

## Spin The Circle

Generated by Doxygen 1.8.10

Mon Apr 18 2016 19:59:16



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	ButtonLeaderboard Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Function Documentation	5
3.1.2.1	OnClickedOpenLeaderboard()	5
3.2	ButtonLike Class Reference	6
3.2.1	Detailed Description	6
3.3	ButtonRate Class Reference	6
3.3.1	Detailed Description	7
3.4	CirclePart Class Reference	7
3.4.1	Detailed Description	7
3.4.2	Member Function Documentation	7
3.4.2.1	GetMiddleAngle()	7
3.4.2.2	Init(float fillAmout, float angle, Color color)	7
3.4.3	Member Data Documentation	8
3.4.3.1	image	8
3.5	ColorManager Class Reference	8
3.5.1	Detailed Description	8
3.6	GameLogic Class Reference	8
3.6.1	Detailed Description	10
3.6.2	Member Function Documentation	10
3.6.2.1	Awake()	10
3.6.2.2	BuildCircle()	10
3.6.2.3	CheckIfBallColorEqualCircleColor()	10
3.6.2.4	DefineLevel()	10
3.6.2.5	DOColorBall()	10

3.6.2.6	DORotateCircle(int direction)	11
3.6.2.7	InstantiateCircle()	11
3.6.2.8	InstantiateCircle(float fillAmout, float angle, Color c)	11
3.6.2.9	Start()	11
3.6.2.10	Update()	11
3.6.3	Member Data Documentation	11
3.6.3.1	allCircles	11
3.6.3.2	ball	11
3.6.3.3	circlePrefab	11
3.6.3.4	firstMove	11
3.6.3.5	lastColor	11
3.6.3.6	listColorReordered	11
3.6.3.7	numOfColor	12
3.6.3.8	numOfPart	12
3.6.3.9	partParent	12
3.6.3.10	rotateTween	12
3.6.3.11	speedCircle	12
3.7	GameManager Class Reference	12
3.7.1	Detailed Description	13
3.7.2	Member Function Documentation	14
3.7.2.1	Awake()	14
3.7.2.2	DOMoveLevelIn(Action callback)	14
3.7.2.3	DOMoveLevelOut(Action callback)	14
3.7.2.4	GameOver()	14
3.7.2.5	MoveDone()	14
3.7.2.6	SetNewGame()	14
3.7.2.7	ShowAds()	14
3.7.2.8	Start()	14
3.7.3	Member Data Documentation	14
3.7.3.1	_soundManager	14
3.7.3.2	isGameOver	14
3.7.3.3	levelCenterScreen	15
3.7.3.4	numberOfPlayToShowInterstitial	15
3.7.3.5	RESET_PLAYER_PREF	15
3.7.3.6	VerySimpleAdsURL	15
3.7.4	Property Documentation	15
3.7.4.1	point	15
3.8	InputTouch Class Reference	15
3.8.1	Detailed Description	16
3.9	PlayerPrefsX Class Reference	16

3.9.1 Detailed Description . . . . .	18
3.10 SoundManager Class Reference . . . . .	18
3.10.1 Detailed Description . . . . .	18
3.10.2 Member Function Documentation . . . . .	19
3.10.2.1 Awake() . . . . .	19
3.10.2.2 PlayFail() . . . . .	19
3.10.2.3 PlaySuccess() . . . . .	19
3.10.2.4 PlayTouch() . . . . .	19
3.10.3 Member Data Documentation . . . . .	19
3.10.3.1 audioSource . . . . .	19
3.10.3.2 soundFail . . . . .	19
3.10.3.3 soundSuccess . . . . .	19
3.10.3.4 soundTouch . . . . .	19
3.11 Util Class Reference . . . . .	19
3.11.1 Detailed Description . . . . .	20
3.11.2 Member Function Documentation . . . . .	20
3.11.2.1 CleanMemory() . . . . .	20
3.11.2.2 IsEqual(this Color c, Color o) . . . . .	20
3.11.2.3 ReloadLevel() . . . . .	20
3.11.2.4 RestartFromGameOver() . . . . .	20
3.11.2.5 Shuffle< T >(this IList< T > list) . . . . .	20
<b>Index</b>	<b>21</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

MonoBehaviour	
ButtonLeaderboard . . . . .	5
ButtonLike . . . . .	6
ButtonRate . . . . .	6
CirclePart . . . . .	7
ColorManager . . . . .	8
GameLogic . . . . .	8
GameManager . . . . .	12
InputTouch . . . . .	15
SoundManager . . . . .	18
PlayerPrefsX . . . . .	16
Util . . . . .	19





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">ButtonLeaderboard</a>	Class attached to the leaderboard button. Works only on mobile (iOS & Android), with Very Simple Leaderboard : <a href="http://u3d.as/qxf">http://u3d.as/qxf</a> . . . . .	5
<a href="#">ButtonLike</a>	Attached to like button . . . . .	6
<a href="#">ButtonRate</a>	Attached to rate button . . . . .	6
<a href="#">CirclePart</a>	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 . . .	7
<a href="#">ColorManager</a>	Class with an array of color. Change the array to customize the colors. Attached to the Canvas game object . . . . .	8
<a href="#">GameLogic</a>	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".	8
<a href="#">GameManager</a>	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: <a href="http://u3d.as/oWD">http://u3d.as/oWD</a> . . .	12
<a href="#">InputTouch</a>	Class in charge to manage input touch and desktop input in the game . . . . .	15
<a href="#">PlayerPrefsX</a>	A player pref extension . . . . .	16
<a href="#">SoundManager</a>	Class in charge to play FX in the game. Attached to the Canvas game object. Change the audioSource to customize the sounds. . . . .	18
<a href="#">Util</a>	Utility class. This class is static, so you can use it in all your projects! . . . . .	19



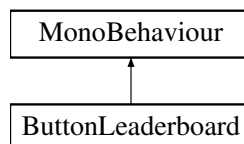
## Chapter 3

# Class Documentation

### 3.1 ButtonLeaderboard Class Reference

Class attached to the leaderboard button. Works only on mobile (iOS & Android), with Very Simple Leaderboard : <http://u3d.as/qxf>

Inheritance diagram for ButtonLeaderboard:



#### Public Member Functions

- void [OnClickedOpenLeaderboard](#) ()

*If player clicks on the leaderbord button, we call this method. Works only on mobile (iOS & Android) if using Very Simple Leaderboard by App Advisory : <http://u3d.as/qxf>*

#### 3.1.1 Detailed Description

Class attached to the leaderboard button. Works only on mobile (iOS & Android), with Very Simple Leaderboard : <http://u3d.as/qxf>

#### 3.1.2 Member Function Documentation

##### 3.1.2.1 void ButtonLeaderboard.OnClickedOpenLeaderboard ( ) [inline]

If player clics on the leaderbord button, we call this method. Works only on mobile (iOS & Android) if using Very Simple Leaderboard by App Advisory : <http://u3d.as/qxf>

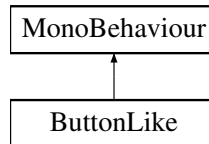
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/UIScripts/ButtonLeaderboard.cs

## 3.2 ButtonLike Class Reference

Attached to like button

Inheritance diagram for ButtonLike:



### Public Member Functions

- void **OnClickedFacebookLikeButton** ()

### Public Attributes

- string **facebookApp** = "fb://profile/515431001924232"
- string **facebookAddress** = "https://www.facebook.com/appadvisory"

### 3.2.1 Detailed Description

Attached to like button

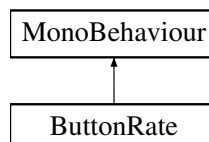
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/UIScripts/ButtonLike.cs

## 3.3 ButtonRate Class Reference

Attached to rate button

Inheritance diagram for ButtonRate:



### Public Member Functions

- void **OnClickedRate** ()

### Public Attributes

- string **iosRateURL** = "fb://profile/515431001924232"
- string **androidRateURL** = "https://www.facebook.com/appadvisory"

### 3.3.1 Detailed Description

Attached to rate button

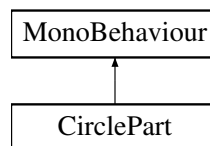
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/UIScripts/Button↵  
Rate.cs

## 3.4 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

Inheritance diagram for CirclePart:



### Public Member Functions

- [CirclePart Init](#) (float fillAmount, 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*
- float [GetMiddleAngle](#) ()  
*Get the angle of the middle of the part of circle*

### Public Attributes

- Image [image](#)  
*The image = a simple circle*

### 3.4.1 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

### 3.4.2 Member Function Documentation

#### 3.4.2.1 float CirclePart.GetMiddleAngle ( ) [inline]

Get the angle of the middle of the part of circle

#### 3.4.2.2 CirclePart CirclePart.Init ( float fillAmount, 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

### 3.4.3 Member Data Documentation

#### 3.4.3.1 Image CirclePart.image

The image = a simple circle

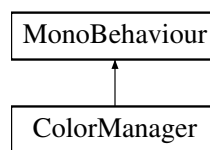
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/CirclePart.cs

## 3.5 ColorManager Class Reference

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

Inheritance diagram for ColorManager:



### Public Attributes

- Color[] **colors**

#### 3.5.1 Detailed Description

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

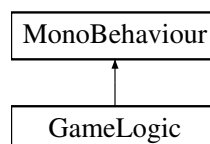
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/Color↔Manager.cs

## 3.6 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".

Inheritance diagram for GameLogic:



### Public Member Functions

- void **DOColorBall** ()

*Change the color of the ball = color to find*

- bool [CheckIfBallColorEqualCircleColor](#) ()

*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*

## Public Attributes

- [CirclePart](#) [circlePrefab](#)

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

- Transform [circlesParent](#)
- RectTransform [partParent](#)

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

- Image [ball](#)

*Reference to the ball Image = player*

- List< Color > [listColorReordered](#) = new List<Color>()

*Reference to a list of color built for a level*

- float [speedCircle](#) = 0.05f

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

## Properties

- [GameManager](#) [gameManager](#) [get]

## Private Member Functions

- void [Awake](#) ()

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

- void [OnEnable](#) ()
- void [OnDisable](#) ()
- void [OnTouchLeft](#) ()
- void [OnTouchRight](#) ()
- void [DOStart](#) ()
- void [Update](#) ()

*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 opposite direction*

- void [DORotateCircle](#) (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.*

- void [Start](#) ()

*Place the border and the border shadow at the good place*

- void [DefineLevel](#) ()

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

- [CirclePart](#) [GetSelection](#) ()
- void [BuildCircle](#) ()

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

- [CirclePart](#) [InstantiateCircle](#) ()

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

- [CirclePart](#) [InstantiateCircle](#) (float fillAmount, float angle, Color c)

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

## Private Attributes

- int `numOfPart` = 12  
*Number of parts in the circle, for the current level*
- int `numOfColor` = 3  
*Number of colors in the circle, for the current level*
- List< `CirclePart` > `allCircles` = new List<`CirclePart`>()  
*Reference to all the parts contained in the circle, for the current level*
- Color `lastColor`  
*Reference to the last color to find, to avoid duplicate check*
- bool `shuffleColorArray` = true
- `GameManager` `_gameManager`
- bool `firstMove` = true  
*Is it the first time we start the rotation for the level?*
- Tweener `rotateTweener`  
*Reference to the tweener who rotate the circle*

### 3.6.1 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".

### 3.6.2 Member Function Documentation

#### 3.6.2.1 void GameLogic.Awake ( ) [inline],[private]

Create a new list of corlors for this level, randomly : listColorReordered and save it in `PlayerPrefsX` to use the same list of colors in case of game over

#### 3.6.2.2 void GameLogic.BuildCircle ( ) [inline],[private]

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

#### 3.6.2.3 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

#### 3.6.2.4 void GameLogic.DefineLevel ( ) [inline],[private]

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

#### 3.6.2.5 void GameLogic.DOColorBall ( ) [inline]

Change the color of the ball = color to find



**3.6.2.6 void GameLogic.DORotateCircle ( int *direction* ) [inline], [private]**

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.

**3.6.2.7 CirclePart GameLogic.InstantiateCircle ( ) [inline], [private]**

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

**3.6.2.8 CirclePart GameLogic.InstantiateCircle ( float *fillAmout*, float *angle*, Color *c* ) [inline], [private]**

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

**3.6.2.9 void GameLogic.Start ( ) [inline], [private]**

Place the border and the border shadow at the good place

**3.6.2.10 void GameLogic.Update ( ) [inline], [private]**

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 opposite direction

**3.6.3 Member Data Documentation****3.6.3.1 List<CirclePart> GameLogic.allCircles = new List<CirclePart>() [private]**

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

**3.6.3.2 Image GameLogic.ball**

Reference to the ball Image = player

**3.6.3.3 CirclePart GameLogic.circlePrefab**

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

**3.6.3.4 bool GameLogic.firstMove = true [private]**

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

**3.6.3.5 Color GameLogic.lastColor [private]**

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

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

Reference to a list of color built for a level

**3.6.3.7** `int GameLogic.numOfColor = 3` [private]

Number of colors in the circle, for the current level

**3.6.3.8** `int GameLogic.numOfPart = 12` [private]

Number of parts in the circle, for the current level

**3.6.3.9** `RectTransform GameLogic.partParent`

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

**3.6.3.10** `Tweener GameLogic.rotateTweener` [private]

Reference to the tweener who rotate the circle

**3.6.3.11** `float GameLogic.speedCircle = 0.05f`

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

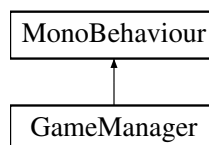
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/GameLogic.cs

## 3.7 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>

Inheritance diagram for GameManager:



### Public Member Functions

- void [MoveDone](#) ()  
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 [GameOver](#) ()  
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 [ShowAds](#) ()  
Show Ads - Interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>

## Public Attributes

- string `VerySimpleAdsURL` = "http://u3d.as/oWD"  
*If you want to monetize this game, get VERY SIMPLE ADS at this URL: <http://u3d.as/oWD>*
- int `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 `RESET_PLAYER_PREF` = false  
*to reset the player pref. Use if for debug only!!*
- bool `isGameOver` = false  
*True if game over*
- Text `levelCenterScreen`  
*Text in the center of the screen = number of colors to find to clear the level*
- Text `textLastScore`
- Text `textBestScore`

## Properties

- `SoundManager soundManager` [get]
- int `point` [get, set]  
*The number of move we have to do to clear this level = the level number*

## Private Member Functions

- void `Awake` ()  
*Clean the memory and place the circleparent at the good place*
- void `Start` ()  
*Clean the memory and place the circleparent at the good place*
- void `SetNewGame` ()  
*Create a new game: Set the texts, the numTotalOfMove and if the last game was not a game over : do the animation in*
- void `DOMoveLevelOut` (Action callback)  
*Animation out of the circle (from center to left)*
- void `DOMoveLevelIn` (Action callback)  
*Animation in of the circle (from right to center)*

## Private Attributes

- `SoundManager _soundManager`  
*Reference to circle parent, to do the animation in and out for transition between level*
- int `m_point`

### 3.7.1 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, everything is already coded for you. You just need to get VERY SIMPLE ADS here: <http://u3d.as/oWD>

### 3.7.2 Member Function Documentation

#### 3.7.2.1 void GameManager.Awake ( ) [inline],[private]

Clean the memory and place the circleparent at the good place

#### 3.7.2.2 void GameManager.DOMoveLevelIn ( Action callback ) [inline],[private]

Animation in of the circle (from right to center)

#### 3.7.2.3 void GameManager.DOMoveLevelOut ( Action callback ) [inline],[private]

Animation out of the circle (from center to left)

#### 3.7.2.4 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>

#### 3.7.2.5 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

#### 3.7.2.6 void GameManager.SetNewGame ( ) [inline],[private]

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

#### 3.7.2.7 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>

#### 3.7.2.8 void GameManager.Start ( ) [inline],[private]

Clean the memory and place the circleparent at the good place

### 3.7.3 Member Data Documentation

#### 3.7.3.1 SoundManager GameManager.\_soundManager [private]

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

#### 3.7.3.2 bool GameManager.isGameOver = false

True if game over

## 3.7.3.3 Text GameManager.levelCenterScreen

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

## 3.7.3.4 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>

## 3.7.3.5 bool GameManager.RESET\_PLAYER\_PREF = false

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

## 3.7.3.6 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>

## 3.7.4 Property Documentation

## 3.7.4.1 int GameManager.point [get], [set]

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

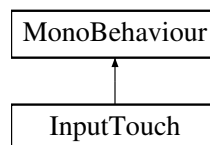
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/GameManager.cs

## 3.8 InputTouch Class Reference

Class in charge to manage input touch and desktop input in the game

Inheritance diagram for InputTouch:



## Public Member Functions

- delegate void **TouchLeft** ()
- delegate void **TouchRight** ()
- delegate void **TouchScreen** ()

## Events

- static TouchLeft **OnTouchLeft**
- static TouchRight **OnTouchRight**
- static TouchScreen **OnTouchScreen**

## Private Member Functions

- void **Update** ()
- void **\_OnTouchLeft** ()
- void **\_OnTouchRight** ()

### 3.8.1 Detailed Description

Class in charge to manage input touch and desktop input in the game

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

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/Input↵  
Touch.cs

## 3.9 PlayerPrefsX Class Reference

A player pref extension

### 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[] > 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**

### 3.9.1 Detailed Description

A player pref extension

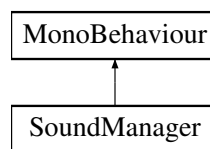
The documentation for this class was generated from the following file:

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/PlayerPrefsX.cs

## 3.10 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.

Inheritance diagram for SoundManager:



### Public Member Functions

- void [PlaySuccess](#) ()  
*Method called when the level is clear = success*
- void [PlayFail](#) ()  
*Method called when game over*
- void [PlayTouch](#) ()  
*Method called when the player tap at the good moment on the screen*

### Private Member Functions

- void [Awake](#) ()  
*Find the audiosource attached to the same game object*

### Private Attributes

- AudioSource [audioSource](#)  
*Reference to the audiosouce use to play fx, attached to the same game object*
- AudioClip [soundSuccess](#)  
*Sound played when the level is clear = success*
- AudioClip [soundFail](#)  
*Sound played when game over*
- AudioClip [soundTouch](#)  
*Sound played when the player tap at the good moment on the screen*

### 3.10.1 Detailed Description

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



### 3.10.2 Member Function Documentation

#### 3.10.2.1 void SoundManager.Awake ( ) [inline],[private]

Find the audiosource attached to the same game object

#### 3.10.2.2 void SoundManager.PlayFail ( ) [inline]

Method called when game over

#### 3.10.2.3 void SoundManager.PlaySuccess ( ) [inline]

Method called when the level is clear = success

#### 3.10.2.4 void SoundManager.PlayTouch ( ) [inline]

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

### 3.10.3 Member Data Documentation

#### 3.10.3.1 AudioSource SoundManager.audioSource [private]

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

#### 3.10.3.2 AudioClip SoundManager.soundFail [private]

Sound played when game over

#### 3.10.3.3 AudioClip SoundManager.soundSuccess [private]

Sound played when the level is clear = success

#### 3.10.3.4 AudioClip SoundManager.soundTouch [private]

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

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

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/SoundManager.cs

## 3.11 Util Class Reference

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

### Static Public Member Functions

- static bool [IsEqual](#) (this Color c, Color o)  
*Compare two colors*
- static void [Shuffle< T >](#) (this IList< T > list)

- Real shuffle of List*
- static bool **SetLastScore** (int score)
- static int **GetLastScore** ()
- static int **GetBestScore** ()
- static void **ReloadLevel** ()
- Clean the memory and reload the scene*
- static void **CleanMemory** ()
- Clean the memory*
- static bool **RestartFromGameOver** ()
- Resturn true if last time we play we lose (= Game Over)*

### Static Private Attributes

- static System.Random **rng** = new System.Random()

### 3.11.1 Detailed Description

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

### 3.11.2 Member Function Documentation

#### 3.11.2.1 static void Util.CleanMemory ( ) [inline],[static]

Clean the memory

#### 3.11.2.2 static bool Util.IsEqual ( this Color c, Color o ) [inline],[static]

Compare two colors

#### 3.11.2.3 static void Util.ReloadLevel ( ) [inline],[static]

Clean the memory and reload the scene

#### 3.11.2.4 static bool Util.RestartFromGameOver ( ) [inline],[static]

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

#### 3.11.2.5 static void Util.Shuffle< T > ( this IList< T > list ) [inline],[static]

Real shuffle of List

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

- /Volumes/LaCie/Dropbox/Anthony/\_\_\_AppAdvisory/SpinTheCircle/Assets/SpinTheCircle/Scripts/Util.cs

# Index

- [\\_soundManager](#)
    - [GameManager, 14](#)
- [allCircles](#)
  - [GameLogic, 11](#)
- [audioSource](#)
  - [SoundManager, 19](#)
- [Awake](#)
  - [GameLogic, 10](#)
  - [GameManager, 14](#)
  - [SoundManager, 19](#)
- [ball](#)
  - [GameLogic, 11](#)
- [BuildCircle](#)
  - [GameLogic, 10](#)
- [ButtonLeaderboard, 5](#)
  - [OnClickedOpenLeaderboard, 5](#)
- [ButtonLike, 6](#)
- [ButtonRate, 6](#)
- [CheckIfBallColorEqualCircleColor](#)
  - [GameLogic, 10](#)
- [CirclePart, 7](#)
  - [GetMiddleAngle, 7](#)
  - [image, 8](#)
  - [Init, 7](#)
- [circlePrefab](#)
  - [GameLogic, 11](#)
- [CleanMemory](#)
  - [Util, 20](#)
- [ColorManager, 8](#)
- [DOColorBall](#)
  - [GameLogic, 10](#)
- [DOMoveLevelIn](#)
  - [GameManager, 14](#)
- [DOMoveLevelOut](#)
  - [GameManager, 14](#)
- [DORotateCircle](#)
  - [GameLogic, 10](#)
- [DefineLevel](#)
  - [GameLogic, 10](#)
- [firstMove](#)
  - [GameLogic, 11](#)
- [GameLogic, 8](#)
  - [allCircles, 11](#)
  - [Awake, 10](#)
  - [ball, 11](#)
- [BuildCircle, 10](#)
- [CheckIfBallColorEqualCircleColor, 10](#)
- [circlePrefab, 11](#)
- [DOColorBall, 10](#)
- [DORotateCircle, 10](#)
- [DefineLevel, 10](#)
- [firstMove, 11](#)
- [InstantiateCircle, 11](#)
- [lastColor, 11](#)
- [listColorReordered, 11](#)
- [numOfColor, 11](#)
- [numOfPart, 12](#)
- [partParent, 12](#)
- [rotateTween, 12](#)
- [speedCircle, 12](#)
- [Start, 11](#)
- [Update, 11](#)
- [GameManager, 12](#)
  - [\\_soundManager, 14](#)
  - [Awake, 14](#)
  - [DOMoveLevelIn, 14](#)
  - [DOMoveLevelOut, 14](#)
  - [GameOver, 14](#)
  - [isGameOver, 14](#)
  - [levelCenterScreen, 14](#)
  - [MoveDone, 14](#)
  - [numberOfPlayToShowInterstitial, 15](#)
  - [point, 15](#)
  - [RESET\\_PLAYER\\_PREF, 15](#)
  - [SetNewGame, 14](#)
  - [ShowAds, 14](#)
  - [Start, 14](#)
  - [VerySimpleAdsURL, 15](#)
- [GameOver](#)
  - [GameManager, 14](#)
- [GetMiddleAngle](#)
  - [CirclePart, 7](#)
- [image](#)
  - [CirclePart, 8](#)
- [Init](#)
  - [CirclePart, 7](#)
- [InputTouch, 15](#)
- [InstantiateCircle](#)
  - [GameLogic, 11](#)
- [IsEqual](#)
  - [Util, 20](#)
- [isGameOver](#)
  - [GameManager, 14](#)

- lastColor
  - GameLogic, 11
- levelCenterScreen
  - GameManager, 14
- listColorReordered
  - GameLogic, 11
- MoveDone
  - GameManager, 14
- numOfColor
  - GameLogic, 11
- numOfPart
  - GameLogic, 12
- numberOfPlayToShowInterstitial
  - GameManager, 15
- OnClickedOpenLeaderboard
  - ButtonLeaderboard, 5
- partParent
  - GameLogic, 12
- PlayFail
  - SoundManager, 19
- PlaySuccess
  - SoundManager, 19
- PlayTouch
  - SoundManager, 19
- PlayerPrefsX, 16
- point
  - GameManager, 15
- RESET\_PLAYER\_PREF
  - GameManager, 15
- ReloadLevel
  - Util, 20
- RestartFromGameOver
  - Util, 20
- rotateTweenner
  - GameLogic, 12
- SetNewGame
  - GameManager, 14
- ShowAds
  - GameManager, 14
- Shuffle< T >
  - Util, 20
- soundFail
  - SoundManager, 19
- SoundManager, 18
  - audioSource, 19
  - Awake, 19
  - PlayFail, 19
  - PlaySuccess, 19
  - PlayTouch, 19
  - soundFail, 19
  - soundSuccess, 19
  - soundTouch, 19
- soundSuccess
  - SoundManager, 19
- soundTouch
  - SoundManager, 19
- speedCircle
  - GameLogic, 12
- Start
  - GameLogic, 11
  - GameManager, 14
- Update
  - GameLogic, 11
- Util, 19
  - CleanMemory, 20
  - IsEqual, 20
  - ReloadLevel, 20
  - RestartFromGameOver, 20
  - Shuffle< T >, 20
- VerySimpleAdsURL
  - GameManager, 15