SPIN THE CIRCLE

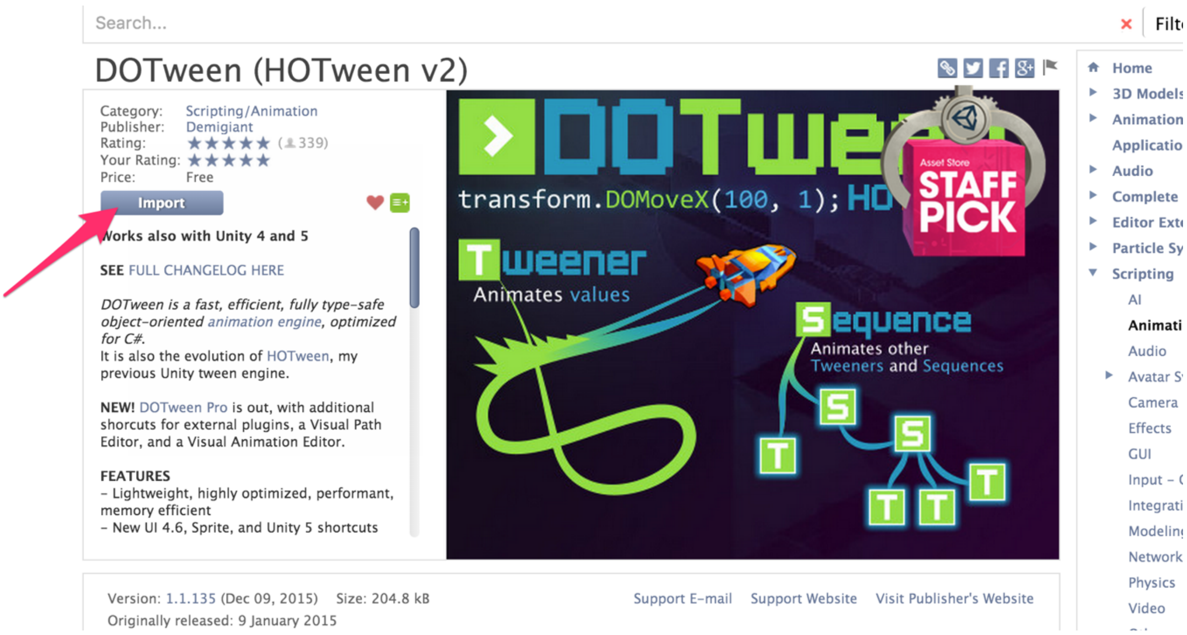
Thanks for your purchase.

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



DOTween (HOTween v2)

Category: Scripting/Animation
Publisher: Demigiant
Rating: ★★★★★ (339)
Your Rating: ★★★★★
Price: Free

Works also with Unity 4 and 5

SEE FULL CHANGELOG HERE

DOTween is a fast, efficient, fully type-safe object-oriented animation engine, optimized for C#.
It is also the evolution of HOTween, my previous Unity tween engine.

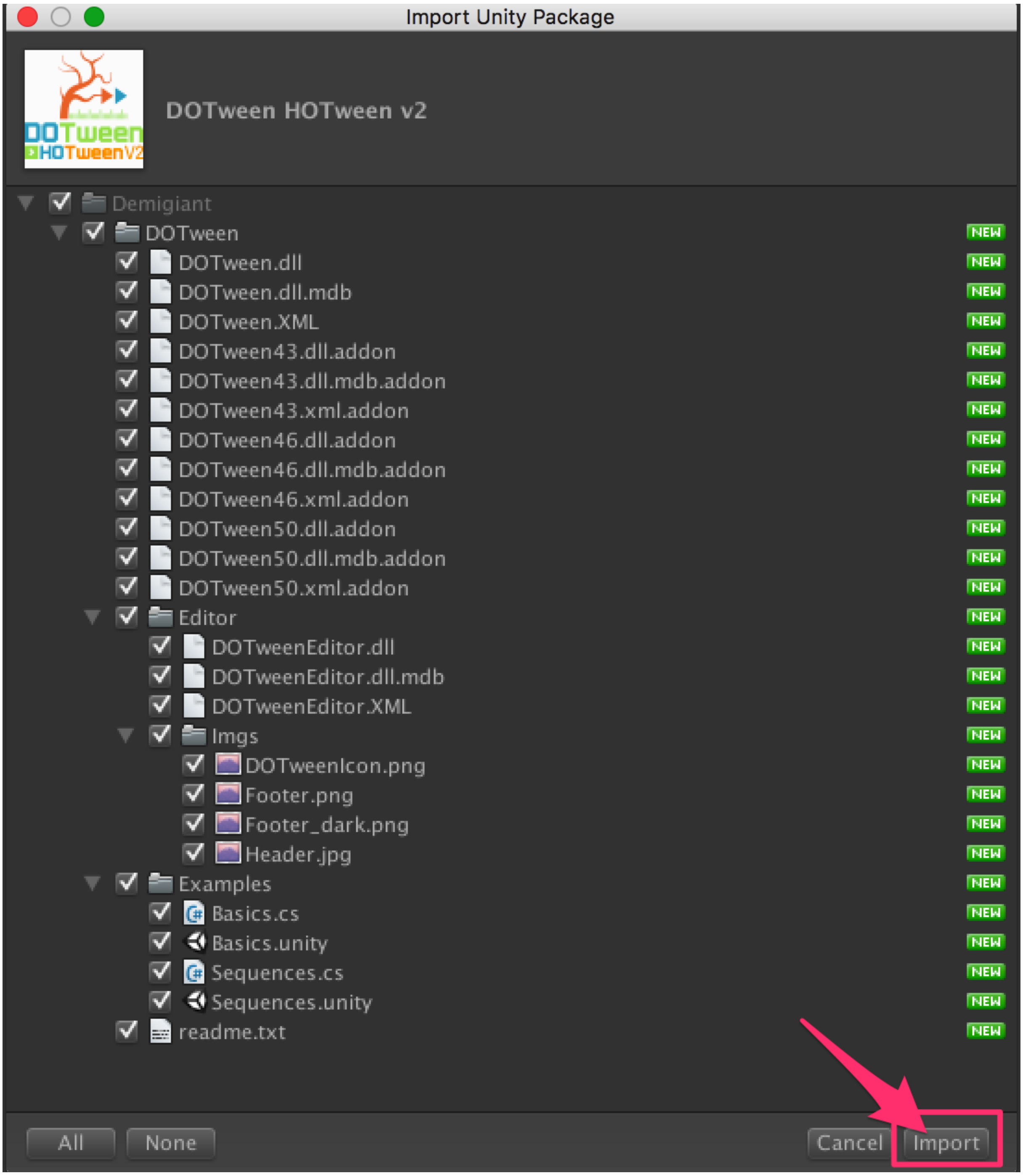
NEW! DOTween Pro is out, with additional shorcuts for external plugins, a Visual Path Editor, and a Visual Animation Editor.

FEATURES
– Lightweight, highly optimized, performant, memory efficient
– New UI 4.6, Sprite, and Unity 5 shortcuts

Version: 1.1.135 (Dec 09, 2015) Size: 204.8 kB
Originally released: 9 January 2015

Support E-mail Support Website Visit Publisher's Website

2/ Import the package into Unity

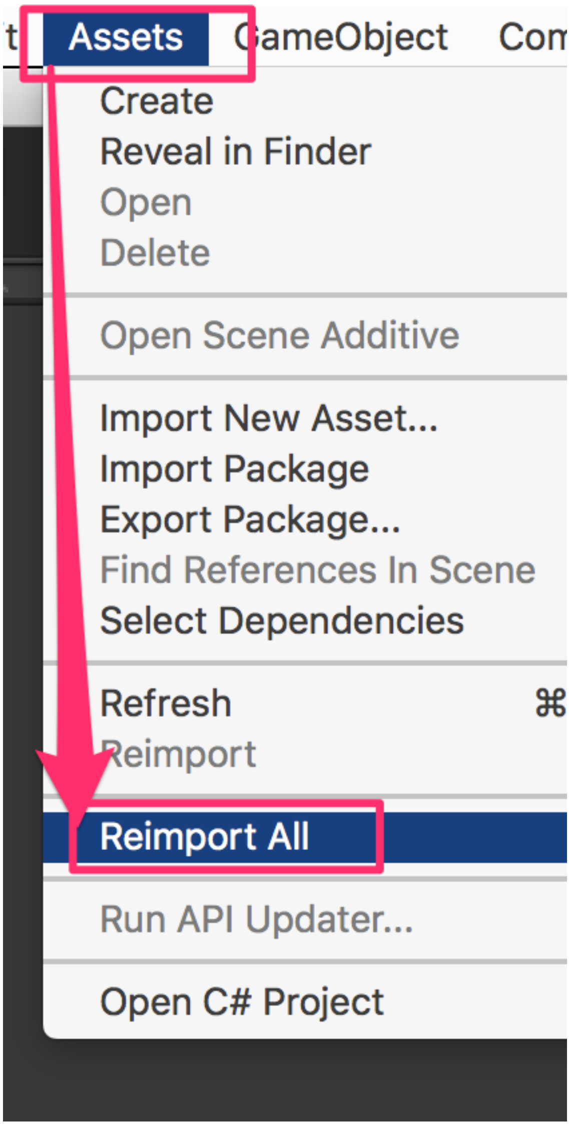


Import Unity Package

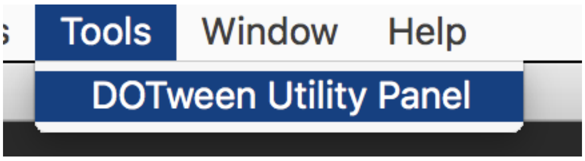
DOTween HOTween v2

```
☑ Demigiant
  ☑ DOTween                          NEW
    ☑ DOTween.dll                    NEW
    ☑ DOTween.dll.mdb                NEW
    ☑ DOTween.XML                    NEW
    ☑ DOTween43.dll.addon            NEW
    ☑ DOTween43.dll.mdb.addon        NEW
    ☑ DOTween43.xml.addon            NEW
    ☑ DOTween46.dll.addon            NEW
    ☑ DOTween46.dll.mdb.addon        NEW
    ☑ DOTween46.xml.addon            NEW
    ☑ DOTween50.dll.addon            NEW
    ☑ DOTween50.dll.mdb.addon        NEW
    ☑ DOTween50.xml.addon            NEW
    ☑ Editor                         NEW
      ☑ DOTweenEditor.dll            NEW
      ☑ DOTweenEditor.dll.mdb        NEW
      ☑ DOTweenEditor.XML            NEW
      ☑ Imgs                         NEW
        ☑ DOTweenIcon.png            NEW
        ☑ Footer.png                 NEW
        ☑ Footer_dark.png            NEW
        ☑ Header.jpg                 NEW
    ☑ Examples                       NEW
      ☑ Basics.cs                    NEW
      ☑ Basics.unity                 NEW
      ☑ Sequences.cs                 NEW
      ☑ Sequences.unity              NEW
    ☑ readme.txt                     NEW
```

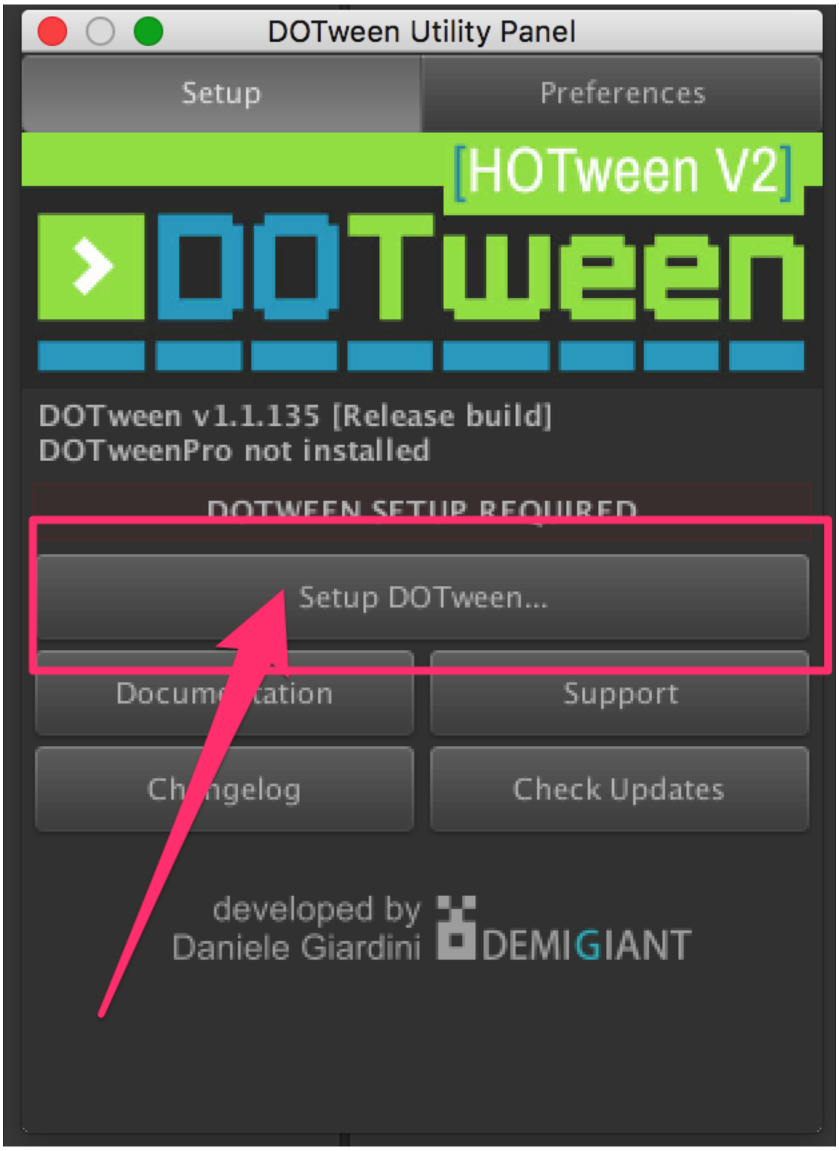
3/If you don’t see the « Tools » in the top of the Unity Screen, please do this :



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



5/Click on « Setup DOTween ».

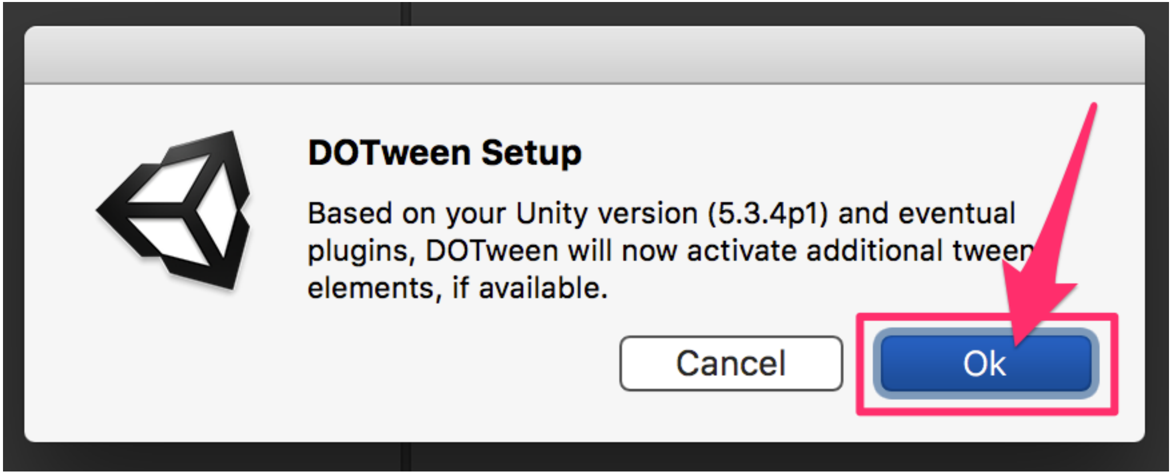


DOTween Utility Panel

DOTween v1.1.135 [Release build]
DOTweenPro not installed

DOTWEEN SETUP REQUIRED

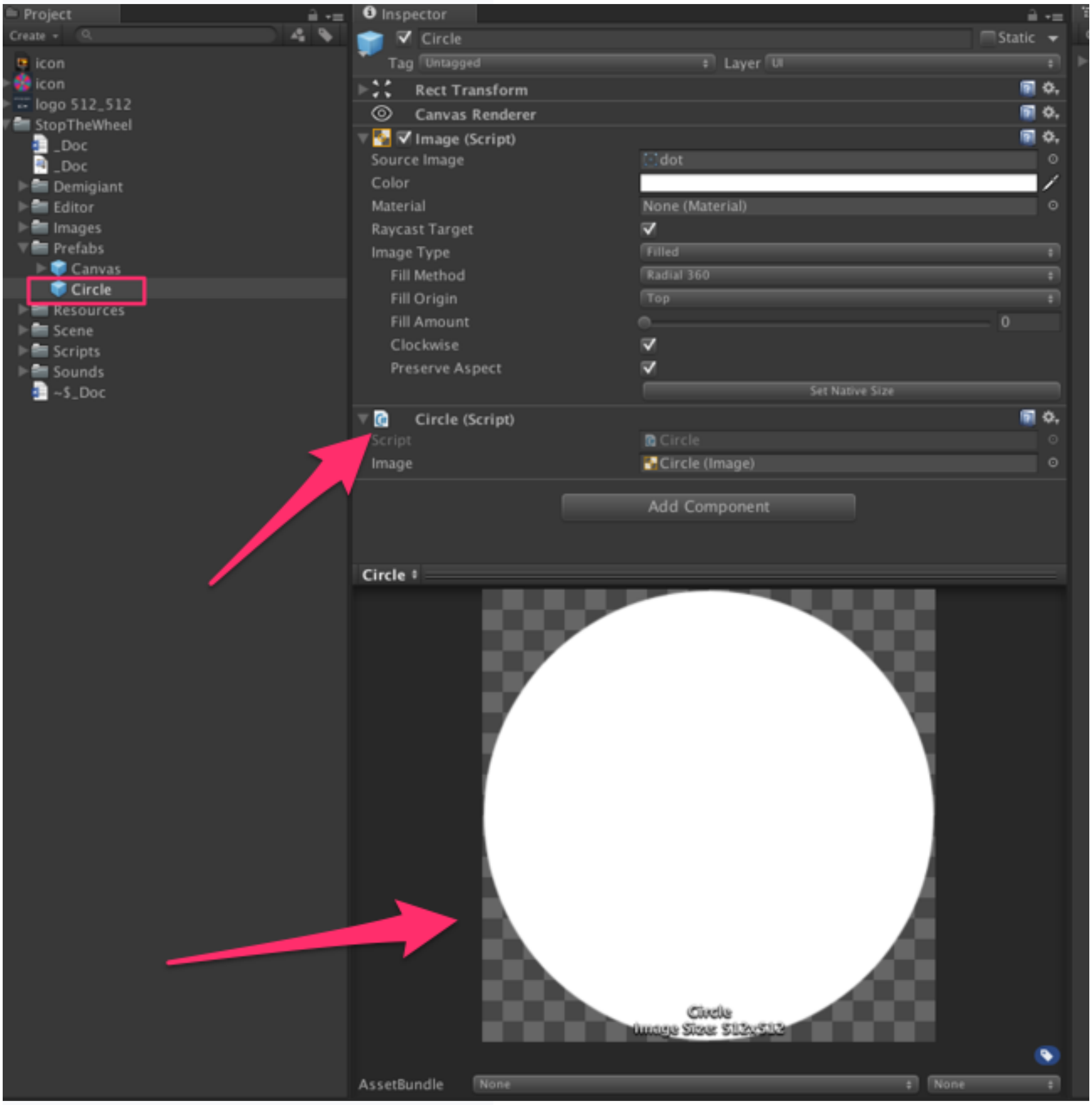
6/And to finish, click on the « OK » button.



DOTween Setup

Based on your Unity version (5.3.4p1) and eventual plugins, DOTween will now activate additional tween elements, if available.

CirclePart Class Reference



Circle

Tag Untagged Layer UI

Rect Transform

Canvas Renderer

Image (Script)

Source Image: dot

Color

Material: None (Material)

Raycast Target ☑

Image Type: Filled

Fill Method: Radial 360

Fill Origin: Top

Fill Amount: 0

Clockwise ☑

Preserve Aspect ☑

Circle (Script)

Script: Circle

Image: Circle (Image)

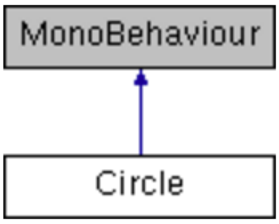
Circle

Circle
Image Size: 512x512

AssetBundle None None

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...

Inheritance diagram for CirclePart:



Public Member Functions

CirclePart	Init (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...
float	GetMiddleAngle ()
	Get the angle of the middle of the part of circle More...

Public Attributes

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

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 CirclePart.Init (float fillAmout,
float angle,
Color color
) inline

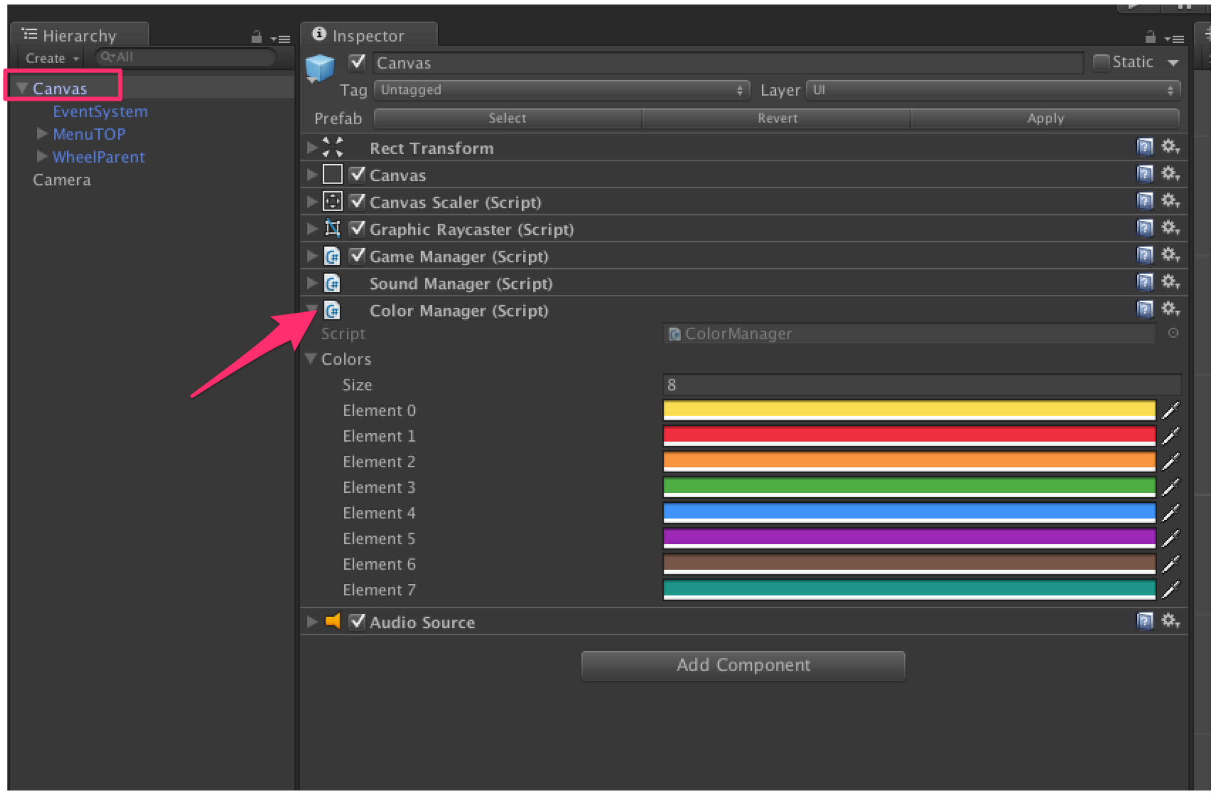
Init the circle = the part of the circle. Each part is defined with a fillAmount = 1 / number of part in the circle, an angle and a color

Member Data Documentation

Image CirclePart.image

The image = a simple circle

ColorManager Class Reference



☑ Canvas □ Static

Tag Untagged Layer UI

Prefab Select Revert Apply

Script ColorManager

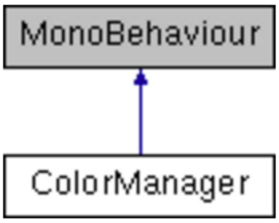
Colors

Size 8

Element 0 🟨
Element 1 🟥
Element 2 🟧
Element 3 🟩
Element 4 🟦
Element 5 🟪
Element 6 🟫
Element 7 🟩

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

Inheritance diagram for ColorManager:



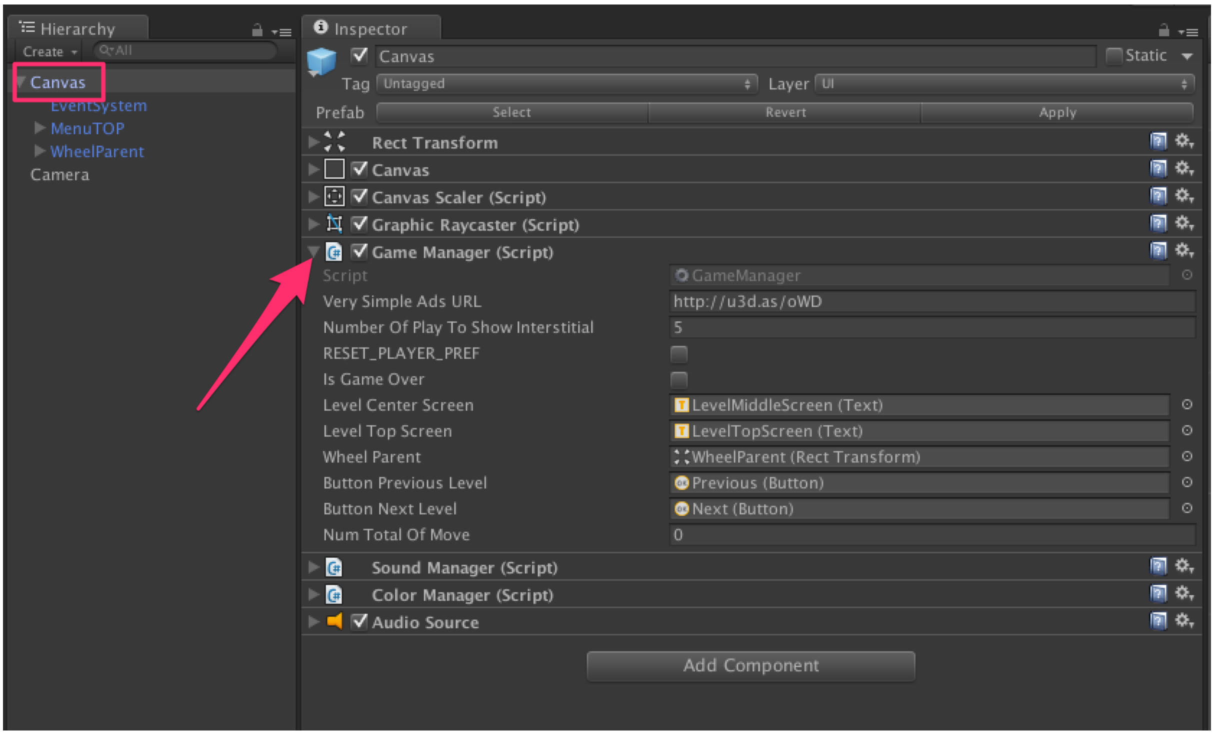
Public Attributes

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



☑ Canvas

Tag Untagged Layer UI

Prefab

Script GameManager

Very Simple Ads URL http://u3d.as/oWD

Number Of Play To Show Interstitial 5

RESET_PLAYER_PREF □

Is Game Over □

Level Center Screen LevelMiddleScreen (Text)

Level Top Screen LevelTopScreen (Text)

Wheel Parent WheelParent (Rect Transform)

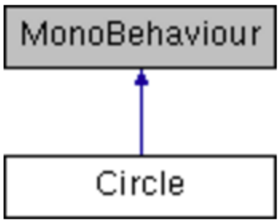
Button Previous Level Previous (Button)

Button Next Level Next (Button)

Num Total Of Move 0

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...

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 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...
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	**textLastScore**
Text	**textBestScore**

Properties

int	**point** `[get, set]`

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

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"

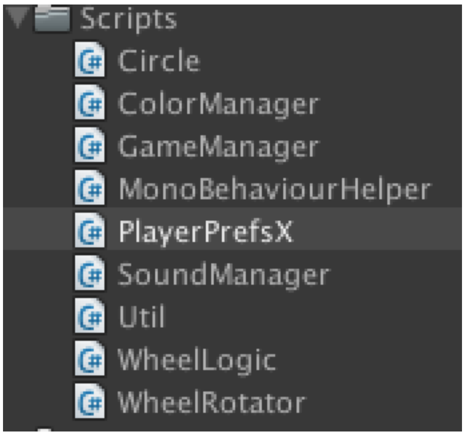
If you want to monetize this game, get VERY SIMPLE ADS at this URL: http://u3d.as/oWD

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...

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)

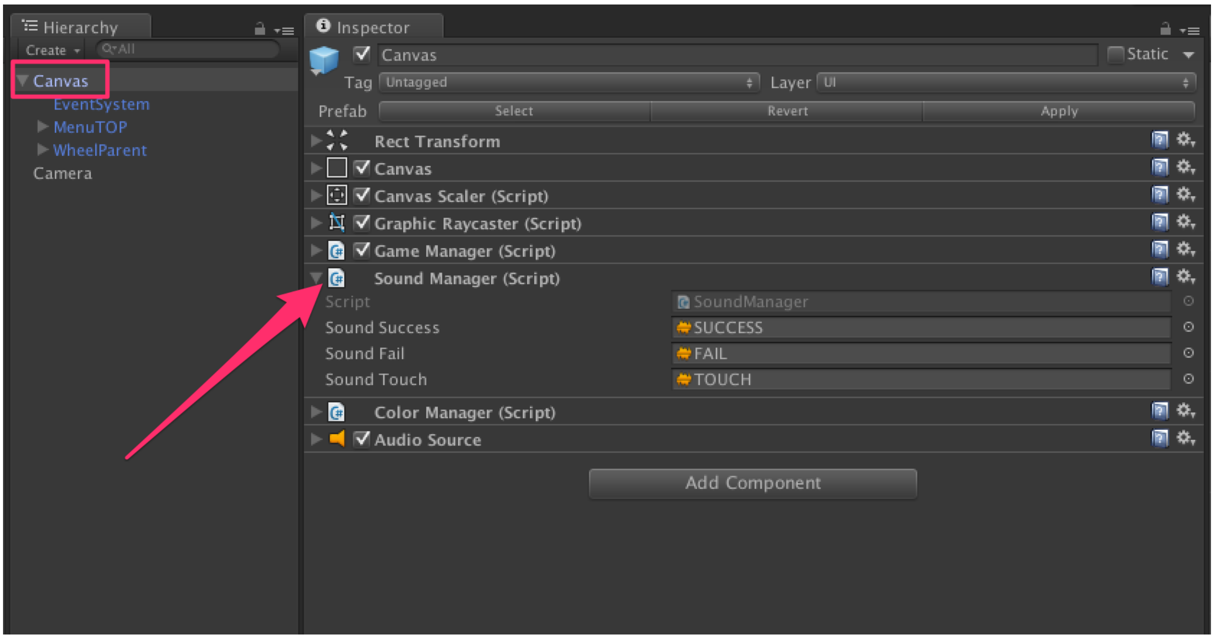
Static Private Attributes

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

Detailed Description

A player pref extension

SoundManager Class Reference



☑ Canvas

Tag Untagged

Layer UI

Prefab

Script SoundManager

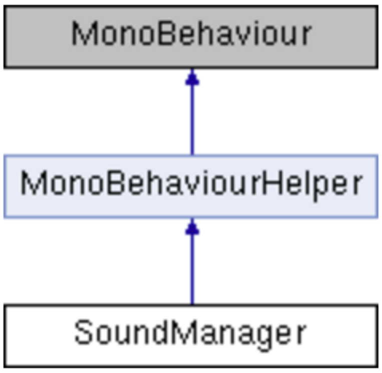
Sound Success SUCCESS

Sound Fail FAIL

Sound Touch TOUCH

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

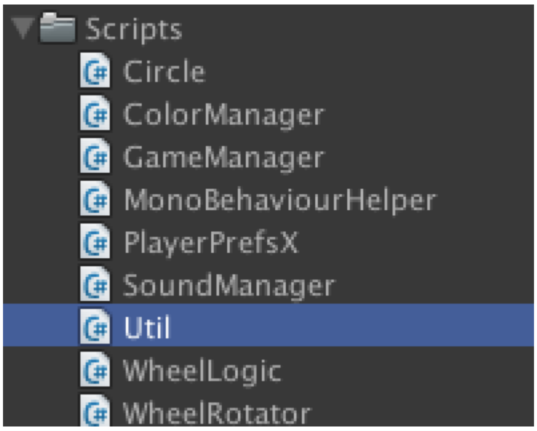
AudioClip SoundManager.soundSuccess private

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



Scripts
Circle
ColorManager
GameManager
MonoBehaviourHelper
PlayerPrefsX
SoundManager
Util
WheelLogic
WheelRotator

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 bool	**SetLastScore** (int score)
static int	**GetLastScore** ()
static int	**GetBestScore** ()
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 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

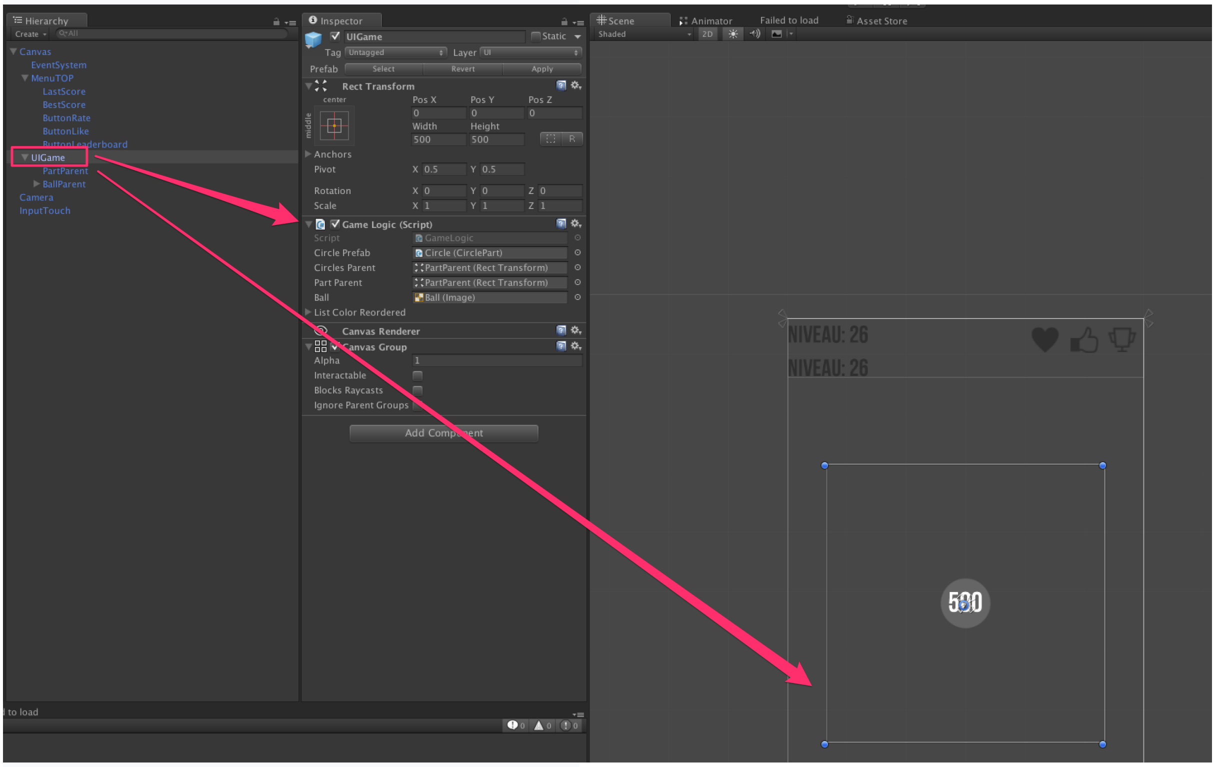
static bool Util.RestartFromGameOver ( ) inlinestatic

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



UIGame ☐ Static
Tag Untagged Layer UI
Prefab

Rect Transform
center / middle
Pos X 0 Pos Y 0 Pos Z 0
Width 500 Height 500
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

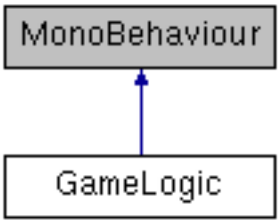
☑ Game Logic (Script)
Script GameLogic
Circle Prefab Circle (CirclePart)
Circles Parent PartParent (Rect Transform)
Part Parent PartParent (Rect Transform)
Ball Ball (Image)
List Color Reordered

Canvas Renderer

☑ Canvas Group
Alpha 1
Interactable ☐
Blocks Raycasts ☐
Ignore Parent Groups ☐

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...

Inheritance diagram for GameLogic:



Public Member Functions

void	DOColorBall ()
	Change the color of the ball = color to find More...
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 More...

Public Attributes

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

Properties

GameManager	**gameManager** `[get]`

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	**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 oposite direction More...
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. 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...
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 amout property to cretae the parts of the circle More...
CirclePart	InstantiateCircle ()
	Method to create a new circle = new part of the circle More...
CirclePart	InstantiateCircle (float fillAmout, float angle, Color c)
	Method to create a new circle = new part of the circle More...

Private Attributes

int	numOfPart = 12
	Number of parts in the circle, for the current level More...
int	numOfColor = 3
	Number of colors in the circle, for the current level More...
List< CirclePart >	allCircles = new List<CirclePart>()
	Reference to all the parts contained in the circle, for the current level More...
Color	lastColor
	Reference to the last color to find, to avoid duplicate check More...
bool	**shuffleColorAray** = true
GameManager	**_gameManager**
bool	firstMove = 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 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** 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 GameLogic.InstantiateCircle ( ) inlineprivate

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

```
CirclePart 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> GameLogic.allCircles = new List<CirclePart>() private

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

Image GameLogic.ball

Reference to the ball Image = player

CirclePart 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