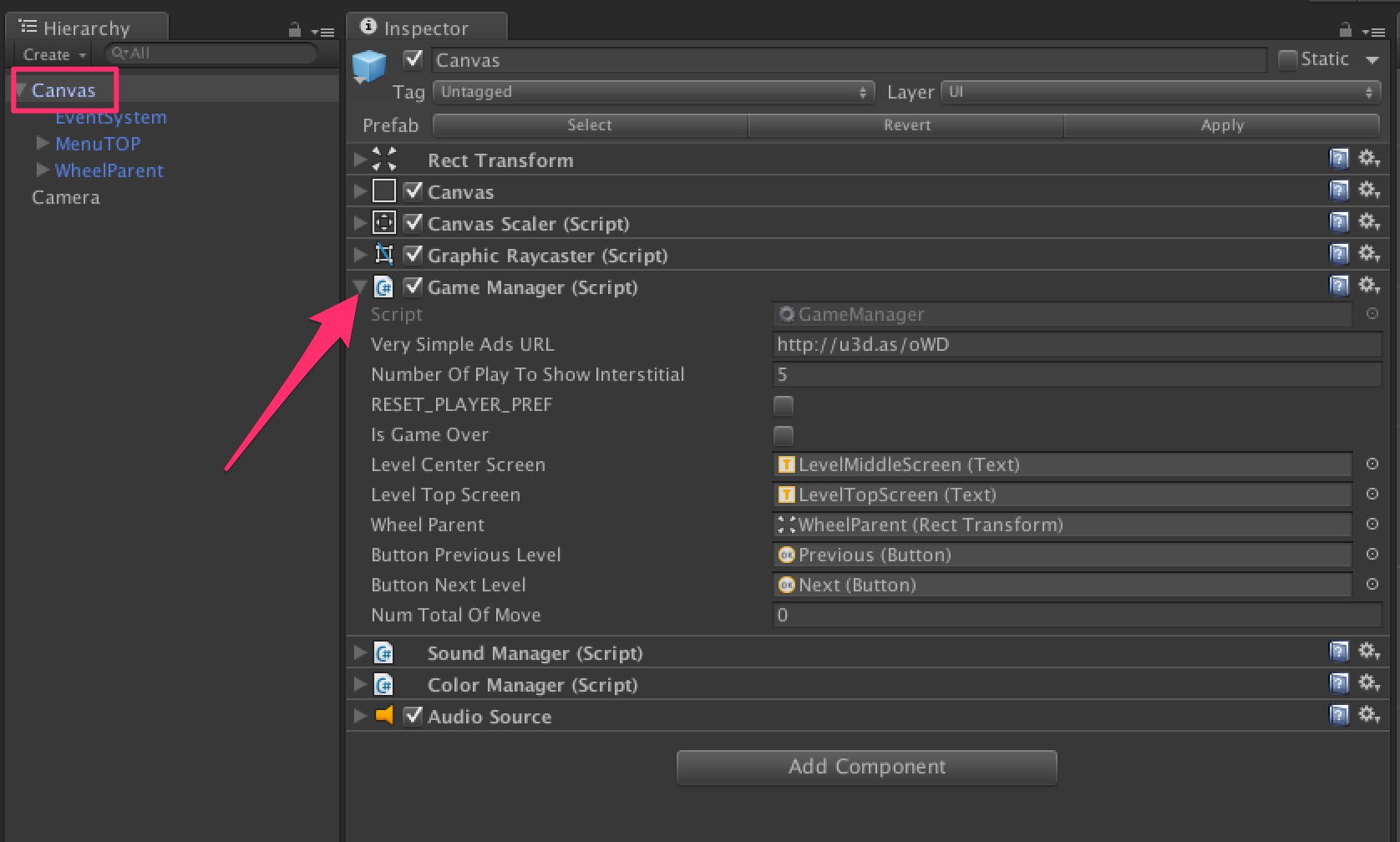


|  |  |
| --- | --- |
| Public Attributes | |
| Color[] | **colors** |
|  | |

Detailed Description

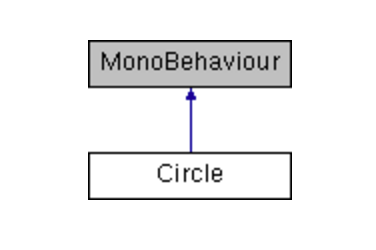
Class with an array of color. Change the array to customize the colors. Attached to the Canvas game object

**GameManager Class Reference**



In charge of the game logic: Game Start, Game Over, Score, Ads etc... Attached to the Canvas game object. In Charge to all the game management (game over, point, restart etc..) and in charge to show interstitial in the game. For monetizing this game with ads, everythign is already coded for you. You just need to get VERY SIMPLE ADS here: <http://u3d.as/oWD> [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 ball is not equal of the color of the part of the circle 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 | [**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 | **textLastScore** |
|  | |
| Text | **textBestScore** |
|  | |

|  |  |
| --- | --- |
| Properties | |
| int | [**point**](class_game_manager.html#afdd70504096fdc4919c45b386744a5d8) [get, set] |
|  | The number of move we have to do to clear this level = the level number [More...](#afdd70504096fdc4919c45b386744a5d8) |
|  | |

## Detailed Description

In charge of the game logic: Game Start, Game Over, Score, Ads etc... Attached to the Canvas game object. In Charge to all the game management (game over, point, restart etc..) and in charge to show interstitial in the game. FOr monetizing this game with ads, everythign is already coded for you. You just need to get VERY SIMPLE ADS here: <http://u3d.as/oWD>

## Member Function Documentation

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

When a move is done, ie. player tap on the screen and the color of the ball is not equal of the color of the part of the circle 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.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.ShowAds** | **(** |  | **)** |  | | inline |

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

## Member Data Documentation

|  |
| --- |
| **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

|  |
| --- |
| **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>

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

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

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

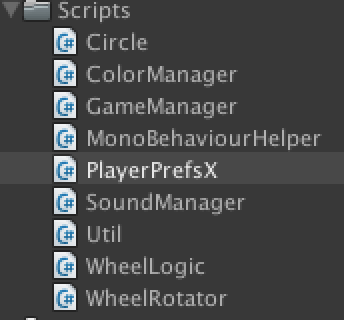
If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>

## Property Documentation

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **int GameManager.point** | | getset |

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

**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**  } |
|  | |

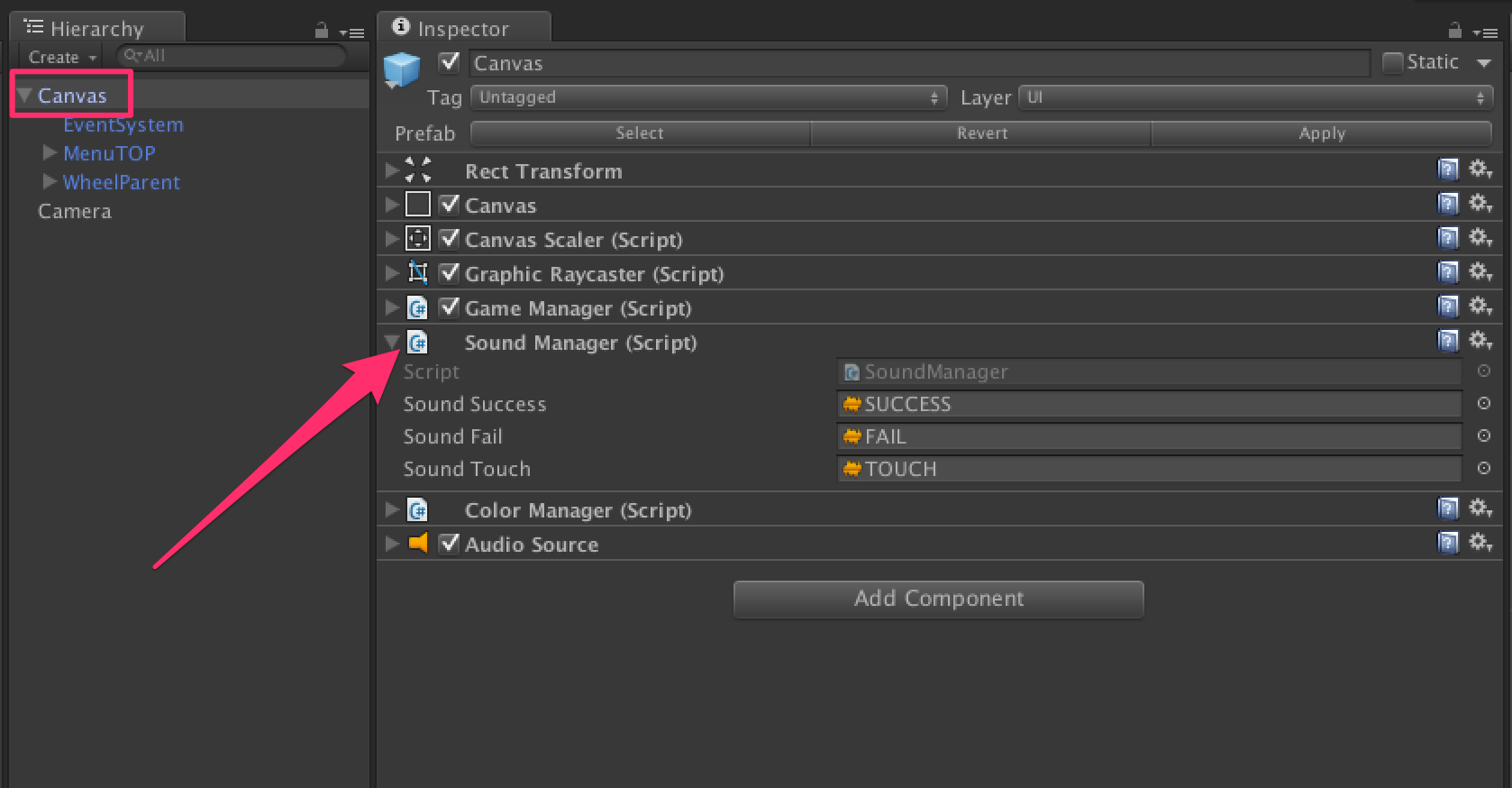
|  |  |
| --- | --- |
| Static Private Member Functions | |
| static void | **SplitLong** (long input, out int lowBits, out int highBits) |
|  | |
| static Vector2 | **GetVector2** (String key) |
|  | |
| static bool | **SetValue< T >** (String key, T array, ArrayType arrayType, int vectorNumber, Action< 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

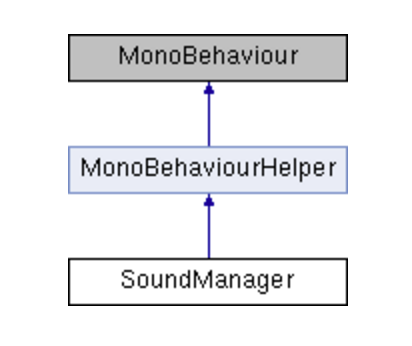
A player pref extension

**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

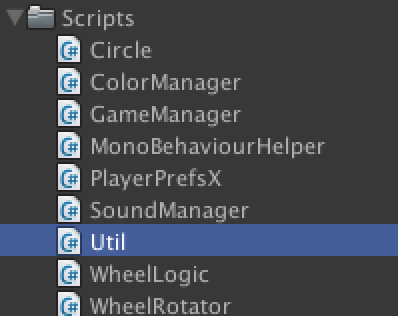
|  |  |  |
| --- | --- | --- |
| |  | | --- | | **AudioClip SoundManager.soundSuccess** | | private |

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 bool | **SetLastScore** (int score) |
|  | |
| static int | **GetLastScore** () |
|  | |
| static int | **GetBestScore** () |
|  | |
| 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 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 bool Util.IsEqual** | **(** | **this Color** | **c,** | |  |  | **Color** | **o** | |  | **)** |  |  | | inlinestatic |

Compare two colors

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

Clean the memory and reload the scene

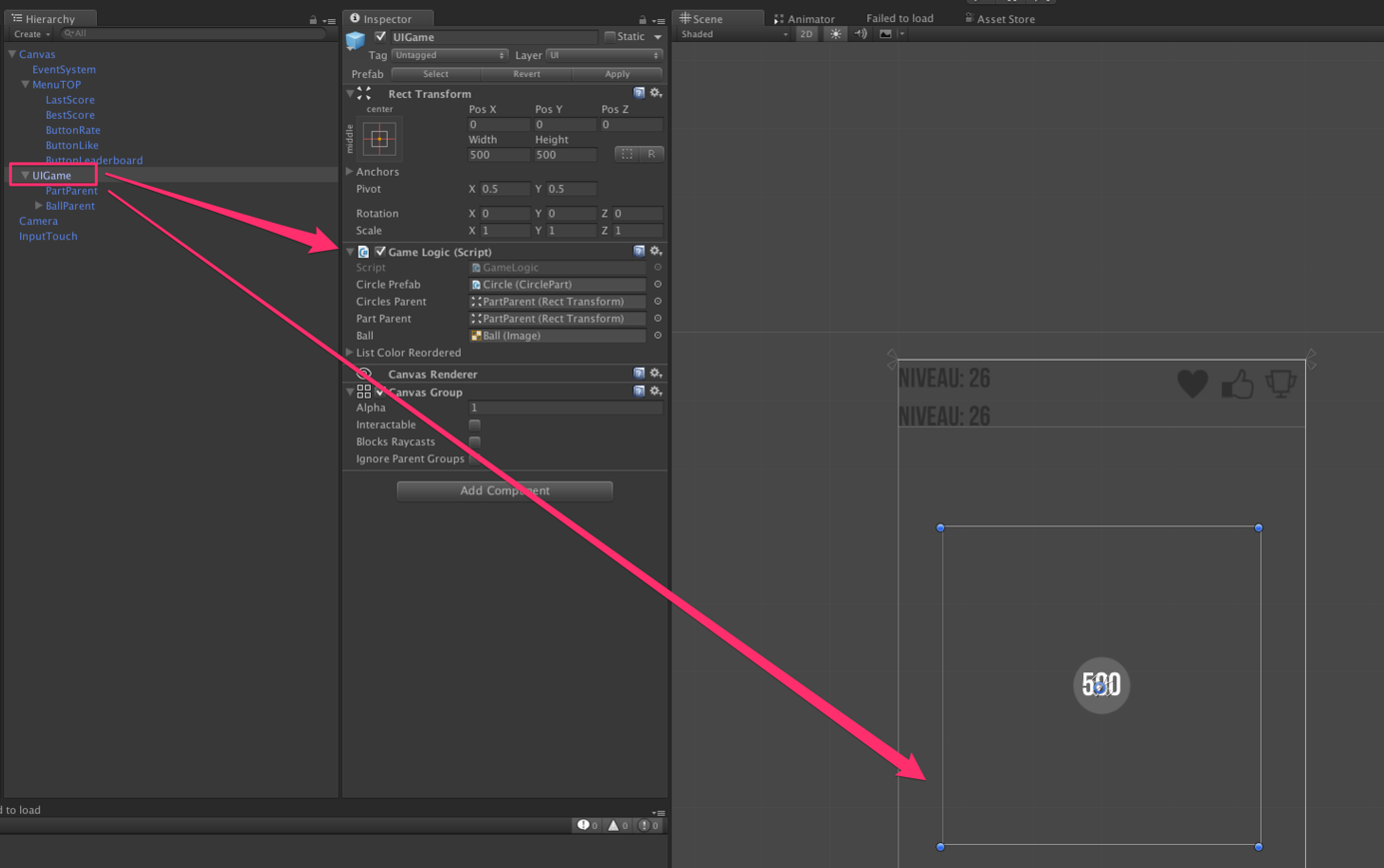
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **static bool Util.RestartFromGameOver** | **(** |  | **)** |  | | inlinestatic |

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

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

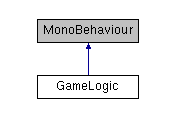
Real shuffle of List

**GameLogic Class Reference**



In charge of all the circle logic. Attached to the game object: "CircleParent". Create the colors, Spawn each element of the circle. Check the color when the player tap the screen etc... In charge of the rotation of the circle 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_game_logic.html#details)

Inheritance diagram for GameLogic:



|  |  |
| --- | --- |
| Public Member Functions | |
| void | [**DOColorBall**](class_game_logic.html#a72a5014a940f6abe52d81384e88d974e) () |
|  | Change the color of the ball = color to find [More...](#a72a5014a940f6abe52d81384e88d974e) |
|  | |
| bool | [**CheckIfBallColorEqualCircleColor**](class_game_logic.html#ad107ec88b03a42a129380b0156d24f7c) () |
|  | Check if the player tap at the good moment on the screen, ie. check if the color of the ball = the color of the part of the circle below the ball [More...](#ad107ec88b03a42a129380b0156d24f7c) |
|  | |

|  |  |
| --- | --- |
| Public Attributes | |
| [**CirclePart**](class_circle_part.html) | [**circlePrefab**](class_game_logic.html#a7a936705d6bd434286bcb77793d72249) |
|  | Prefab of Circle. Use to create the circle. Each part is a UI Image with a certain fillAmount [More...](#a7a936705d6bd434286bcb77793d72249) |
|  | |
| Transform | **circlesParent** |
|  | |
| RectTransform | [**partParent**](class_game_logic.html#a3f118c8432939c3c8f9584969e441469) |
|  | Reference to the GameObject who contains all the part of the circle we will spawn [More...](#a3f118c8432939c3c8f9584969e441469) |
|  | |
| Image | [**ball**](class_game_logic.html#a2ffff75d8677bb0419c8b87efe1771ea) |
|  | Reference to the ball Image = player [More...](#a2ffff75d8677bb0419c8b87efe1771ea) |
|  | |
| List< Color > | [**listColorReordered**](class_game_logic.html#a3753f9bc4673eef36746daa104352613) = new List<Color>() |
|  | Reference to a list of color built for a level [More...](#a3753f9bc4673eef36746daa104352613) |
|  | |
| float | [**speedCircle**](class_game_logic.html#aea6404740b61accca83a20011e664ece) = 0.05f |
|  | Speed of the circle, in seconds (total time in seconds to make 360 degree rotation), for the current level [More...](#aea6404740b61accca83a20011e664ece) |
|  | |

|  |  |
| --- | --- |
| Properties | |
| [**GameManager**](class_game_manager.html) | **gameManager** [get] |
|  | |

|  |  |
| --- | --- |
| Private Member Functions | |
| void | [**Awake**](class_game_logic.html#a2fd501368611e89c933725401db48a6b) () |
|  | 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...](#a2fd501368611e89c933725401db48a6b) |
|  | |
| void | **OnEnable** () |
|  | |
| void | **OnDisable** () |
|  | |
| void | **OnTouchLeft** () |
|  | |
| void | **OnTouchRight** () |
|  | |
| void | **DOStart** () |
|  | |
| void | [**Update**](class_game_logic.html#afc1960f0d415a35e84302f4eb1080ec1) () |
|  | Listen if the player tap or click, and if the game is not game over after the click (so ball color = part of the circle color) launch again the rotation but in the oposite direction [More...](#afc1960f0d415a35e84302f4eb1080ec1) |
|  | |
| void | [**DORotateCircle**](class_game_logic.html#a8e801f0f1eebe7a9a0516f3bef7309e0) (int direction) |
|  | Start the rotation of the circle. Check in each updates if the ball enter a part of the circle 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 ball go out of the part with the same color, so it's game over. [More...](#a8e801f0f1eebe7a9a0516f3bef7309e0) |
|  | |
| void | [**Start**](class_game_logic.html#a5c8a5055ff4a003a255bcdf52cbd4d2c) () |
|  | Place the border and the border shadow at the good place [More...](#a5c8a5055ff4a003a255bcdf52cbd4d2c) |
|  | |
| void | [**DefineLevel**](class_game_logic.html#a2ffd078e7be05a0f07b41c900f3d6f3a) () |
|  | IMPORTANT ==> It's here we define the levels. Change the formulas if you want. [More...](#a2ffd078e7be05a0f07b41c900f3d6f3a) |
|  | |
| [**CirclePart**](class_circle_part.html) | **GetSelection** () |
|  | |
| void | [**BuildCircle**](class_game_logic.html#a009833e8cfe5f54068067a33cced80f4) () |
|  | Method to build the circle. Each part of the circle is an UI Image, type = fill image. We use the fill amout property to cretae the parts of the circle [More...](#a009833e8cfe5f54068067a33cced80f4) |
|  | |
| [**CirclePart**](class_circle_part.html) | [**InstantiateCircle**](class_game_logic.html#a740be6f5d0e2fb833cd62364a7abdeb0) () |
|  | Method to create a new circle = new part of the circle [More...](#a740be6f5d0e2fb833cd62364a7abdeb0) |
|  | |
| [**CirclePart**](class_circle_part.html) | [**InstantiateCircle**](class_game_logic.html#af06ab79355414ad3f088b8448df19675) (float fillAmout, float angle, Color c) |
|  | Method to create a new circle = new part of the circle [More...](#af06ab79355414ad3f088b8448df19675) |
|  | |

|  |  |
| --- | --- |
| Private Attributes | |
| int | [**numOfPart**](class_game_logic.html#a25d78922a4f1f097a5db54a5f9f1bf29) = 12 |
|  | Number of parts in the circle, for the current level [More...](#a25d78922a4f1f097a5db54a5f9f1bf29) |
|  | |
| int | [**numOfColor**](class_game_logic.html#a04fa16afcc7e88b5ccf333d9d408bc03) = 3 |
|  | Number of colors in the circle, for the current level [More...](#a04fa16afcc7e88b5ccf333d9d408bc03) |
|  | |
| List< [**CirclePart**](class_circle_part.html) > | [**allCircles**](class_game_logic.html#a89e1833ece0013e9173327d2fbcfa0b9) = new List<[**CirclePart**](class_circle_part.html)>() |
|  | Reference to all the parts contained in the circle, for the current level [More...](#a89e1833ece0013e9173327d2fbcfa0b9) |
|  | |
| Color | [**lastColor**](class_game_logic.html#ad3673abc99cfb413017a2b173eb20d0a) |
|  | Reference to the last color to find, to avoid duplicate check [More...](#ad3673abc99cfb413017a2b173eb20d0a) |
|  | |
| bool | **shuffleColorAray** = true |
|  | |
| [**GameManager**](class_game_manager.html) | **\_gameManager** |
|  | |
| bool | [**firstMove**](class_game_logic.html#a9948663afc6d28ebacfbb56538e3d49f) = true |
|  | Is it the first time we start the rotation for the level? [More...](#a9948663afc6d28ebacfbb56538e3d49f) |
|  | |
| Tweener | [**rotateTweener**](class_game_logic.html#a06ba92d803290d588e411e699d836cf0) |
|  | Reference to the tweener who rotate the circle [More...](#a06ba92d803290d588e411e699d836cf0) |
|  | |

## Detailed Description

In charge of all the circle logic. Attached to the game object: "CircleParent". Create the colors, Spawn each element of the circle. Check the color when the player tap the screen etc... In charge of the rotation of the circle 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 GameLogic.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 GameLogic.BuildCircle** | **(** |  | **)** |  | | inlineprivate |

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **bool GameLogic.CheckIfBallColorEqualCircleColor** | **(** |  | **)** |  | | inline |

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

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

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **void GameLogic.DOColorBall** | **(** |  | **)** |  | | inline |

Change the color of the ball = color to find

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **void GameLogic.DORotateCircle** | **(** | **int** | **direction** | **)** |  | | inlineprivate |

Start the rotation of the circle. Check in each updates if the ball enter a part of the circle 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 ball go out of the part with the same color, so it's game over.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | [**CirclePart**](class_circle_part.html)**GameLogic.InstantiateCircle** | **(** |  | **)** |  | | inlineprivate |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | [**CirclePart**](class_circle_part.html)**GameLogic.InstantiateCircle** | **(** | **float** | **fillAmout,** | |  |  | **float** | **angle,** | |  |  | **Color** | **c** | |  | **)** |  |  | | inlineprivate |

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

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

Place the border and the border shadow at the good place

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

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

## Member Data Documentation

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **List<**[**CirclePart**](class_circle_part.html)**> GameLogic.allCircles = new List<**[**CirclePart**](class_circle_part.html)**>()** | | private |

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

|  |
| --- |
| **Image GameLogic.ball** |

Reference to the ball Image = player

|  |
| --- |
| [**CirclePart**](class_circle_part.html)**GameLogic.circlePrefab** |

Prefab of Circle. Use to create the circle. Each part is a UI Image with a certain fillAmount

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **bool GameLogic.firstMove = true** | | private |

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

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Color GameLogic.lastColor** | | private |

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

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

Reference to a list of color built for a level

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

Number of colors in the circle, for the current level

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

Number of parts in the circle, for the current level

|  |
| --- |
| **RectTransform GameLogic.partParent** |

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

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Tweener GameLogic.rotateTweener** | | private |

Reference to the tweener who rotate the circle

|  |
| --- |
| **float GameLogic.speedCircle = 0.05f** |

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

**ADS :**

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

Get it here : http://u3d.as/oWD

**Leaderboard :**

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

Get it here : <http://u3d.as/qxf>

Thanks !

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

Inquiries : https://appadvisory.zendesk.com