STOP THE WHEEL

Thanks for your purchase.

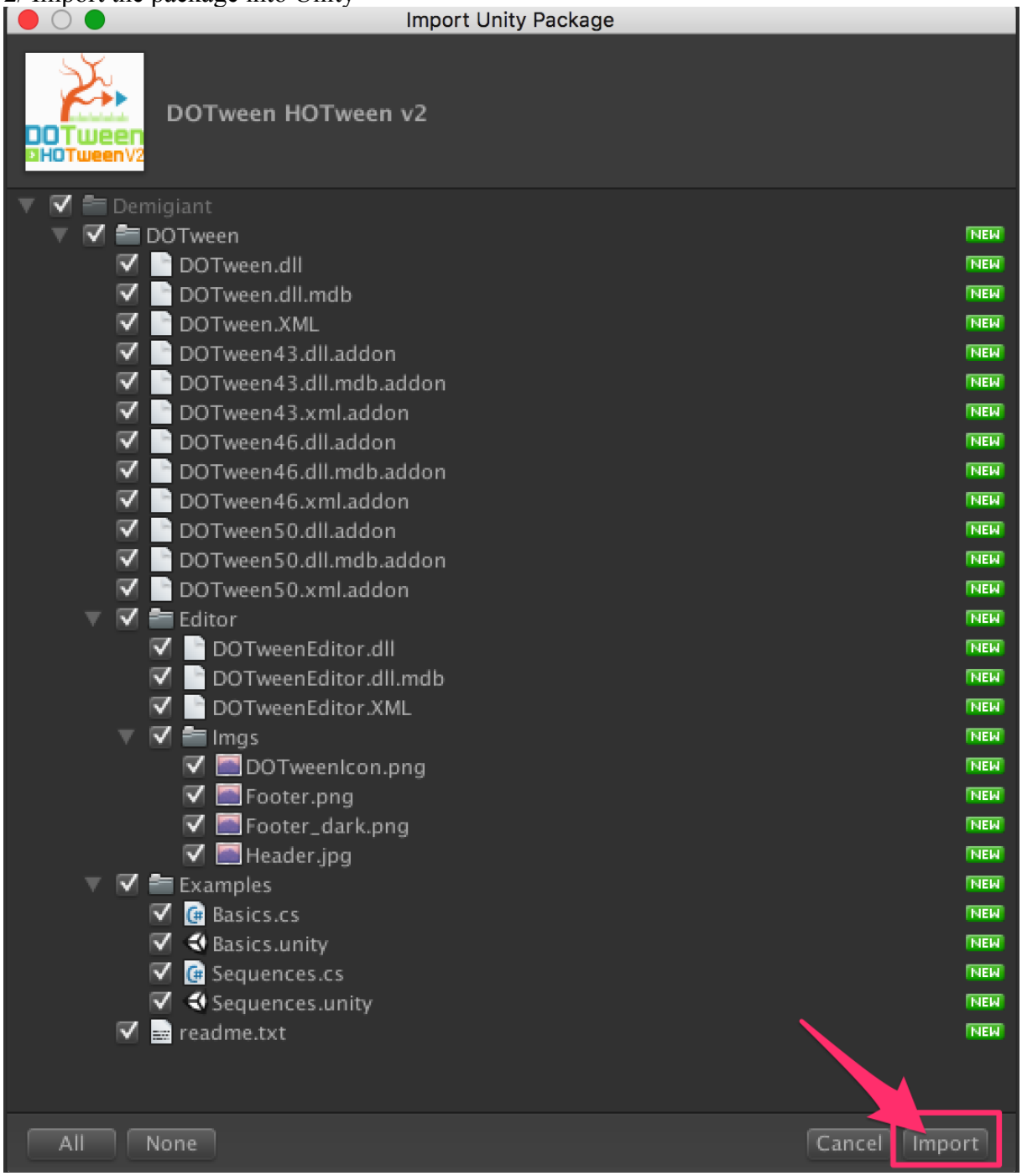
First of all, you have to get DOTWEEN from the Asset Store :

http://u3d.as/aZ1

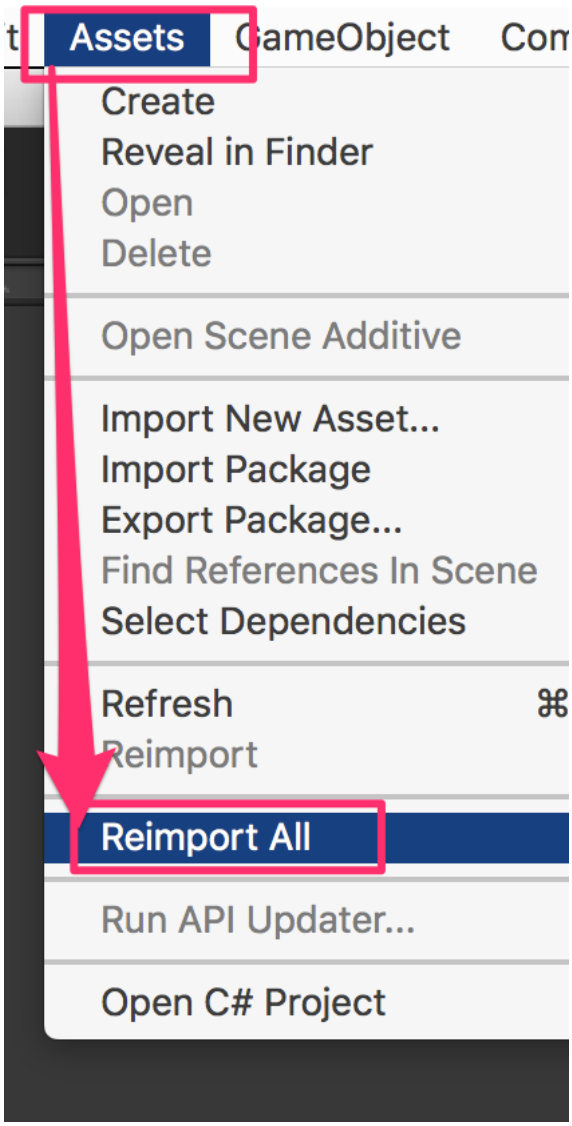
(it's free)

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

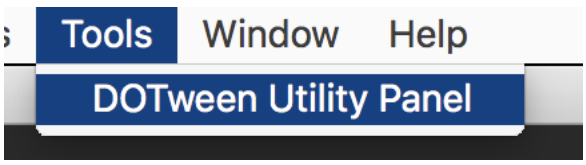
2/ Import the package into Unity



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

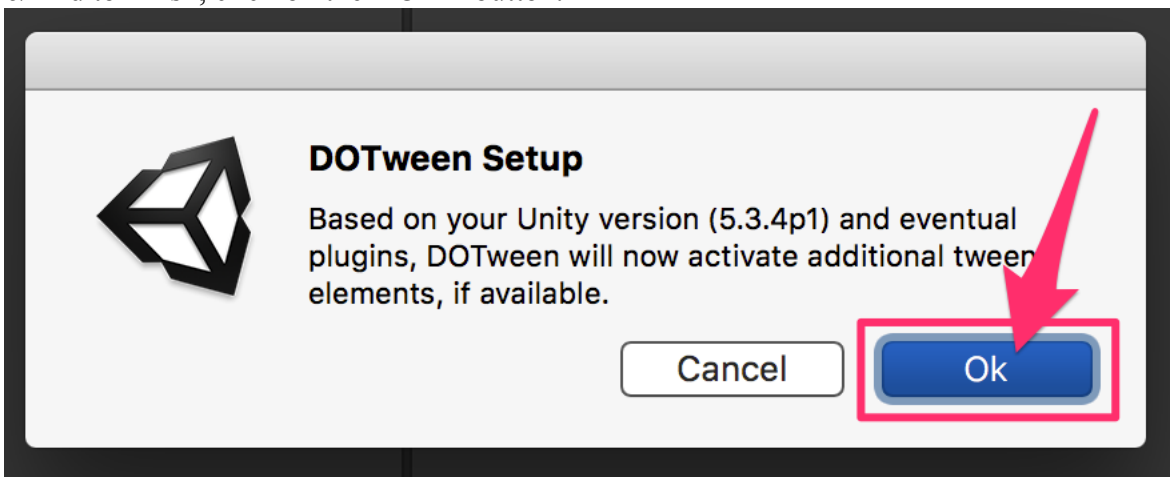


4/Now you have the « Tools ». Open it and click on « DOTween Utility Panel ».

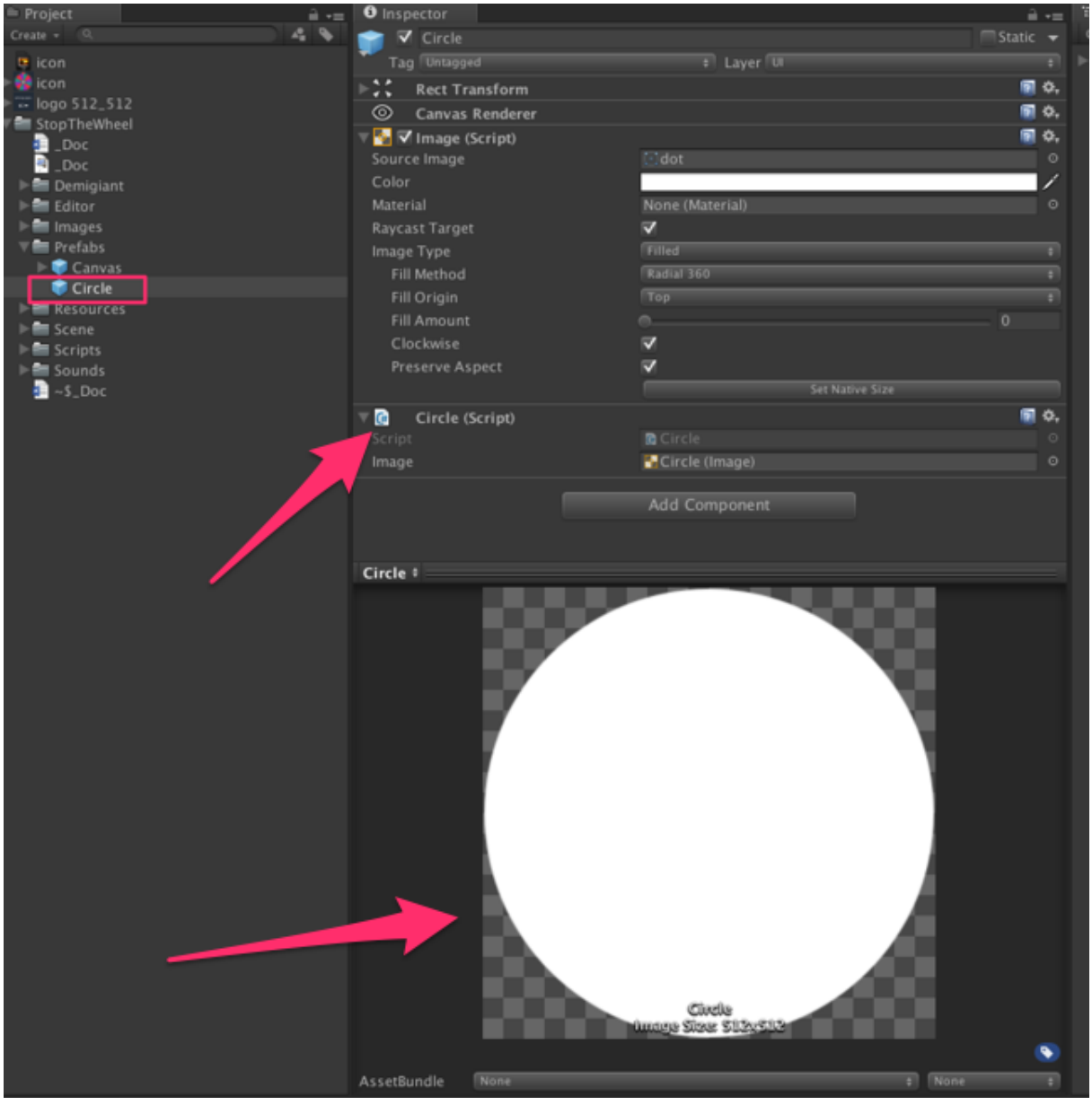


5/Click on « Setup DOTween ».

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

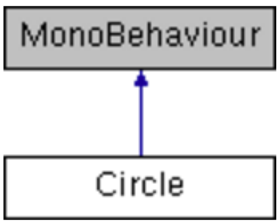


Circle Class Reference



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**). The **Circle** prefab is in the Prefabs folder. Each Circles are instantiate in the **WheelLogic** at the start of each level More...

Inheritance diagram for Circle:



Public Member Functions

Circle	**Init** (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...

float	**GetMiddleAngle** ()
	Get the angle of the middle of the part of wheel More...

Public Attributes

Image	**image**
	The image = a simple circle More...

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**). The **Circle** prefab is in the Prefabs folder. Each Circles are instantiate in the **WheelLogic** 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 Circle.Init (float fillAmout, inline

```
    float  angle,
    Color  color
)
```

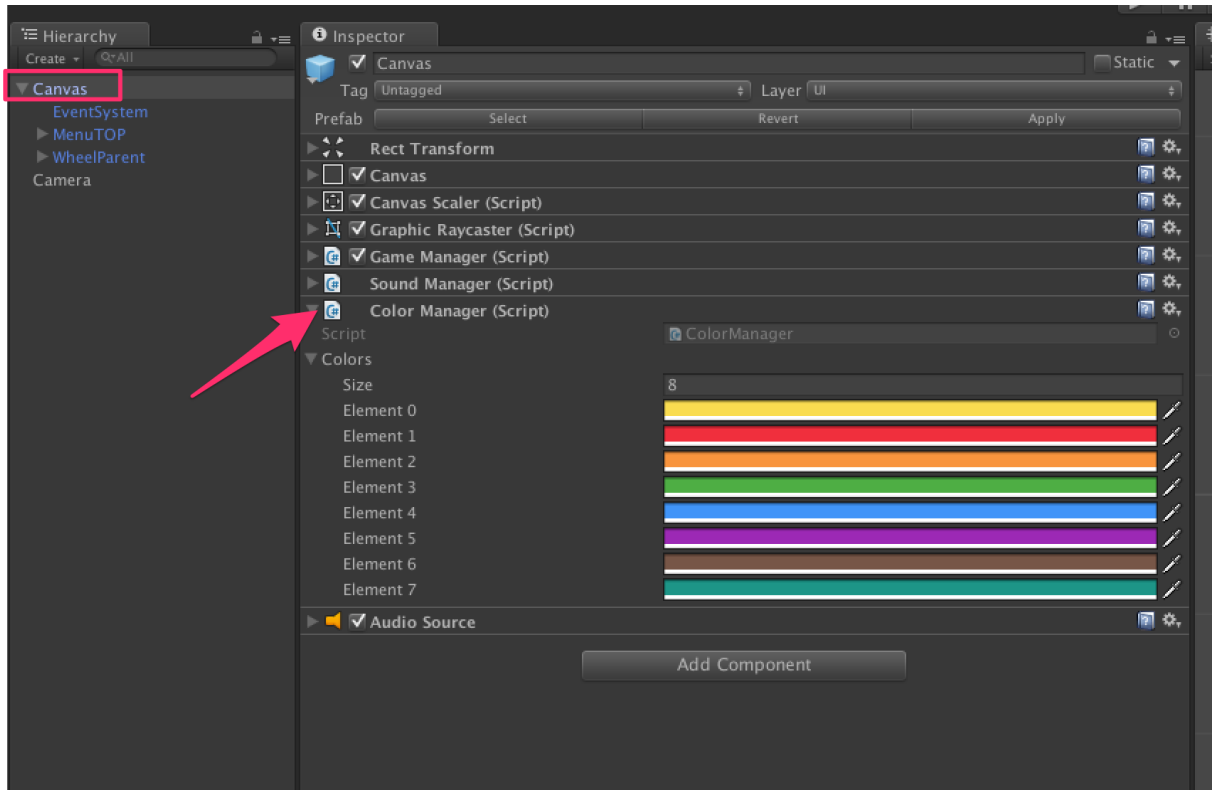
Init the circle = the part of the wheel. Each part is defined with a fillAmount = 1 / number of part in the wheel, an angle and a color

Member Data Documentation

Image Circle.image

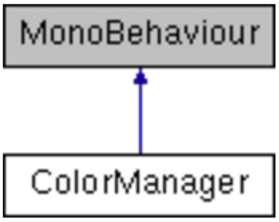
The image = a simple circle

ColorManager Class Reference



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

Inheritance diagram for ColorManager:



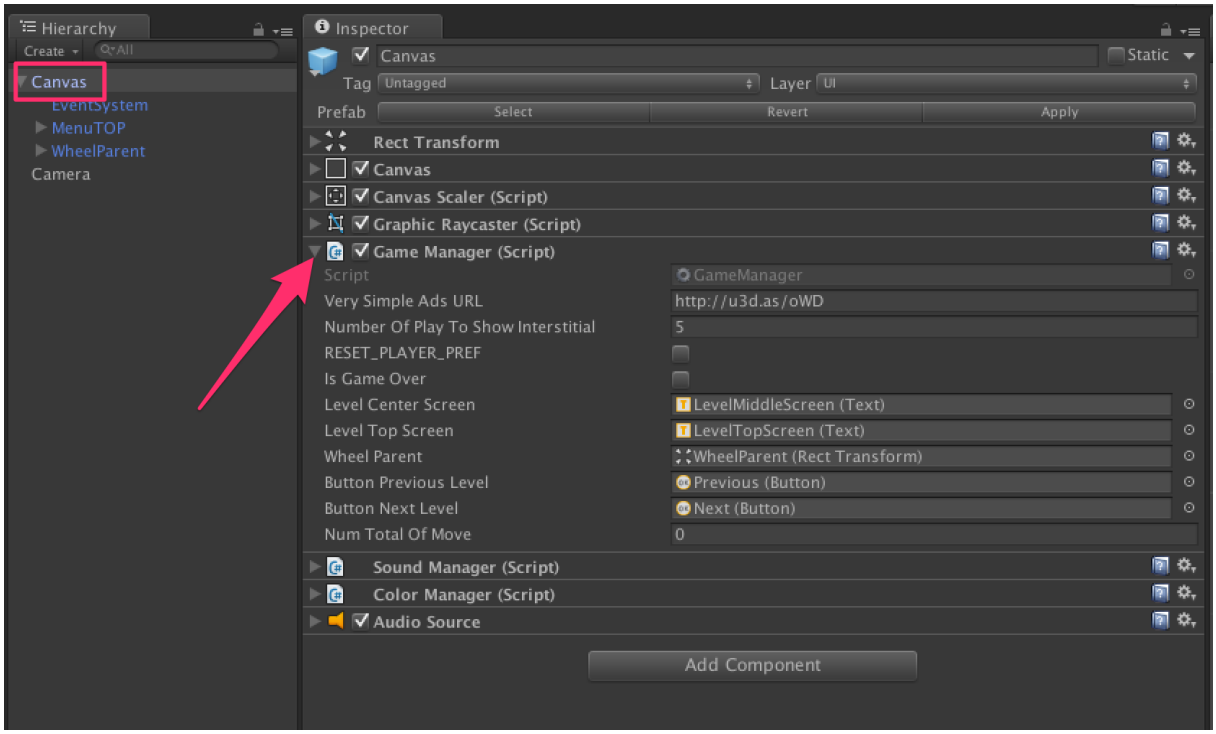
Public Attributes

Color[]	**colors**

Detailed Description

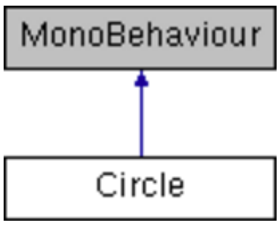
Class with an array of color. Change the array to customize the colors. Attached to the Canvas game object

GameManager Class Reference



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

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 More...
void	**GameOver** ()
	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...
void	**LevelCleared** ()
	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...
void	**ShowAds** ()
	Show Ads - Interstitial. If you want to monetize this game, get VERY SIMPLE ADS at this URL: http://u3d.as/oWD More...

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 More...
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 More...
bool	**RESET_PLAYER_PREF** = false

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

bool	**isGameOver** = false
	True if game over More...

Text	**levelCenterScreen**
	Text in the center of the screen = number of colors to find to clear the level More...

Text	**levelTopScreen**
	Text in the center of the screen = number of colors to find to clear the level More...

RectTransform	**wheelParent**
	Reference to wheel parent, to do the animation in and out for transition between level More...

int	**numTotalOfMove** = 0
	The number of move we have to do to clear this level = the level number More...

Private Member Functions

void	**Awake** ()
	Clean the memory and place the wheelparent at the good place More...

void	**Start** ()
	Clean the memory and place the wheelparent at the good place More...

void	**OnClickedPreviousLevel** ()
	Called when player tap the previous button More...

void	**OnClickedNextLevel** ()
	Called when player tap the next button More...

void	**OnClick** (bool isNext)
	Called when player tap the next or previous button More...

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 More...

void	**UpdateButton** ()
	Update the button previous and next More...

void	**DOMoveLevelOut** (Action callback)
	Animation out of the wheel (from center to left) More...

void	**DOMoveLevelIn** (Action callback)
	Animation in of the wheel (from right to center) More...

Private Attributes

Button	**buttonPreviousLevel**
	Reference to the button to go to the previous level More...

Button	**buttonNextLevel**
	Reference to the button to go to the next level, if the next level is already unlocked More...

Additional Inherited Members

Properties inherited from MonoBehaviourHelper

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!!

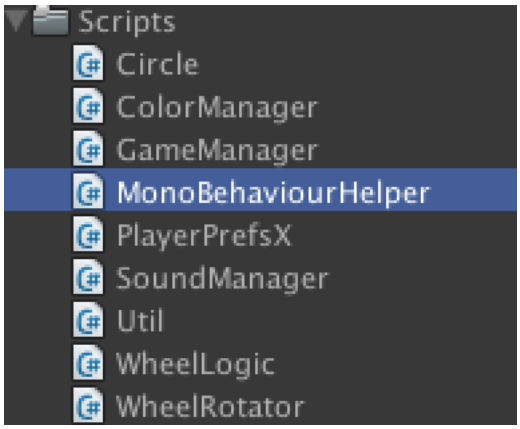
string GameManager.VerySimpleAdsURL = "http://u3d.as/oWD"

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

RectTransform GameManager.wheelParent

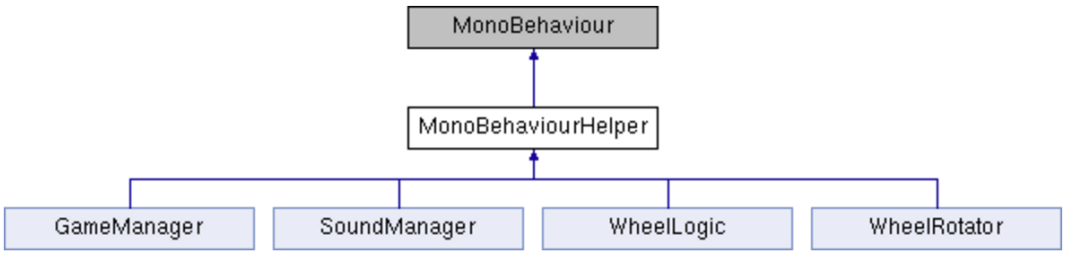
Reference to wheel parent, to do the animation in and out for transition between level

MonoBehaviourHelper Class Reference



Script to avoid duplicate code. More...

Inheritance diagram for MonoBehaviourHelper:



Properties

WheelLogic	**wheelLogic** `[get]`
WheelRotator	**wheelRotator** `[get]`
GameManager	**gameManager** `[get]`
SoundManager	**soundManager** `[get]`
ColorManager	**colorManager** `[get]`

Private Attributes

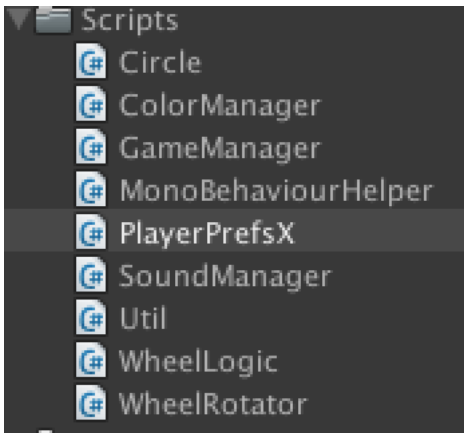
WheelLogic	**_wheelLogic**

WheelRotator	**_wheelRotator**
GameManager	**_gameManager**
SoundManager	**_soundManager**
ColorManager	**_colorManager**

Detailed Description

Script to avoid duplicate code.

PlayerPrefsX Class Reference



A player pref extension More...

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)

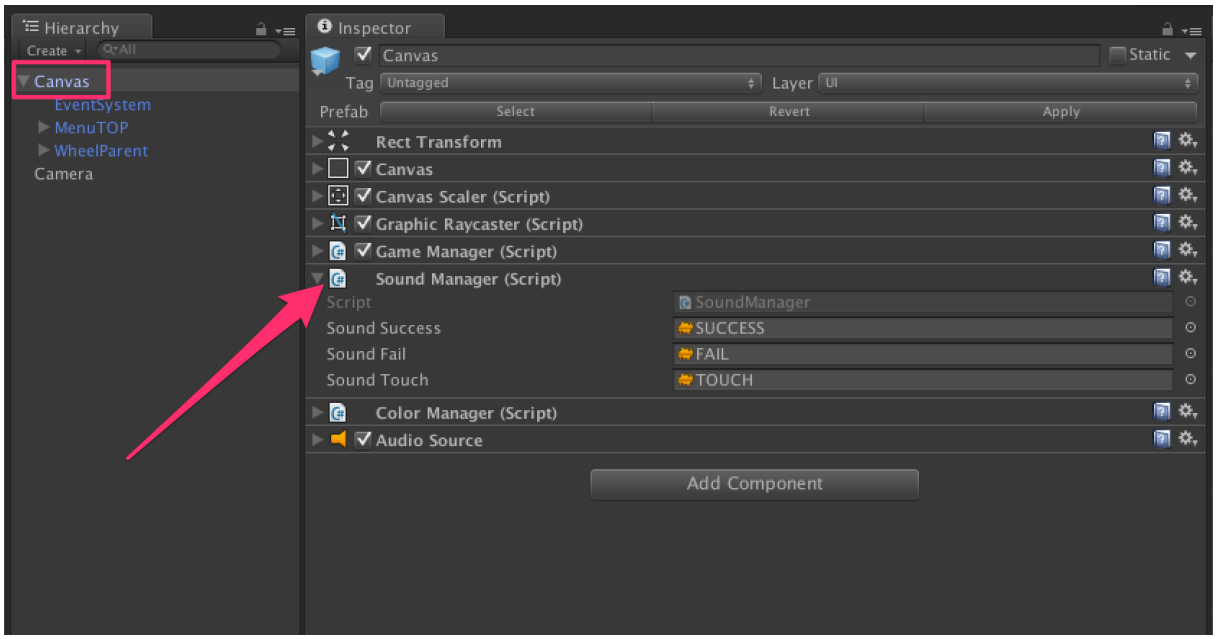
Static Private Attributes

static int	**endianDiff1**
static int	**endianDiff2**
static int	**idx**
static byte[]	**byteBlock**

Detailed Description

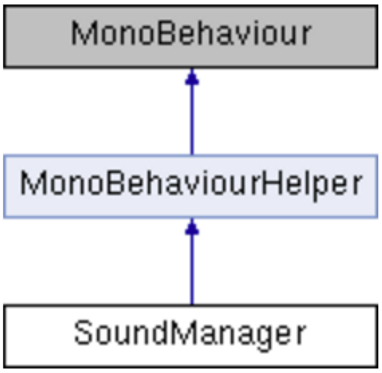
A player pref extension

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...

Inheritance diagram for SoundManager:



Public Member Functions

void	**PlaySuccess** ()
	Method called when the level is clear = success More...
void	**PlayFail** ()
	Method called when game over More...

void	**PlayTouch** ()
	Method called when the player tap at the good moment on the screen More...

Private Member Functions

void	**Awake** ()
	Find the audiosource attached to the same game object More...

Private Attributes

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

Additional Inherited Members

Properties inherited from MonoBehaviourHelper

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

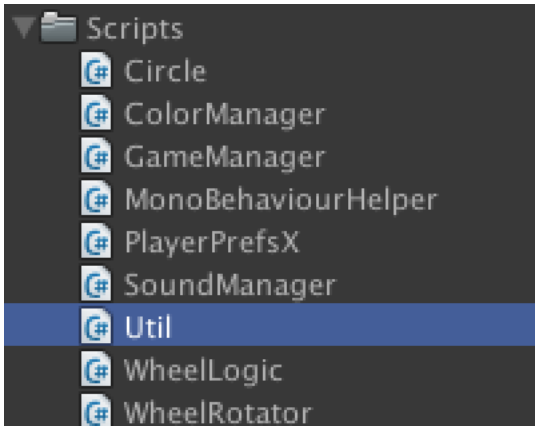
AudioClip SoundManager.soundSuccess private

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...

Static Public Member Functions

static bool	**IsEqual** (this Color c, Color o)
	Compare two colors More...
static void	**Shuffle< T >** (this IList< T > list)
	Real shuffle of List More...
static int	**GetCurrentLevel** ()
	Get the current level More...
static bool	**HavePreviousLevel** ()
	Check if there is a previous level More...
static bool	**HaveNextLevel** ()
	Check if there is a next level, ie. if the next level is unlocked by the player More...

static int	**GetMaxLevel** ()
	Get the max level unlocked by the player More...
static void	**SetMaxLevel** (int level)
	Set the max level unlocked by the player More...
static void	**SetCurrentLevel** (int level)
	Set the current played level More...
static void	**ReloadLevel** ()
	Clean the memory and reload the scene More...
static void	**CleanMemory** ()
	Clean the memory More...
static bool	**RestartFromGameOver** ()
	Resturn true if last time we play we lose (= Game Over) More...
static void	**SetRestartFromGameOver** ()
	Set restart from game over = true More...
static void	**SetNotRestartFromGameOver** ()
	Set restart from game over = false More...

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

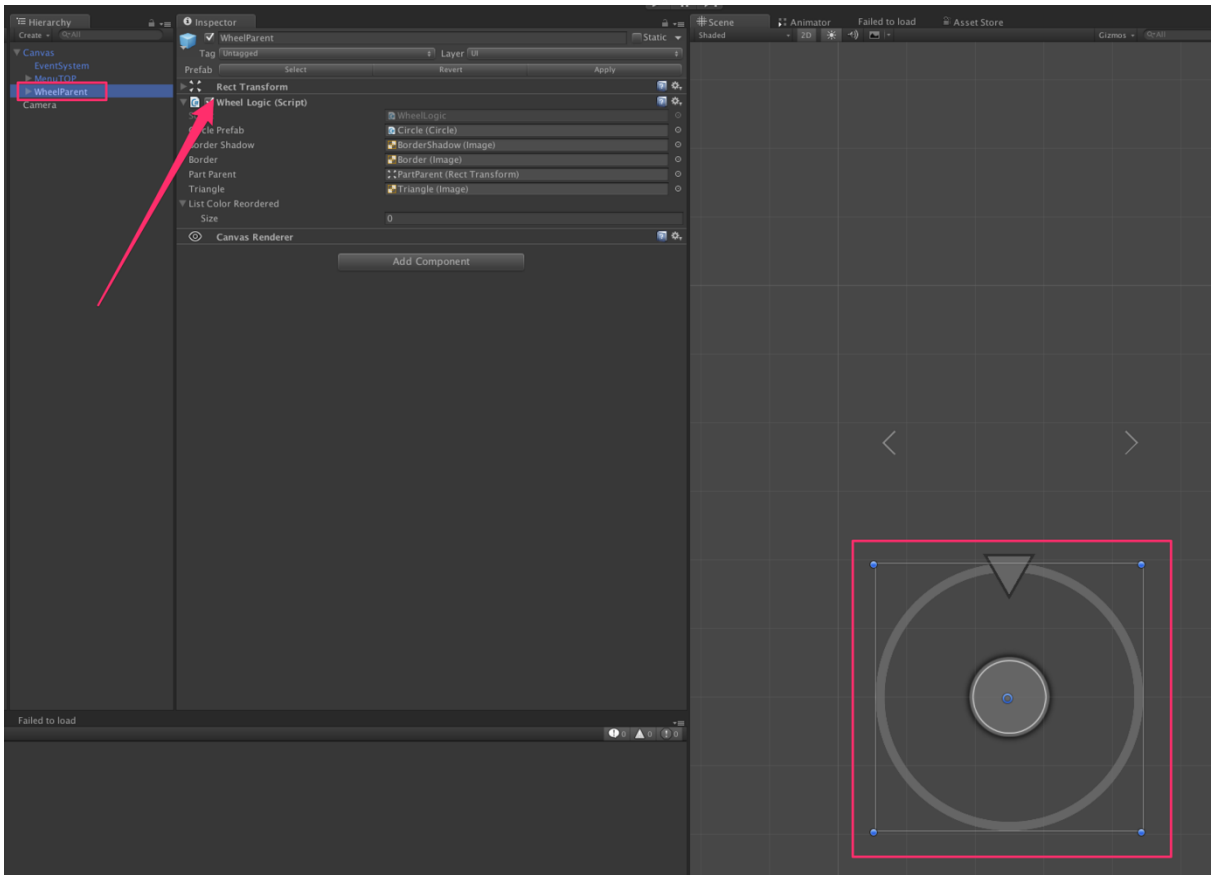
static void Util.SetRestartFromGameOver ( ) inlinestatic

Set restart from game over = true

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

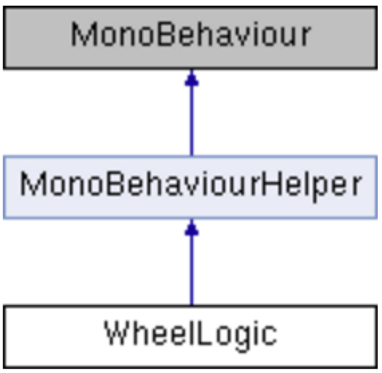
Real shuffle of List

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...

Inheritance diagram for WheelLogic:



Public Member Functions

void	**DOColorTriangle** ()

	Change the color of the triangle = color to find More...

bool	**CheckIfTriangleEqualWheelColor** ()
	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...

bool	**DOCheck** ()
	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...

Public Attributes

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

float	**speedWheel** = 3f
	Sped of the wheel, in seconds (total time in seconds to make 360 degree rotation), for the current level More...

Image	**borderShadow**
	Image of the shadow of border of the wheek More...

Image	**border**
	Image of the border of the wheek More...

RectTransform	**partParent**
	Reference to the GameObject who contains all the part of the wheel we will spawn More...

Image	**triangle**
	Reference to the Triangle Image on the top of the wheel More...

List< Color >	**listColorReordered** = new List<Color>()
	Reference to a list of color built for a level More...

Private Member Functions

void	**Awake** ()
	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 More...

void	**Start** ()
	Place the border and the border shadow at the good place More...

void	**DefineLevel** ()
	IMPORTANT ==> It's here we define the levels. Change the formulas if you want. More...

void	**BuildWheel** ()
	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...

Circle	**InstantiateCircle** ()
	Method to create a new circle = new part of the wheel More...

Circle	**InstantiateCircle** (float fillAmout, float angle, Color c)
	Method to create a new circle = new part of the wheel More...

Private Attributes

int	**numOfPart** = 12
	Number of parts in the wheel, for the current level More...

int	**numOfColor** = 3

	Number of colors in the wheel, for the current level More...
List< Circle >	allCircles = new List<Circle>()
	Reference to all the parts contained in the wheel, for the current level More...
Color	lastColor
	Reference to the last color to find, to avoid duplicate check More...
bool	firstChangeColor = 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...

Additional Inherited Members

Properties inherited from MonoBehaviourHelper

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 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 WheelLogic.InstantiateCircle ( ) inlineprivate

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

```
Circle 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> WheelLogic.allCircles = new List<Circle>() 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 WheelLogic.circlePrefab

Prefab of Circle. 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

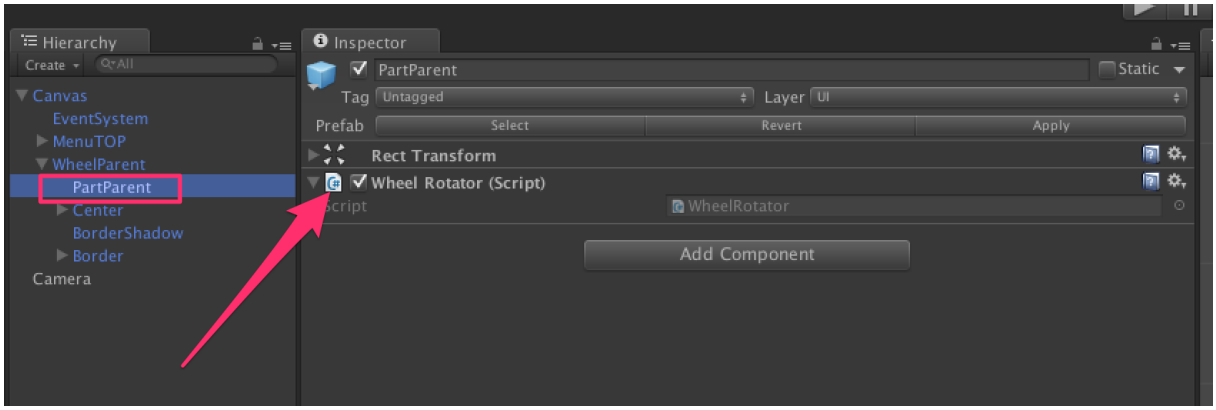
float WheelLogic.speedWheel = 3f

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

Image WheelLogic.triangle

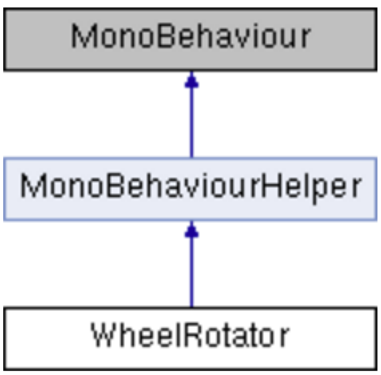
Reference to the Triangle Image on the top of the wheel

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...

Inheritance diagram for WheelRotator:



Public Attributes

int	**direction** = 1
	Two directions: left and right (1 and -1) More...

Properties

bool	**isSuccess** `[get]`
	Check if success = number of moove to do = 0 More...

Properties inherited from MonoBehaviourHelper

Private Member Functions

void	**Awake** ()
	Choose the start direction randmly and set firstStart to true More...

void	**Update** ()
	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...

void	**DORotateWheel** ()
	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...

Private Attributes

bool	**firstStart** = true
	Is it the first time we start the rotation for the level? More...

Tweener	**rotateTweener**

	Reference to the tweener who rotate the circle More...

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