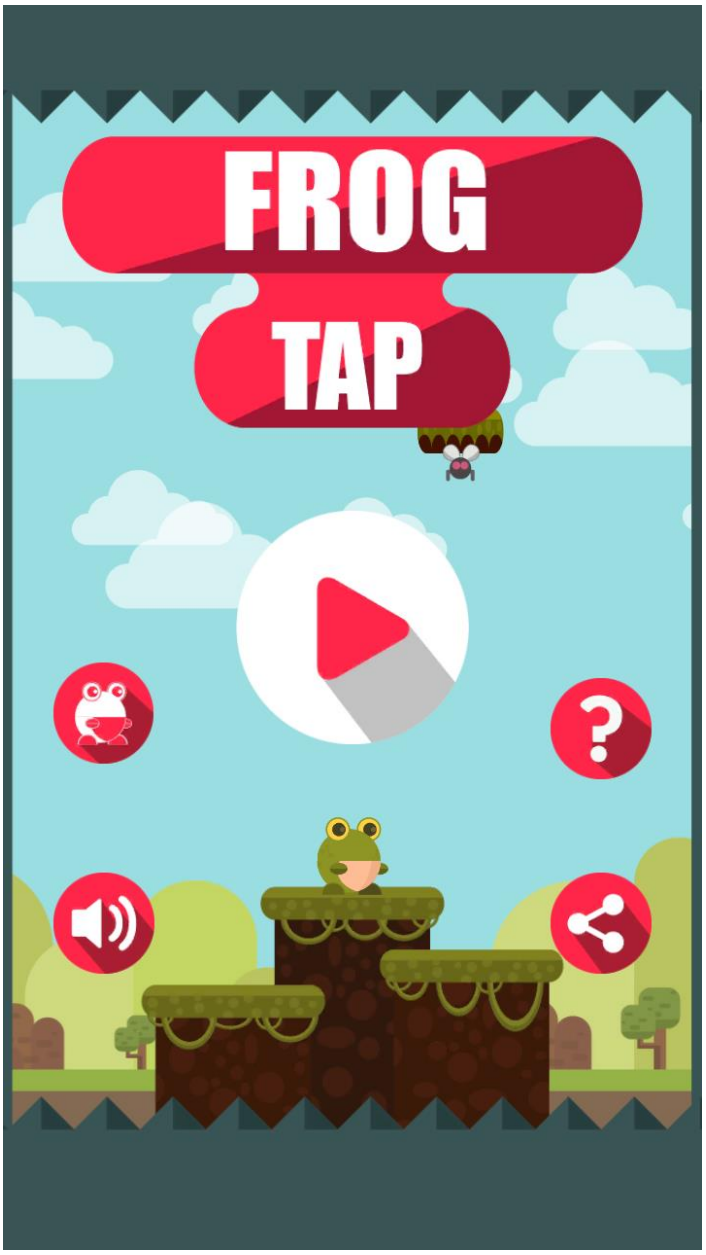
Frog Tap Complete Game Template

Documentation



Version 1.0

Thanks for your purchase!

Index

Introduction

This asset is a complete game template that you can modify as much as you want and this document will help you to find and understand the main parts of this one.

The core of this asset are three scripts:

- **Game controller:** Basically controls the game's life and the game's navigation.
- **Level manager:** It's the level generation system.
- **Reskin manager:** Used to change the game's appearance when a new character is selected.

This asset includes two important tasks for the end user and you need to see the **Project's structure** section and the **Native Share Setup** section to configure some values for your game:

- Native Share game
- Rate game

Some resources like sounds, one font and one animation were downloaded from external pages. You can use these resources for free and for commercial use as long as the author is credited. In the **Credits** section you will find all the information and the license for each resource used.

What is frog tap?

Frog tap is an infinite game type where our main character will move from platform to platform trying to reach a highest score while this one eats flies along the way. With a simple tap on the screen the character will launch its tongue to hook it to the next platform.

Project's structure

The project is structured in a main folder named FrogTap and then in descriptive folders to find the specific resource you want, you can see the folder FrogTap into the folder Assets which is located into the project window. See the image 1.

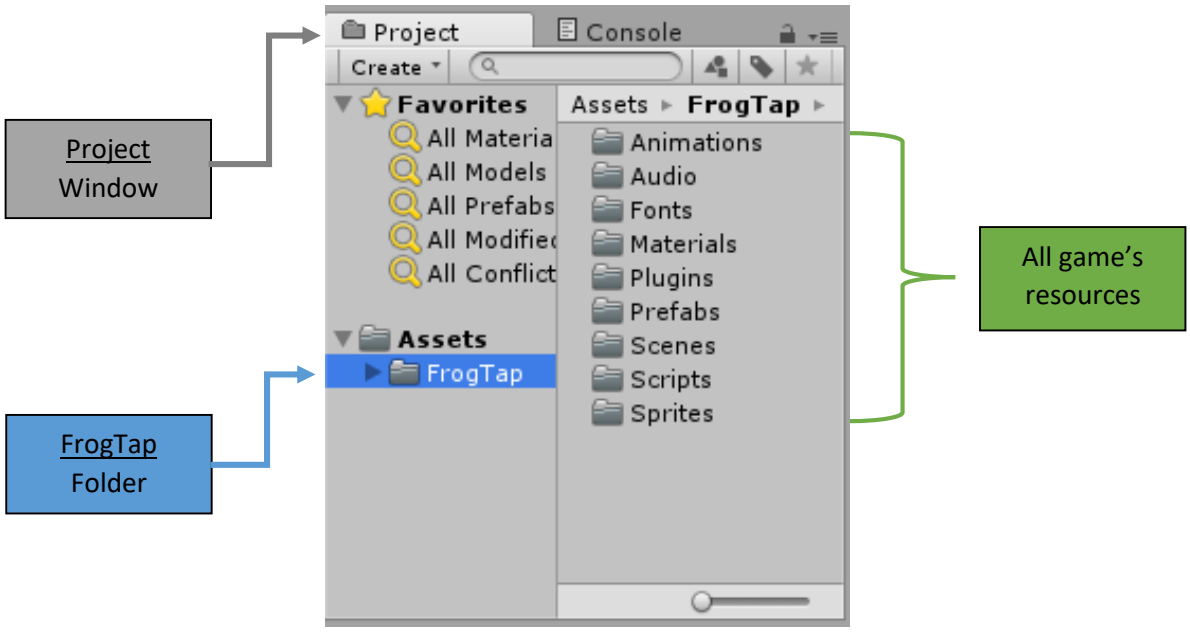
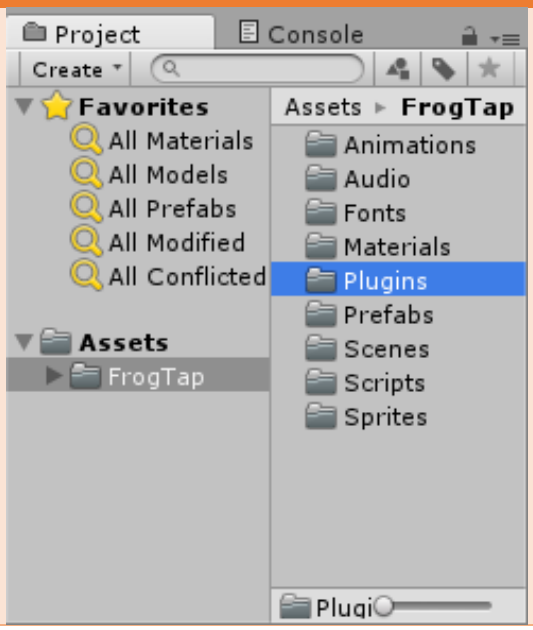
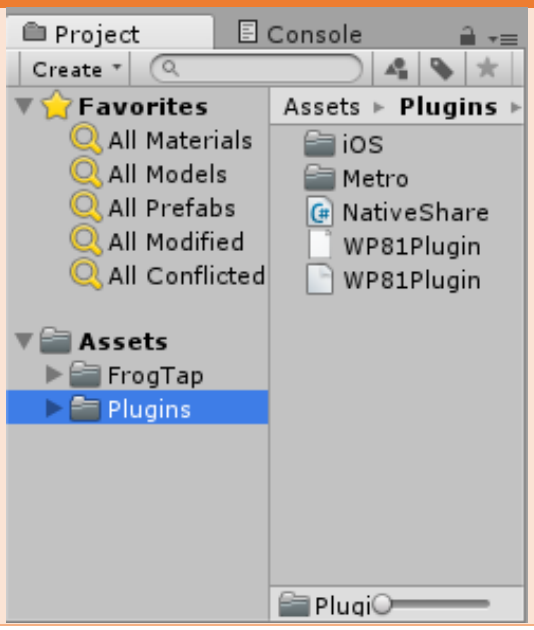


Image 1 Game's resources

NOTE: To use the social share and rate game functions, you have to move the folder Plugins from the folder FrogTap to the root folder (Assets). See the below table:

Before	After
	
Folder **Plugins** into the folder **FrogTap**	Folder **Plugins** in the root folder **Assets**

To see the game in action you need to open the scene MainGame which is located in the folder Scenes located into the folder FrogTap. See the image 2.

Image 2 Main game scene

Managers

The managers are the responsible scripts of make the game works correctly, you can see them in the inspector window selecting the game object Managers in the hierarchy window. In the image 3 you can see the three managers:

- Game controller
- Level manager
- Reskin manager

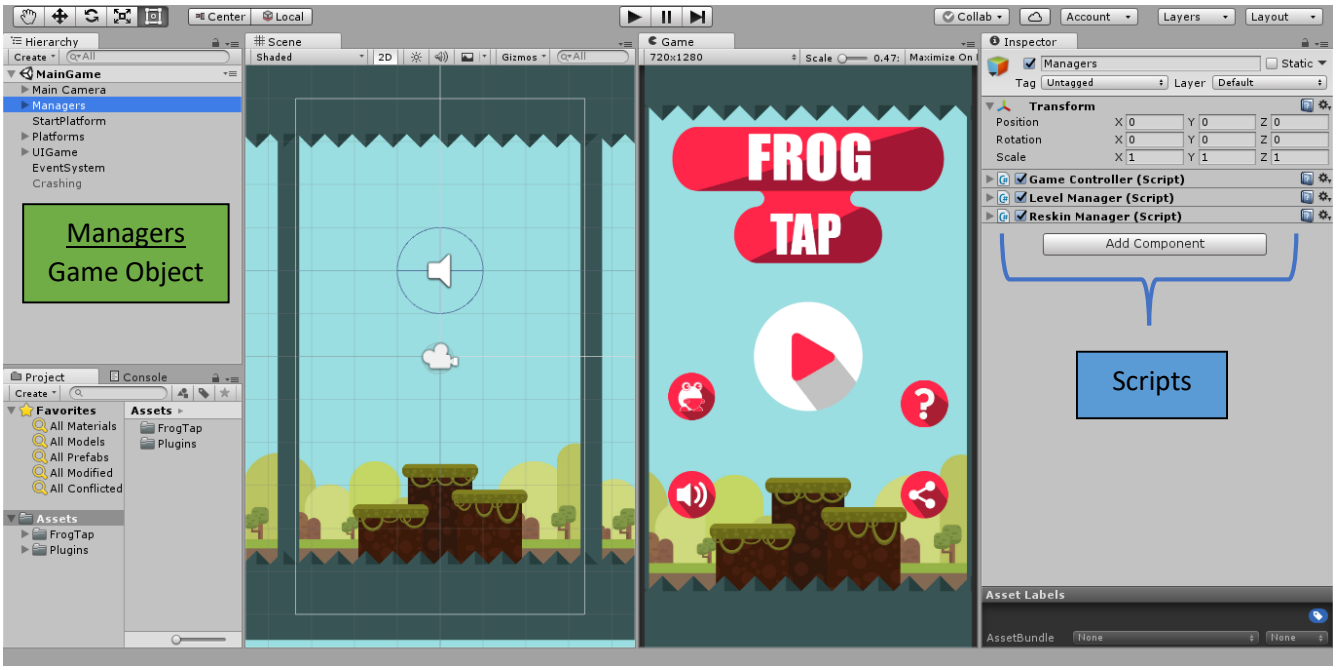


Image 3 Game's managers

Game controller

The game controller script is accessible for any object in the game, for that reason as you can see in the image 4 it has references for the rest of the managers (Level manager and Reskin manager).

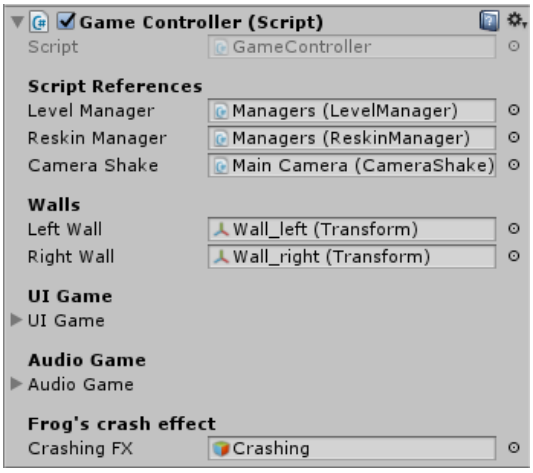


Image 4 Game controller script

This script has the following responsibilities:

- **Control the life of game:**
  - Start game
  - Pause game
  - Resume game
  - Game over
- **UI (User Interface) Navigation**: You can navigate into the game using the buttons of each menu or with the back button in case of the mobile devices.
  All menus of game are listed below:
  - Menu main
  - Menu credits

- Menu game
- Menu pause
- Menu game over
- Menu shop

❖ **Others:**

- Share game (*This method needs configuration, please see the section **Native Share Setup***).
- Rate game (*This method needs configuration, please see the section **Native Share Setup***).
- Increase, display and store the score and best score of player.
- Increase, display and store the flies counter.
- Manage the sound effects.

NOTE: All values such as the score, the flies counter and the characters purchased are stored in the application using **PlayerPrefs,** if you want to reset all values, you have to open the ***GameController*** script, then find the line of code number 47 and uncomment it, save the file, execute the game and finally comment the line again.

Game controller's sections

Script References: In this section there are references to level manager and reskin manager with the goal of making all the managers are available for the rest of the objects, in this way there is only one reference per manager and the camera shake script is used to make a shake effect when the character dies. See the Image 5.

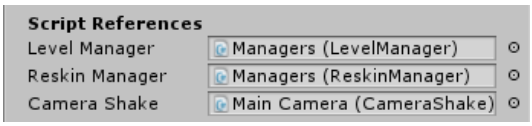
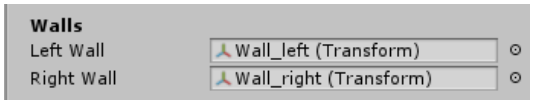


Image 5 Scripts references

Walls: The walls are used to limit the space where the frog can launch its tongue, when the frog's tongue collide with a wall, this one gets back to its origin. See the image 6.



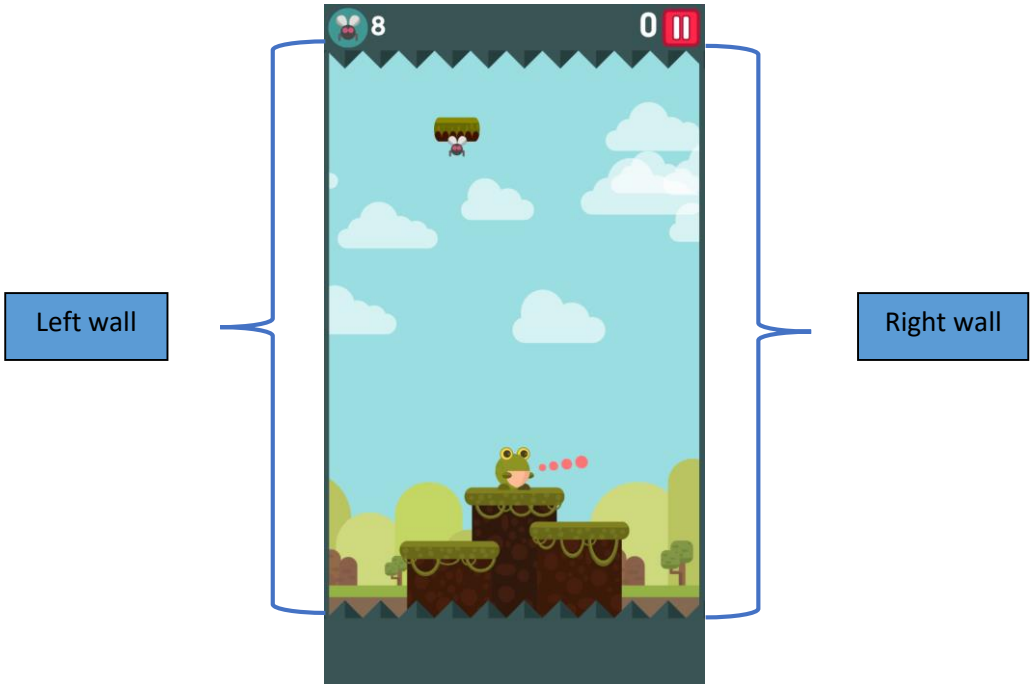


Image 6 Walls to limit the game's space

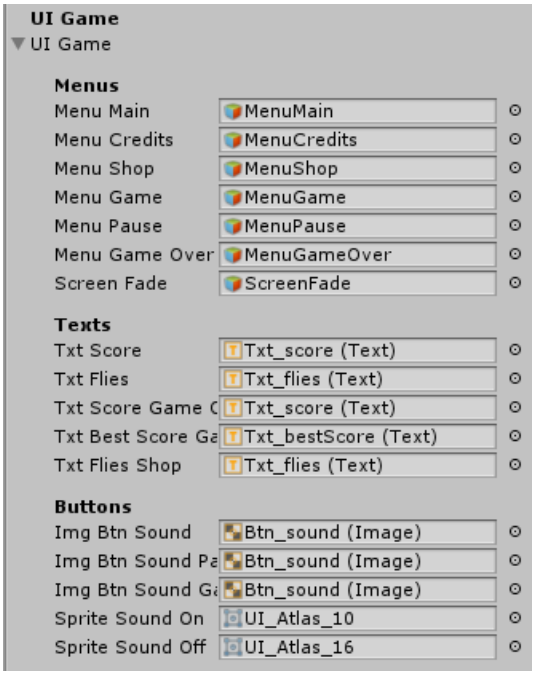


Image 7 UI game section

UI Game: This section contain the references of all menus, texts and buttons you see in the game, this references are required to create the game's navigation. See the image 7.

You can modify each menu in the hierarchy window. See the image 8.

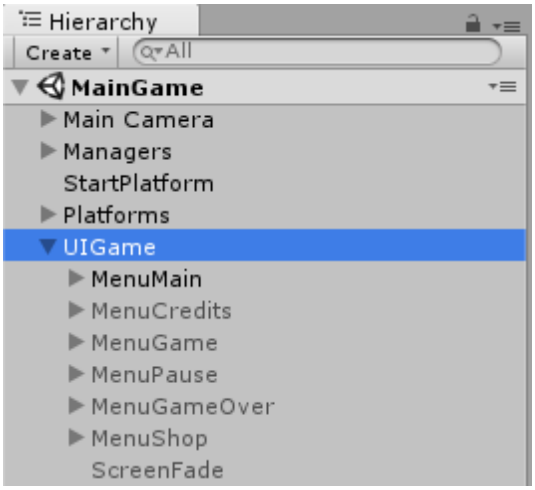
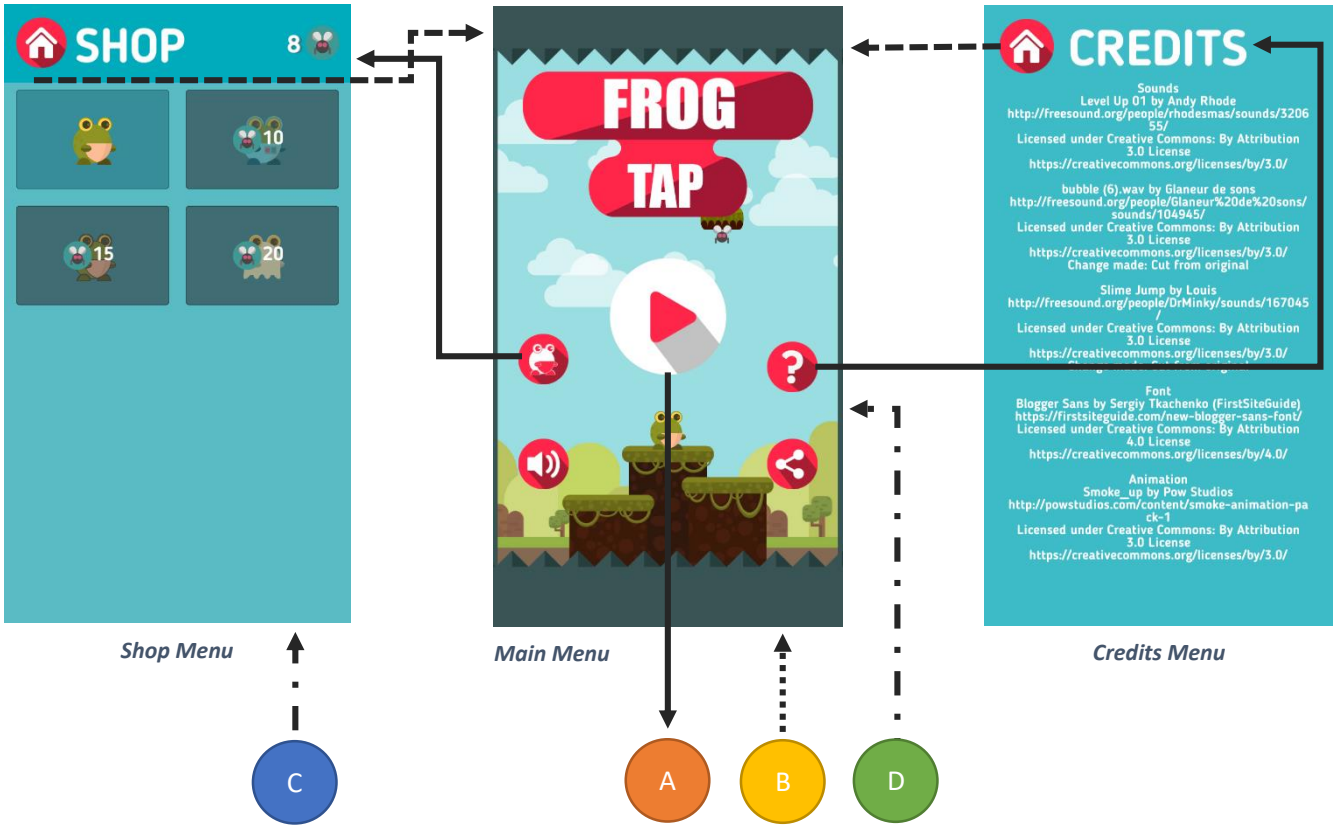


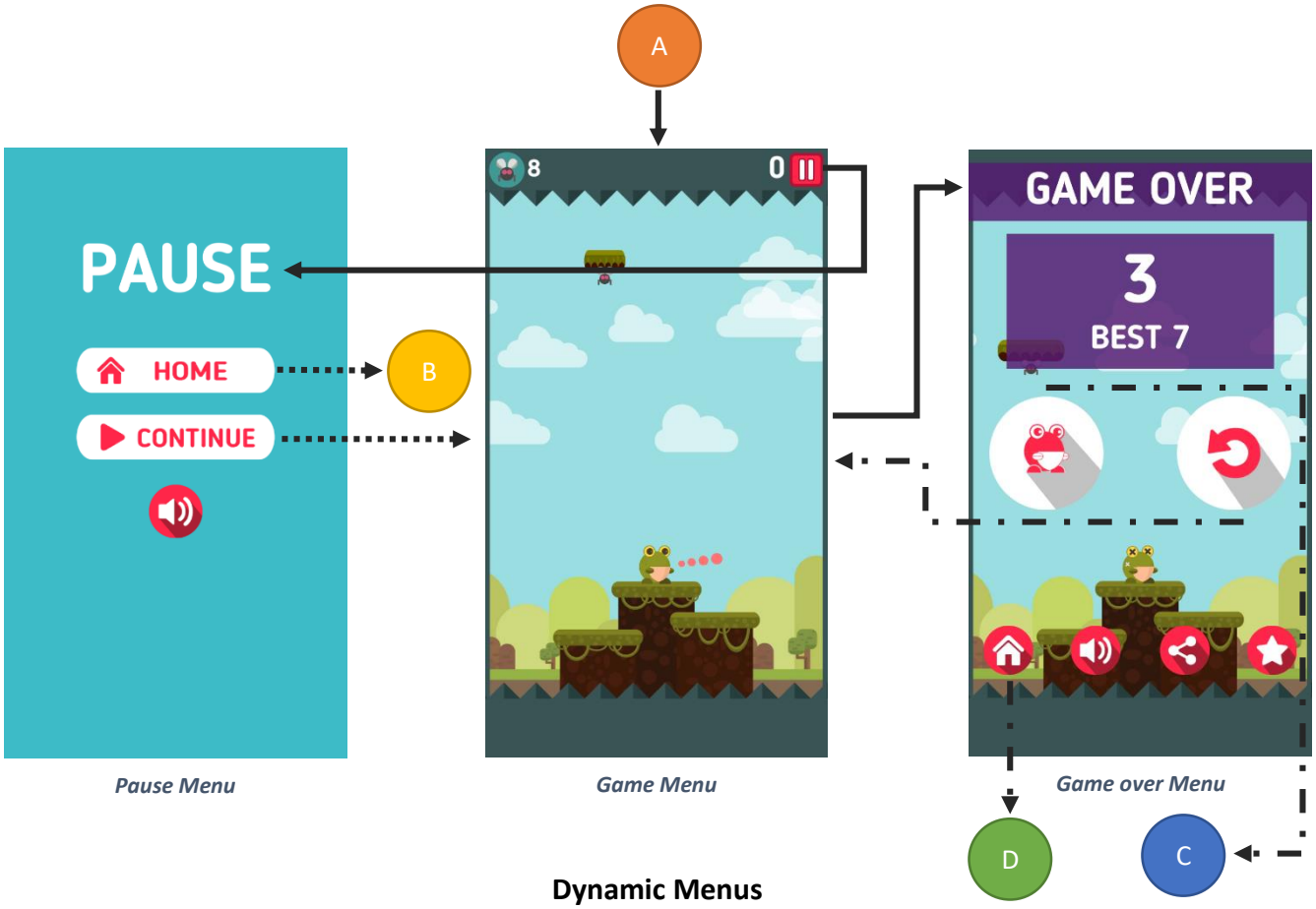
Image 8 Game's menus

Game Navigation

- If the user is in the main menu, this one can go to the shop menu or credits menu and then get back.
- If the user press the play button, this one goes to the game menu.



- If the user is in the game menu, this one can go to the pause menu or if the character dies the game over menu is displayed.
- If the user is in the pause menu, this one can get back to the game menu or go to the main menu.
- If the user is in the game over menu, this one can get back to the game menu using the restart button or open the shop menu or go to the main menu.



Pause Menu *Game Menu* *Game over Menu*

Dynamic Menus

Some menus have animation, you can modify, delete or add more animations just selecting a menu in the inspector and open the animation window (Window->Animation), then you can modify the properties you want. See the image 9.

Image 9 Animation window

Audio Game: This section contain the references to the audio sources and audio clips. The audio sources are used to reproduce the sounds (audio clips). See the image 10.

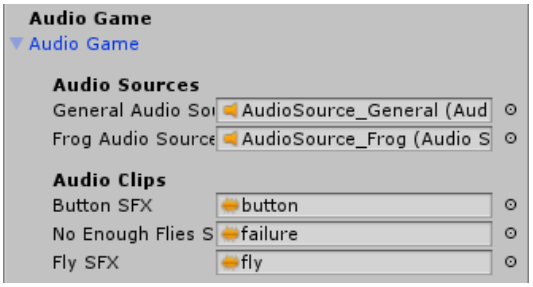


Image 10 Audio game section

The General Audio Source is used to reproduce general sounds effects for the game:

- Button SFX: Used when the user interacts with a button.
- No enough flies SFX: Used when the user wants to buy a character (frog) but the user not has enough flies to buy it.
- Fly SFX: Used when the character eats a fly.

The Frog Audio Source reproduces sounds effects just for the character, the references for the audio clips are in the script ***Player Control***:

- Launch tongue SFX: Played when the frog launch its tongue.
- Crash SFX: Played when the frog crashing.
- Score SFX: Played when the frog reaches a platform.

Frog's crash effect: Effect used when the character crashing with the spikes. You can find it in the hierarchy window. See the image 12.

Image 11 Frog's crash effect in the inspector window

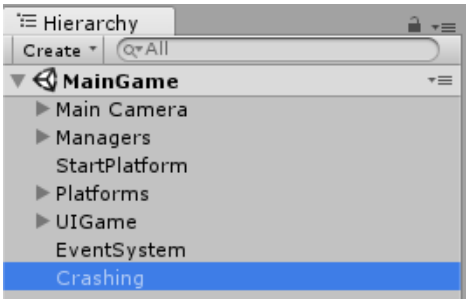


Image 12 Frog's crash effect in the hierarchy window

NOTE: In the folder FrogTap there is a folder named Sprites which contain an image named **"UI_Atlas"** as you can see in the image 13, this atlas has the UI used for this game, but this one has extra buttons you can use to represent other functions like:

- Leaderboard
- Achievements
- Music on
- Music off
- Settings
- Delete ADS
- Restore IAP (In-App Purchasing)
- Rewarded Video Ads

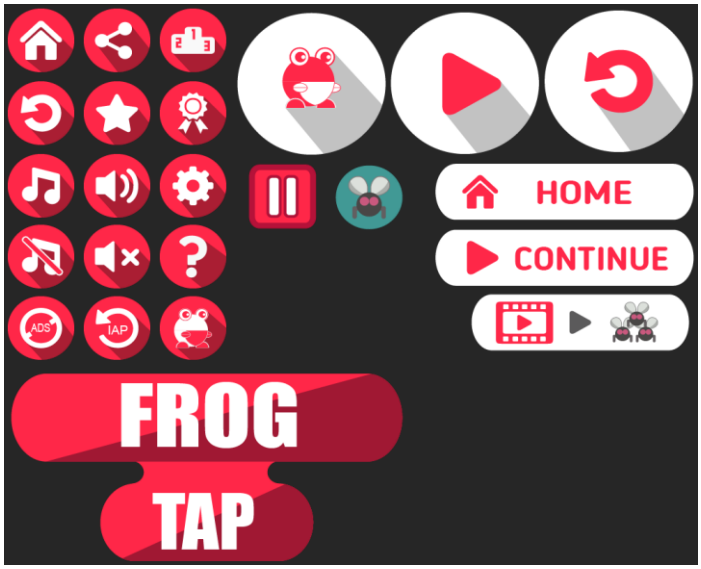


Image 13 Atlas UI Game

Level Manager

The level manager script is the level generation system with the following responsibilities:

- ❖ Create platforms and pooled (reuse them).
- ❖ Create flies provided by ***Reskin manager*** and pooled.
- ❖ Create the game's level.
- ❖ Increase the game's difficulty.

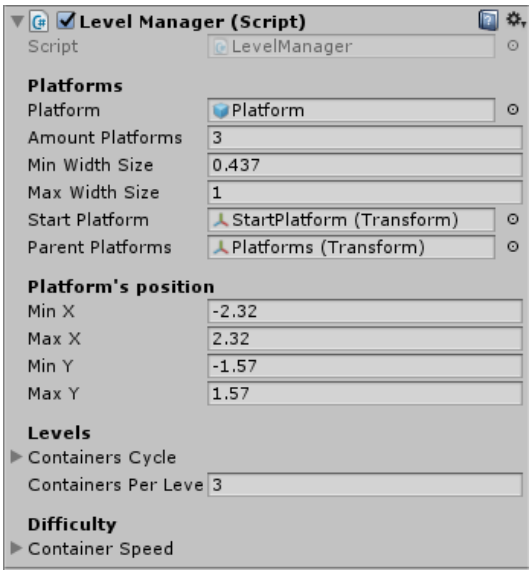


Image 14 Level manager script

Level manager's sections

Platforms: In this section all the platforms used along the game are configured. The necessary settings are listed below:

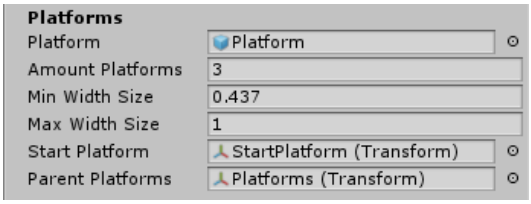


Image 15 Platform section

- Platform: The prefab game object used as a template to be created, see the image 16. All prefabs are in the folder *Prefabs* located into the folder *FrogTap*.

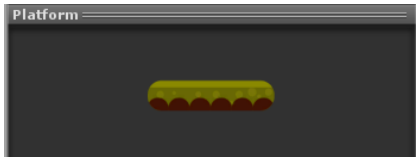


Image 16 Platform prefab

- Amount platforms: How many platforms will be created? , for optimization instead of create a platform each time a new platform is needed, we reuse them along the game, for that reason when this script is loaded just three platforms are instantiated. If you need more platforms just increase this variable.

- Min width size: The minimum width size the platform will have, this value is used as a reference to create a random size.

- Max width size: The maximum width size the platform will have, this value is used as a reference to create a random size.

- Start platform: The start platform is the platform where the character starts as you can see in the image 17.

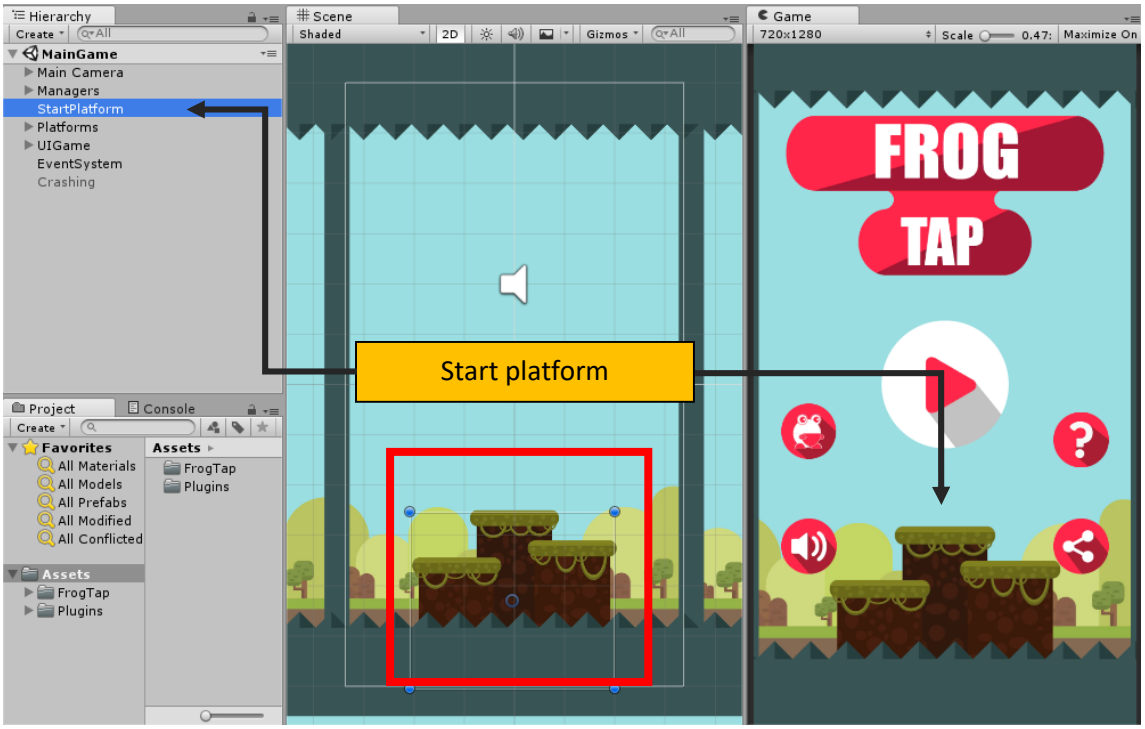


Image 17 Start platform object

- Parent platform: The parent platform is used to attach each platform created to this object, this will keep the hierarchy clean. This object also contain the objects ***containers***, as you can see in the image 18.

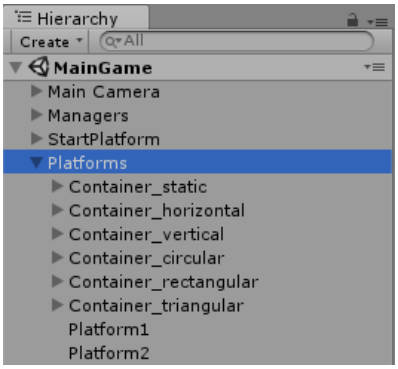


Image 18 Parent platform object

Containers

The containers are objects with an animator component attached (except static container) and one child named PointPlatform, with the animation window we created different movements for the object PointPlatform, and then in the ***level manager*** script, we attach a platform to the PointPlaform object, it means that the PointPlatform object will be the parent of the platform. See the image 19.

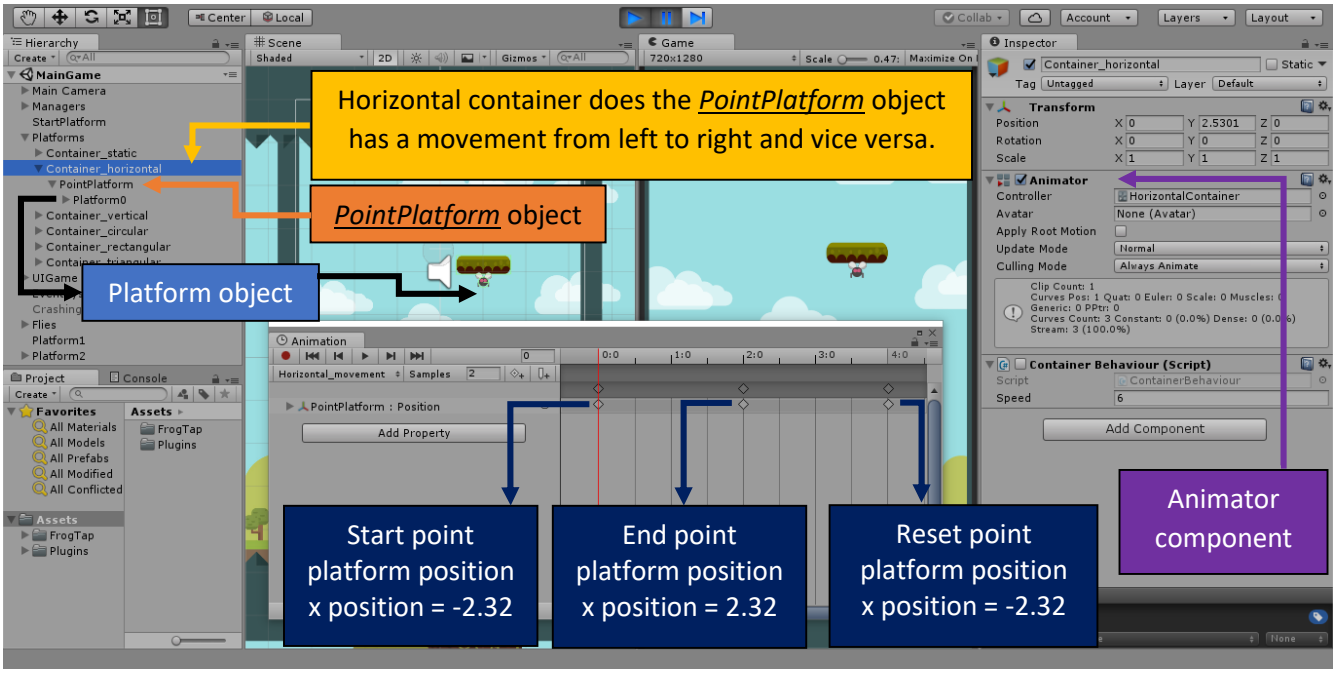


Image 19 PointPlatform object

Platform's position

Min X	-2.32
Max X	2.32
Min Y	-1.57
Max Y	1.57

Image 20 Platform's position section

Platform's position: In this section we set the minimum and maximum local position the platform will have, taking the PointPlatform object as a reference (PointPlatform object will be the parent of the platform) and therefore, for the platform the pointPlatform object is the point zero (x = 0, y = 0, z = 0) as you can see in the image 21.

The platform's position values are just used to static, horizontal and vertical containers.

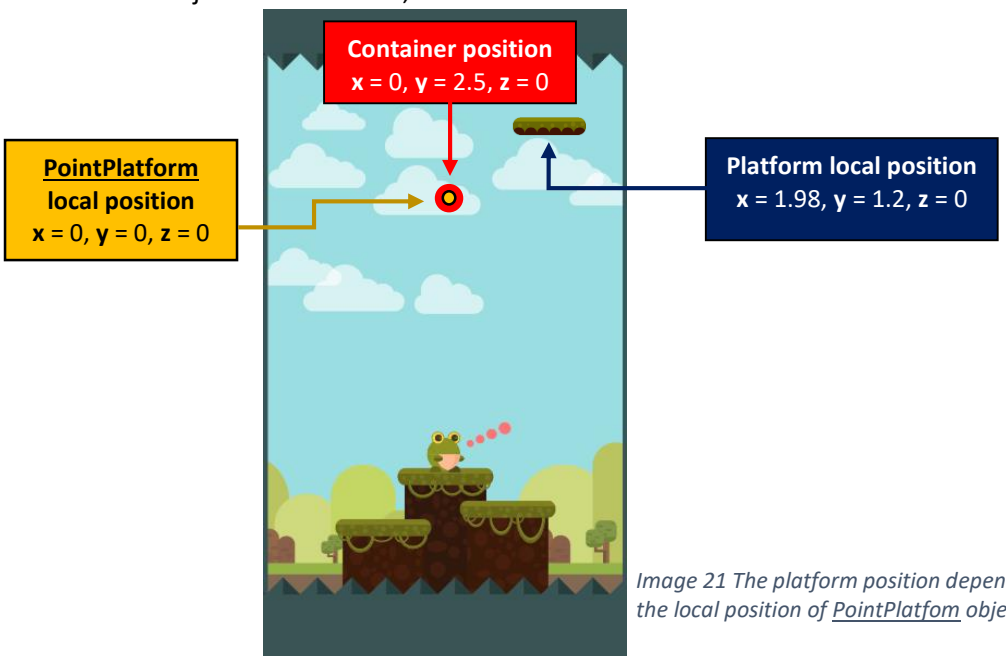


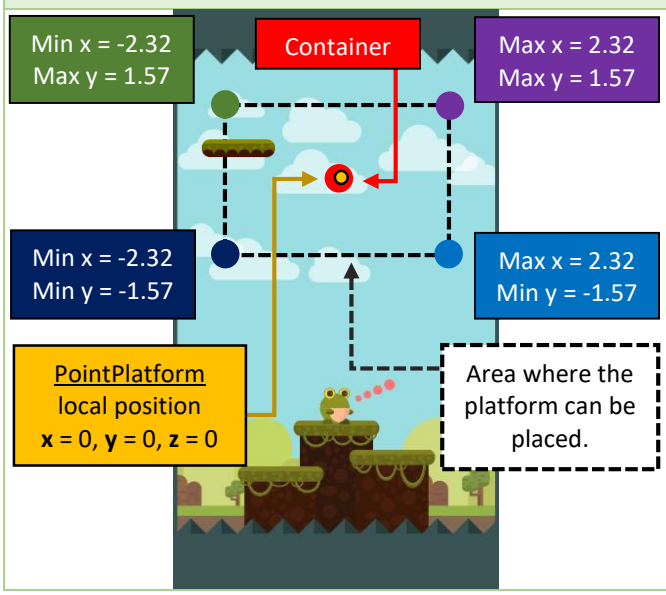
Image 21 The platform position depends on the local position of PointPlatfom object

NOTE: All containers will have this position: x = 0, y = 2.5, z = 0, you can modify this value selecting any container on the hierarchy window and open the script named Container Behaviour which is attach it to the container, then change the value of the variable positionToReach on start method.

Kind of containers

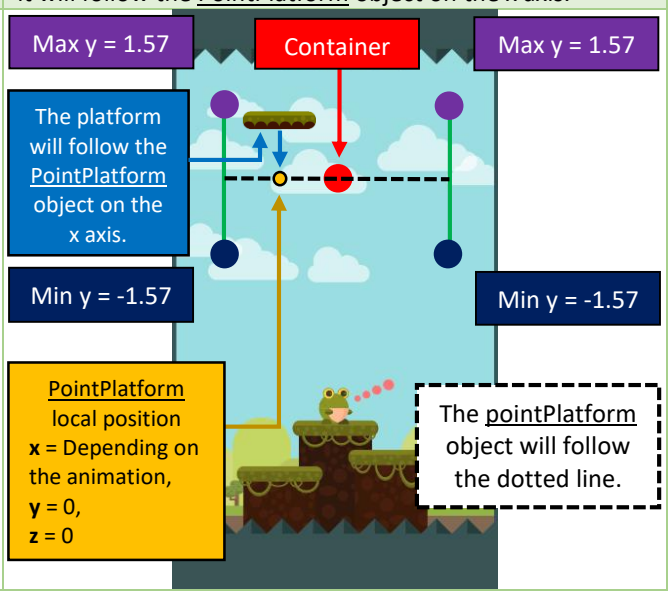
Static container

This container doesn't have animations therefore to place the platform the four values of platform's position section will be used. The platform will be positioned randomly into the dotted area.



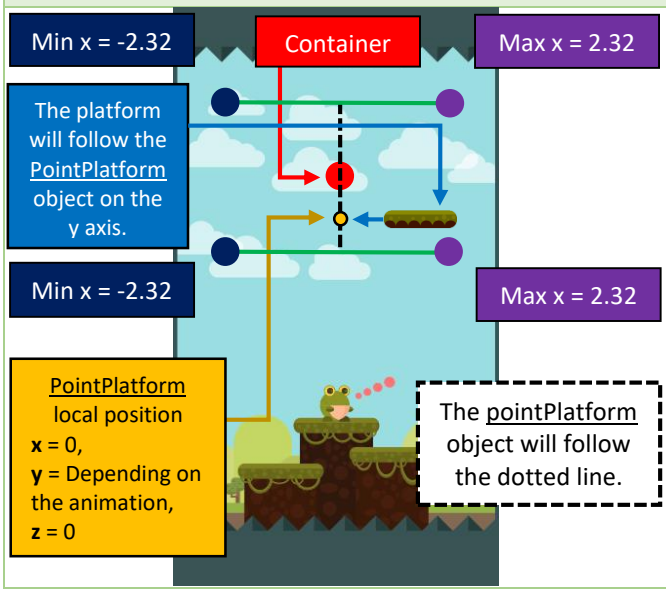
Horizontal container

This container does the PointPlatform object has a movement from left to right and vice versa, for this container we just need the values min y and max y of platform's position section. With the previous two values the platform will be positioned randomly on the y axis and it will follow the PointPlatform object on the x axis.



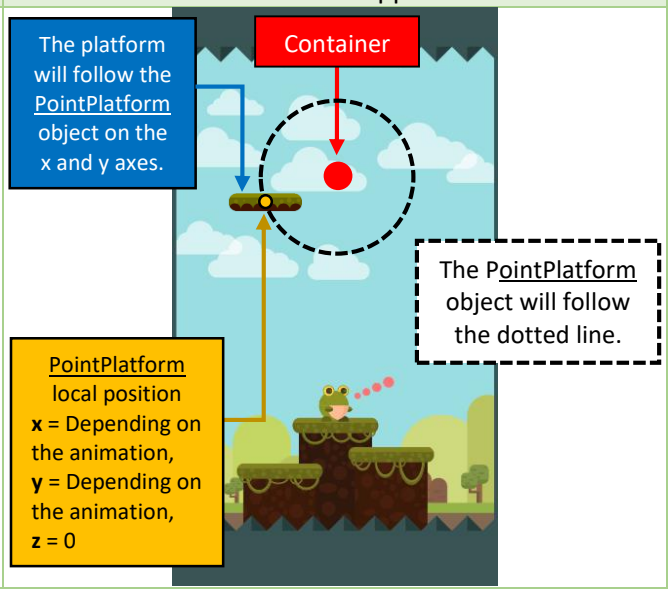
Vertical container

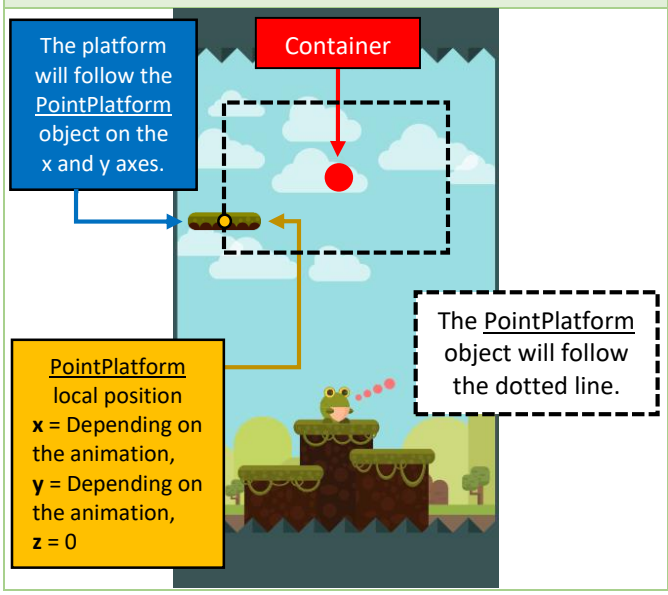
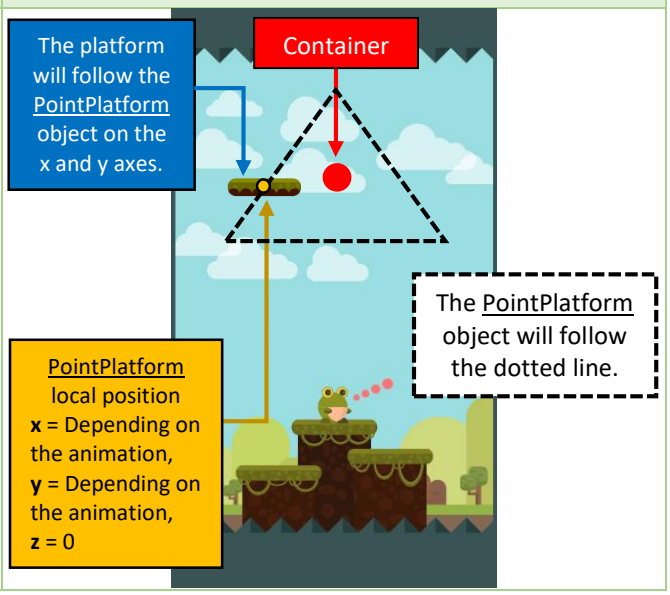
This container does the PointPlatform object has a movement from top to bottom and vice versa, for this container we just need the values min x and max x of platform's position section. With the previous two values the platform will be positioned randomly on the x axis and it will follow the PointPlatform object on the y axis.



Circular container

This container does the PointPlatform object has a circular movement, for this container we don't need any value of platform's position section. The platform will be positioned on the center of the PointPlatform object and randomly the platform will follow the direction of clockwise or the opposite direction.



Rectangular container	Triangular container
This container does the PointPlatform object has a rectangular movement, for this container we don't need any value of platform's position section. The platform will be positioned on the center of the PointPlatform object and randomly the platform will follow the direction of clockwise or the opposite direction.	This container does the PointPlatform object has a triangular movement, for this container we don't need any value of platform's position section. The platform will be positioned on the center of the PointPlatform object and randomly the platform will follow the direction of clockwise or the opposite direction.
	

NOTE: To modify the movement of each container, you have to select the container in the hierarchy window and then open the animation window. See the image 22.

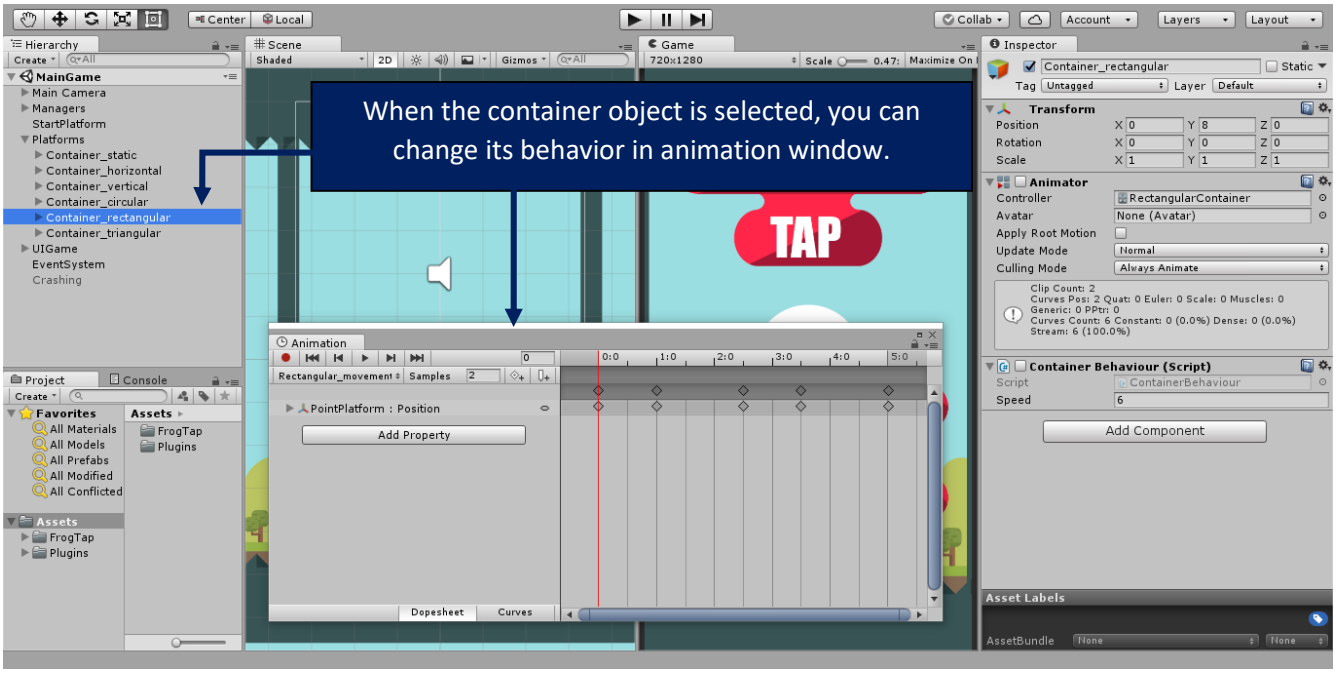


Image 22 Animation of rectangular container

Levels: In this section we set the order and the number of containers that will be used per level along the game.

- **Containers Cycle**: It is a collection of containers and the order in which they will appear. If you want to add more containers just increase the size of this list.
- **Containers Per Level:** The number of containers per level.

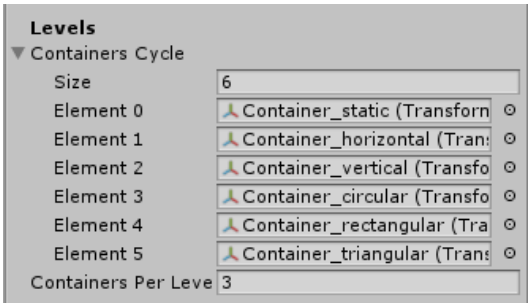


Image 23 Levels section

Difficulty: Each container has the value 1 as a default speed to do its animation, now to increase the difficulty of game the ***level manager*** script will increase the speed of each container.

- **Container Speed:** The order in which the container's speed will increase. If you want to add more variations (speed values) just increase the size of this collection.

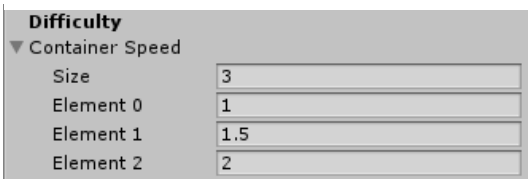


Image 24 Difficulty section

How the levels and difficulty sections works?

- Each element (container) you see in the Containers cycle will be used for a whole level.
- All levels will have a number of containers that you can change in the Containers Per Level value.
- When the exact number of containers per level has appeared, the ***level manager*** script will get the next element (container) in the collection Containers cycle to create the next level.
- When the last element in the collection Containers cycle is used, the collection is restart it to use the first element again, then we say that the cycle has finished.
- When the cycle has finished, the game's difficulty will increase, here the ***level manager*** script will get the next value in the Container Speed collection and it will change the speed of each container.
- When the last value (speed) in the collection Container Speed is used, the collection is restart it to use the first value again.

Example of the level generation system				
# Cycle	**# Level**	**# Containers**	**Container type**	**Animation's speed**
1	1	3	Static	1
	2	3	Horizontal	
	3	3	Vertical	
	4	3	Circular	
	5	3	Rectangular	
	6	3	Triangular	
	Total levels: 6	**Total containers:** 18		**First speed value**
The cycle ended				
2	7	3	Static	1.5
	8	3	Horizontal	
	9	3	Vertical	
	10	3	Circular	
	11	3	Rectangular	
	12	3	Triangular	
	Total levels: 6	**Total containers:** 18		**Second speed value**

The cycle ended				
3	13	3	Static	2
	14	3	Horizontal	
	15	3	Vertical	
	16	3	Circular	
	17	3	Rectangular	
	18	3	Triangular	
	Total levels: 6	**Total containers:** 18		**Third speed value**
The cycle ended				
4	19	3	Static	1
	20	3	Horizontal	
	21	3	Vertical	
	22	3	Circular	
	23	3	Rectangular	
	24	3	Triangular	
	Total levels: 6	**Total containers:** 18		**First speed value used again**
The cycle ended				
5	25	3	Static	1.5
	26	3	Horizontal	
	27	3	Vertical	
	28	3	Circular	
	29	3	Rectangular	
	30	3	Triangular	
	Total levels: 6	**Total containers:** 18		**Second speed value used again**

Reskin Manager

The reskin manager script has the main responsibility of change the game's skin when necessary.

Image 25 Reskin manager script

All the tasks of this script are listed below:

- If there is information stored of a character saved as selected, this script will create that character saved, if there is not information saved the default character will be created.
- Change the game's skin if the game loads for the first time and if the character saved as selected is different to default character.
- Change the game's skin when the user selects a new character in shop menu.
- Depending on the character selected, create background elements like clouds, planets or ghosts.
- Depending on the character selected, create the items (flies) which will be collected along the game and then used by the player to buy new characters.

Reskin manager's sections

Background: Here the script needs a reference to background object which is located in the hierarchy window, as you can see in the image 27. The reference is necessary to change the sprite (image) of the object when a new character is selected.

Image 26 Background sprite renderer reference

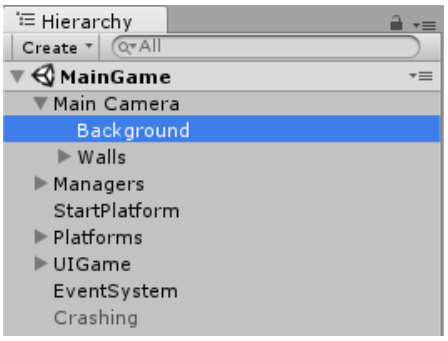


Image 27 Background object in hierarchy window

Worlds: It is the collection of worlds that will be used along the game, here there is a world per character as you can see at the below table, if you want to add more worlds just increase the size of this list.

Currently the game have four worlds and the first element you see in the Worlds collection will be taken as the default world. See the image 28.

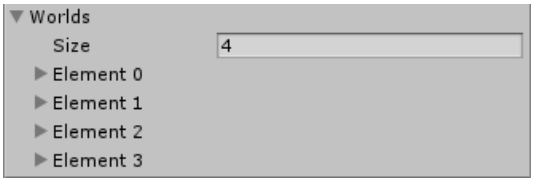
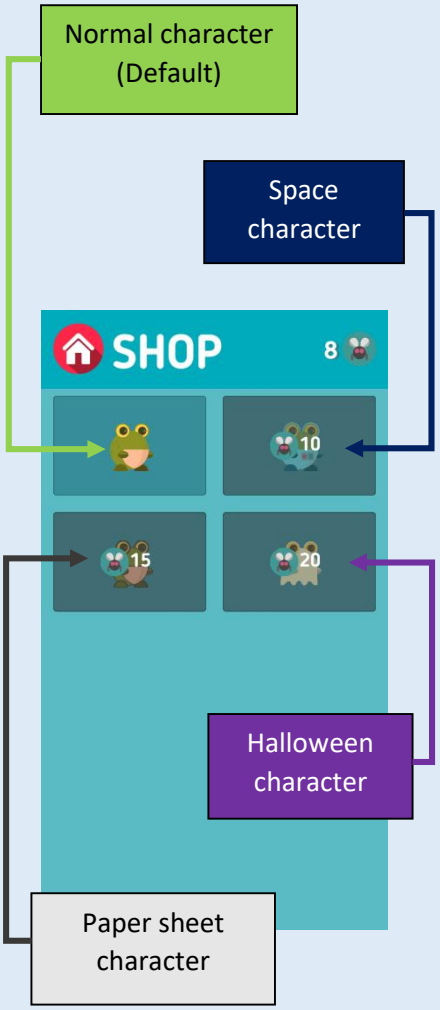
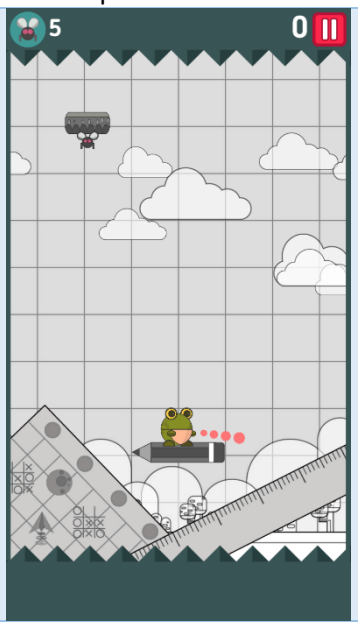


Image 28 Worlds collection

Game's worlds

Characters in shop menu	Game's Worlds	
	Normal world – Default world	Space world
	Paper sheet world	Halloween world





World element

A world element is an object that contain all the information necessary to make the game changes its appearance but while retaining the game's mechanic as you can see in the below table.

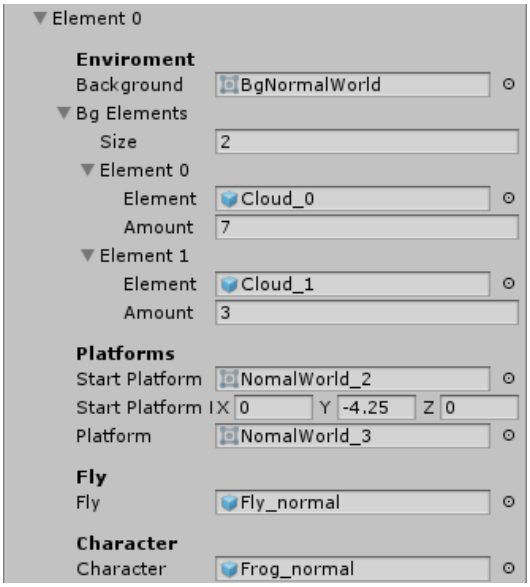


Image 29 World element

World's sections

Environment

- Background: The sprite (image) for the background.

Bg Elements: Collection of backgrounds elements like: clouds, planets and ghosts.

- Element
  - ❖ Element: Background element prefab to instantiate.
  - ❖ Amount: The number of background elements to create.

Platforms

- Start platform: The sprite for the start platform.
- Start platform position: The start position for the start platform.
- Platform: The sprite for all platforms except the start platform.

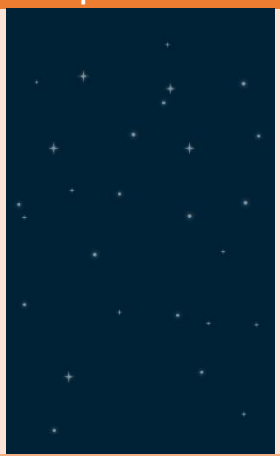
Fly

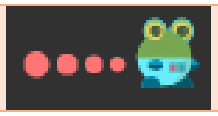
- Fly: The fly prefab to instantiate, the ***level manager*** script will create one fly per platform.

Character

- Character: The character prefab to instantiate.

NOTE: All the images used in the game are in the folder Sprites located into the folder FrogTap.

World's elements	Normal world	Space world	Halloween world
Background			
Elements	 		 

Start platform			
Platform			
Fly			
Character			

How the shop menu works?

If you have added a new world and you want to sell it as you can see in the image 30, you need to add the world's character into the shop menu, following the next steps:

- Into the hierarchy window display the UIGame object.
- Display the MenuShop object.
- Display Scroll View object.
- Display Content Panels object.
- Duplicate the object Panel Frogs (1), to duplicate the object just select the object and use the commands: Ctrl + D.

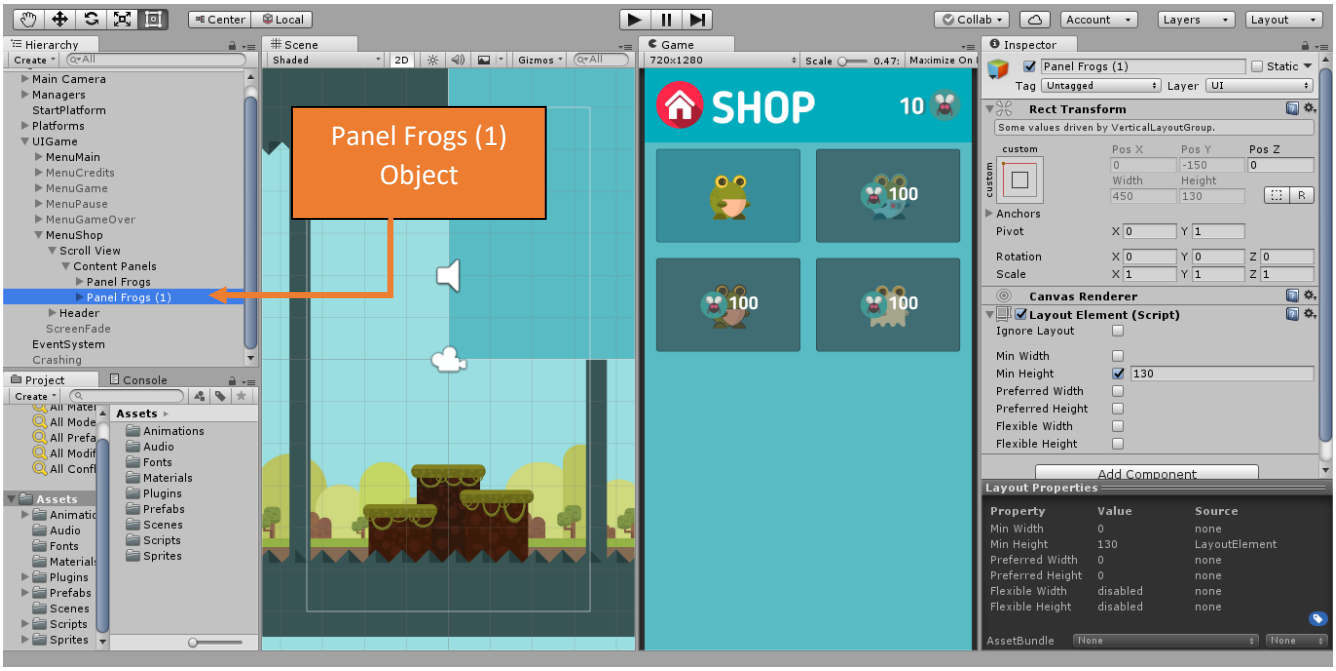


Image 30 Shop menu items

- Display the object duplicated Panel Frogs (2), then select the Frog object and in the bottom of the inspector window, you will see a script attached, see the image 31. This script requires the following information:
    - ❖ **Panel price:** This object is required to see the character as locked. For the default character select this object and disable it.
    - ❖ **Txt price:** This object is required to show the price of this product.
    - ❖ **Frog:** The character prefab to instantiate, this one needs to be the same character you set in the world created.
    - ❖ **Price:** The price for this product. For the default character set the value zero.

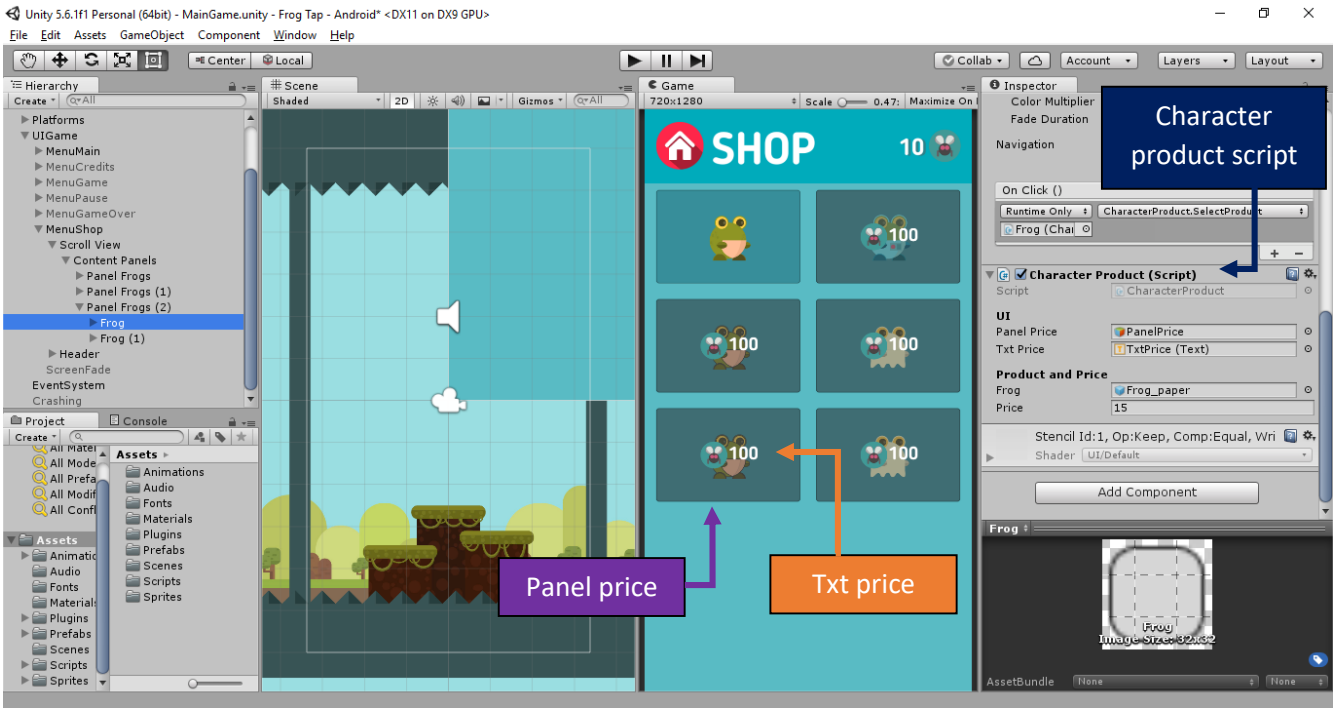


Image 31 Character product script

Native Share Setup

The native share script will give the end user the possibility to make two tasks:

- Share a screenshot of your game with a descriptive text.
- Rate your game in the store.

NOTE: The native share works on Android, IOS and Windows phone 8.1 Applications.

In the Project window there is a folder named Plugins which contain a script called **"NativeShare"**, open that script and update the following variables with the information of your game, see the image 32.

- **_applicationAndroidID**: This variable is the name of your package, you can find it following this route: menu Edit -> Project Settings -> Player -> Player Settings -> Other settings -> Identification -> Package name.
- **_applicationIOSID**: When you send the game to the App Store this variable is provided.
- **_applicationWindowsPhoneID**: When you send the game to the Windows Store this variable is provided.

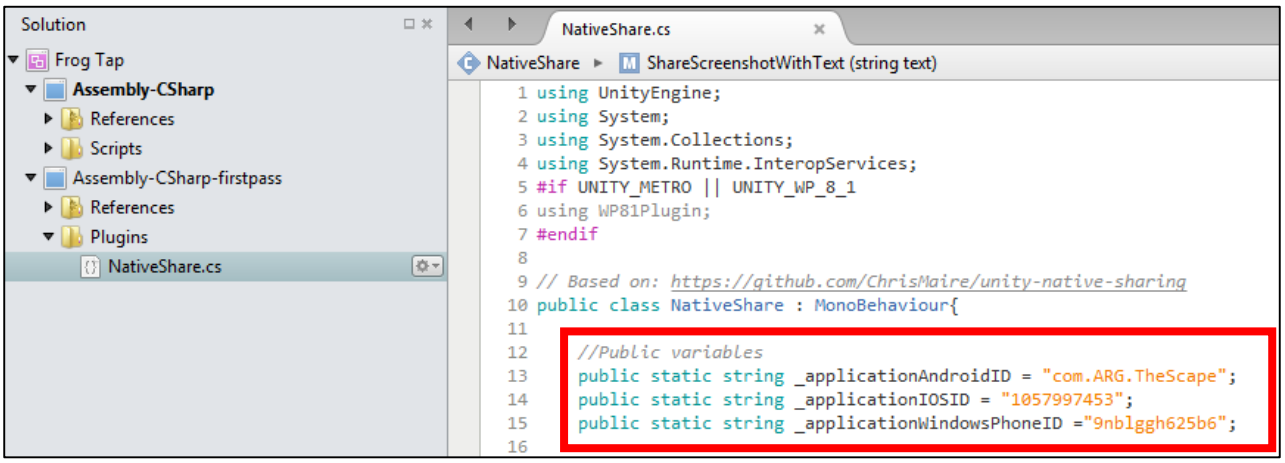


Image 32 Public variables of Native share script

Script's methods

In the ***Game controller*** script you will find two public methods Rate Game and Share Game.

- The rate game function will open the virtual store application to rate this game.

```
public void RateGame(){
    PlayButtonSound ();
    NativeShare.Rate ();
}
```

Image 33 Rate game method

- The share game function will take a screenshot of game and then it will add the text provided (for example the score reached) plus the URL of the store where this game is published.

```
public void ShareGame(){
    PlayButtonSound ();                          //Play click button sound.
    //This method will take a screenshot of game and it will add text with the url of the store where
    StartCoroutine (NativeShare.ShareScreenshotWithText ("OMG! I have reached " + score.ToString () +
    " points in Frog Tap! Can you beat my score?"));
}
```

Image 34 Share game method

With the previous method you can get the image 35.

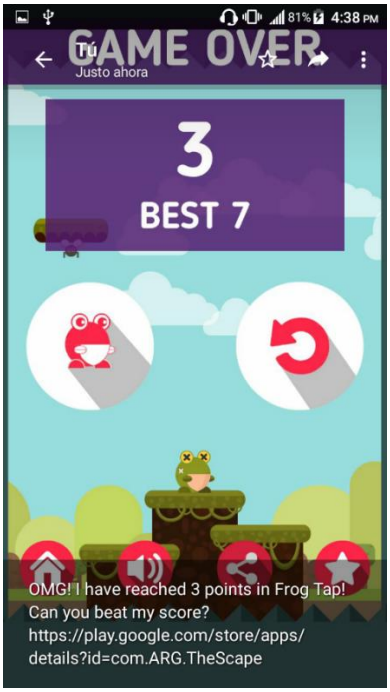


Image 35 Game's screenshot

Specific setup

Android setup

For android build, the game needs write permission to store the screenshot, by default the option *"External (SDCard)"* is selected, you need to change this option by *"External (SDCard)"* to make this change follow this route: File -> Build Settings -> Player Settings -> Other Settings -> Write Permission -> External (SDCard).

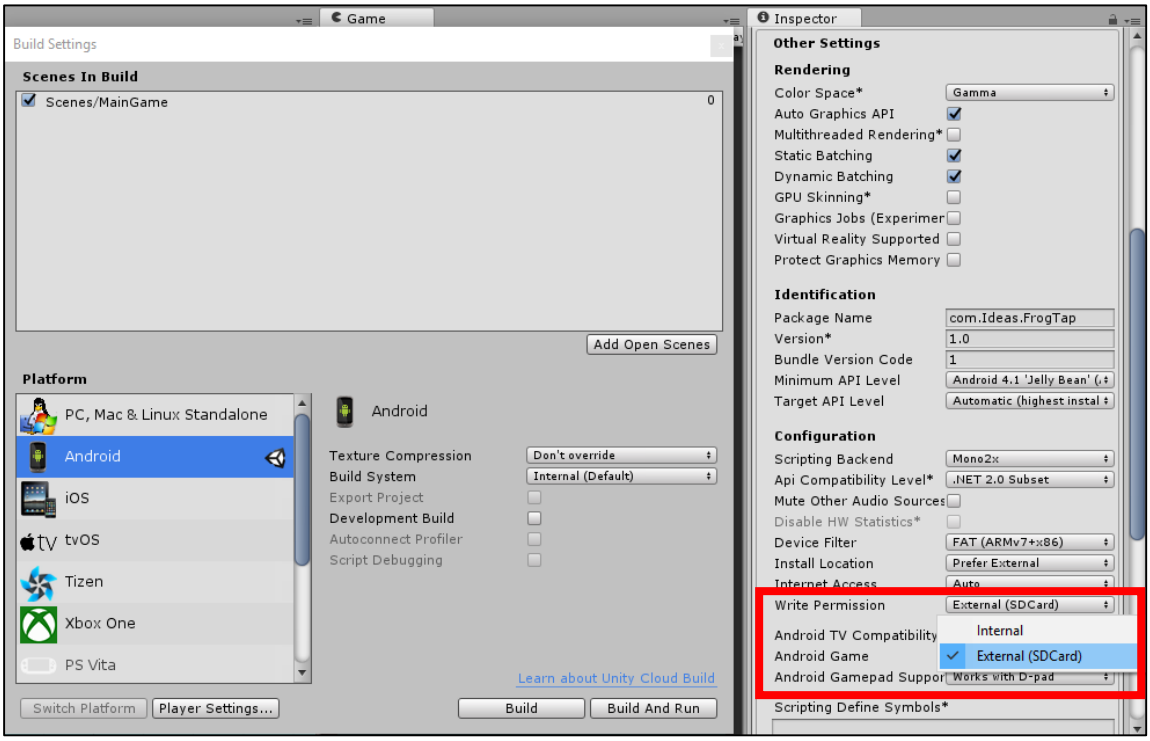


Image 36 Write permissions options

Windows Phone 8.1 Setup

Build the project for windows phone 8.1 as you can see in the image 37.

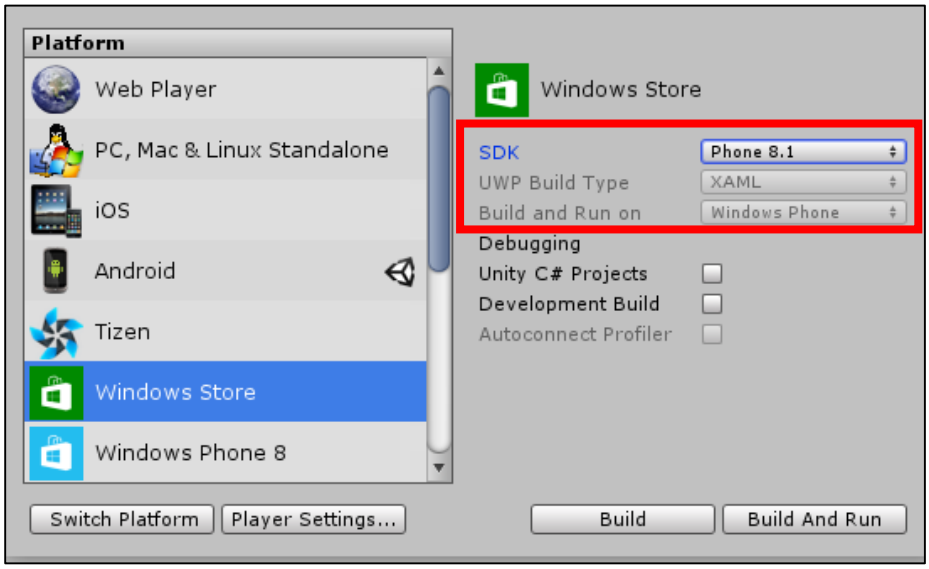


Image 37 Windows phone 8.1 settings

Finally In visual studio, select the option *"Master".* See the image 38.

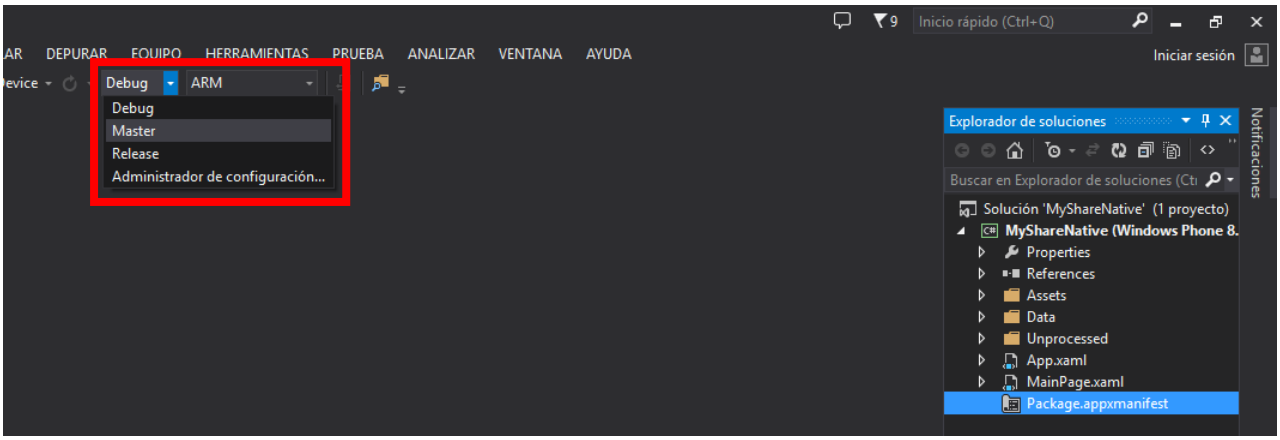


Image 38 Visual studio setting

Credits

Some resources like sounds, one font and one animation were downloaded from external pages. You can use these resources for free and for commercial use as long as the author is credited.
In this section you will find all the information and the license for each resource used.

Sounds

Level Up 01 by Andy Rhode http://freesound.org/people/rhodesmas/sounds/320655/
Licensed under Creative Commons: By Attribution 3.0 License
https://creativecommons.org/licenses/by/3.0/

bubble (6).wav by Glaneur de sons http://freesound.org/people/Glaneur%20de%20sons/sounds/104945/
Licensed under Creative Commons: By Attribution 3.0 License
https://creativecommons.org/licenses/by/3.0/
Change made: Cut from original

Slime Jump by Louis http://freesound.org/people/DrMinky/sounds/167045/
Licensed under Creative Commons: By Attribution 3.0 License
https://creativecommons.org/licenses/by/3.0/
Change made: Cut from original

Font

Blogger Sans by Sergiy Tkachenko (FirstSiteGuide) https://firstsiteguide.com/new-blogger-sans-font/
Licensed under Creative Commons: By Attribution 4.0 License
https://creativecommons.org/licenses/by/4.0/

Animation

Smoke_up by Pow Studios http://powstudios.com/content/smoke-animation-pack-1
Licensed under Creative Commons: By Attribution 3.0 License
https://creativecommons.org/licenses/by/3.0/

Support

For questions about this asset you can contact me by e-mail.
pablo.huaxteco@gmail.com

Please rate my asset on the asset store, I will appreciate it.